Prasenjit Majumder Mandar Mitra
Pushpak Bhattacharyya L. Venkata Subramaniam
Danish Contractor Paolo Rosso (Eds.)

LNCS 7536

Multilingual Information Access in South Asian Languages

Second International Workshop, FIRE 2010 Gandhinagar, India, February 2010 and Third International Workshop, FIRE 2011 Bombay, India, December 2011, Revised Selected Papers

Springer

Lecture Notes in Computer Science 7536

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Prasenjit Majumder Mandar Mitra
Pushpak Bhattacharyya
L. Venkata Subramaniam Danish Contractor
Paolo Rosso (Eds.)

Multilingual Information Access in South Asian Languages

Second International Workshop, FIRE 2010
Gandhinagar, India, February 19-21, 2010 and
Third International Workshop, FIRE 2011
Bombay, India, December 2-4, 2011
Revised Selected Papers

Springer

Volume Editors

Prasenjit Majumder
Dhirubhai Ambani Institute of Information and Communication Technology
Gujarat, India
E-mail: prasenjit.majumder@gmail.com

Mandar Mitra
Indian Statistical Institute, Kolkata, India
E-mail: mandar.mitra@gmail.com

Pushpak Bhattacharyya
Indian Institute of Technology, Bombay, India
E-mail: pushbakbh@gmail.com

L. Venkata Subramaniam
Danish Contractor
IBM Research, New Delhi, India
E-mail:{lvsubram, dcontrac}@in.ibm.com

Paolo Rosso
NLE Lab - ELiRF, Universitat Politècnica de València, Spain
E-mail: prosso@dsic.upv.es

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-40086-5 e-ISBN 978-3-642-40087-2
DOI 10.1007/978-3-642-40087-2
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013944224

CR Subject Classification (1998): H.3, I.2.7, I.7

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper
Springer is part of Springer Science+Business Media (www.springer.com)

Multilingual Information Access in South Asian Languages

Preface

The Forum for Information Retrieval Evaluation (FIRE — `http://www.isical.\breakac.in/fire`) aims to provide a common platform for evaluating information access technologies with a focus on South Asian languages, by creating Cranfield-style test collections in the same spirit as TREC, CLEF, NTCIR, etc. The first evaluation exercise conducted by FIRE was held in 2008. This volume brings together revised and expanded versions of 29 papers that were presented at the second and third FIRE workshops held in 2010 and 2011.

For FIRE 2010 (held during February 19–21, 2010), four tasks were planned, including two pilot tracks. Eventually, however, submissions were received for only the ad-hoc monolingual and cross-lingual document retrieval task.

A total of seven tasks were planned for FIRE 2011 (held during December 2–4, 2011). Two tasks were dropped later. This volume includes papers from the following tasks:

1. Ad-hoc. Its objective was to evaluate the effectiveness of retrieval systems in retrieving accurate and complete ranked lists of documents in response to 50 one-time information needs.
2. CLITR (Cross-Language Indian Text Reuse). This task dealt with the identification of highly similar journalistic articles and news stories in a cross-language setting.
3. SMS-based FAQ retrieval. The goal of this task was to find a question from a collection of FAQs (frequently asked questions) that best answers/matches a query received via SMS.
4. RISOT (Retrieval from Indic script OCR'd text). This task looks at retrieval from a (noisy) document collection created using OCR.
5. Personalized IR. The primary objective of the task was to retrieve more relevant information for a particular user by making use of the logged actions of other users who had entered similar queries to the system.

FIRE is coordinated by the Information Retrieval Society of India (www.irsi.res.in) and supported by the Department of Information Technology, Government of India, and has also received funding from Google, HP, IBM Research, Microsoft Research Society for Natural Language Technology Research, and Yahoo! India Research and Development. We should like to thank the members of the FIRE Steering Committee for their advice and encouragement. The invaluable assistance provided by Sauparna Palchowdhury and Rashmi Sankepally in preparing

this volume is gratefully acknowledged. Finally, we are thankful to all our participants, and particularly to the contributors of this volume. Our apologies to them for the inordinate delay in publishing this collection of papers.

April 2013

Prasenjit Majumder
Mandar Mitra
Pushpak Bhattacharyya
L. Venkata Subramaniam
Danish Contractor
Paolo Rosso

Table of Contents

FIRE 2011

Adhoc Track

CLiTR - Cross Lingual Text Reuse Track

SMS Based FAQ Retrieval Track

RISOT - Retrieval from Indic Script OCR'd Text Track

RISOT - Retrieval from Indic Script OCR'd Text Track

Simple Transliteration for CLIR

FIRE 2010

Overview of FIRE 2011

Sauparna Palchowdhury[1], Prasenjit Majumder[2], Dipasree Pal[1],
Ayan Bandyopadhyay[1], and Mandar Mitra[1]

[1] Indian Statistical Institute, Kolkata, India
[2] DA-IICT, Gujarat, India
{sauparna.palc,prasenjit.majumder,bandyopadhyay.ayan,
mandar.mitra}@gmail.com,
dipasree_t@isical.ac.in

Abstract. We provide an overview of FIRE 2011, the third evaluation exercise conducted by the Forum for Information Retrieval Evaluation (FIRE). Our main focus is on the Adhoc task. We describe how the FIRE 2011 test collections were constructed. We also provide a brief overview of the approaches adopted by the Adhoc task participants.

1 Introduction

The third FIRE workshop was held at IIT Bombay, from 2nd – 4th December, 2011, bringing together a growing IR community in India. The large number of downloads of the FIRE test collections over the web — more than in any of the previous iterations of the FIRE campaign — is an indication of the community's interest. The following six tracks were offered this time:

- Adhoc monolingual / cross-lingual retrieval
  - documents in Bengali, Gujarati, Hindi, Marathi, Tamil and English
  - queries in Bengali, Gujarati, Hindi, Marathi, Tamil, Telugu and English
- SMS-based FAQ Retrieval
- Cross-Language Indian Text Reuse (CL!TR)
- Personalised IR (PIR)
- Retrieval from Indic Script OCR'd Text (RISOT)
- WSD for IR
- Adhoc Retrieval from Mailing Lists and Forums (MLAF).

The last two tracks were eventually discontinued because of a lack of manpower / participation.

FIRE is funded primarily by the TDIL (Technology Development in Indian Languages) group, housed within the Department of Information Technology, Government of India, with a mandate of creating, in phases, test collections for the 23 "official" languages used in India. Corpus creation in Indian languages requires some amount of familiarity with these languages. Multiple Indian institutes, representing different language verticals, constitute a consortium that

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 1–12, 2013.

produces evaluation resources — corpora, queries and relevance judgements — for these languages.

In the next section (Section 2) we provide statistics about the Adhoc collections, including an institute wise break-up of the sources of these data. Section 2.3 attempts to take a closer look at the FIRE queries and the relevance judgements by providing a set of charts profiling the queries. This gives us some idea about how the FIRE collections have evolved over three campaigns. Section 3 provides a survey of the Adhoc runs submitted. Section 4 concludes this overview with a discussion.

2 Adhoc Task Data

This is the third year of the Adhoc task. Five Indian language collections (Bengali, Gujarati, Hindi, Marathi and Tamil) — out of an eventual target of 23 — were offered. The size and coverage of document collections have increased significantly this year. For example, the Bengali and Hindi document collections have grown about three-fold. Bengali news articles from Bangladeshi sources have been included in the collection. Usage of Bengali in Bangladesh (a Bengali speaking country) is somewhat different from its usage in the Eastern Indian states. The augmented corpus should thus provide some flavour of the intra-lingual diversity present in Bengali. The Hindi corpus has been a matter of concern over the last two years because of the presence of noise, arising mostly due to errors that occurred while converting from proprietary character encoding schemes to UTF-8. This year, a completely new Hindi corpus sourced from one of the largest national Hindi dailies replaced the previous corpus. Further, two new languages, Gujarati and Tamil were added. Like the other corpora, the Gujarati corpus was also created from online news articles crawled from a major Gujarati daily. This corpus is comparable to the Hindi and Bengali corpora in terms of size, timeline and coverage. Table 1 shows an institute wise break-up of the sources of data used in the Adhoc task.

2.1 Query Formulation

A pool of 100 topics related to various social, political, cultural and other events was generated. Manual interactive search was performed using those topics on the Bangla and Hindi corpora. Each collection was indexed using the desktop version of Terrier [1] and manual judgements were done for all of the 100 topics to get a fair estimate of the recall base. The TREC norm of having at least 5 relevant documents per topic was followed while shortlisting the topics. Finally, 50 topics were selected based on their complexity in terms of information need and recall base. Topics were then translated manually into English, Gujarati, Marathi, Tamil and Telugu.

Table 1. Corpus sources

Language	Source	Time span	Institute
Bengali	*Anandabazar Patrika*[1]	2004-2010	ISI Kolkata
	BDNews24	2006-2010[2]	
English	*BDNews24*	2007-2010[2]	ISI Kolkata
	The Telegraph	2001-2010	
Gujarati	*Gujarat Samachar*	2002-2010	DAIICT
Hindi	*Amar Ujala*	2004-2008	DAIICT
	Navbharat Times	2002-2010	
Marathi	*eSakal*	2004,Sep-2007,Sep	IIT Bombay
	Maharashtra Times		
Tamil	*BBC*	2004-2011	IIT Kharagpur
	Dinamalar	2010-2011	

Table 2. Corpus statistics

Language	# docs.	Size (GB)	# Distinct words	Median doc. size (bytes/chars)	Min. doc. size (bytes/chars)	Max. doc. size (bytes/chars)
Bengali	457370	3.4	1251522	4860/1876	67/67	293590/138132
English	392577	1.8	483889	2445	99	35734
Gujarati	313163	2.7	2054427	5019/1993	200/129	151529/58003
Hindi	331599	1.9	442836	3224/1354	152/126	264831/264139
Marathi	99275	0.7	854506	4353/1701	590/296	93272/35496
Tamil	194483	1.0	659476	2666/1032	72/72	29078/10866

2.2 Relevance Judgements

Although there was a great deal of interest in downloading the FIRE data, the number of Adhoc track participants came down significantly this time. As there were chances that the official submissions would not be sufficient for the creation

[1] Documents in the range September 2004 – December 2004 and January 2007 – September 2007 were inadvertently left out of the collection this year. The set in its entirety will be offered next year (2012).

[2] The date for each document is specified in the file name, having been extracted from the HTML web page. Oddly, 1970 and 1999, appear. We believe these were erroneous annotations introduced by the source systems, even more so, as 1970 is the beginning of the UNIX epoch.

of a diverse pool, we did pre-pooling for Gujarati, Hindi and English using the official queries. Stopwords were removed and YASS [2] was used for stemming the Indian language texts. Official submissions include 14 Bengali, 18 Marathi and 2 English runs. After the submission deadline, 4 runs each for Bengali and Hindi, and 6 Gujarati runs were added. The pool was constructed using both official and "unofficial" submissions. A pool depth of 130 was chosen for Bengali and Hindi keeping in mind the available manpower, whereas for Gujarati, the pool depth was 200. The pools for Bengali and Hindi were further enriched through manual interactive retrieval by the assessors. No relevance assessments could be done for Tamil due to the lack of manpower.

Table 3 summarises the pooling statistics. The minimum and maximum pool size across queries are shown in Table 4. For Bengali and Hindi, each document was judged by two assessors and conflicts were resolved over several sessions to completely remove disagreements across assessors. Table 5 shows the number of relevant documents in Bengali, English, Gujarati, Hindi and Marathi.

Table 3. Pooling at FIRE 2011

Lang.	Preliminary		FIRE Submissions	
	# runs	depth	# runs	depth
Bengali	-	-	14	130
English	2	130	2	130
Gujarati	12	200	15 (unofficial)	200
Hindi	4	130	-	-
Marathi	-	-	18	20

Table 4. Pool size across queries

	Bengali	English	Gujarati	Hindi	Marathi
Minimum	174	154	126	0	32
Maximum	484	297	380	0	151
Total	15561+	10601	11712	0	3503

Table 5. Number of relevant documents in various languages

	Bengali	English	Gujarati	Hindi	Marathi
Minimum	7	11	4	7	0 (14)
Maximum	199	123	97	404	62
Mean	55.56	55.22	33.18	81.68	7.08
Median	49	53	30	63	2
Total	2778	2761	1659	4084	354
FIRE 2010	510	653	-	915	621
FIRE 2008	1863	3779	-	3436	1095

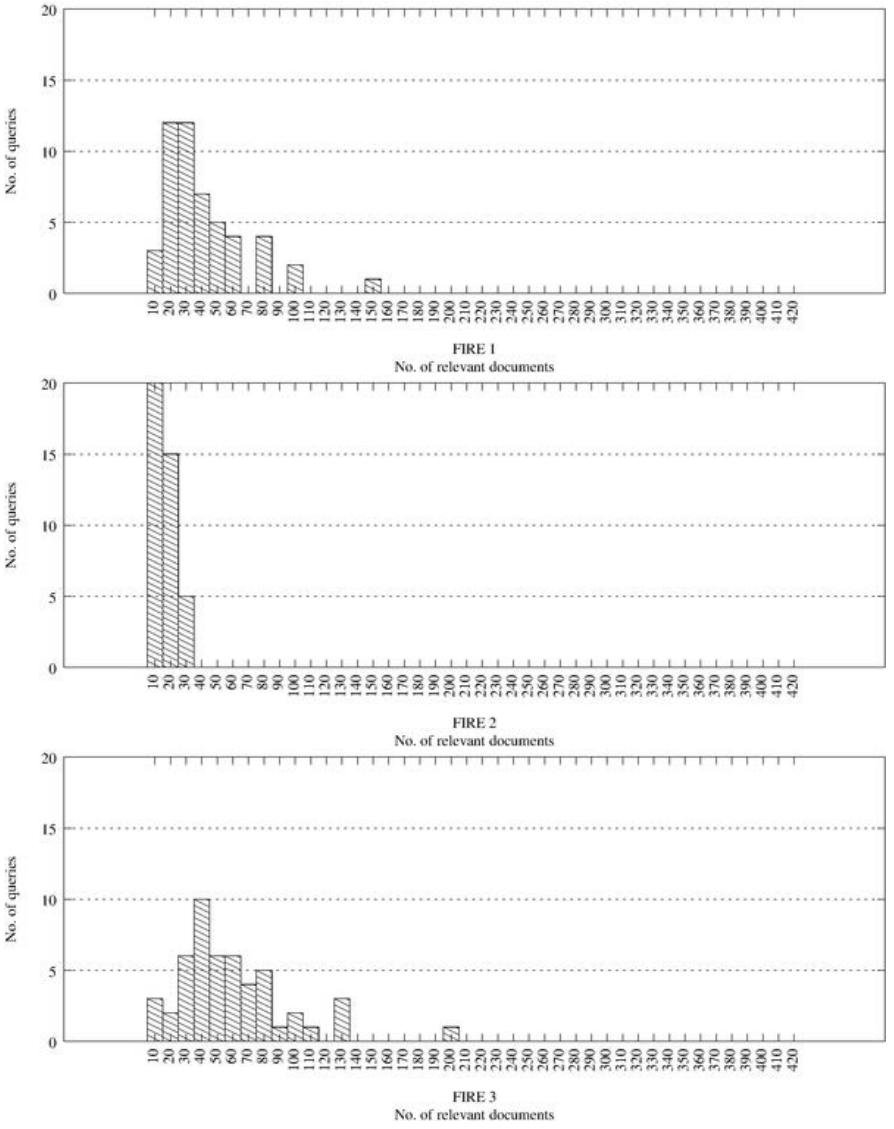


Fig. 1. Bengali

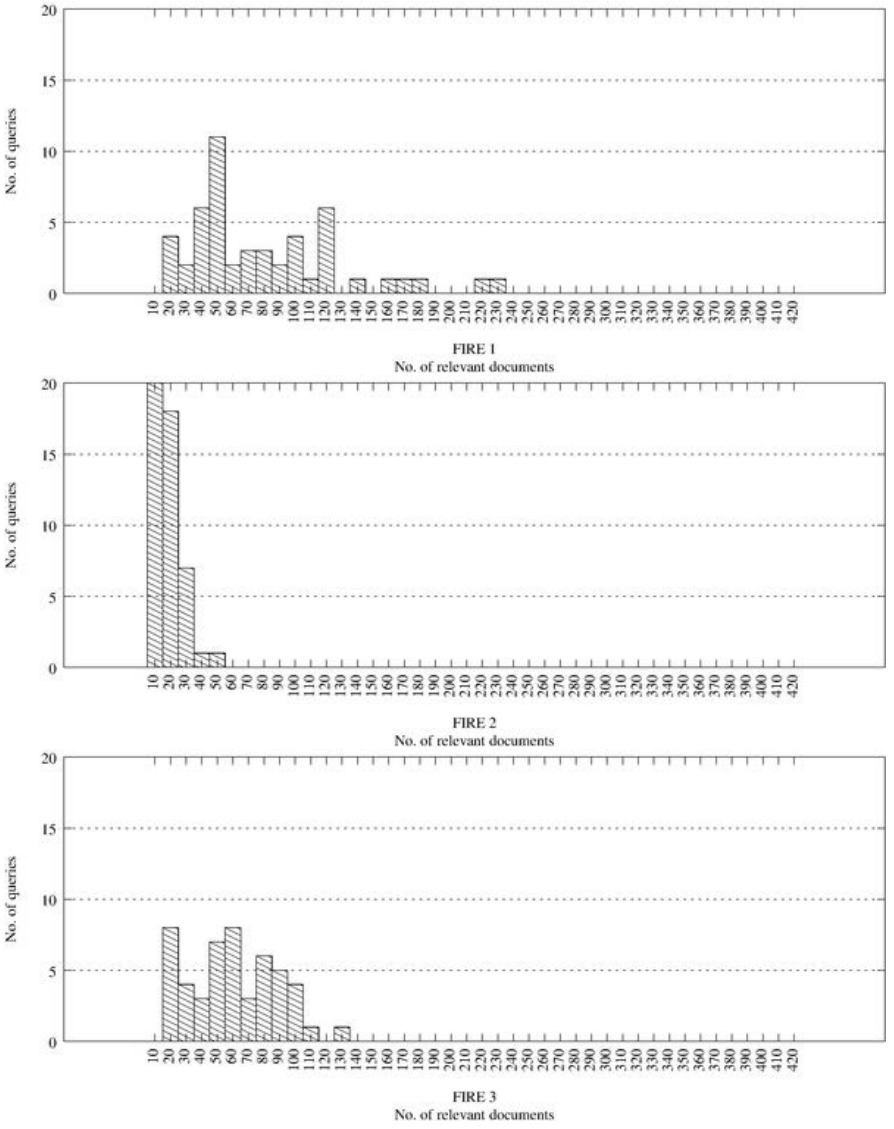


Fig. 2. English

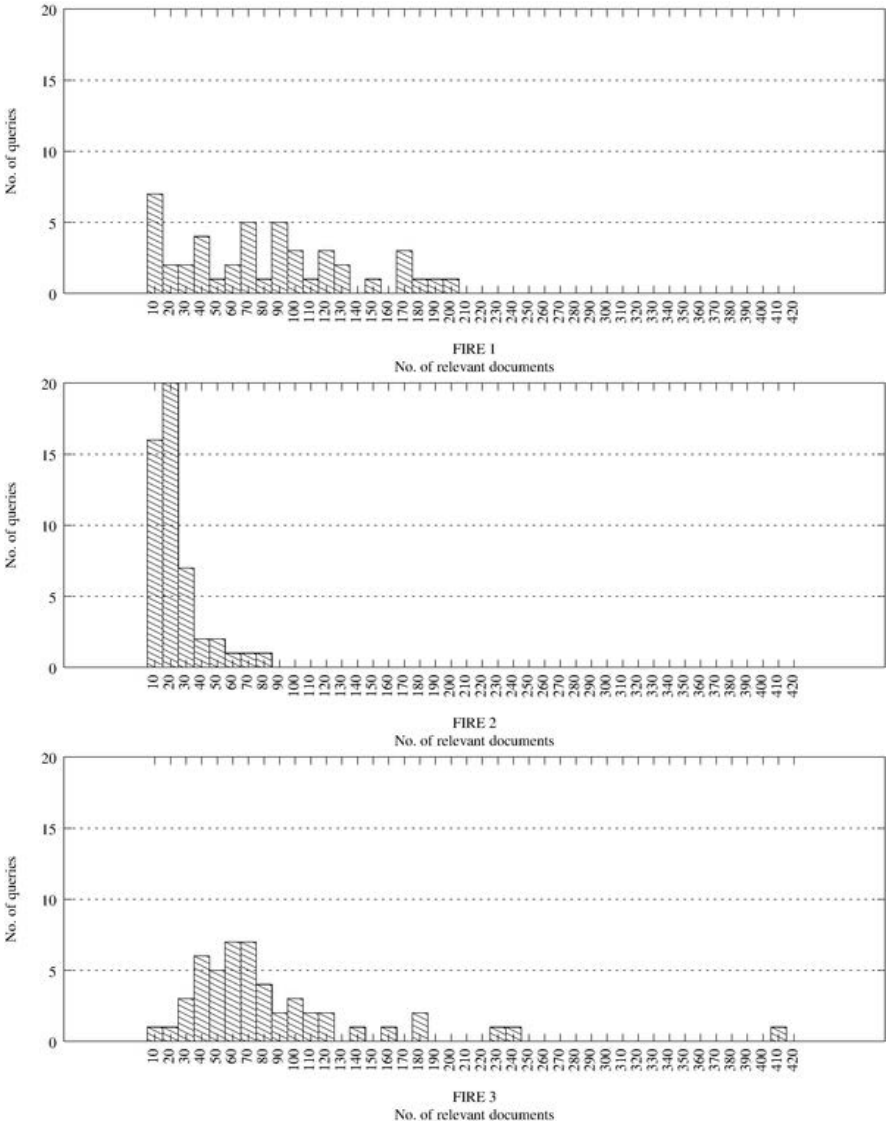

No. of queries
FIRE 1
No. of relevant documents
No. of queries
FIRE 2
No. of relevant documents
No. of queries
FIRE 3
No. of relevant documents

Fig. 3. Hindi

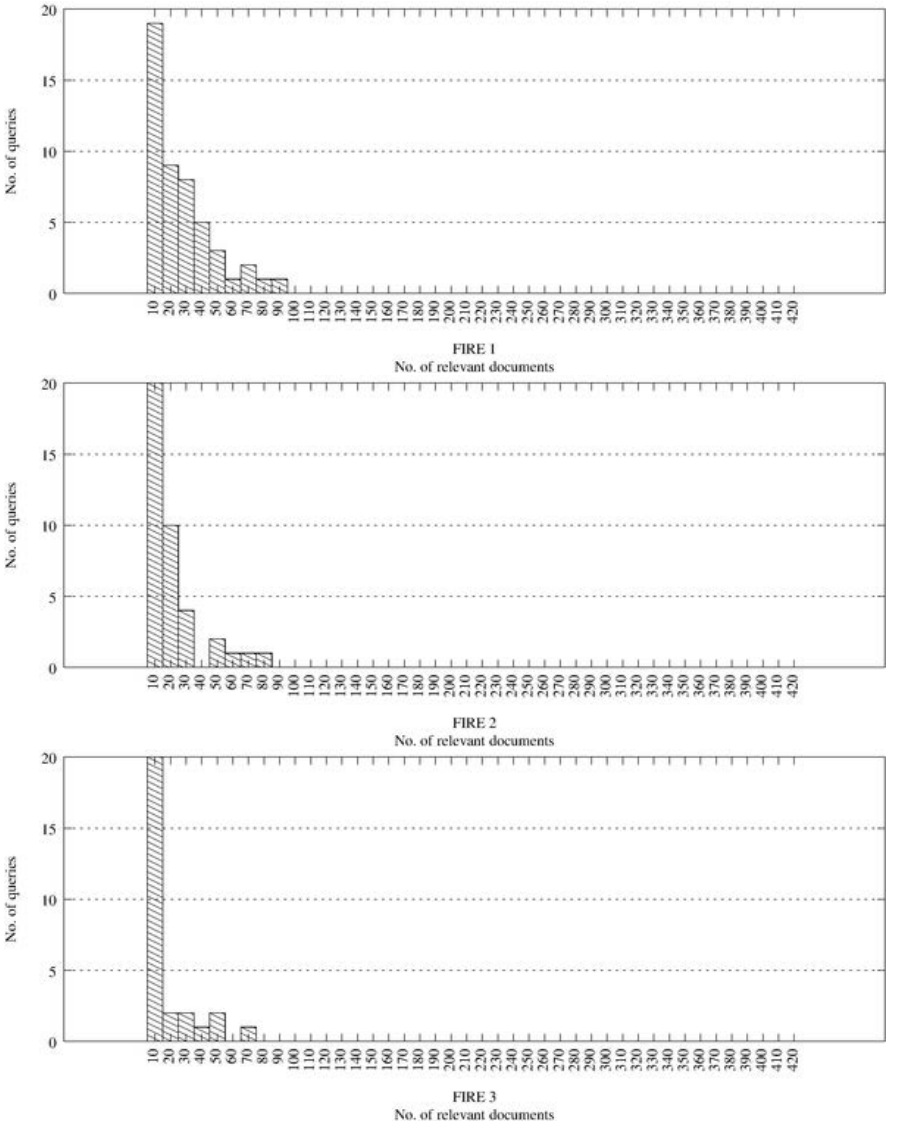


Fig. 4. Marathi

2.3 Query Profile

Voorhees and Harman ([3](Section 3.2.2), [4](Section 5.4)) analysed the Adhoc query sets for TREC 5 and TREC 6 on the basis of the number of relevant documents (NR), number of relevant retrieved documents (NRR), and MAP values. A query-hardness factor was computed, but this was not found to be correlated with NRR.

In this section, we make a *very* preliminary attempt in this direction, by looking at the number of relevant documents across various queries and across the FIRE campaigns. Figures 1– 4 show graphs generated from the FIRE qrel files. The x-axis corresponds to the number of relevant documents (NR), while the y-axis corresponds to the number of queries. More precisely, the bars corresponding to 10, 20, 30, ... on the x-axis show the number of queries with, respectively, less than 10, 10–19, 20–29, ... relevant documents. For each language, data for the three FIRE years — 2008, 2010 and 2011 — have been shown in order.

Of course, a direct comparison across years would not be justified because the target corpus was modified over the years. In order to obtain insights about the collections, a more careful analysis is needed.

3 Runs Submitted

There were 34 official runs submitted and all were monolingual. It appears that the dearth of translation resources was responsible for the lack of cross-lingual experiments. This section presents a brief summary of the runs submitted.

Paik et al. identified morphological variations in Bengali and Marathi by Frequent Case Generation (FCG) [5]. This method is based on the skewed distribution of word forms in natural languages and is suitable for languages that are morphologically rich. The evaluation results show that their approach yields an improvement in Mean Average Precision (MAP) of 30% and 50% for Bengali and Marathi respectively. They submitted 3 FCG runs for each language (6 runs in all), based on the *title-and-description* (TD) fields of topics. There was no significant difference in MAP between the three Bengali FCG runs. However, in Marathi, increase in case form coverage gave rise to improvements in MAP, with FCG-80 giving 13% better MAP than FCG-60, and 63% better MAP than un-stemmed runs.

Akasereh and Savoy [6] tested their light and aggressive stemmers along with a stopword list for Marathi. The stemmers were compared with n-grams and trunc-n language-independent indexing strategies.[3] A number of retrieval models were also tried. Their results showed that for Marathi, DFR-I(ne)C2, DFR-PL2 and Okapi IR models performed best, while the trunc-4 indexing strategy gave the best retrieval effectiveness compared to other stemming and indexing approaches. Pseudo-relevance-feedback also tended to improve retrieval effectiveness. Their findings also confirm that longer queries give better results.

[3] The effectiveness of n-gram based indexing for Indian language IR was reported earlier by McNamee et al. at FIRE 2008 and later in SIGIR 2009 [7].

Compared to title-only queries, title+description queries yield an improvement of 21.44% in MAP, while complete queries (including the title, description, and narrative fields) improve MAP by as much as 40.50% for the best performing model (I(ne)C2).

Vaidyanathan et al. [8] tested an extension of the Positional Relevance Model [9] for Pseudo Relevance Feedback on the FIRE 2011 English corpus. An improvement of 5.64% and 6.08% was observed over the default query expansion methods, BO1 and BO2. It would have been interesting if the authors reported results for Indian languages too.

The Indian School of Mines, Dhanbad [10], participated for the first time, contributing 3 runs to the task. They used the YASS stemmer and did some manual cleaning of the YASS output. The improvement was about 27% on their chosen baseline. For baseline runs, they used the DFR-PL2 model available within Terrier.

Table 6. Runs by participating groups

Institute	Country	# runs submitted
MANIT	India	2
ISI Kolkata (1) and UTA	India and Finland	9 (3 unofficial)
IIT Bombay	India	1
U. Neuchatel	Switzerland	22
ISM, Dhanbad	India	3
ISI, Kolkata (2)	India	36 (all unofficial)

Table 7. FIRE 2011 runs by task

Query language	Docs retrieved	# runs
Bengali	Bengali	14 + 4 unofficial
Hindi	Hindi	0 + 4 unofficial
Marathi	Marathi	18
English	English	2
Gujarati	Gujarati	0 + 7 unofficial
Bengali	Hindi	0 + 4 unofficial
Bengali	Gujarati	0 + 4 unofficial
Gujarati	Bengali	0 + 4 unofficial
Gujarati	Hindi	0 + 4 unofficial
Hindi	Bengali	0 + 4 unofficial
Hindi	Gujarati	0 + 4 unofficial

Table 8. Results for the Adhoc Bengali monolingual task

RunID	Group	MAP
qListDFR_IneC2-c1d5-NNN.trec(4)	UniNE	0.3798
qListOkapi-b0d75k1d2-NPN.trec(4)	UniNE	0.3768
fcg-80	ISI and UTA	0.3457
fcg-60	ISI and UTA	0.3447

Table 9. Results for the Adhoc Marathi monolingual task

RunID	Group	MAP
qListDFR_IneC2-c1d5-NNN.trec_2	UniNE	0.2350
qListOkapi-b0d75k1d2-NPN.trec_2	UniNE	0.2318
fcg-80	ISI and UTA	0.2223
qListDFR_PB2-c1d5-NNN.trec	UniNE	0.2222
qListDFR_PB2-c1d5-NNN.trec_3	UniNE	0.2222

4 Discussion

The main impression that we came away with from FIRE 2011 is that interest in the Adhoc tasks seems to be on the wane. This is perhaps not unexpected, since this task is, in many ways, fairly well-understood, with the languages of the collections providing the only novelty. We believe, however, that the mandate for creating more Adhoc test collections remains valid, since these collections provide the experimental basis for preliminary explorations that must precede more sophisticated studies. Of course, for these collections to be reliable, we need to create a diverse, representative and robust pool of documents that will be judged for relevance. Given the lack of participation, we will probably need to continue creating artificial pools by combining results obtained using various retrieval models, different parameter settings, and, above all, the results of manual interactive retrieval by expert searchers.

A second problem that we have not been able to address effectively is the bias inherent in the query formulation process. The creation of a topic set is left to an institute where language competence is confined to Bengali and Hindi. The topics chosen for inclusion in the final query collection are thus skewed towards the Bengali and Hindi corpora. The number of relevant documents found for these queries in other corpora is often alarmingly low. More active participation by all consortium members in the topic creation process should lead to more balanced topic sets. The approach adopted at CLEF appears to confirm this hypothesis. We hope to try this approach in future years, even though its practical implementation is likely to be non-trivial (due largely to non-technical and non-academic factors).

References

1. Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., Lioma, C.: Terrier: A High Performance and Scalable Information Retrieval Platform. In: Proceedings of ACM SIGIR 2006 Workshop on Open Source Information Retrieval, OSIR 2006 (2006)
2. Majumder, P., Mitra, M., Parui, S.K., Kole, G., Mitra, P., Datta, K.: YASS: Yet another suffix stripper. ACM Trans. Inf. Syst. 25(4) (2007)
3. Voorhees, E., Harman, D.: Overview of the Fifth Text Retrieval Conference (TREC-5). In: NIST Special Publication 500-238: The Fifth Text Retrieval Conference, TREC-5 (1996)
4. Voorhees, E., Harman, D.: Overview of the Sixth Text Retrieval Conference (TREC-6). In: NIST Special Publication 500-240: The Sixth Text Retrieval Conference, TREC-6 (1997)
5. Kettunen, K., Airio, E., Järvelin, K.: Restricted inflectional form generation in management of morphological keyword variation. Inf. Retr. 10(4-5), 415–444 (2007)
6. Akasereh, M., Savoy, J.: Ad Hoc Retrieval with Marathi Language. In: Majumder, P., Mitra, M., Bhattacharyya, P., Subramaniam, L.V., Contractor, D., Rosso, P. (eds.) FIRE 2010 and 2011. LNCS, vol. 7536, pp. 23–37. Springer, Heidelberg (2013)
7. McNamee, P., Nicholas, C.K., Mayfield, J.: Addressing morphological variation in alphabetic languages. In: Allan, J., Aslam, J.A., Sanderson, M., Zhai, C., Zobel, J. (eds.) SIGIR, pp. 75–82. ACM (2009)
8. Vaidyanathan, R., Das, S., Srivastava, N.: Query Expansion based on Equi-Width and Equi-Frequency Partition. In: Majumder, P., Mitra, M., Bhattacharyya, P., Subramaniam, L.V., Contractor, D., Rosso, P. (eds.) FIRE 2010 and 2011. LNCS, vol. 7536, pp. 13–22. Springer, Heidelberg (2013)
9. Lv, Y., Zhai, C.: Positional relevance model for pseudo-relevance feedback. In: Crestani, F., Marchand-Maillet, S., Chen, H.H., Efthimiadis, E.N., Savoy, J. (eds.) SIGIR, pp. 579–586. ACM (2010)
10. Banerjee, R., Pal, S.: ISM@FIRE-2011 bengali monolingual task: A frequency-based stemmer. In: Majumder, P., Mitra, M., Bhattacharyya, P., Subramaniam, L.V., Contractor, D., Rosso, P. (eds.) FIRE 2010 and 2011. LNCS, vol. 7536, pp. 51–58. Springer, Heidelberg (2013)

Query Expansion Based on Equi-Width and Equi-Frequency Partition

Rekha Vaidyanathan, Sujoy Das, and Namita Srivastava

Department of Computer Applications, Maulana Azad National Institute of Technology, Bhopal, M.P. India
v_rekha@hotmail.com, {sujdas,sri.namita}@gmail.com

Abstract. Query Expansion has been widely used to improve the effectiveness of conceptual search. In this paper pseudo relevance feedback is used along with equi-width and equi-frequency partition technique. The proposed method effectively uses the position and frequency of the query terms for identifying a region within the retrieved documents, which is expected to contain expansion terms. This region is an *intersecting* region obtained by partitioning the retrieved documents using equi-width and equi-frequency partition techniques. Initial results indicate that words falling in the intersecting region contain good candidate terms for query expansion. The experiments are performed on FIRE 2011's Ad-hoc Hindi and English Data using Terrier as the retrieval engine. The initial experiments show an improvement in average precision of 12-14% in case of English data and 12.75% in case of Hindi data set.

Keywords: Pseudo Relevance Feedback, Local Analysis, Query Expansion, Equi-width partitioning, Equi-frequency partitioning, position, frequency.

1 Introduction

Query Expansion has been widely used by many researchers to improve the effectiveness of conceptual search. In Query expansion, keyword of the source query is supplemented with plurals, modifiers, category keywords, business names, Jargons, acronyms, homonyms, synonyms etc., for improving the effectiveness of conceptual search. It is considered that it may reduce paraphrase problem as it takes into account words that are similar to the key word of query.

Query Expansion can be done manually, automatically or interactively [1]. Pseudo Relevance feedback, Latent Semantic Indexing, Statistical Analysis of Word-Co-occurrence, Using Thesauri [1], [2], [6], [8] are some of the well known techniques for query expansion. The expanded word can be obtained from corpus through global or local analysis. These two analyses have their pros and cons. In case of global analysis, statistics building through entire corpus is time consuming whereas, in case of local analysis, the scope might be limited. Relevance feedback, Pseudo Relevance feedback are most common methods for local analysis [1]. Local analysis may reduce time, effort and complexity as number of documents considered is much less and more relevant compared to global analysis. Jones et.al [4] stated that Pseudo

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 13–22, 2013.

Relevance feedback based models are costly models. Ballesteros et.al [13] used Local Context Analysis for expanding the query and has observed that it is more effective than local feedback. It uses both global and local document analysis for query expansion.

In this paper, we propose a Pseudo Relevance Feedback method of partitioning the top 10 retrieved documents, first using equi-width and then using equi-frequency techniques. A superimposition of the results of both of these techniques is found to give good candidate-words for query expansion. Here we use frequency statistic and positions of words to achieve this result. Henceforth, this intersection is referred to as the *overlapped region* throughout this paper.

2 Motivation

The motivation behind proposing a method of Query Expansion by partitioning the document is mainly drawn from the *structure of the document* itself. Typically, a document is divided into sections such as titles, paragraphs, headers and footers. The sections within a document could be specific observations or related observations about the topic. Some documents may even contain multiple topics such as news pages, letters to the editor or even topic digression by the author [15].

The document as a whole, may give variety of information that may be relevant or irrelevant to the query. In this paper, an attempt is made to find *that* section of a document which has greater concentration of relevant information. One of the underlying experiences is that people while reading, tend to *intuitively* pick or capture words surrounding a keyword for further reading or exploration. Several researchers have observed the importance of position of words and their proximity to the keywords. Yuanhua Lv et al. [5] observed that topically related contents and words that are closer to query terms are more likely to be relevant to a query. Hawking et.al [10] observed that distance measurement between terms occurrences can give a good relevance assessment.

In this context, it is assumed that words that are nearer to keywords and also occurring more number of times (but may or may not co-occur) may be considered as good hint-words. Hence we have used the frequency statistic and position of the keywords to implement this partitioning technique effectively. As pseudo relevance is feedback along with partitioning technique is considered for finding relevant section for expansion terms, chances of digression is also reduced.

3 Related Works

In Pseudo Relevance Feedback, the user submits a query and retrieves a ranked list of relevant documents. The initial result set of documents is used to extract terms using some or the other method for query expansion. The original query is then expanded and resubmitted to get more relevant result [4], [14]. Many relevance feedback models with a few relevant documents have been proposed by many researchers. Mitra et.al [8] observed that if there are large fractions of non-relevant documents that

are assumed to be relevant for relevance feedback, the expanded query is likely to be poor. Yuanhua Lv et al. [5] used positional relevance model and proposed that words appearing together closely with the query term are more relevant for query expansion. They exploited the *positions* of words and *proximity* of words to the query terms for their method, assuming that topically related contents and words that are closer to query terms are more likely to be relevant to the query. They observed that the position-based relevance model outperforms document-based or passage based feedback.

Galeas et al. [3], [9] observed that two words that are near to each other in a set of documents may possess some semantic relationship. They have used the position and frequency of words in a document to calculate their inter-quartile ranges to get the dispersion of the words within a document using Fourier series expansion. Tao et.al [7] proposed a mixed model based on statistical language models for Pseudo Relevance feedback. Hawking et.al [10] observed that distance measurement between terms occurrences can give a good relevance assessment.

It is observed that finding co-occurring terms in a document is a complex process. Therefore, it is proposed to partition a document in a two stage process with the help of equi-width and equi-frequency partition. Hoppner et.al [11] used partitions for obtaining information granules in their research work on Granular Computing. The granules are arranged for their similarity using clustering algorithms by adopting several one-dimensional and multi dimensional partitioning. In our method we try to find a region in the document that can have words that are similar to the keyword, capable of augmenting the query.

4 Methodology

In this research, pseudo relevance feedback technique is used for expanding the query terms. The words are extracted from the title and (OR) description field of FIRE 2011 topic and are submitted to TERRIER search engine. The initial result set of 10 documents is used to extract terms using equi-width and equi-frequency technique.

Firstly, the documents are partitioned using equi-width with an equal interval of c where c= n/10, deciles, where n= total words in the document. The range of partition P_i, that has the maximum appearance of keywords (one or more keywords occurring most times) is chosen as that ***area of the document where appearance of keyword(s) is maximum.*** For example, Fig 1 represents a 60 worded document partitioned equally in the range c=6 obtained by deciles. Here the range of words from position 20 to 30, denoted as [20, 30] has the maximum occurrence of keywords.

Fig. 1. Equi-Width Partition with shaded region as the portion that has maximum number of keywords

Secondly, the same document is subjected to equi-frequency partition, where each partition, P_i, has equal number of appearance of the keywords but the range of each partition varies. Here the *frequency statistic* for partition is obtained by the formula c= N/f, obtained by *total number of keywords in the document/max frequency in a decile*. In this case, the total number of words in the document, N=60 and total keywords =9 and c is *rounded down* to obtain 2. We choose the partition that has the minimum range which is that ***area of the document where the appearance of keyword is dense.*** In our example from Fig 2, the Range is from the position 25 to 29 denoted as [25, 29].

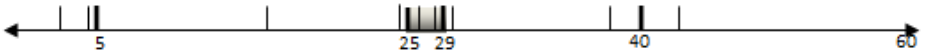


Fig. 2. Equi-Frequency Partition with shaded region as the portion that has maximum number of keywords

Finally, super-imposing the two regions, i.e., words obtained from [20, 30] and [25, 29], we get a resultant area that is observed to have greater concentration of similar words that are relevant to the query. The final selection of upper and lower bound of the intersection obtained from the two techniques is discussed in detail in *Section 4.1.*

The step by step algorithm of query expansion process is as follows

1. Extract title and description field from FIRE 2011 topic. Obtain top 10 retrieved documents (Set N=10).
2. *Repeat step from 3 to 6 for all the top 10 retrieved documents*
3. Partition each retrieved document into 10 partitions using equi-width partitioning. (*P_i is referred as a partition of a Document D).*

    a. For each partition P_i, calculate total frequency of query terms falling within that partition.
    b. Obtain range, $\mathbf{Range_{max} = [P_{w1}, P_{w2}]}$ of that partition in which query terms occur maximum number of times is the word offsets.

    $\mathbf{P}_{w1}$ = Lower bound of the Range: refers to the position of the word starting in that range.
    $\mathbf{P}_{w2}$= Upper bound of the Range: refers to the position of the word ending in that range and ($\mathbf{P}_{w1} < \mathbf{P}_{w2}$).
    Set the frequency of the keywords appearing in this region as $\mathbf{f_{max}}$. (From the example, **Fig1**, $Range_{max}$= [20,30] , f_{max}=4)

4. Partition each retrieved document using equi-frequency partitioning. The frequency statistic **c** (the frequency of the keywords that must appear in all partitions) is calculated using formula : $\mathbf{c=\sum f_i / f_{max}}$, where $\sum f_i$ is the total frequency of keywords in the document D and f_{max} is *maximum times* the keyword(s) have occurred in a decile from equi-width partition, [P_{w1}, P_{w2}].

    a. Obtain the smallest range of partition with frequency **c**.

b. Set it as $\mathbf{Range_{min}= [P_{f1}, P_{f2}]}$, where $\mathbf{P_{f1}}$ and $\mathbf{P_{f2}}$ are lower and upper bound of the selected partition referring to the positions of the words starting and ending in that range. (*As an example, from Fig 2, the* $Range_{min}$*=[25,29]*)

5. Obtain the *intersection of* $Range_{max}$ *and* $Range_{min}$. The two areas are superimposed to get the terms that appear maximum number of times. This is the *denser* region. The selection of this intersection's lower and upper bound is discussed in detail in Section 4.1
6. Extract the words obtained from this intersected range by removing stop-words and save them in a separate file *F*.
7. From this collection of words, obtain the words that are occurring more than twice in file *F* as expansion word.

4.1 Selection of Lower and Upper Bound of Intersected Region

The documents are initially partitioned using equi-width and equi-frequency technique separately and the intersection of these two regions is obtained. The lower and upper bound of the region is decided by calculating the boundaries of the word offsets obtained from the two partitioning techniques.

Let N be total number of words and f be total frequency of keywords in a document D and selected Partition be denoted as $\mathbf{[P_{I1}, P_{I2}]}$

$\mathbf{[P_{I1}, P_{I2}] = [P_{w1}, P_{w2}] \Lambda [P_{f1}, P_{f2}]}$ i.e., $Range_{max} \Lambda Range_{min}$

Where $\mathbf{P_{I1}}$ can take the one of the following values, depending on the following conditions

Case 1: $\mathbf{P_{I1}=P_{w1}}$, if $\mathbf{P_{w1} < P_{f1}}$ and $\{(\mathbf{P_{w1}- P_{f1}}) < (N/f)\}$;
Case 2: $\mathbf{P_{I1}=P_{f1}}$, if $\mathbf{P_{f1}< P_{w1}}$ and $\{(\mathbf{P_{f1}- P_{w1}}) < (N/f)\}$;
Case 3: $\mathbf{P_{I1}=}$ $\{(\mathbf{P_{w1}+ P_{f1}})/2\}$, if $\{|\mathbf{P_{w1}- P_{f1}}| > (N/f)\}$; *this holds good only if* $\mathbf{[P_{w1}, P_{w2}]}$ *and* $\mathbf{[P_{f1}, P_{f2}]}$ *have intersecting regions.*

Similarly, $\mathbf{P_{I2}}$ can take one of the following values depending on the following conditions.

Case 1: $\mathbf{P_{I2}= P_{w2}}$, if $\mathbf{P_{w2} > P_{f2}}$ and $\{(\mathbf{P_{w2}- P_{f2}}) < (N/f)\}$;
Case 2: $\mathbf{P_{I2} = P_{f2}}$, if $\mathbf{P_{f2}> P_{w2}}$ and $\{(\mathbf{P_{f2} - P_{w2}}) < (N/f)\}$;
Case 3: $\mathbf{P_{I2}}$ = $\{(\mathbf{P_{w2}+ P_{f2}})/2\}$, if $\{|\mathbf{P_{w2}- P_{f2}}| > (N/f)\}$; *this holds good only if* $\mathbf{[P_{w1}, P_{w2}]}$ *and* $\mathbf{[P_{f1}, P_{f2}]}$ *have intersecting regions.*

If these two ranges *do not have any common region*, then the Document is not considered for Query expansion. It is observed that in all the cases75-80% of the documents produced intersections.

5 Experiment and Discussion

The experiments were performed on both Ad hoc Hindi and Ad hoc English FIRE 2011 data to test their performance. Two Runs for English data set and one run for

Hindi data set were obtained. For Hindi, only "Title" and for English, "Description" and "Title + Description" were used for performing the experiments and the runs were referred as **HTRUN, EDRUN** and **ETDRUN** respectively**.** After evaluation, we observe that the average precision improves anywhere between 12-14% for English Data and there is 12.75% improvement with regards to precision for Hindi data.

Sample of expansion words obtained after applying the technique in case of **ETRUN** is shown in Table 1.

Table 1. Sample Expansion words ontained from Adhoc English FIRE 2011 data set in **EDRUN**

Topic	**Description**
145	**Research on the existence of life or water in space**
Keywords	life, outer, astronauts, question, liquid, potential,find, water, necessity, scientists, research, space, people, evidence, observatories, quest, mars
152	**Successful missile tests in India and contemporaneous response**
Keywords	bhubaneswar, musharraf, tests, advanced, nuclear,agni, defence, scientists,tests, india, pakistan, cruise, country, system, delhi, army,nirbhay, drdo, test,next, strategic, nations, first

Table 2. Sample Expansion words ontained from Adhoc Hindi FIRE 2011 data set in **HTRUN**

Topic	**Title**
130	**संवाददाताओं का अपहरण तथा हत्या**
Keywords	अदालती, हत्या, शेख, आतंकवाद, पर्ल, आतंकवादियों, अदालत, आरोप, , अपहरण, खिलाफ, पुलिस, प्रदेशकराची, पुलिसकर्मियों, अधिकारी, स्तान, पहले,, मामले, नाम, दो, उमर, अन्य
Drifting words	**स्तान**, **पहले**, **मामले**, **नाम**, **दो**, **उमर**, **अन्य**
165	**विवाह या तलाक कानून**
Keywords	दोनों, पक्षों, पक्ष, समाज, संशोधन, अदालत, सुप्रीम, दायर, आधार, तलाक, संबंध, कोर्ट, अहम, शादी, आयोग, विवाह, हिंदू, जाती, महिला, कानून, बाल, , रिपोर्ट, देखी, तय, केंद्रीय, बारे, रोकने, अलग, जाने, बनाए,
Drifting words	**बाल**, **रिपोर्ट**, **देखी**, **तय**, **बारे**, **रोकने**, **अलग**, **जाने**, **बनाए**

5.1 Results and Discussion

The 50 FIRE 2011 Ad hoc English and Hindi Topics and Terrier Retrieval systems were used for performing experiments. The comparison of Mean Average Precision at different level is shown in Table 3, 4 and 5 respectively for **EDRUN**, **ETDRUN**. For

English Data, the average precision shows an improvement of 14.3% (Description only) and 12.9% (Title and Description) over the original.

Ad Hoc English Data

Table 3. . Comparison of Average Precision incase of before and after Query Expansion for **EDRUN**

	Before Expansion *(only Description)*	**After Expansion** *(only Description)*
No. Of Queries	50	50
Retrieved	50000	50000
Relevant	2761	2761
Relevant retrieved	2570	2629
Average Precision	**0.3726**	**0.4260**

Average precision shows an improvement of 14.3%

Table 4. Comparison of Average Precision incase of before and after Query Expansion for **ETDRUN**

	Before Expansion *(Title and Description)*	**After Expansion** *(Title and Description)*
No. Of Queries	50	50
Retrieved	50000	50000
Relevant	2761	2761
Relevant retrieved	2627	2706
Average Precision	**0.3873**	**0.4376**

Average precision shows an improvement of **12.98**%

Table 5. Precision at different levels for both the Runs (**EDRUN,ETDRUN**) *before* and *after* applying the technique

Precision @	**Before Expansion** *(Desc)*	**After Expansion** *(Desc)*	**Before Expansion** *(Title and Desc)*	**After Expansion** *(Title and Desc)*
0%	0.8067	0.7481	0.7770	0.7430
10%	0.5919	0.6234	0.5911	0.6400
20%	0.5069	0.5726	0.5265	0.5749
30%	0.4444	0.5133	0.4784	0.5236
40%	0.4093	0.4774	0.4266	0.4835
50%	0.3780	0.4562	0.3966	0.4547
60%	0.3428	0.4212	0.3593	0.4182
70%	0.3046	0.3739	0.3297	0.3855
80%	0.2623	0.3127	0.2908	0.3414
90%	0.2008	0.2387	0.2268	0.2831
100%	0.0916	0.1177	0.0990	0.1613

The proposed approach was tested on Hindi FIRE 2011 data set. It is observed that there is similar improvement in case of Hindi data set using the same technique. The comparison of Mean average precision and average precision is shown in Table 6, 7 and 8 respectively.

Ad Hoc Hindi Data

Table 6. Comparison of Average Precision incase of before and after Query Expansion for **HTRUN**

	Before Expansion *(only Title)*	**After Expansion** *(only Title)*
No. Of Queries	50	50
Retrieved	49907	50000
Relevant	2885	2885
Relevant Retrieved	2059	2142
Average Precision	**0.2453**	**0.2766**

Average precision shows an improvement of **12.76**%

Table 7. Precision at different levels for **HTRUN** *before* and *after* applying the technique

Precision @	**Before Expansion** *(only Title)*	**After Expansion** *(only Title)*
0%	1.0172	1.1524
10%	0.7677	0.8404
20%	0.6066	0.6768
30%	0.5185	0.5987
40%	0.4316	0.4894
50%	0.3570	0.4257
60%	0.2940	0.3751
70%	0.2202	0.2814
80%	0.1590	0.1750
90%	0.1080	0.1018
100%	0.0470	0.0409

Table 8. Baseline Comparison at Different Document Levels for **HTRUN**

Documents	**Before Expansion** *(only Title)*	**After Expansion** *(only Title)*
100	0.1940	0.2318
200	0.2246	0.2562
500	0.2419	0.2719

The experimental results on both English and Hindi dataset show that the technique is language independent in nature. The Adhoc Hindi task fetched approximately 13% improvement and Adhoc English tasks fetched 12-14% for the two runs. The results

show fair amount of improvement but our technique need more fine-tuning to get better results. Further study is on in this direction.

The Precision at 0% exceeds 1 for the HTRUN (Table 7) and we are still investigating the reason behind this abnormal value.

6 Conclusion

The proposed fully automatic query expansion method, based on equi-width and equi-frequency partitioning technique using pseudo relevance feedback is able to identify good candidate words for query expansion. It is observed that overlapped region yields good candidates for query expansion. Further, if a document is not having overlapped region then the document is *invariably* irrelevant for query expansion. The initial result of our experiments is encouraging. It is observed that this technique performed in a more or less consistent manner on both the languages used, *without any modifications*. Research in fine-tuning the method is in progress.

Acknowledgements. We would like to thank the FIRE forum for providing the researchers with data that is easily available in their website free of cost. We are grateful to the Terrier™ team members who have extended their support during the experimental stage, clarifying doubts and for providing the software for the purpose.

References

1. Aly, A.A.: Using a Query Expansion Technique to improve Document Retrieval. International Journal "Information Technologies and Knowledge" 2, 343 (2008)
2. Matsuo, Y., Ishizuka, M.: Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information. International Journal on Artificial Intelligence Tools (2004)
3. Galeas, P., Kretschmer, R., Freisleben, B.: Document Relevance Assessment via Term Distribution Analysis Using Fourier Series Expansion. In: Proceedings of the 2009 Joint International Conference on Digital Libraries, JCDL 2009, pp. 277–284. ACM, New York (2009)
4. Jones, R., Rey, B., Madani, O.: Generating Query Substitutions. In: Proceedings of the 15th International Conference on World Wide Web. ACM (2006)
5. Lv, Y., Zhai, C.: Positional Relevance Model for Pseudo-Relevance Feedback. In: SIGIR 2010, Geneva, Switzerland, July 19-23 (2010)
6. Xu, J., BruceCroft, W.: Improving the Effectiveness of Information Retrieval with Local Context Analysis. Journal ACM Transactions on Information Systems (TOIS) 18(1) (January 2000)
7. Tao, T., Zhai, C.: Regularized Estimation of Mixture Models for Robust Pseudo Relevance Feedback. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (2006)
8. Mitra, M., Singhal, A., Buckley, C.: Improving Automatic Query Expansion. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (1998)

9. Galeas, P., Freisleben, B.: Word Distribution Analysis for Relevance Ranking and Query Expansion. In: Gelbukh, A. (ed.) CICLing 2008. LNCS, vol. 4919, pp. 500–511. Springer, Heidelberg (2008)
10. Hawking, D., Thistlewaite, P.: Relevance Weighting Using Distance Between Term Occurrences, Cooperative Research Centre For Advanced Computational Systems (January 25, 1996)
11. Höppner, F., Klawonn, F.: Systems of Information Granules. Wiley Online Library (published online July 16, 2008)
12. Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., Lioma, C.: Terrier: A High Performance and Scalable Information Retrieval Platform. In: 2006 ACM SIGIR Conference on OSIR (2006)
13. Ballesteros, L., Croft, W.B.: Phrasal Translation and Query Expansion Techniques for Cross-Language Information Retrieval. ACM SIGIR Forum (1997)
14. Strzalkowski, T., Wang, J., Wise, B.: Summarization-based Query Expansion in Information Retrieval. In: COLING 1998 Proceedings of the 17th International Conference on Computational Linguistics, vol. 2 (1998)
15. Yu, S., Cai, D., Wen, J.-R., Ma, W.-Y.: Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation. In: The Twelfth International World Wide Web Conference (WWW 2003) (May 2003)

Ad Hoc Retrieval with Marathi Language

Mitra Akasereh and Jacques Savoy

Computer Science Department,
University of Neuchatel,
Rue Emile Argand 11, 2000 Neuchatel, Switzerland
{Mitra.Akasereh,Jacques.Savoy}@unine.ch

Abstract. Our goal in participating in FIRE 2011 evaluation campaign is to analyse and evaluate the retrieval effectiveness of our implemented retrieval system when using Marathi language. We have developed a light and an aggressive stemmer for this language as well as a stopword list. In our experiment seven different IR models (language model, DFR-PL2, DFR-PB2, DFR-GL2, DFR-I(n_e)C2, *tf idf* and Okapi) were used to evaluate the influence of these stemmers as well as *n*-grams and trunc-*n* language-independent indexing strategies, on retrieval performance. We also applied a pseudo relevance-feedback or blind-query expansion approach to estimate the impact of this approach on enhancing the retrieval effectiveness. Our results show that for Marathi language DFR-I(n_e)C2, DFR-PL2 and Okapi IR models result the best performance. For this language trunc-*n* indexing strategy gives the best retrieval effectiveness comparing to other stemming and indexing approaches. Also the adopted pseudo-relevance feedback approach tends to enhance the retrieval effectiveness.

Keywords: Marathi information retrieval, retrieval effectiveness with Indian languages, FIRE evaluation campaign, automatic indexing.

1 Introduction

One of our main objectives in the IR group of University of Neuchâtel is to design, implement and evaluate various indexing and search strategies that work with different non-English languages (monolingual IR). More specifically, in this part we begin with less frequently used languages (and new from an IR perspective), such as Persian, Turkish, Polish, Hindi, Marathi, Bengali and other Indian languages (e.g., Punjabi, Tamil, Telugu) [1]. This set of languages covers various branches of the Indo-European family, while we also tackle popular European [2] as well as Far-East (e.g., Chinese, Japanese, and Korean) [3] languages in order to provide a basis of comparison for our tests. Our objective also includes bilingual and multilingual IR systems. In our participation in the FIRE campaign (www.isical.ac.in/~fire/), our main motivation is to promote new tools and to evaluate and improve existing ones for monolingual IR when facing with Hindi, Marathi and Bengali languages. We applied our tests on the three above mentioned languages while in this paper we talk only about Marathi language. The reason is that our results for Bengali and Hindi

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 23–37, 2013.

languages were not completely reliable due to some mistakes while applying stopword removal on these two languages.

The rest of this paper is organized as follows: Section 2 contains a brief introduction to Marathi language, Section 3 gives an overview of the corpus used in the FIRE-2011 *ad hoc* task. Section 4 represents an overview of our experiment setup and introduces different IR models used in the experiment, the developed stopword list as well as applied stemming and indexing strategies and finally the evaluation method used to evaluate our results. Section 5 presents the results obtained during the experiment and contains the analysis on the usage of different query formulations, different stemming and indexing strategies, various IR models and the impact of query expansion on obtained results. Finally, Section 6 concludes the experiment.

2 Marathi Language

Within the Indian languages studied in FIRE 2011 evaluation campaign, the Marathi owns a special place due to its complex inflectional morphology compared to the Hindi or Bengali languages. In fact, the Marathi grammar has three genders (masculine, feminine, and neuter), two numbers (singular, and plural) and eight grammatical cases (nominative, accusative, genitive, dative, ablative, locative, instrumental, and oblique). At the limit, we may have $3 \times 2 \times 8 = 48$ different suffixes. As for other Indo-European languages, however, this theoretical limit is not reached and several combinations of gender, number and grammatical case own the same suffix. Nevertheless, this highly inflected language raises challenges when designing a light stemmer. To be precise, this is not directly related to the number of possible suffixes. For example, the Hungarian morphology owns 23 grammatical cases while the Finnish language has 15. For an IR point of view, developing an effective Hungarian stemmer is possible [4] while for the Finnish language none algorithmic stemmer is able to produce useful stems. Within this language, adding a given suffix may alter letters inside the word to facilitate its pronunciation (and governed by various phonetic rules such as the vowel harmony) [5]. The linguistic equation "stem + suffix = surface word" possesses numerous exceptions in Finnish while it is relatively direct in the Hungarian language.

As for other languages, the Marathi morphology owns some irregularities (e.g., in plural form, the noun "child" gives "children" and not "childs"). In the best of our knowledge, these exceptions are less frequent than in the Finnish language but more frequent than in other languages (e.g., Hungarian, or German).

3 Overview of the Corpus

The test-collection used for this experiment is the collection made available during the FIRE 2011. The Marathi collection is a collection of about 618 MB of data made up of 99,270 news articles with the average of 266 terms per document (215 terms per document after stopword removal). The articles are extracted from two sources: Esakal between April 2007 to september 2007 and Maharashtratimes between

September 1st 2004 and December 31st 2006. The corpus is coded in UTF-8 and each article is marked up using following tags:

<DOC> : Starting tag of a document.
<DOCNO> </DOCNO> : Contains document identifier.
<TEXT> </TEXT> : Contains document text.
</DOC> : Ending tag of a document.

In this corpus we can find 50 topics (from Topic #126 to Topic #175). Among which fourteen topics have no relevant document in the collection (Topic #126, #129, #132, #137, #141, #145, #151, #154, #155, #156, #158, #159, #160 and #162). The rest thirty six topics have a total number of 354 relevant documents with mean of 9.83 items/topic and a median of 4 (standard deviation 14.26). Topics #130, #133, #139, #143, #146, #149, #150, #164, and #167 own one relevant item which is the smallest number of pertinent documents. Topic #170 with 62 relevant items has the greatest number of relevant documents.

Following the TREC model [6], each topic is divided into three sections: the title (T) which is a brief title, the description (D) that gives a one-sentence description, and the narrative part (N) which specifies the relevance assessment criteria. Topic #145 is shown below as an example. This topic holds "Benazir Bhutto murder" in its title, "Benazir Bhutto murder enquiry" in its description, while more details on the subject is given in its narrative section.

<top lang='mr'>
<num>145</num>
<title>बेनजीर भुट्टो यांची हत्या</title>
<desc>बेनजीर भुट्टो यांच्या हत्येची चौकशी</desc>
<narr>प्रासंगिक लेखांत बेनजीर भुट्टो यांच्या हत्येची चौकशी आणि चौकशीच्या कार्यपद्धतीबद्दल (चौकशीच्या प्रणालीबद्दल) विविध लोक, गट आणि संघटना ह्यांचे विचार व मते यांविषयी माहिती असावी.</narr>
</top>

4 Experiment Architecture

This section describes the setup of our experiment. Section 4.1 describes the adopted IR models, Section 4.2 explains the stopword list used for stopword removal, Section 4.3 describes the applied stemming and indexing strategies while Section 4.4 explains the measurements used to evaluate our system.

4.1 IR Models

In the experiment we analysed and compared different stemming and indexing strategies. To achieve this, seven different IR models are implemented and evaluated. The models are the following:

The first model is the classical *tf idf* model, where the weight for each indexing term t_i is the product of its term frequency in the document (tf_{ij}) and the logarithm of its inverse document frequency ($idf_j = \log\left[\frac{n}{df_j}\right]$ where n indicates the number of documents in the collection and df_j the number of documents which indexed the term t_i). The index weights normalized using cosine normalization. To compute the similarity between a document and a given query we have adopted the inner product given in Equation 1.

$$sim(d_i, q) = \sum_{t_i \in q} w_{ij} \cdot w_{qj} \tag{1}$$

As the first probabilistic model we have adopted the Okapi model (BM25) [7]. To evaluate the score of the similarity between the query and the document the Okapi function is described in Equation 2. In this formulation l_i indicates the length of document d_i (number of indexing terms). In our experiment the *advl* (average document length) is set to 265, b to 0.55 and k_1 to 1.2.

$$sim(d_i, q) = \sum_{t_i \in q} qtf_j \cdot \log\left[\frac{n - df_j}{df_j}\right] \cdot \frac{(k_1 + 1) \cdot tf_{ij}}{K + tf_{ij}} \tag{2}$$

$$K = k_1 \cdot \left((1 - b) + b \cdot \frac{l_i}{advl}\right)$$

As other probabilistic models, we have used the DFR-PL2, DFR-I(n_e)C2, DFR-PB2 and DFR-GL2. These models are derived from the Divergence From Randomness (DFR) family [8]. The indexing term weight (weight of term t_j in document d_i) is calculated as:

$$w_{ij} = Inf_{ij}^1 \cdot Inf_{ij}^2 = -\log_2\left(Prob_{ij}^1\left(tf_{ij}\right)\right) \cdot \left(1 - Prob_{ij}^2\left(tf_{ij}\right)\right) \tag{3}$$

In this experiment the DFR-PL2 is implemented as in Equations 4 and 5.

$$Prob_{ij}^1 = \frac{e^{-\lambda_j} \cdot \lambda_j^{tfn_{ij}}}{tfn_{ij}!} \tag{4}$$

$$Prob_{ij}^2 = \frac{tfn_{ij}}{tfn_{ij} + 1} \tag{5}$$

where:

- tf_n is the normalized term frequency
- $\lambda_j = \frac{tc_j}{n}$ (tc_j is the number of occurrence of term t_i in the collection and n is the number of documents in the corpus)
- $tfn_{ij} = tf_{ij} \cdot \log_2\left(1 + \frac{c \cdot mean\_dl}{l_i}\right)$ (in this experiment c is set to 1.5 and $mean\_dl$ (average document length) to 265)

For the DFR-PB2 model, the $Prob_{ij}^{1}$ is calculated as mentioned in Equation 4 and $Prob_{ij}^{2}$ as follows:

$$Prob_{ij}^{2} = 1 - \frac{tc_j + 1}{df_j(tfn_{ij} + 1)} \tag{6}$$

DFR- I(n_e)C2 is defined by the same $Prob_{ij}^{2}$ as in Equation 6 and:

$$Inf_{ij}^{1} = tfn_{ij} \cdot \log\left[\frac{n+1}{n_e + 0.5}\right] \tag{7}$$

$$n_e = n \cdot \left(1 - \left(\frac{n-1}{n}\right)^{tc_j}\right)$$

In the DFR-GL2 model $Prob_{ij}^{1}$ is defined as follows while $Prob_{ij}^{2}$ is defined as in Equation 5.

$$Prob_{ij}^{1} = 1 - \left[\frac{1}{1+\lambda_j}\right] \cdot \left[\frac{\lambda_j}{1+\lambda_j}\right]^{tfn_{ij}} \tag{8}$$

Finally we have also implemented one approach based on language model (LM) [9]. In this case, the underlying paradigm is based on a non-parametric probabilistic model. We have opted the model suggested by Hiemstra [10] using the Jelinek-Mercer smoothing [11] scheme as shown in Equation 9.

$$P(d_i|q) = P(d_i) \cdot \prod_{t_j \epsilon q} \left[\lambda_j \cdot P\left(t_j|d_i\right) + \left(1 - \lambda_j\right) \cdot P\left(t_j|C\right)\right] \tag{9}$$

where:

- $P\left(t_j|d_i\right) = \frac{tf_{ij}}{l_i}$
- $P\left(t_j|C\right) = \frac{df_j}{lc}$ ($lc = \sum_k df_k$)
- λ_i is a smoothing factor (here set to 0.30 for all index terms)
- *lc* is an estimation of the corpus *C* length.

4.2 Stopword List

Different words do not have the same importance in describing the semantic content of a document (or a query). Therefore it is usually a good practice to remove very frequent terms having no precise meaning (stopwords). Accordingly before applying the indexing strategies we eliminate the stopwords. To do so we used a stopword list. This list is generated using the approach explained in [12] containing 99 terms. The list is freely available on http://www.unine.ch/info/clef/.

4.3 Stemming and Indexing Strategies

To represent the documents and the topics, different automatic indexing methods can be applied. As a first strategy, we can use some language-independent indexing representations. The two possible strategies are the *n*-gram and trunc-*n* approaches. *N*-grams approach is the act of producing, for each word, the overlapping sequences of *n* characters [13]. With the word "computer" and defining *n*=3, we obtain {com, omp, mpu, put, ute, ter} indexing terms. The trunc-*n* is the process of truncating a word by keeping its first *n* characters and cutting of the remaining letters. With our previous example we obtain "comp" with n=4. In our experiment different values for *n*, for both *n*-gram and trunc-*n* are tested to find which value of *n* gives the best performance.

Having some morphological variations do not usually change the meaning of a word. For example a document with the word "houses" may be a good answer to a query containing the word "house". As a way to conflate word variations under the same form, we can apply a stemmer that removes the final letters of a word. To do so, in this experiment a light suffix-stripping algorithm is used which removes the inflectional suffixes. In our implementation, the removal is focused on nouns and adjectives and verbal suffixes are not taken into account. The reason for ignoring the verbal suffixes is mainly due to the following hypothesis. We believe that in a given text verbs convey less important semantic information than nouns and adjectives thus the retrieval based on match between different verbal forms is less useful. There are more details on this assumption in our previous experiments on other languages [14]. On the other hand, considering the fact that our light stemmer does not take into account part of speech and it does not apply any morphological analysis, verbal suffix removal would not help the retrieval effectiveness and might even reduce the mean average precision. Moreover previous experiments show that stemmers based on deep morphological analysis do not give better retrieval results than simpler light stemmers [15], [16], [17]. As a variant, we will also apply a more aggressive stemmer which removes also some derivational suffixes apart from inflectional ones. In this case we can conflate words like "computational" and "computer" under the same root.

4.4 Evaluation

To evaluate the retrieval performance we have adopted the mean average precision (MAP) measurement (as calculated by TREC_EVAL program [18] based on the 1000 retrieved documents per query). Using the average means that we attach the same importance to all queries. It is important to mention that in our calculations the queries with no relevant items were not taken into account. So for Marathi language we considered only 36 queries (for which there is at least one relevant document in the collection) while the official measure takes the whole 50 queries into account. As a result there are some differences between values presented in this paper and the ones computed according to the official measure.

5 Results and Analysis

This section evaluates and analyses the results obtained during our experiment. Section 5.1 contains the overall results obtained during the experiment. Sections 5.2 and 5.3 discuss the performance of the seven adopted IR models and different query formulation (T, TD and TDN). Section 5.4 compares the different stemming strategies used in the experiment (light and aggressive) with no stemming approach. While Section 5.5 discusses the effectiveness of using indexing strategies (*n*-gram and trunc-*n*) and compare the results obtained for different values for *n*. Finally Section 5.6 shows the results when applying a blind-query expansion technique.

5.1 Experiment Results

The MAPs of our different experiments are depicted in Table 1 (Title only), Table 2 (Title and Description) and Table 3 (Title, Description and Narrative). In these tables, we can find the retrieval effectiveness values obtained by applying different stemming and indexing approaches to seven different IR models. For indexing approaches like *n*-gram and trunc-*n* different values of *n* are selected and evaluated in order to define which value for *n* gives a better overall performance.

The first row of the three tables shows the results when the retrieval is done without applying either a stemming strategy (NoStem) or stopword removal (NoSL). While the next row shows the retrieval performance where stopword removal is applied but still stemming is ignored. The next two rows show the effects of applying a light and a more aggressive stemmer. Afterwards the results of performing various *n*-gram approaches with two different values for *n* are depicted. Finally the last three rows report the results for trunc-*n* approach with three different values of *n*.

Table 1. MAP of different IR models and different stemmers for T query formulation

	Mean Average Precision (T)							
	Okapi	**DFR-I(n_e)C2**	**DFR-PL2**	**DFR-GL2**	**LM**	***tf idf***	**DFR-PB2**	**Average**
NoStem/NoSL	0.2038	0.2147	0.2028	0.2038	0.2035	0.1422	0.2074	0.1969
NoStem	0.2044	0.2128	0.2013	0.1957	0.2011	0.1418	0.2057	0.1947
Light Stem.	0.2044	0.2128	0.2013	0.1957	0.2011	0.1418	0.2057	0.1947
Aggressive	0.2044	0.2129	0.2015	0.1961	0.2011	0.1418	0.2057	0.1948
3-grams	0.2733	0.2635	0.2579	0.2574	0.2131	0.1700	0.1297	0.2236
4-grams	0.2454	0.2433	0.2424	0.2476	0.2379	0.1607	0.2249	0.2289
trunc-3	0.2239	0.2113	0.2290	0.2197	0.2014	0.1332	0.1978	0.2023
trunc-4	0.2767	0.2682	0.2878	0.2704	0.2614	0.1918	0.2669	**0.2605**
trunc-5	0.2501	0.2410	0.2501	0.2461	0.2422	0.1551	0.2393	0.2320
Average	0.2269	**0.2278**	0.2251	0.2204	0.2150	0.1511	0.2086	

Table 2. MAP of different IR models and different stemmers for TD query formulation

	Mean Average Precision (TD)							
	Okapi	**DFR-I(n_e)C2**	**DFR-PL2**	**DFR-GL2**	**LM**	***tf idf***	**DFR-PB2**	**Average**
NoStem/NoSL	0.2360	0.2396	0.2303	0.2258	0.2367	0.1640	0.2375	0.2243
NoStem	0.2351	0.2381	0.2304	0.2239	0.2368	0.1639	0.2368	0.2236
Light Stem.	0.2351	0.2381	0.2304	0.2239	0.2368	0.1639	0.2368	0.2236
Aggressive	0.2351	0.2383	0.2304	0.2240	0.2368	0.1639	0.2375	0.2237
3-grams	0.2728	0.3121	0.2869	0.3163	0.2569	0.1856	0.0738	0.2435
4-grams	0.2632	0.2774	0.2619	0.2733	0.2436	0.1735	0.2558	0.2498
trunc-3	0.2800	0.2750	0.2743	0.2681	0.2640	0.1385	0.2531	0.2504
trunc-4	0.3226	0.3257	0.3381	0.3089	0.3215	0.2139	0.3186	**0.3070**
trunc-5	0.2928	0.2871	0.2869	0.2900	0.2938	0.1729	0.2758	0.2713
Average	0.2585	**0.2643**	0.2573	0.2547	0.2546	0.1698	0.2363	

Table 3. MAP of different IR models and different stemmers for TDN query formulation

	Mean Average Precision (TDN)							
	Okapi	**DFR-I(n_e)C2**	**DFR-PL2**	**DFR-GL2**	**LM**	***tf idf***	**DFR-PB2**	**Average**
NoStem/NoSL	0.2792	0.2638	0.2640	0.2729	0.2707	0.1789	0.2542	0.2548
NoStem	0.2800	0.2769	0.2753	0.2829	0.2824	0.1771	0.2686	0.2633
Light Stem.	0.2800	0.2769	0.2753	0.2829	0.2824	0.1771	0.2686	0.2633
Aggressive	0.2800	0.2765	0.2754	0.2832	0.2824	0.1771	0.2686	0.2633
3-grams	0.2597	0.3456	0.3019	0.2738	0.3114	0.2081	0.0232	0.2462
4-grams	0.2974	0.3169	0.3025	0.3004	0.2919	0.1874	0.3156	0.2874
trunc-3	0.3138	0.3204	0.3449	0.2835	0.3232	0.1478	0.2713	0.2864
trunc-4	0.3788	0.3768	0.3722	0.3801	0.3713	0.2261	0.3704	**0.3537**
trunc-5	0.3199	0.3188	0.3210	0.3075	0.3217	0.1842	0.3046	0.2968
Average	0.2953	**0.3024**	0.2985	0.2939	0.3002	0.1834	0.2620	

Referring to these results, a general overview of the issues that are addressed to analyze is as follows:

1. Comparing the performance of different IR models and discussing which retrieval model performs the best for different stemmers.
2. Between different stemming and indexing methods, which one is the most effective one and what are the reasons for the weak performance of certain strategies.
3. Whether applying stemming or indexing strategies has practically a better effect on performance than non-stemming methods.
4. Evaluating the *n*-grams and trunc-*n* indexing strategies to identify which value of *n* is the most appropriate one.
5. Verifying whether stopword removal helps to achieve a better retrieval performance comparing to approaches that ignore this operation.

5.2 IR Models Evaluation

Referring to Tables 1 through 3, we can see that DFR-I(n_e)C2 model has the best average performance for any given stemming or indexing strategy and any query formulation. This model is followed by Okapi and DFR-PL2 model for T and TD query formulations and by LM, DFR-PL2 and Okapi for TDN query formulation. We can also see that the classical *tf idf* vector space model has always the worst performance for any applied stemming and indexing strategy.

Although, we believe that in some cases the average measurements do not precisely describe the overall performance (e.g., it is known that extreme values influence the average). To overcome this inadequacy of average values, applying a query-by-query analysis could help to have a more precise understanding of the reasons behind the obtained results.

5.3 Query Formulation Evaluation

From the Tables 1, 2 and 3 we can see that expanding a query by adding the description and then the narrative logical sections improves the performance for any stemming and indexing strategy. To be more precise and considering the trunc-4 strategy as the best performing one, Table 4 shows the change in percentage when considering the TD and TDN query formulations. We can see that this query expansion makes a positive average improvement for all IR models. The results show that adding the description to the query makes an average improvement of +17.89% in performance while adding the narrative section as well as the description (TDN) improves the average performance for +35.78%.

Looking at the different IR models separately we can see that expanding the query has the most impact for LM model where the performance changes for +23.01% and +42.03% for TD and TDN respectively over T formulation.

Considering only the best performing model (DFR-I(n_e)C2) in our experiment we can see that the change percentage over T is +21.44% and +40.5% for TD and TDN respectively.

As for query formulation using only title (T) has obviously week retrieval performance, for the rest of our evaluation we consider only TD and TDN formulations. Also we omitted *tf idf* and DFR-PB2 IR models in further evaluations as according to the results they are clearly the models with the weakest performance.

Table 4. MAP of different query formulation (with trunc-4 and different IR models) & its change percentage for TD and TDN over T

	Mean Average Precision trunc-4				
	T	**TD**	**TDN**	**% of Change TD over T**	**% of Change TDN over T**
Okapi	0.2767	0.3226	0.3788	+16.62%	+36.90%
DFR-I(n_e)C2	0.2682	0.3257	0.3768	+21.44%	+40.50%

Table 4. (*continued*)

DFR-PL2	0.2878	0.3381	0.3722	+17.47%	+29.34%
DFR-GL2	0.2704	0.3089	0.3801	+14.23%	+40.58%
LM	0.2614	0.3215	0.3713	**+23.01%**	**+42.03%**
tf idf	0.1918	0.2139	0.2261	+11.51%	+17.89%
DFR-PB2	0.2669	0.3186	0.3704	+19.35%	+38.76%
Average	0.2605	0.3070	0.3537		
% of Change over T	base	+17.89%	+35.78%		

5.4 Stemming Strategies Evaluation

Tables 5 and 6 depict the MAP under different stemming strategies and stopword removal. The results show that removing the stopwords improves the average performance when using TDN query formulation while it does not have a positive impact when we have TD query formulation. This might be due to our stopword list which is quite a short list. As stopword removal plays an important role in retrieval effectiveness enhancement [19] it seems important to consider increasing the number of terms present in the stopword list.

Table 5. MAP of various IR models with different stemming strategies (TD query formulation)

	Mean Average precision (TD)			
	NoStem/NoSL	**NoStem**	**Light Stem**	**Aggressive**
Okapi	0.2360	0.2351	0.2351	0.2351
DFR-I(n_e)C2	0.2396	0.2381	0.2381	0.2383
DFR-PL2	0.2303	0.2304	0.2304	0.2304
DFR-GL2	0.2258	0.2239	0.2239	0.2240
LM	0.2367	0.2368	0.2368	0.2368
Average	0.2337	0.2329	0.2329	0.2329

Table 6. MAP of various IR models with different stemming strategies (TDN query formulation)

	Mean Average precision (TDN)			
	NoStem/NoSL	**NoStem**	**Light Stem**	**Aggressive**
Okapi	0.2792	0.2800	0.2800	0.2800
DFR-I(n_e)C2	0.2638	0.2769	0.2769	0.2765
DFR-PL2	0.2640	0.2753	0.2753	0.2754
DFR-GL2	0.2729	0.2829	0.2829	0.2832
LM	0.2707	0.2824	0.2824	0.2824
Average	0.2701	0.2795	0.2795	0.2795

The values in Tables 5 and 6 also show that applying either light or aggressive stemmers does not change the performance and the MAP stays almost the same with our without applying these stemmers. As Marathi language has a complex inflectional morphology the structure of applied stemmers should be reconsidered so that stemming would help to improve the performance. A more complex stemmer might help to augment the retrieval performance.

5.5 Indexing Strategies Evaluation

Referring to Tables 7 and 8 we will find that applying indexing strategies like *n*-grams or trunc-*n* clearly increases the retrieval performance comparing to no stemming method.

Table 7. MAP of various IR models with different indexing strategies (TD query formulation) and its change % over no-stemming approach

	Mean Average precision (TD)					
	NoStem	**3-gram**	**4-gram**	**trunc-3**	**trunc-4**	**trunc-5**
Okapi	0.2351	0.2728	0.2632	0.2800	0.3226	0.2928
DFR-I(n_e)C2	0.2381	0.3121	0.2774	0.2750	0.3257	0.2871
DFR-PL2	0.2304	0.2869	0.2619	0.2743	0.3381	0.2869
DFR-GL2	0.2239	0.3163	0.2733	0.2681	0.3089	0.2900
LM	0.2368	0.2569	0.2436	0.2640	0.3215	0.2938
Average	0.2329	0.2890	0.2639	0.2723	0.3234	0.2901
% of Change over base	base	+24.1%	+13.3%	+16.9%	**+38.9%**	+24.6%

Table 8. MAP of various IR models with different indexing strategies (TDN query formulation) and its change % over no-stemming approach

	Mean Average precision (TDN)					
	NoStem	**3-grams**	**4-grams**	**trunc-3**	**trunc-4**	**trunc-5**
Okapi	0.2800	0.2597	0.2974	0.3138	0.3788	0.3199
DFR-I(n_e)C2	0.2769	0.3456	0.3169	0.3204	0.3768	0.3188
DFR-PL2	0.2753	0.3019	0.3025	0.3449	0.3722	0.3210
DFR-GL2	0.2829	0.2738	0.3004	0.2835	0.3801	0.3075
LM	0.2824	0.3114	0.2919	0.3232	0.3713	0.3217
Average	0.2795	0.2985	0.3018	0.3171	0.3758	0.3178
% of Change over base	base	+6.8%	+8.0%	+13.5%	**+34.5%**	+13.7%

From the obtained values we can see that between *n*-grams and trunc-*n* strategies (with different values for *n*), trunc-4 method clearly increases the mean performance the most and gives, almost always, the best performance (except for DFR-GL2 model in TD query formulation where 3-gram gives a better result).

We can say that for the trunc-*n* approach the best value for *n* is 4. For the *n*-gram models, 3-gram tends to have a better performance than 4-gram. While both language-independent indexing strategies increase the performance comparing to no stemming approach and even stemming approach (applying the proposed light and aggressive stemmers).

5.6 Pseudo-Relevance Feedback

According to our previous experiments with different languages we see that applying a blind-query expansion (or pseudo-relevance feedback) (PRF) might help to improve the mean retrieval effectiveness.

In this expansion the original query is reformulated by adding *m* terms extracted from the *k* top ranked documents. In this experiment, we applied Rocchio's approach [20] with $\alpha = 0.75$, $\beta = 0.75$. The expansion is applied to both TD and TDN query formulations. The results are shown in Table 9 and Table 10.

Table 9. MAP of Different Blind-Query Expansions, Rocchio's method, TD queries

	Mean Average Precision (TD)		
	NoStem	**3-gram**	**trunc-3**
Okapi	0.2351	0.2728	0.2800
3 docs / 20 terms	0.2372	0.2944	0.2833
3 docs / 50 terms	0.2406	0.2849	**0.3122**
3 docs / 70 terms	0.2421	0.2860	0.3088
3 docs / 100 terms	0.2335	0.2889	0.3004
3 docs / 150 terms	0.2258	0.2804	0.2788
5 docs / 20 terms	0.2406	0.2957	0.2878
5 docs / 50 terms	0.2144	0.2892	0.2856
5 docs / 70 terms	0.2440	0.2913	0.2874
5 docs / 100 terms	0.2382	0.2882	0.3036
5 docs / 150 terms	0.2304	0.2899	0.2960
10 docs / 20 terms	0.2407	0.2967	0.2894
10 docs / 50 terms	0.2431	0.2874	0.2896
10 docs / 70 terms	0.2465	0.2874	0.3064
10 docs / 100 terms	0.2446	0.2938	0.3050
10 docs / 150 terms	0.2442	0.2932	0.2954
15 docs / 20 terms	0.2349	0.2972	0.2853
15 docs / 50 terms	0.2440	0.2908	0.2894
15 docs / 70 terms	0.2454	0.2882	0.3046
15 docs / 100 terms	0.2478	0.2922	0.3003
15 docs / 150 terms	0.2468	0.2954	0.2967

Table 10. MAP of Different Blind-Query Expansions, Rocchio's method, TDN queries

	Mean Average Precision (TDN)		
	NoStem	**3-gram**	**trunc-3**
Okapi	0.2800	0.2597	0.3138
3 docs / 20 terms	0.2614	0.2944	0.3239
3 docs / 50 terms	0.2736	0.3231	0.3290
3 docs / 70 terms	0.2838	0.3325	0.3279
3 docs / 100 terms	0.2810	0.3454	0.3269
3 docs / 150 terms	0.2685	**0.3461**	0.3218
5 docs / 20 terms	0.2613	0.2943	0.3239
5 docs / 50 terms	0.2718	0.3203	0.3322
5 docs / 70 terms	0.2725	0.3393	0.3310
5 docs / 100 terms	0.2808	0.3354	0.3314
5 docs / 150 terms	0.2786	0.3414	0.3279
10 docs / 20 terms	0.2622	0.2900	0.3292
10 docs / 50 terms	0.2700	0.3179	0.3358
10 docs / 70 terms	0.2731	0.3392	0.3336
10 docs / 100 terms	0.2738	0.3383	0.3310
10 docs / 150 terms	0.2790	0.3413	0.3303
15 docs / 20 terms	0.2618	0.2957	0.3271
15 docs / 50 terms	0.2742	0.3188	0.3311
15 docs / 70 terms	0.2753	0.3421	0.3265
15 docs / 100 terms	0.2763	0.3382	0.3229
15 docs / 150 terms	0.2757	0.3404	0.3192

In our experiment the results show that applying the pseudo-relevance feedback approach enhance the mean retrieval performance. The best results were gained for the Okapi model.

When using TD query formulation (see Table 9), the best enhancement is for the trunc-3 model when the queries are expanded adding 50 terms selected from the first 3 retrieved documents. Here the MAP is increased for +11.5% (from 0.2800 to 0.3122).

When it comes to TDN query formulation (see Table 10) the enhancement of performance becomes even bigger. Here for the best case the MAP is increased for +33.30% (from 0.2597 to 0.3461) for 3-grams strategy. As Table 10 shows for TDN query formulation the blind-query expansion improves a lot the retrieval effectiveness for Okapi model using 3-grams indexing strategy.

The results show that adding more than 100 terms does not help any better the enhancement. We believe that this is due to the fact that including more terms causes noises for the system [21].

6 Conclusion

The results of our experiment in FIRE 2011 evaluation campaign show that in general the IR models DFR-I(n_e)C2 and DFR-PL2, both based on Divergence From Randomness paradigm, are giving the best retrieval results for any stemming or indexing strategies. These models are followed by Okapi model. The classical *tf idf* model tends to offer lower performance levels.

The results also show that in general expanding the query by adding the description (D) and narrative (N) sections to it improves the retrieval effectiveness comparing to using only the title (T) part of the query. For the best performing model (DFR-I(n_e)C2), enlarging the query from T to TD improves the retrieval effectiveness up to +21.41%. This improvement increases to +40.5% when changing the query formulation to TDN. In average for all seven models there were between +17.89% and +35.78% enhancement in performance while using TD and TDN respectively (over Title only formulation).

We can see from the results that the light and aggressive stemmers, proposed in this experiment, did not change the performance comparing to no stemming approach. But applying *n*-gram or trunc-*n* indexing strategies clearly increases the retrieval performance comparing to no stemming method. In our experiment trunc-4 approach tends to result the best MAP.

The results after applying the Rochio's approach as the adopted approach for blind-query expansion show that this expansion tends to help the retrieval enhancement. Here the blind-query expansion increases the retrieval effectiveness the most for trunc-3 and 3-gram strategies while using the Okapi IR model.

Acknowledgements. This work was supported in part by the Swiss National Science Foundation under Grant #200020-129535/1.

References

1. Dolamic, L., Savoy, J.: UniNE at FIRE 2008: Hindi, Marathi and Bengali IR. FIRE 2008 Working Notes (2008)
2. Savoy, J.: Combining Multiple Strategies for Effective Monolingual and Cross-Lingual Retrieval. IR Journal 7, 121–148 (2004)
3. Savoy, J.: Comparative Study of Monolingual and Multilingual Search Models for Use with Asian Languages. ACM - Transactions on Asian Languages Information Processing 4, 163–189 (2005)
4. Savoy, J.: Searching Strategies for the Hungarian Language. Information Processing & Management 44(1), 310–324 (2008)
5. Koskenniemi, K., Church, K.W.: Complexity Two-Level Morphology and Finnish. In: Proceedings COLING, Budapest, pp. 1–9 (1988)
6. Voorhees, E.M., Harman, D.K. (eds.): TREC. Experiment and Evaluation in Information Retrieval. The MIT Press, Cambridge (2005)
7. Robertson, S.E., Walker, S., Beaulieu, M.: Experimentation as a Way of Life: Okapi at TREC. Information Processing & Management 36, 95–108 (2002)

8. Amati, G., van Rijsbergen, C.J.: Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness. ACM Transactions on Information Systems 20, 357–389 (2002)
9. Hiemstra, D.: Using Language Models for Information Retrieval. Ph.D. Thesis (2000)
10. Hiemstra, D.: Term-Specific Smoothing for the Language Modeling Approach to Information Retrieval. In: Proceedings of the ACM-SIGIR, pp. 35–41. The ACM Press (2002)
11. Zhai, C., Lafferty, J.: A Study of Smoothing Methods for Language Models Applied to Information Retrieval. ACM Transactions on Information Systems 22, 179–214 (2004)
12. Fox, C.: A Stop List for General Text. ACM-SIGIR Forum 24, 19–35 (1990)
13. McNamee, P., Mayfield, J.: Character N-gram Tokenization for European Language Text Retrieval. IR Journal 7, 73–97 (2004)
14. Savoy, J.: Light Stemming Approaches for the French, Portuguese, German and Hungarian Languages. In: Proceedings of the ACM-SAC, pp. 1031–1035. The ACM Press (2006)
15. Harman, D.K.: How Effective is Suffxing? Journal of the American Society for Information Science 42, 7–15 (1991)
16. Porter, M.F.: An Algorithm for Suffix Stripping. Program 14, 130–137 (1980)
17. Fautsch, C., Savoy, J.: Algorithmic Stemmers or Morphological Analysis: An Evaluation. Journal of the American Society for Information Sciences and Technology 60, 1616–1624 (2009)
18. Buckley, C., Voorhees, E.M.: Retrieval System Evaluation. In: Voorhees, E.M., Harman, D.K. (eds.) TREC. Experiment and Evaluation in Information Retrieval, pp. 53–75. The MIT Press, Cambridge (2005)
19. Dolamic, L., Savoy, J.: When Stopword Lists Make the Difference. Journal of the American Society for Information Sciences and Technology 61, 200–203 (2010)
20. Buckley, C., Singhal, A., Mitra, M., Salton, G.: New Retrieval Approaches Using SMART. In: Proceedings of the TREC-4, pp. 25–48. NIST Publication #500-236, Gaithersburg (1996)
21. Peat, H.J., Willett, P.: The Limitations of Term Co-Occurrence Data for Query Expansion in Document Retrieval Systems. Journal of the American Society for Information Science 42, 378–383 (1991)

Frequent Case Generation in Ad Hoc Retrieval of Three Indian Languages – Bengali, Gujarati and Marathi

Jiaul H. Paik[1], Kimmo Kettunen[2], Dipasree Pal[1], and Kalervo Järvelin[2]

[1] Indian Statistical Institute, Kolkata
[2] University of Tampere, Finland

Abstract. This paper presents results of a generative method for the management of morphological variation of query keywords in Bengali, Gujarati and Marathi. The method is called Frequent Case Generation (FCG). It is based on the skewed distributions of word forms in natural languages and is suitable for languages that have either fair amount of morphological variation or are morphologically very rich. We participated in the ad hoc task at FIRE 2011 and applied the FCG method on monolingual Bengali, Gujarati and Marathi test collections. Our evaluation was carried out with title and description fields of test topics, and the Lemur search engine. We used plain unprocessed word index as the baseline, and n-gramming and stemming as competing methods. The evaluation results show 30%, 16% and 70% relative mean average precision improvements for Bengali, Gujarati and Marathi respectively when comparing the FCG method to plain words. The method shows competitive performance in comparison to n-gramming and stemming.

1 Introduction

One of the basic problems of full-text retrieval is the variation of word forms that is caused by the morphology of natural languages. Shortly put, this means that one base or dictionary form of a word in language may occur in different (inflected) variant forms in texts. Consequently, the principle one keyword one concept - one match in the textual index of retrieval systems often does not hold due to morphology alone. Therefore something needs to be done to morphological variation so that the performance of information retrieval (IR) systems will not suffer too much if the language has a rich morphology. There are several solutions to this problem, and they can be divided to reductive and generative techniques [3]. Stemming has been the most widely applied reductive morphological technique in IR. In stemming distinct variants of word forms are conflated or reduced (optimally) to one form that may be a base form or just a technical stem. An example would be optimal reduction of the set *cat, cats, cat's, cats'* to one lemma *cat*. A more sophisticated reductive keyword variation management method is lemmatization, which produces base or dictionary forms of inflected word forms using rules and a dictionary [7] or without using a dictionary [10].

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 38–50, 2013.

Another option for keyword variation management is generation of variant word forms for query keys. In this case variant forms of the keyword are generated using a base form. Given a base form, like *cat*, the generator produces all the variant forms of it, in this case *cat, cats, cat's, cats'* or just a subset of the forms if the number of variant forms is very high. These produced variant forms are then given to the retrieval engine which matches them in the plain inflected word form text index that has not been analyzed with language technology tools, i.e. the index contains the textual words as such.

Kettunen and Airio [5] and Kettunen et al. [6] have developed a linguistic frequency based method called Frequent Case Generation (FCG) for word form variation management in information retrieval. For languages with large or moderately large number of variant word forms, the number of generated forms is restricted to only the most frequent forms that are first determined with statistical analysis of corpora. By restricting the number of generated keyword forms to only the most frequent forms Kettunen, Airio and Järvelin [6] were able to achieve IR results that are 86-95% of the best available competing method, the use of a lemmatizer or stemmer for Finnish, Swedish, German and Russian in the Inquery search engine. They constructed queries manually, simulating an automatic method.. All of the languages evaluated were at least morphologically moderately complex. Automatic generation of FCG style query keywords was proven feasible with English, Finnish, German and Swedish in Lemur query system [4]. Leturia et al. [8,9] have used same type of approach, usage of the most common inflected word forms, in their web-search enhancement for the Basque language.

The FCG method and its language specific evaluation procedure can be characterized as follows:

1. For a morphologically sufficiently complex language the distribution of nominal case/other word forms is first studied through corpus analysis. The used corpus can be quite small, because variation at this level of language can be detected reliably enough even from smaller corpora [5,6]. Variation in textual styles may affect slightly the results, so a style neutral corpus, such as a collection of newspaper articles, is the best.

   By morphological complexity of a language we mean that a language uses extensively inflectional morphology. This in turn leads to high variation, i.e. different surface forms of words. English, for example, has four differing noun forms (nominative and genitive in singular and plural). We consider English to be an example of a simple inflectional system. Finnish, on the other hand, is an example of a complex inflectional system with 14 different case forms in singular and plural [4,6,7]. Many languages fall between these extremes, but there are also more complex inflectional systems.
2. After the most frequent (case) forms for the language have been identified with corpus statistics, the IR results of using only these forms for noun and adjective keyword forms are evaluated in a well-known test collection. The results are compared to the best available reductive keyword and index management method (lemmatization or stemming), if available. The number

of evaluated FCG retrieval procedures depends on the morphological complexity of the language: more procedures can be evaluated for a complex language, only a few for a simpler one.

3. After evaluation, the best FCG procedure with respect to morphological normalization is usually distinguished. The evaluation process will probably also show that more than one FCG procedure is giving quite good results, and thus a varying number of keyword forms can be used for different retrieval purposes, if necessary. This gives the method scalability and flexibility in use [5,6]. For some purposes a smaller number of variant forms might give good enough results, and in some cases a more comprehensive listing is better.[1]

It should be noted, that the FCG method usually does not outperform gold standard, usage of a lemmatizer, for morphologically complex languages. It provides, however, a simple and usually easily implementable competitive alternative for lemmatization for languages that might lack language technology tools for information retrieval. Another important point about FCG is that the identification of the frequent suffixes can be done based on a small corpus (e.g 5K words) once and they can be reused for the different corpora of the same language without further processing.

In this paper we present results of Frequent Case Generation on three Indian languages that are morphologically at least moderately complex: Bengali, Gujarati and Marathi. We compare the FCG method to two standard word form variation management methods, n-gramming [13] and stemming, our stemmer being GRAS [14]. The n-gram approach splits a document to a sequence of overlapping character n-gram tokens. One obvious drawback of n-gram method is that it significantly increases the index size and query processing time. In particular, a passage of k characters contains $k-n+1$ n-grams as compared to only $(k+1)/(l+1)$ words, where l is the average length of words and as a consequence, retrieval is 10 times slower that plain word based retrieval. In contrast, FCG increases query processing time approximately 1.2 times compared to indexing time normalization [6]. The other method, namely GRAS, is an unsupervised language independent stemmer which is based on graph paradigm found to be effective on a number of suffixing languages.

The structure of the paper is as follows: first we characterize morphological properties of the languages shortly, and then the test collections, FCG implementation and results of evaluations are described and discussed.

2 Morphological Properties

Bengali [2,11] is a highly inflectional language where one root can produce 20 or more morphological variants. Unlike English, proper nouns can also have a

[1] It should be noted, that our usage of the FCG method is a simulation, where the right forms are inserted manually to the queries. This might produce slightly better results than fully automated generation, as has been shown in Kettunen et al. [4] with Finnish, German and Swedish.

number of variations. In most cases variants are generated by adding suffixes to the end of the root. Also two or more atomic suffixes can combine to form a single suffix and inflect the root (for example, *samir-der-ke-o*, where *samir* is the root, *der, ke, o* are atomic suffixes). Nouns and pronouns are inflected in four cases: nominative, objective, genitive, and locative. The case marking pattern for each noun depends on the nouns degree of animacy. When a definite article such as *-ta* (singular) or *-gula* (plural) is added, nouns are also inflected for number. There also exists a large number of compound words having more than one root and they have a number of morphological variants. For example the word *dhan* means wealth and *haran* means robbing. These two words combine to form *dhanharan*, meaning robbing of wealth. Now *dhanharan* can produce morphological variants like, *dhanharankari*, *dhanharankarider* where *-kari* and *-der* are suffixes. New words in Bengali are formed by derivation. Derivatives and their stems may belong to different part of speech categories. Derivatives may change their form significantly from the root word and they are also formed through simple suffixation. Derived and root words very often use the same suffixes to generate their inflectional forms. Antonyms are also formed through derivation, and as in English, they are often formed by adding prefixes.

Similar to Bengali, Marathi [2,11] is also morphologically very rich and in fact richer. Its agglutinative nature complicates the vocabulary mismatch in a much broader way for an information retrieval system. Structurally, the morphological variations of Marathi are very similar to those of Bengali. The variants are generally made through adding suffixes at the end of the words. The suffixes in Marathi include indeclinable such as post-positions, adverbial markers, intensifier and case markers. Marathi nominals inflect in three cases, direct, oblique and locative, in singular and plural.

Gujarati is also an agglutinative language. Grammatical information is encoded by way of affixation (largely suffixation), rather than via independent morphemes. There are six oblique forms in Gujarati, corresponding more or less to the case forms nominative, genitive, accusative-dative, instrumental, genitive and locative. All cases are distinguished by means of postpositions. Gujarati verbs inflect for tense, aspect (perfective, imperfective), mood (indicative, imperative, subjunctive, conditional), voice (active, passive), person, number, and gender (the latter in aspectual forms only). In this way, Gujarati verbs agree with their subjects, as is the case with other Indic languages. Adjectives inflect for gender, number, and case, and thus agree with the nouns they modify. Adverbs do not inflect.

3 The Task and Data Sets

FIRE 2011 provides ad hoc mono-lingual task on Bengali, Marathi, Hindi, Tamil and Gujarati. Hindi is not suitable for the FCG method due to its very simple morphology. We also skipped Tamil as we were unable to get linguistic help for Tamil.

Table 1. Statistics on Test Corpora

Corpus →	Bengali	Marathi	Gujarati
No. of documents	457370	99275	313163
No. of unique words	1405303	862335	2104495
Mean words per/doc	337	279	450
No. of queries	50	50	50
No. of rel. doc	2778	354	1659

All of these languages contain articles from respective popular newspapers. The Bengali collection contains almost 10 years (2001-2010) worth of articles, Marathi contains 4 years (2004-2007), and Gujarati has 9 years (2002-2010).

Our experiments are on Bengali, Marathi and Gujarati. We submitted six official runs on Bengali and Marathi, and three unofficial runs on Gujarati. There are 50 queries, translated in each of these languages. In these experiments, only the 'title' and 'description' fields of topics are considered, as these are compulsory for FIRE 2011 official runs.

4 Implementation of the Methods

In order to get the case frequencies for the three languages, we carried out the following analysis. We first computed the frequency of n-gram suffixes from the unique words of the corpus in each language. Then we selected the frequent suffixes from the lexicon and sorted out only the valid case suffixes judged by a native speaker. This gave us the set of case suffixes with their frequencies. The suffixes were then sorted based on their frequency. This way we found out the top k1, k2, and k3 suffixes such that their cumulative frequency in text is 60%, 70% and 80% of the cumulative frequency of all suffixes, respectively. For Bengali, we used a separate corpus other than the retrieval corpus to identify the suffixes. However, for Marathi and Gujarati, since another corpus is not available, we used only a part of the retrieval corpus (10%) for FCG in order to minimize the overestimation.

Having selected the suffix sets, we lemmatized the topic words manually and generated queries by extending each lemma with each suffix to produce a set of inflected word forms, which were then wrapped up by Lemur's syn operator, one for each lemma and its generations. Finally, the synonym enriched queries were run on an un-normalized text index. The synonym classes generated by using 60%, 70%, 80% case suffixes are denoted as FCG-60, FCG-70, FCG-80, respectively. The average synset sizes for Bengali queries are 10.36, 13 and 16 words for FCG-60, FCG-70 and FCG-80. For Marathi queries the synset sizes are 3.5, 4 and 5.25 words, and for Gujarati queries 6.7, 9.04, and 11.6 words. Although Marathi is known to be more morphologically complex than Bengali, the synset sizes are higher for Bengali. This can happen because of the peculiarities of the suffix distributions. Bengali suffix distributions are flat as opposed to the skewed distributions of Marathi and therefore, for Bengali, a higher number of suffixes is needed to reach the required coverage percentages.

Fixed length character n-grams [12], sequences of n consecutive character are known for their language independence and they have been successfully used in quite a number languages both European and Indian. There are many variations on n-gram indexing; in this article we implement overlapping character n-grams of a fixed length. For the text *black, cat* the resulting 4-grams would be *_bla, blac, lack, ack_, ck_c, k_ca, _cat, cat_*. Clearly, the technique provides significant redundancy in text representation. This redundancy has the advantage of not requiring precise identification of root morphemes because a sliding window of length n will be sure to overlap morphemes. In general, lengths of n = 4 and n = 5 have been reported to be the most effective [13] and in our case we found 4-grams are better than 5-grams.

In GRAS [14], we start by considering word pairs of the form $\langle w_1 = ps_1, w_2 = ps_2 \rangle$ that share a sufficiently long common prefix p. We regard s_1 and s_2 as a pair of candidate suffixes only if we find a sufficiently large number of word pairs (usually around 10) of this form. The key idea here is that suffixes are considered in pairs, rather than individually. Once candidate suffix pairs are identified, we look for pairs of words that are potentially morphologically related. Two words are regarded as possibly related if (i) they share a non-empty common prefix, and (ii) the suffix pair that remains after the removal of the common prefix is a candidate pair identified in the first phase of our approach. These word relationships can be modeled by a graph, with the words being mapped to nodes, and potentially related word pairs being connected by edges. We next identify *pivot* nodes - words that are connected by edges to a large number of other words. In the final step, a word that is connected to a pivot is put in the same class as the pivot if it shares many common neighbours with the pivot, i.e. if the words that it is related to are also related to the pivot. Once such word classes are formed, stemming is done by mapping all the words in a class to the pivot for that class.

5 Results

We present our evaluation results under five metrics, namely, MAP (mean average precision), GMAP, relevant returned (Rel-ret), p@10 (precision at 10) and R-precision (average of precison at R, R is the number of relevant document for a query). GMAP [15] is the geometric mean of average precision of individual queries. Our query engine was Lemur[2], version 4.12. Our baseline is plain words, and as comparable keyword variation management methods we use n-gramming (the length of the n-gram being 4 [13]) and GRAS stemming.

5.1 Bengali Results

The retrieval results of Bengali are shown in Table 2. FCG improved the performance very significantly in comparison to baseline. FCG-60 provides relative

[2] `http://www.lemurproject.org/`

mean average precision improvement of nearly 30% over the plain word baseline. The other FCGs did not provide much better performance. Precision at 10 and R-Prec also improved noticeably. One noticeable thing is that, although FCG-60, FCG-70 and FCG-80 performed with very marginal differences in mean average precision with each other, FCG-80 was able to get a slightly better precision at 10 when compared to other FCGs. Overall, n-gramming performed best with most of the measures, but the differences to FCGs and GRAS were small. There were statistically very significant ($p = 0.0001$) differences when the Friedman test [1] was used between the baseline and all the keyword variation management methods. There were no statistically significant differences between the FCGs, n-gramming and the GRAS stemmer.

Table 2. Result Summary for Bengali

Measure	Baseline	n-gram	GRAS	FCG-60	FCG-70	FCG-80
MAP	0.2662	0.3501	0.3460	0.3447	0.3446	0.3457
GMAP	0.1726	0.2925	0.2877	0.2867	0.2865	0.2874
Rel-ret	2032	2389	2452	2381	2380	2386
P10	0.4600	0.5580	0.5500	0.5360	0.5380	0.5480
R-Prec	0.3016	0.3602	0.3608	0.3654	0.3640	0.3654

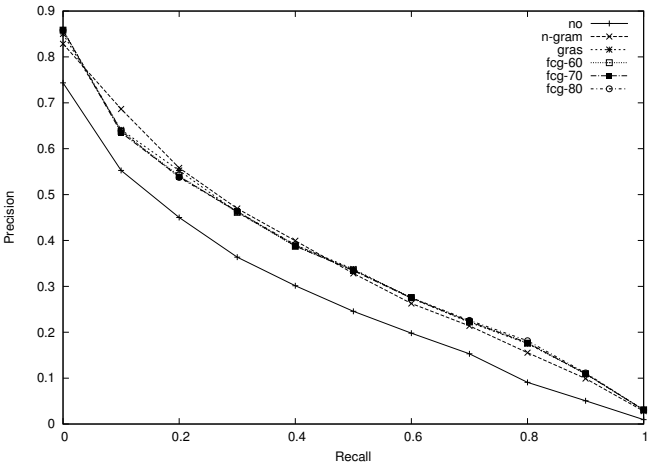


Fig. 1. Bengali P-R Curve

Figure 1 shows that each of the keyword variation management methods improved consistently the precision at every recall point compared to plain words baseline and differences among normalizations are not noticeable. The query-by-query bar graph comparing the ngram run and the best FCG run is given in Figure 2. While on average the differences between FCG-80 and n-grams are

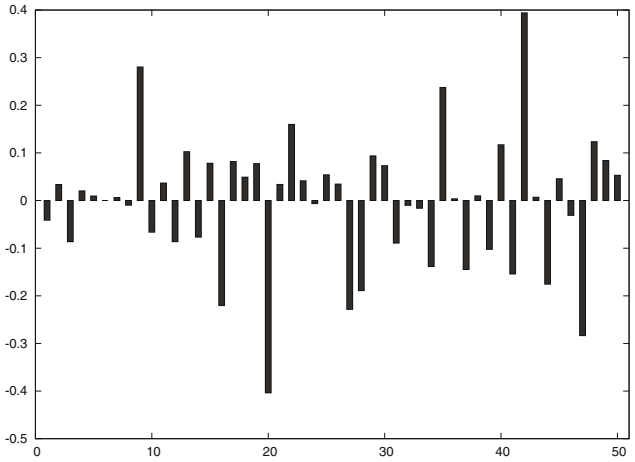


Fig. 2. Bengali query-by-query Plot. FCG-80 vs. n-gram. Relative AP.

negligible, there is a lot of variation at individual query level in both directions. FGC-80 gets a better result in 30 queries and n-gramming in 20 queries.

5.2 Marathi Results

Our next set of experiments explores the impact of FCG on retrieval in Marathi. As expected, each FCG performed noticeably better than the baseline. The relative improvement achieved by FCG-60 is 50% compared to the baseline in mean average precision. In Marathi the three FCGs provide clearly different mean average precisions with FCG-80 having the highest MAP (70 % better than the baseline and 13% and 11% better than FCG-60 and FCG-70 respectively). The same trend is observed for precision at 10 and R-Precision in Table 3. Overall, GRAS performed best with regards to MAP and R-Prec. There were statistically very significant ($p = 0.0001$) differences between the baseline and all the keyword variation management methods. There were no statistically significant differences between FCGs, n-gramming and the GRAS stemmer.

Figures 3 and 4 represent the precision recall curve and the query-by-query performance (gras vs. FCG-80) for Marathi. Following the same trend as in the MAP, FCG-80 remains very consistent in terms of improving precision at various recall points. The analysis of query-by-query performance indicates that there is lot of variation to both directions between FCG and GRAS, but the averages are close to each other. FCG performed better in 35 queries and GRAS in 15 queries.

Table 3. Result Summary for Marathi

Measure	Baseline	n-gram	GRAS	FCG-60	FCG-70	FCG-80
MAP	0.1305	0.2172	0.2432	0.1964	0.2005	0.2223
GMAP	0.0034	0.0076	0.0075	0.0066	0.0067	0.0071
Rel-ret	301	342	343	316	317	319
P10	0.1280	0.1860	0.1840	0.1840	0.1860	0.1900
R-Prec	0.1178	0.1874	0.1983	0.1684	0.1727	0.1890

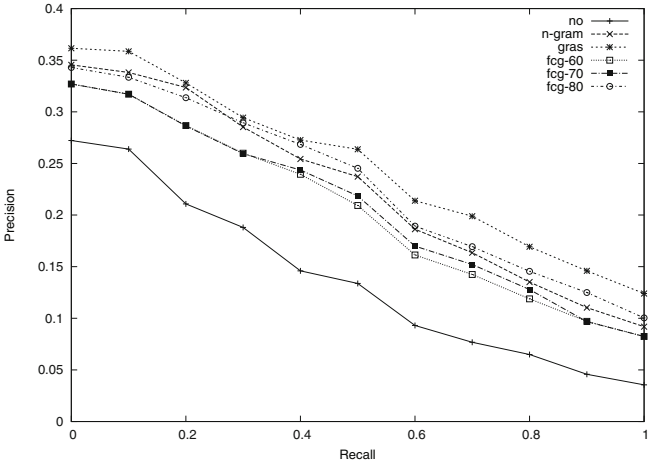


Fig. 3. Marathi P-R Curve

5.3 Gujarati Results

The retrieval results of Gujarati are shown in Table 4. FCGs improved the performance about 11-16% when compared to the baseline and about 21-23% when compared to n-gramming. The GRAS stemmer was also slightly outperformed by FCG-70 and FCG-80. Precision at 10 and R-Prec improved also slightly with all the FCGs. The best FCG outperformed n-gramming and GRAS with MAP, GMAP and R-Prec slightly, but GRAS got the best P@10.

In this case n-gram performs worse that plain word baseline in terms of precision at 10. There were statistically significant ($p = 0.01$) differences when the Friedman test was used between the baseline and n-gramming and all the FCGs. The stemmer was not significantly better than baseline or n-gramming.

Figure 5 shows that FGCs and stemming improved consistently the precision at most of the recall points when compared to no stemming and n-gramming. The query-by-query bar graph between the GRAS stemming run and the best FCG is given in Figure 6. FCG outperformed the stemmer by a small margin in a substantial number of queries with absolute per cents of 5-20, but was drastically worse in two.

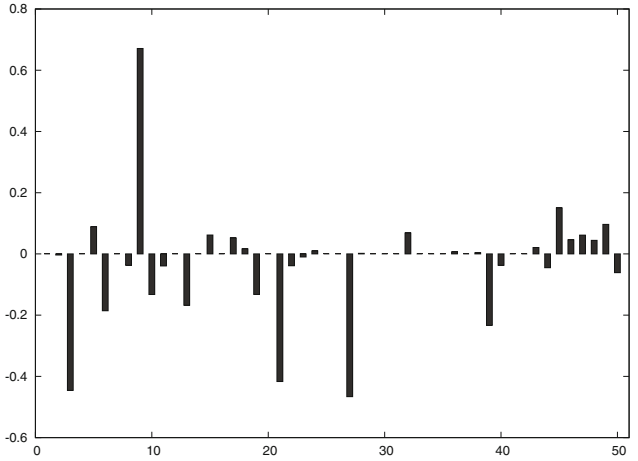


Fig. 4. Marathi query-by-query Plot. FCG-80 vs. GRAS. Relative AP.

Table 4. Result Summary for Gujarati

Measure	Baseline	n-gram	GRAS	FCG-60	FCG-70	FCG-80
MAP	0.296	0.277	0.337	0.333	0.341	0.343
GMAP	0.230	0.200	0.273	0.271	0.277	0.275
Rel-ret	1389	1346	1466	1435	1438	1440
P10	0.460	0.480	0.524	0.500	0.500	0.504
R-prec	0.332	0.313	0.362	0.354	0.366	0.360

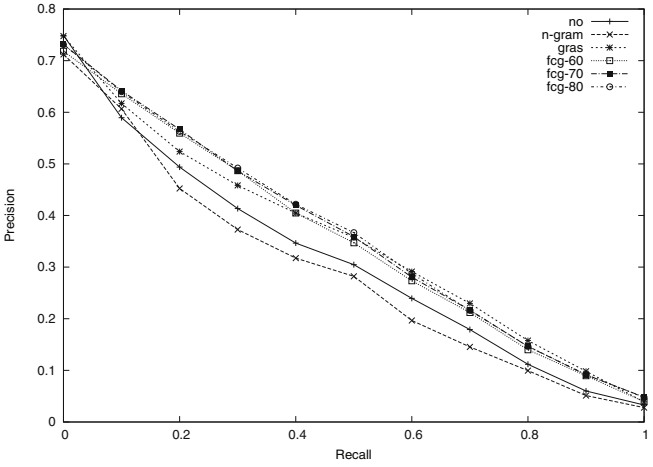


Fig. 5. Gujarati P-R Curve

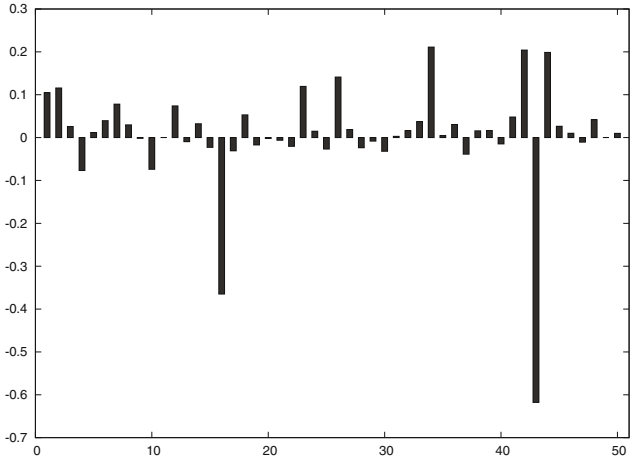


Fig. 6. Gujarati query-by-query Plot. FCG-80 vs. GRAS. Relative AP.

6 Discussions and Conclusions

We participated in the monolingual ad hoc task at FIRE 2011 with three Indian languages: Bengali, Marathi and Gujarati. We performed keyword variation management of the languages by n-gramming, stemming (GRAS stemmer) and frequent case generation, FCG, a novel method applied to the Indian languages. We submitted three FCG runs based on the title and description fields (TD) of topics for each language, totalling nine runs. Our results show that there are significant improvements in performance with FCG - 30% for Bengali, 70% for Marathi, and 16% for Gujarati - when compared to plain runs. With Bengali and Gujarati data we observed that there is no noticeable difference in MAP between the three different FCG runs. In the Marathi collection we noticed a monotonic improvement between case form coverage and mean average precision with FCG-80 giving 13% better MAP than FCG-60, and 70% better than plain runs.

The results of our evaluation show clearly that the generation of variant key word forms yields successful retrieval with Bengali, Gujarati and Marathi in standard text collections using state-of-the-art statistical best-match query engine, Lemur. These findings corroborate earlier findings on FCG in other morphologically rich languages. Our evaluation section showed that different versions of keyword generation produced effective searches that compared well with both n-gramming and stemming, two standard methods used in keyword variation management. There were differences in performance between the languages, but overall FCGs performed evenly, competitively and reliably with all the data.

In particular n-gram is very expensive in terms of consuming disk space and this goes up rapidly with the value of n. GRAS on the other hand creates long equivalence classes, since it considers both derivational and inflectional morphology.

One of the aims of information retrieval research is creation of better tools and methods for the information society. As digital content has become more common in India rather lately, there is obviously a clear need for tools that help to manage the mass of digital content. Based on our empirical results we believe that our generation approach may be useful for management of text information in the Indian languages. The method suits morphologically at least moderately complex languages and offers a simple but effective means to improve searches, both in the Web and other retrieval environments. As language technology tools might be many times missing for Indian languages, the FCG approach offers a simple solution to the management of keyword variants. When keyword variant generation is used, index can be kept un-lemmatized or un-stemmed, which leads to a higher architectural simplicity of the retrieval system, as the management of keyword variation is handled in query processing. We have also shown in our evaluation that generation is usually flexible or scalable: best results are achieved with bigger number of variant forms, but differences are not that great, and even a small number of variant forms leads to significant improvement of results when compared to doing nothing to the word form variation of the three evaluated Indian languages. Most clearly this was shown in the case of Marathi.

References

1. Conover, W.J.: Practical Nonparametric Statistics. John Wiley & Sons (December 1998)
2. Dolamic, L., Savoy, J.: Comparative study of indexing and search strategies for the hindi, marathi, and bengali languages. ACM Trans. Asian Language Information Processing 9, 11:1–11:24 (2010)
3. Kettunen, K.: Reductive and generative approaches to management of morphological variation of keywords in monolingual information retrieval: an overview. Journal of Documentation 65(2), 267–290 (2009)
4. Kettunen, K.: Automatic generation of frequent case forms of query keywords in text retrieval. In: Nordström, B., Ranta, A. (eds.) GoTAL 2008. LNCS (LNAI), vol. 5221, pp. 222–236. Springer, Heidelberg (2008)
5. Kettunen, K., Airio, E.: Is a morphologically complex language really that complex in full-text retrieval? In: Salakoski, T., Ginter, F., Pyysalo, S., Pahikkala, T. (eds.) FinTAL 2006. LNCS (LNAI), vol. 4139, pp. 411–422. Springer, Heidelberg (2006)
6. Kettunen, K., Airio, E., Järvelin, K.: Restricted inflectional form generation in management of morphological keyword variation. Inf. Retr. 10, 415–444 (2007)
7. Koskenniemi, K.: Finite state morphology and information retrieval. Natural Language Engineering 2(04), 331–336 (1996)
8. Leturia, I., Gurrutxaga, A., Areta, N., Alegria, I., Ezeiza, A.: Eusbila, a search service designed for the agglutinative nature of basque. In: Vilares, F., Lazarinis, J., Tait, J.I. (eds.) First Workshop on Improving Non English Web Searching (ACM Sigir 2007 Workshop), Marrakech, Morocco (2007)

9. Leturia, I., Gurrutxaga, A., Areta, N., Pociello, E.: Analysis and performance of morphological query expansion and language-filtering words on basque web searching. In: Choukri, K., Maegaard, B., Mariani, J., Odjik, J., Piperidis, S., Tapias, D., Calzolari, N. (eds.) Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008), Marrakech, Morocco. European Language Resources Association, ELRA (May 2008), http://www.lrec-conf.org/proceedings/lrec2008/
10. Loponen, A., Järvelin, K.: A dictionary- and corpus-independent statistical lemmatizer for information retrieval in low resource languages. In: Agosti, M., Ferro, N., Peters, C., de Rijke, M., Smeaton, A. (eds.) CLEF 2010. LNCS, vol. 6360, pp. 3–14. Springer, Heidelberg (2010)
11. Majumder, P., Mitra, M., Pal, D., Bandyopadhyay, A., Maiti, S., Pal, S., Modak, D., Sanyal, S.: The fire 2008 evaluation exercise. ACM Trans. Asian Language Information Processing 9, 10:1–10:24 (2010)
12. McNamee, P., Mayfield, J.: Character n-gram tokenization for european language text retrieval. Inf. Retr. 7(1-2), 73–97 (2004)
13. McNamee, P., Nicholas, C.K., Mayfield, J.: Addressing morphological variation in alphabetic languages. In: SIGIR, pp. 75–82 (2009)
14. Paik, J.H., Mitra, M., Parui, S.K., Järvelin, K.: Gras: An effective and efficient stemming algorithm for information retrieval. ACM Trans. Inf. Syst. 29(4), 19 (2011)
15. Voorhees, E.M.: Overview of the trec 2004 robust track. In: TREC (2004)

ISM@FIRE-2011 Bengali Monolingual Task: A Frequency-Based Stemmer

Raktim Banerjee and Sukomal Pal

Dept of CSE,
Indian School of Mines, Dhanbad
India
{ranarktm,sukomalpal}@gmail.com

Abstract. This paper describes the work that we did at Indian School of Mines, Dhanbad towards adhoc Bengali monolingual retrieval task for FIRE 2011. During official submissions, we prepared three TD runs using TERRIER search retrieval system without query expansion. When we used YASS stemmer we received substantially improved retrieval performance. Post-submission, we also developed a statistical stemmer based on frequent pattern mining using apriori-like algorithm taken from market basket data analysis. Initial results that we received for our stemmer showed noticeable retrieval performance gain over no-stem runs. Although this performance-gain is lower than that of YASS, we believe that it is promising enough to fine-tune the stemmer towards better results.

1 Introduction

This is our first year at FIRE where we participate in the Bengali adhoc monolingual retrieval task. Information retrieval in Bengali is not mature enough unlike its most European counterparts due to a number of major factors: (a) serious scarcity of digitized texts in Bengali (b) very few language-resources are developed and freely available (c) Bengali is a highly inflectional language (e.g. for a single root word there may be more than 20 morphological variants) (d) also Bengali is not an alphabetic language like English. There are lots of compound characters existing in Bengali language. Handling these compound characters is not straight-forward and therefore very often an off-the-shelf information retrieval (IR) system cannot be readily applied for Bengali retrieval. Nevertheless, the above problems are not specific to Bengali only, rather they can be generalized for a number of other Indian languages as well. Our approach this year was of very preliminary nature. Our objective was to customize a standard IR system for Bengali retrieval. We used freely available open-source TERRIER [1] system with minimal modification and submitted three runs: a) one without stemming and without query expansion (*Ism-PL2c10.99-td-noqe-nost*) b) one without query expansion but using YASS stemmer [1] (*Ism-PL2c10.99-td-noqe-st*) and c) a pruned list of stemmed words (*Ism-PL2c10.99- tdo-noqe-st*). The

[1] http://terrier.org/

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 51–58, 2013.

performance of our official submissions was moderate, but promising. As a first-timer our aim this year was to acquire FIRE relevance data.

Post submissions, we developed a language-independent statistical stemmer based on frequent suffix mining. Initial results show that our stemmer improves the retrieval performance by around 10% over our no-stem run. Although this performance is inferior to that of another statistical stemmer YASS over the same test collection, we believe the performance our stemmer based on frequent suffix pattern can be improved and it could be a potential alternative to other statistical or language-independent stemmers.

The paper is organized as follows. First we briefly summarize the related work. Next we describe our approach in Section 3 followed by data used in Section 4. We discuss our results and observations in Section 5. Finally we conclude with future scope.

2 Related Work

Stemming is a popular technique in IR which has proven to enhance recall in general, but for the morphologically complex languages improves precision as well [2]. Stemming algorithms are broadly classified into two categories, namely rule-based [3] and statistical [4,5,1]. Rule-based stemmers work on a set of pre-defined language-specic rules, whereas statistical stemmers employs statistical information from a large corpus of a given language in order to learn the morphology. A statistical stemmer obviates the need of language-specific expertise, and therefore, is often a preferred choice, specifically in information retrieval [2,4,1].

There are mainly three kinds of approaches to language-independent stemming. The first kind of methods take a set of words and try to find probable stems and suffixes for each word; other methods look for association between lexicographically-similar words by analyzing their co-occurrence in a corpus. The third group of methods is based on character n-grams.

Paik et al. [6] proposed a graph-based statistical stemmer GRAS, where a set of word classes are formed each having a *pivot* word or stem. All the words within a class share a common prefix but have different *valid* suffixes. Valid suffixes are shortlisted pairwise based on their occurrence with other prefix words in the corpus.

Majumder et al. [1] developed a clustering-based unsupervised technique (YASS) where string-distance between two words is used. A long-match in the prefix for a word pair is rewarded while an early mismatch is penalized during the complete-linkage clustering.

Bacchin et al. [5] described a probabilistic model for stem generation based on the mutual reinforcement relationship between stems and suffixes. Stems and suffixes are generated by splitting words at all possible positions. A set *good* stems and *good* suffixes are chosen using HITS algorithm.

In the work by Oard et al. [4], suffixes were discovered statistically in a text collection and the word endings were eliminated. The frequency of every one,

two, three, and four character suffix that would result in a stem of three or more characters for the first 500,000 words of the collection were collected. Then they subtracted the frequency of the most common subsuming suffix of the next longer length from each suffix (for example, frequency of "ing" from the frequency of "ng"). The adjusted frequencies were then used to sort all n-gram suffixes in descending order. The count vs rank plot was found to be convex and the rank where minimum frequency was reached was chosen as the cutoff limit for the number of suffixes for each length. Our approach is to some extent close to this work. However it differs in the fact that we try to discover suffixes based on the concept of frequent itemset generation technique according to apriori algorithm in market basket data analysis [7].

3 Approach

The Information Retrieval task is generally divided into two major components: indexing and retrieval.

3.1 Indexing

The indexing process typically represents documents as a collection of keywords and their corresponding weights. Stemming is one of the vital pre-processing steps in indexing. The words in any natural language text are inflected according to some linguistic rules of that language. Inflection may occur by adding a suffix/prefix to the terms, or in some cases the entire term may be changed. Stemming aims to identify morphological classes that share common roots. For all our runs we used title (T) and description (D) of the query but did not use any relevance feedback and/or query expansion. For the basic run (*Ism-PL2c10.99-td-noqe-nost*) we did not use any stemming either. We used the Batch (TREC) *Terrier software version 1.1.0* with *JAVA OpenJDK 1.6.0_18* on *Fedora Core 11* machine. The TREC collection class used was *TRECUTFcollection* with *UTF-8* encoding. The stopword used was provided by FIRE organizers and used off-the-shelf YASS stemmer for stemming.

We observed that a lot of indexed words are in fact not Bengali valid words. Some of these words even contained non-Bengali characters including English alphabets, digits and numbers. We manually removed these entries from the index. This pruning brought down the size of entire lexicon from 1,270,105 to 848,303. This pruned index was used for the third run (*Ism-PL2c10.99-tdo-noqe-st*).

Post-submission we also attempted to develop a statistical stemmer based on suffix-stripping. First valid suffixes are discovered based on frequent itemset generation technique as used in market basket data mining [7].

In order to find valid suffixes we reversed the strings appearing in the lexicon collected from the given document collection. We tried to find the high-frequency suffixes starting from length of 1 character. For all tokens in the lexicon we collected the collection frequency of the first character of reversed lexicon (i.e.

the last character of original lexicon). Only the characters having frequency higher than a pre-decided threshold qualify as a 1-character suffix. The set of 1-character suffixes also serve as the potential candidates while generating 2-character suffixes. An 1-character suffix which did not cross the threshold can not be part of a 2-character suffix since its frequency is expected to be less or equal to that of 1-character suffix. Hence we discarded such 1-character suffixes during discovery of 2-character suffixes. We scanned the lexicon again and collected frequency for all possible 2-character suffixes. The set of 2-character suffixes crossing the threshold are considered valid 2-character suffixes and form the candidate-set for 3-character suffixes. We continued the process so long we could generate a frequent n-character suffix (n is a dynamic integer ≥ 1). The list of frequent suffixes (having different length) are compared with the reversed list of lexicon to produce stems. Actually stems are generated by subtracting the suffixes from the original words. For tie-breaker, priority is given to suffixes of greater length. For example, if a token t is of the form $t = < p_1 s_1 > = < p_2 s_2 >$ where p_i and s_i are prefixes and frequent suffixes respectively but $len(s_1) > len(s_2)$, then p_1 is taken as the stem for t. These stems were used during document indexing.

We considered threshold-values as the fraction of total number of words (similar to *support* [7]). Note that using a fixed number as threshold may not work for different lexicon built from different corpus. We heuristically varied the threshold in the range $[0.001, 0.04]$. We started with as low as 0.1% of total number of words as a valid suffix and went up to 4%. Threshold values higher than 0.03 seemed to yield diminishing return as we received a steady downward trend in MAP values in the retrieval score.

3.2 Retrieval

We used PL2 (Poisson model with Laplace after-effect and normalization 2) model available within Batch Terrier using title (T) and description (D) part of the topics. We retrieved 1000 documents per query for a total of 50 queries.

4 Data

4.1 Documents

We used the corpus provided by FIRE 2011 organizers taken from the largest circulating Bengali newspaper Ananda Bazar Patrika of last 10 years (2001-2010) along with articles collected from Bangladesh news (BD news 24 of time span 2006-2010).

4.2 Topics

We also got fifty queries (126-175) from the FIRE 2011 website. Each query contains the information need expressed in natural-language divided into three parts title (T), description (D) and narrative (N).

4.3 Qrels

Runs were evaluated against the relevance judgment data provided by the organizers for all the 50 topics (Topic-id 126-175). These qrels are at the document level with binary relevance.

5 Results

Once we received the query relevance assessments file from the organizers, we performed evaluation for our runs. Following table (Table 1) summarizes the performance of our official submissions vis-a-vis the best performer of FIRE 2011 Bengali monolingual task. We tabulated the best result we received using our naive statistical stemmer as well.

Table 1. Performance for FIRE 2011 Bengali monolingual retrieval runs: #Topics = 50, #Reldocs = 2778

	Run-id	Relevant-Ret	MAP	R-precision
Official	Ism-PL2c10.99-td-noqe-nost	2026	0.2297	0.2773
	Ism-PL2c10.99-td-noqe-st	2026	0.2297	0.2773
	Ism-PL2c10.99-tdo-noqe-st	2157	0.2929	0.3294
	DFR_IneC2-c1d5-NNN (Best)	2534	0.3798	0.3859
Post-submission	PL2c10.99-td-noqe-yass-1.5	2423	0.3435	0.3668
	PL2c10.99-td-noqe-freq-0.030	2302	0.3182	0.3516

Post-submission we realized that we made mistake in including YASS stemmer and ended up in actually not involving the stemmer at all. Our first two official runs (*Ism-PL2c10.99-td-noqe-nost* and *Ism-PL2c10.99-td-noqe-st*), therefore, produced identical results. After using YASS stemmer, we actually received tremendous improvement. We also report here the best performance (using threshold = 0.030) that we received so far using our naive frequent-suffix-stripper stemming algorithm. Overall, our stemmer yielded about 9% improvement compared to the best no-stem run (*Ism-PL2c10.99-tdo-noqe-st*). Also, the improvement was at all recall points as evident from the Figure 1. Although this performance is inferior to that of YASS, we believe it can be further improved by fine-tuning our algorithm.

At the time of inception, we did not have any idea what can be a good operating threshold value for frequent suffix generation. We started with 0.001 or 0.1% of the total lexicon. We increased the threshold and observed almost monotonic increase (except a few variations) in MAP values (Figure 2). However after reaching 0.03, we encountered a pattern of diminishing returns in MAP. However, barring a few exceptions, in all cases, we received improvement over the no-stem run, as far as MAP values are concerned.

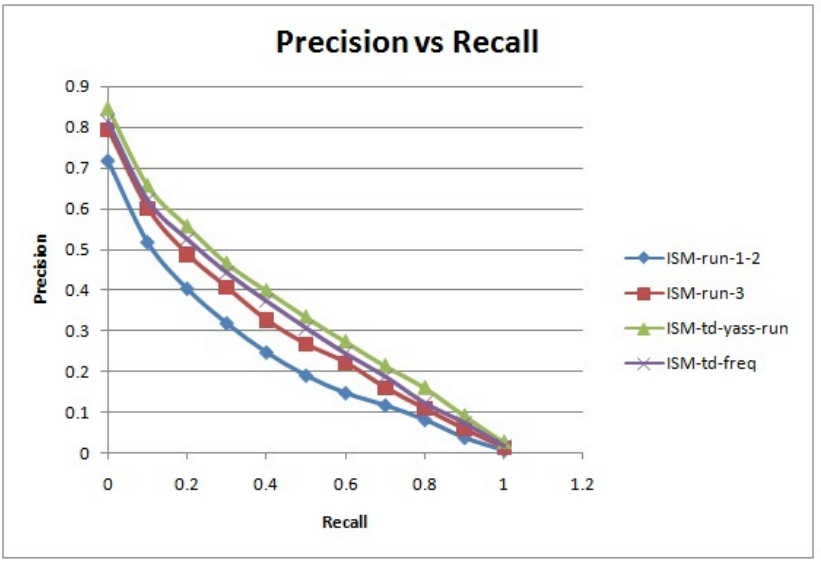


Fig. 1. Precision-Recall graph of our runs

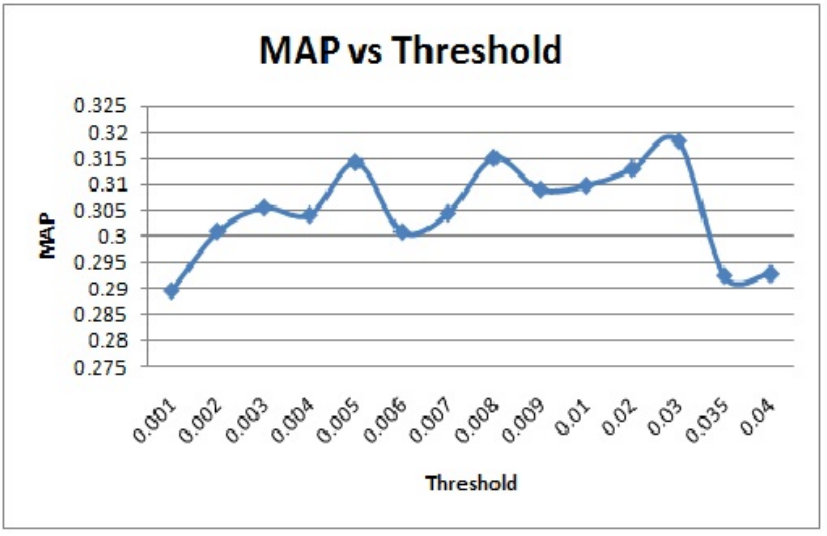


Fig. 2. Variation of MAP over threshold values

When we used very low value for threshold (e.g. 0.001) performance decreased as we generated a large number of suffixes. Some of these suffixes reduced some terms to a stem which happens to be a valid word with different meaning altogether (e.g. *akshay*, a noun (meaning something which does not decay), maps to *aksha* if *ya* is a frequent suffix, but *aksha* is a word having different meaning (means axis).

On the other hand, choosing a large threshold value (say, 0.04) actually blocks the possibility of finding any frequent suffix at all and therefore it falls flat to the no-stem case.

6 Conclusion

This paper describes our participation at FIRE 2011 Bengali Monolingual task. As a new entrant this year our aim was to gain access of a standard benchmark data needed for IR and NLP tasks in Indian languages. We participated with a bare-minimum setup based on open source tools. However post-submission we developed a naive statistical stemmer applying the notion of frequent itemset generation borrowed from market basket data analysis. Initial results that we received using Bengali test collection are interesting enough, although not overwhelming. There is obviously scope of further improvement for the stemmer. The longest suffixes are presently being given priority during tie-breaking. This adhoc solution makes several words carrying altogether different meanings mapped to the same stem. On the other hand, related words having similar prefix are being mapped to different stems. Both the scenarios add to the query drift or the vulnerability of retrieving non-relevant documents. We believe that appropriate tie-breaking scheme will augment the performance of our stemmer. Secondly, we did not consider the effect of common prefix during stem generation. Third, we tested our stemming algorithm with the Bengali test collection. We believe that it can be used and tested with other inflectional languages as well, especially other Indian languages which lack necessary language processing tools at the moment. Finally, we need to compare the performance of our stemmer with that of Oard et al. [4] or of GRAS [6] or YASS [1]. We would like to follow up these works in future.

References

1. Majumder, P., Mitra, M., Parui, S.K., Kole, G., Mitra, P., Datta, K.: Yass: Yet another suffix stripper. ACM Trans. Inf. Syst. 25(4) (October 2007)
2. Xu, J., Croft, W.B.: Corpus-based stemming using cooccurrence of word variants. ACM Trans. Inf. Syst. 16(1), 61–81 (1998)
3. Porter, M.F.: Readings in information retrieval, pp. 313–316. Morgan Kaufmann Publishers Inc., San Francisco (1997)
4. Oard, D.W., Levow, G.-A., Cabezas, C.I.: CLEF experiments at maryland: Statistical stemming and backoff translation. In: Peters, C. (ed.) CLEF 2000. LNCS, vol. 2069, pp. 176–187. Springer, Heidelberg (2001)

5. Bacchin, M., Ferro, N., Melucci, M.: A probabilistic model for stemmer generation. Inf. Process. Manage. 41(1), 121–137 (2005)
6. Paik, J.H., Mitra, M., Parui, S.K., Järvelin, K.: Gras: An effective and efficient stemming algorithm for information retrieval. ACM Trans. Inf. Syst. 29(4), 19:1–19:24 (2011)
7. Pal, S., Bagchi, A.: Association against dissociation: some pragmatic considerations for frequent itemset generation under fixed and variable thresholds. SIGKDD Explor. Newsl. 7(2), 151–159 (2005)

PAN@FIRE: Overview of the Cross-Language !ndian Text Re-Use Detection Competition

Alberto Barròn-Cedeño[1], Paolo Rosso[2], Sobha Lalitha Devi[3], Paul Clough[4], and Mark Stevenson[4]

[1] LSI, Universitat Politècnica de Catalunya, Spain
[2] NLE Lab–ELiRF, Universitat Politècnica de València, Spain
[3] AU-KBC Research Centre, Chennai, India
[4] University of Sheffield, UK
albarron@lsi.upc.edu, prosso@dsic.upv.es, sobha@au-kbc.org, p.d.clough@sheffield.ac.uk, M.Stevenson@dcs.shef.ac.uk

Abstract. The development of models for automatic detection of text re-use and plagiarism across languages has received increasing attention in recent years. However, the lack of an evaluation framework composed of annotated datasets has caused these efforts to be isolated. In this paper we present the CL!TR 2011 corpus, the first manually created corpus for the analysis of cross-language text re-use between English and Hindi. The corpus was used during the Cross-Language !ndian Text Re-Use Detection Competition. Here we overview the approaches applied the contestants and evaluate their quality when detecting a re-used text together with its source.

1 Introduction

Text re-use occurs when pre-existing written material is consciously used again during the creation of a new text or version [9,5]. This might include the re-use of an entire text (e.g. duplicate web pages), or smaller segments (e.g. chunks, paragraphs and sentences) from one or more existing text. Plagiarism, perhaps the most widely known example of text re-use, can be defined as "the reuse of someone else's prior ideas, processes, results, or words without explicitly acknowledging the original author and source" [17]. The problem has received attention from various research areas and even generated new terms such as *copy-paste syndrome* [31,18] and *cyberplagiarism* [11]. The increased availability and accessibility of content online (e.g. texts, images, videos and sounds) is making text re-use easier than ever before and subsequently the automatic detection of text re-use, and in particular plagiarism detection, is of considerable importance.[1]

Recent efforts have focussed on developing datasets with which to evaluate text re-use and plagiarism detection. The PAN International Competition on

[1] See [7,8,21,29] for an overview of the state of the art in automatic plagiarism detection.

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 59–70, 2013.

Plagiarism Detection (PAN@CLEF) [25,29][2], held in conjunction with the Cross-Language Evaluation Forum (CLEF), is perhaps the most widely-known example. Benchmarks that have been developed for the PAN competitions include corpora containing examples of automatically generated and simulated plagiarism[3] and evaluation metrics [28]. As a result, for the first time it has been possible to objectively evaluate and compare diverse methods for plagiarism detection.

The PAN@FIRE track focuses on *cross-language text re-use*, a more specific form of text re-use.[4] In the cross-language text re-use scenario the re-used text fragment and its source(s) are written in different languages, making the detection of re-use harder than when both texts are in the same language. Cross-language text re-use is an emerging research area that has begun to receive attention in recent years [4,6,19,26]. There are various motivations for this interest: (*i*) speakers of under-resourced languages [3] are often forced to consult documentation in a foreign language; (*ii*) people immersed in a foreign country can still consult material written in their native language and (*iii*) cross-language text re-use, and in particular plagiarism, is becoming a problem. However, benchmarks are needed to assist in the development and evaluation of methods for detecting cross-language text re-use. The Cross-Language !ndian Text Re-Use detection task (CL!TR)[5] at FIRE addresses this issue.

CL!TR focused on the re-use of Wikipedia articles as they are often a preferred source for plagiarised examples [16,20]. Therefore, the collection of potential sources for a given case of re-use in CL!TR is composed of Wikipedia articles on several topics, including computer science and tourism (the latter from Incredible !ndia). From a realistic point of view, a corpus with actual cases of plagiarism and text re-use would be the best option. However, two factors argued against using this kind of texts: (*i*) including actual cases of plagiarism would prevent the free distribution of the corpus (mainly for ethical and legal issues) and (*ii*) all the re-used fragments are required to be identified beforehand in order to perform an objective evaluation. Therefore, as in PAN@CLEF, PAN@FIRE relies upon artificially generated cases of re-use.

2 Corpus

A set of potential source documents written in English, D_{en}, and a set of potentially re-used documents written in Hindi, D_{hi}, were provided to participants.

[2] `http://pan.webis.de`

[3] Automatically generated plagiarism is created without any human involvement by altering a source text automatically, for example by deleting words or replacing them with equivalent terms (e.g. synonyms). Simulated plagiarism is generated manually by asking people to re-use text. PAN used automatically generated and simulated examples of plagiarism since cases of true plagiarism, where the writer has re-used text with the intent of claiming authorship, are difficult to identify and distribute.

[4] `http://www.dsic.upv.es/grupos/nle/fire-workshop-clitr.html`

[5] The name of our initiative is partially inspired by the *Incredible !ndia* campaign name (`http://www.incredibleindia.org/`).

D_{en} included a total of 5,032 Wikipedia articles; D_{hi} a total of 388 documents (shown in Table 1). As the languages of D_{hi} and D_{en} are different, the detection of text re-use becomes a more difficult task.

Table 1. CL!TR 2011 corpus statistics. The figures are shown for the two sets D_{en} and D_{hi}. The column headers stand for: $|D|$ number of documents in the corpus (partition), $|D_{tokens}|$ total number of tokens, $|D_{types}|$ total number of types. k= thousand, M = million.

Partition	$\|D\|$	$\|D_{tokens}\|$	$\|D_{types}\|$
D_{hi}	388	216 k	5 k
D_{en}	5,032	9.3 M	644 k

All the (potentially) re-use cases were manually created. The topics included are computer science and tourism. For experimental purposes, D_{hi} was divided into two: training partition (198 documents), and test partition (190 documents). D_{en} remained the same for both training and test stages. The distribution of simulated-plagiarism and original documents in D_{hi} is shown in Table 2.

Table 2. CL!TR 2011 potentially re-used documents distribution

Training partition		**Test Partition**	
Re-used	130	Re-used	146
– Light revision	30	– Light revision	69
– Heavy revision	55	– Heavy revision	43
– Exact copy	45	– Exact copy	34
Original	68	Original	44
Total	198	Total	190

The generation of the 388 potentially re-used documents in D_{hi} was inspired in the approach of [10]. Participants were provided with a set of questions and asked to write a short answer, either by re-using text from a source provided (Wikipedia) or by looking at learning material (e.g. textbook, lecture notes, or websites). To simulate different degrees of obfuscation participants were asked to use one of four methods to write the answer:

Near Copy. Participants were asked to answer the question by simply copying text from the relevant Wikipedia article (i.e. performing cut–and–paste actions). No instructions were given about which parts of the article to copy (selection had to be performed to produce a short answer of the required length, 200-300 words). Using automatic translation was mandatory.

Light Revision. Participants were asked to base their answer on text found in the Wikipedia article and were, once again, given no instructions about which parts of the article to copy. They were instructed that they could alter

the text in some basic ways including substituting words and phrases with synonyms and altering the grammatical structure (i.e. paraphrasing). Participants were also instructed not to radically alter the order of information found in sentences. Participants were allowed to use automatic translators.

Heavy Revision. Participants were once again asked to base their answer on the relevant Wikipedia article but were instructed to rephrase the text to generate an answer with the same meaning as the source text, but expressed using different words and structure. This could include splitting source sentences into one or more individual sentences, or combining more than one source sentence into a single sentence. No constraints were placed on how the text could be altered. Participants were not allowed to use automatic translation.

Non-plagiarism. Participants were provided with learning materials in the form of either lecture notes, sections from textbooks, or web pages from Incredible !ndia that could be used to answer the relevant question. Participants were asked to read these materials and then attempt to answer the question using their own knowledge (including what they had learned from the materials provided). They were also told that they could look at other materials to answer the question but explicitly instructed not to look at Wikipedia.

The first three methods are designed to generate examples of simulated-plagiarism in which the source text has been obfuscated to different levels. The final method, Non-plagiarism, generates answers which are not plagiarised to be used for comparison. This approach was originally developed for the creation of a monolingual corpus of simulated plagiarism [10]: the sources (i.e. Wikipedia articles and learning materials) were in English and participants were asked to write answers in English. The approach was adapted for CL!TR to create a cross-lingual version: participants were provided with source text in English and asked to provide answers in Hindi. Volunteers were allowed to use automatic translators when generating some of the cases, either modifying the resulting translation or not.

3 Task

The focus of CL!TR is on cross-language text re-use detection. This year we target two languages: Hindi and English. The potentially re-used documents are all written in Hindi, whereas the potential source documents are written in English (cf. Section 2).

The task is to identify those documents in D_{hi} that were created by re-using fragments from a document $d \in D_{en}$. It can be described as follows:

> Let D_{en} be a collection of documents (Wikipedia articles). Let $d_q \in D_{hi}$ be a re-used document. Given d_q, retrieve those documents $d \in D_{en}$ that are likely source texts of d_q. Afterwards determine whether the pair $p(d_q, d)$ compose a case of re-use together with its source.

This is a document level task; no specific fragments inside of the documents are expected to be identified. Determining either a text has been re-used from its corresponding source is enough. Specifying the level of re-use (Exact, Heavy, or Light) was not necessary.

For the training phase we provided an annotated corpus. The actual cases of re-use (re-used and source document) were labelled, as well as the specific kind of re-use they composed. During the test phase no annotation or hints about the cases were provided.

4 Submissions Overview

Six teams from five different countries (India, Spain, Ireland, Hong Kong, and Ukraine) participated in the competition. They were allowed to submit up to three runs in order to encourage them to considering different approaches or parameters. A total of 15 text re-use detection runs were submitted.

Most of the participants opted for a "traditional" cross-language information retrieval approach. They translated the suspicious documents in D_{hi} into English in order to perform a monolingual similarity estimation [2,14,15,22,30]. Most of these approaches exploit the Google or Bing translation services.

The prototypical —information retrieval— process that follows the language normalisation is as follows. D_{en} is indexed into a search engine (most of the participants use Nutch/Lucene) and a document d_{hi} is queried to the search engine in order to retrieve the most similar documents $d \in D_{en}$.

We now describe the information retrieval processes used by three approaches.

[2] do not apply any pre-processing to the documents in D_{en}, which are directly submitted to the index. Afterwards, the documents d_{hi} are queried against the index and the most relevant retrieved document is considered a candidate source document for d_{hi}.

[14] splits the documents in D_{en} into paragraphs and expands their vocabulary on the basis of WordNet relationships (hyponyms, hypernyms and synsets). The enriched representation of each paragraph is fed to the index. The sentences in a d_{hi} are queried against the index and the top 10 source paragraphs are retrieved. The best matches are considered in order to select pairs of re-used and source (entire) documents.

[30] used an information retrieval process for their third run. After indexing D_{en}, key phrases were extracted from d_{hi} in order to independently query the index. The most frequently retrieved document $d_{en} \in D_{en}$ by the different key phrases in d_{hi} is selected as the source document.

Instead of translating the documents, [15] use a bilingual dictionary in order to map Hindi to English words. Words for which no possible translation exists in the dictionary are transliterated. Afterwards, a similarity estimation is carried out between the representations of d_{hi} and d_{en}. [15] submitted three runs that incrementally added processing stages: (*i*) for run 1, only dictionary based mapping is applied to d_{hi}; (*ii*) for run 2 mapping and transliteration are applied to d_{hi}; and (*iii*) for run 3, in addition to the mapping and transliteration processes,

a minimal similarity threshold has to be surpassed in order to consider that d_{hi} is re-used from d_{en}.

Instead of assessing the similarity between the vocabulary in d_{hi} and d_{en}, [22] applies a fingerprinting model in order to detect exact string matches. After discarding non alpha-numeric characters, chunks of 5 words with a sliding window of 4 are hashed as in [23]. All the matches between d_{en} to d_{hi} are merged and used to estimate whether a case of re-use is at hand. The three runs of [22] consider different parameters for the fingerprinting process. The best settings are those just described.

In addition to the approach based on a search engine that was just described, [30] also submitted two more approaches based on machine learning. The model is based on a J48 decision tree classifier. For run 1 the features for the classifier were composed of the cosine similarity estimated over stemmed word 3-grams. For run 2 stopwords were removed and key phrases extracted. The relevance and length of the sequences compose the features for the classifier.

The approach of [1] is based on machine learning as well. This approach uses an SVM classifier considering features of statistical machine translation and sentence alignment models. The features for the classification process are three: (*i*) and (*ii*) are the score of the most likely alignments at sentence and paragraph level between d_{hi} and d_{en}, respectively. These scores were computed with the length based alignment algorithm proposed by [13]. (*iii*) is a lexical feature: A Hindi-English dictionary was used to gloss the Hindi documents and calculate an idf-based cosine similarity between suspicious and potential source documents.

5 Evaluation

The success of a text re-use detection model was measured in terms of Precision (P), Recall (R), and F_1-measure (F_1) —the harmonic mean of P and R— on detecting the re-used documents together with their source in the test corpus. A detection is considered correct if the re-used document d_{hi} is identified together with its corresponding source document d_{en}. For the P, R and F_1 computation, we consider three sets:

- *total detected* is the set of suspicious-source pairs detected by the system,
- *correctly detected* is the subset of pairs detected by the system which actually compose cases of re-use, and
- *total re-used* is the gold standard, which includes all those pairs which compose actually re-used cases.

P, R and F_1 are defined as follows:

$$P = \frac{\text{correctly detected}}{\text{total detected}} \qquad R = \frac{\text{correctly detected}}{\text{total re-used}} \qquad F_1\text{- measure} = \frac{2 \cdot R \cdot P}{R + P}$$

F_1-measure is used in order to compose the competition ranking.

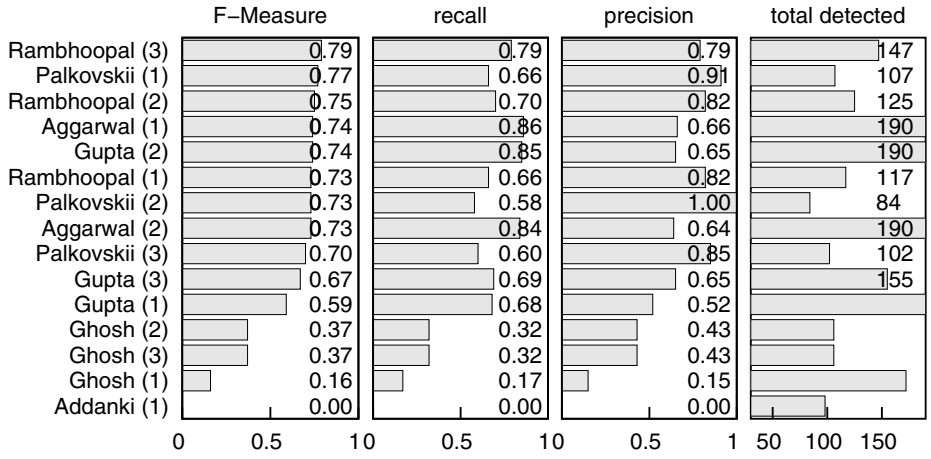


Fig. 1. *Overall* evaluation results. The left hand side information corresponds to *(run number)* and *team leader*. Additionally to rank and evaluation of the runs, *total detected* shows the total amount of candidates detected as re-used (regardless the detection was correct or not). Link between participant and citation identifier:

[30] Rambhoopal [15] Gupta
[22] Palkovskii [14] Ghosh
[2] Aggarwal [1] Addanki

The evaluation results are presented in Figure 1.[6] The most successful approaches for this task are based on standard cross-language information retrieval techniques. After translating the suspicious documents into English and building a search engine, [30] compose the queries by selecting a set of key phrases from the suspicious document. This approach strikes a good balance between recall and precision, with an F-measure of 0.79. The second best approach considers word 5-grams as terms [22]. This kind of representation is very sensitive to changes in the original text and is better suited to identifying exact matches. As a result, their obtained precision is among the highest: 0.91, with still a reasonable level of recall: 0.66. Note that in terms of F-measure, the difference between the top three approaches is only of 0.04.

On the other hand, the highest recall value is obtained by [2]: 0.86, at the cost of a slightly reduced precision: 0.66. They opt for a full representation of d_{hi} when generating the queries to the search engine [2]. Moreover, [2] decided to assume that every document in D_{hi} was re-used and simply retrieved the most similar document in D_{en}. This assumption was made by [15] as well (note that in total four submissions reported 190 documents as re-used).

In order to perform a type-wise evaluation (i.e., regarding at exact, light and heavy revisions in isolation), the actual cases of re-use and the detections of a given detector were sub-sampled as follows. Let G be the set of actual cases of

[6] Link between participants and submission reference shown in Figure 1's caption.

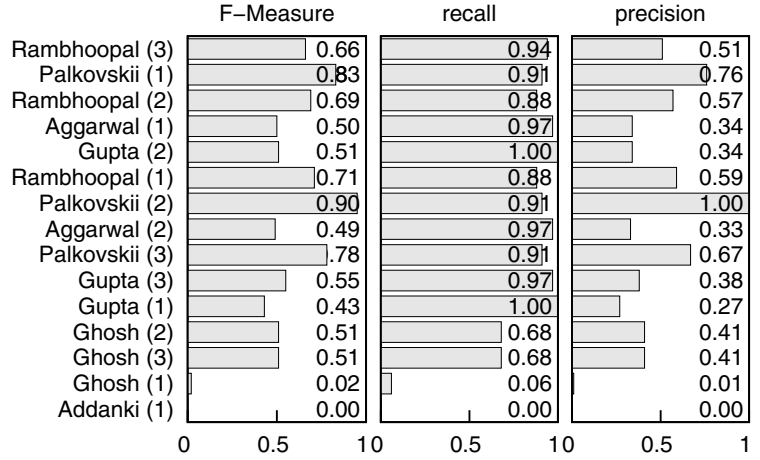


Fig. 2. Evaluation results for *exact* cases

re-use (composed of re-used and original text). Let E, L, and H be the actual cases of exact, light and heavy revisions in G, i.e.,

$$G = \{E \cup L \cup H\} \ .$$

Let P_d be the set of cases identified as re-used by a given detector. The gold standard partition considered for the evaluation when analysing exact cases is simply $G_E = E$. The partition of detections considered is defined as:

$$P_{d,E} = P_d \setminus \{p_d \mid p_d \in P_d \cap (L \cup H)\} \ ,$$

i.e., those properly detected cases that correspond to light and heavy revisions are discarded. The same procedure is followed when sub-sampling for evaluating cases of light and heavy revision. However, those cases in P_d which are not actual cases of re-use are considered in every resulting partition: $P_{d,E}$, $P_{d,L}$, and $P_{d,H}$. This does not effect recall, but reduces precision, and therefore F-measure.[7]

Figure 2 shows the results when considering the cases of *exact* cross-language re-use only, Figure 3, for *light*, and Figure 4 for *heavy* revisions. As aforementioned, these results have to be observed with caution. The precision bias caused by our sub-sampling strategy causes the approach of [22] to outperform the other participants in the three cases. Once again this was expected as they pay special attention on precision.

The rest of our type-wise analysis of the results is centred in recall. As expected, the values of recall for the exact cases are the highest, as they are the easiest to detect. Indeed, [15] reached $R = 1.0$ in two of their runs and many other participants obtained values above 0.9. Higher levels of paraphrasing cause

[7] This strategy for computing type-wise evaluations is similar to that used at PAN@CLEF.

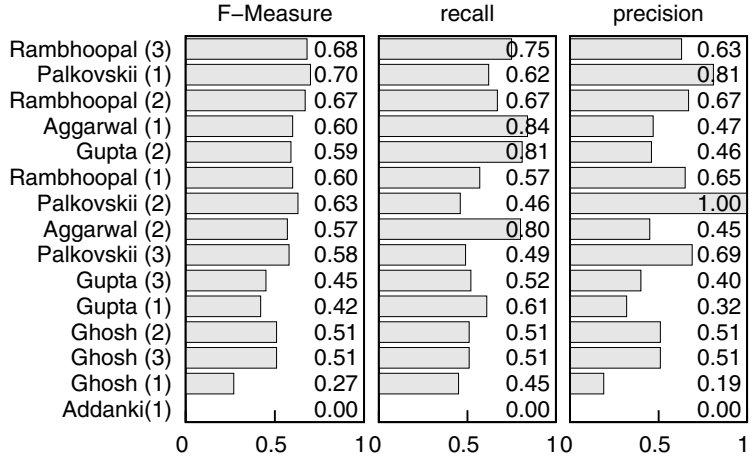


Fig. 3. Evaluation results for *light* cases

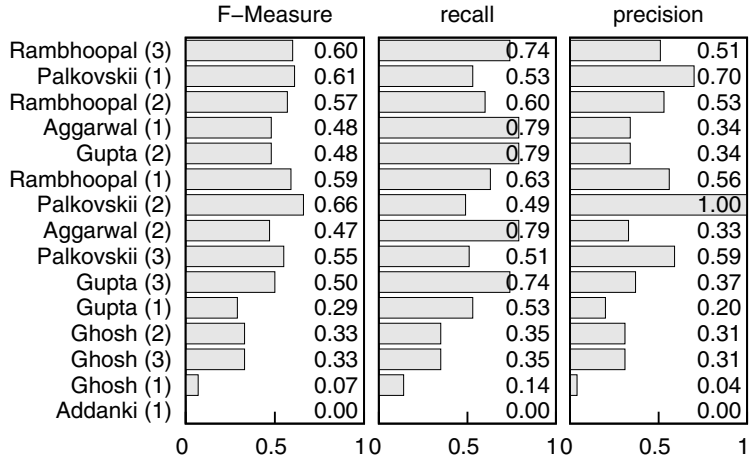


Fig. 4. Evaluation results for heavy cases

these values to decrease. Whereas the average recall of the submissions (that managed to detect at least one case), on exact cases is of 0.84, for light and heavy revisions is of 0.61 and 0.57 only. Paraphrases, also cross-language, cause problems for detectors. However, regardless the level of paraphrasing, most of the top approaches still manage to properly retrieve more than half of the cases.

The surprisingly high results obtained by some of the approaches have to be read with caution. Most of them perform language normalisation based on online translators (such as that offered by Google). When generating the cases, the volunteers were allowed to use these and other automatic tools to translate the contents they had selected to answer a given question and further modify it.

6 Final Remarks

In this paper we presented an overview of the Cross-Language !ndian Text Re-Use Detection Competition. The challenge consisted of identifying, among a set of short documents written in Hindi, those texts that had been generated by re-use and the corresponding source document written in English.

Taking advantage of the first text collection of this nature, fifteen approaches were compared. Most of them were based on standard cross-language information retrieval and some other on statistical machine translation and machine learning techniques.

As in other tasks, such as automatic plagiarism detection [27], participants obtain surprisingly high results when looking for cases of exact (cross-language) re-use, but face problem to detect further paraphrased borrowings, which represents an open issue in this task. It remains pending to observe how these systems perform when facing actual cases of re-use.

Acknowledgements. We would like to thank Pattabhi RK Rao and Mona Parakh from AU-KBC for their help in setting up the necessary resources for the competition. We also thank the volunteers from the Indian School of Mines (ISM), Dhanbad, other institutions, and AU-KBC that helped to manually creating the re-use cases in the CL!TR corpus.

This research work is partially funded by the WIQ-EI (IRSES grant n. 269180) and ACCURAT (grant n. 248347) projects, and the Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 246016 from the European Union. The first author was partially funded by the CONACyT-Mexico 192021 grant and currently works under the ERCIM "Alain Bensoussan" Fellowship Programme. The research of the second author is in the framework of the VLC/Campus Microcluster on Multimodal Interaction in Intelligent Systems and partially funded by the MICINN research project TEXT-ENTERPRISE 2.0 TIN2009-13391-C04-03 (plan I+D+i). The research from AU-KBC Centre is supported by the Cross Lingual Information Access (CLIA) Phase II Project.

References

1. Addanki, K., Wu, D.: An Evaluation of MT Alignment Baseline Approaches upon Cross-Lingual Plagiarism Detection. In: FIRE [12]
2. Aggarwal, N., Asooja, K., Buitelaar, P.: Cross Lingual Text Reuse Detection Using Machine Translation & Similarity Measures. In: FIRE [12]
3. Alegria, I., Forcada, M., Sarasola, K. (eds.): Proceedings of the SEPLN 2009 Workshop on Information Retrieval and Information Extraction for Less Resourced Languages. University of the Basque Country, Donostia, Donostia (2009)
4. Barrón-Cedeño, A., Rosso, P., Pinto, D., Juan, A.: On Cross-Lingual Plagiarism Analysis Using a Statistical Model. In: Stein, B., Stamatatos, E., Koppel, M. (eds.) ECAI 2008 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 2008), vol. 377, pp. 9–13. CEUR-WS.org, Patras (2008), http://ceur-ws.org/Vol-377

5. Bendersky, M., Croft, W.: Finding Text Reuse on the Web. In: Baeza-Yates, R., Boldi, P., Ribeiro-Neto, B., Cambazoglu, B. (eds.) Proceedings of the Second ACM International Conference on Web Search and Web Data Mining, pp. 262–271. ACM, Barcelona (2009)
6. Ceska, Z., Toman, M., Jezek, K.: Multilingual Plagiarism Detection. In: Proceedings of the 13th International Conference on Artificial Intelligence (ICAI 2008), pp. 83–92. Springer, Varna (2008)
7. Clough, P.: Plagiarism in Natural and Programming Languages: an Overview of Current Tools and Technologies. Research Memoranda: CS-00-05, Department of Computer Science. University of Sheffield, UK (2000)
8. Clough, P.: Old and new challenges in automatic plagiarism detection. National UK Plagiarism Advisory Service (2003), http://ir.shef.ac.uk/cloughie/papers/pasplagiarism.pdf
9. Clough, P., Gaizauskas, R.: Corpora and Text Re-Use. In: Lüdeling, A., Kytö, M., McEnery, T. (eds.) Handbook of Corpus Linguistics. Handbooks of Linguistics and Communication Science, pp. 1249–1271. Mouton de Gruyter (2009)
10. Clough, P., Stevenson, M.: Developing a Corpus of Plagiarised Examples. Language Resources and Evaluation 45(1), 5–24 (2011)
11. Comas, R., Sureda, J.: Academic Cyberplagiarism: Tracing the Causes to Reach Solutions. In: Comas, R., Sureda, J. (eds.) Academic Cyberplagiarism [online dossier], Digithum. Iss, vol. 10, pp. 1–6. UOC (2008), http://bit.ly/cyberplagiarism_cs
12. Majumder, P., Mitra, M., Bhattacharyya, P., Subramaniam, L., Contractor, D., Rosso, P. (eds.): FIRE 2010 and 2011. LNCS, vol. 7536. Springer, Heidelberg (2013)
13. Gale, W., Church, K.: A Program for Aligning Sentences in Bilingual Corpora. Computational Linguistics 19, 75–102 (1993)
14. Ghosh, A., Bhaskar, P., Pal, S., Bandyopadhyay, S.: Rule Based Plagiarism Detection using Information Retrieval. In: Petras, et al. [24]
15. Gupta, P., Singhal, K.: Mapping Hindi-English Text Re-use Document Pairs. In: FIRE [12]
16. Head, A.: How today's college students use Wikipedia for course-related research. First Monday 15(3) (March 2010), http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/2830/2476
17. IEEE: A Plagiarism FAQ (2008), http://bit.ly/ieee_plagiarism (published: 2008; accessed March 3, 2010)
18. Kulathuramaiyer, N., Maurer, H.: Coping With the Copy-Paste-Syndrome. In: Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2007 (E-Learn 2007), pp. 1072–1079. AACE, Quebec City (2007)
19. Lee, C., Wu, C., Yang, H.: A Platform Framework for Cross-lingual Text Relatedness Evaluation and Plagiarism Detection. In: Proceedings of the 3rd International Conference on Innovative Computing Information (ICICIC 2008). IEEE Computer Society (2008)
20. Martínez, I.: Wikipedia Usage by Mexican Students. The Constant Usage of Copy and Paste. In: Wikimania 2009, Buenos Aires, Argentina (2009), http://wikimania2009.wikimedia.org
21. Maurer, H., Kappe, F., Zaka, B.: Plagiarism - a survey. Journal of Universal Computer Science 12(8), 1050–1084 (2006)
22. Palkovskii, Y., Belov, A.: Exploring Cross Lingual Plagiarism Detection in Hindi-English with n-gram Fingerprinting and VSM based Similarity Detection. In: FIRE [12]

23. Palkovskii, Y., Belov, A., Muzika, I.: Using WordNet-based Semantic Similarity Measurement in External Plagiarism Detection - Notebook for PAN at CLEF 2011. In: Petras, et al. [24]
24. Petras, V., Forner, P., Clough, P. (eds.): Notebook Papers of CLEF 2011 LABs and Workshops, Amsterdam, The Netherlands (September 2011)
25. Potthast, M., Stein, B., Eiselt, A., Barrón-Cedeño, A., Rosso, P.: Overview of the 1st international competition on plagiarism detection. In: Stein, B., Rosso, P., Stamatatos, E., Koppel, M., Agirre, E. (eds.) SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 2009), vol. 502, pp. 1–9. CEUR-WS.org, San Sebastian (2009), http://ceur-ws.org/Vol-502
26. Potthast, M., Barrón-Cedeño, A., Stein, B., Rosso, P.: Cross-Language Plagiarism Detection. Language Resources and Evaluation (LRE), Special Issue on Plagiarism and Authorship Analysis 45(1), 1–18 (2011)
27. Potthast, M., Eiselt, A., Barrón-Cedeño, A., Stein, B., Rosso, P.: Overview of the 3rd International Competition on Plagiarism Detection. In: Petras, et al. [24]
28. Potthast, M., Stein, B., Barrón-Cedeño, A., Rosso, P.: An Evaluation Framework for Plagiarism Detection. In: Huang, C.R., Jurafsky, D. (eds.) Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010), pp. 997–1005. COLING 2010 Organizing Committee, Beijing (2010)
29. Potthast, M., Barrón-Cedeño, A., Eiselt, A., Stein, B., Rosso, P.: Overview of the 2nd International Competition on Plagiarism Detection. In: Braschler, M., Harman, D. (eds.) Notebook Papers of CLEF 2010 LABs and Workshops, Padua, Italy (September 2010)
30. Rambhoopal, K., Varma, V.: Cross-Lingual Text Reuse Detection Based On Keyphrase Extraction and Similarity Measures. In: FIRE [12]
31. Weber, S.: Das Google-Copy-Paste-Syndrom. Wie Netzplagiate Ausbildung und Wissen gefahrden. Telepolis (2007)

Cross Lingual Text Reuse Detection Based on Keyphrase Extraction and Similarity Measures*

Rambhoopal Kothwal and Vasudeva Varma

International Institute of Information Technology, Hyderabad
bhupal_iiit@research.iiit.ac.in, vv@iiit.ac.in

Abstract. Information on web in various languages is growing fast, but large amount of content still exists in English. There are several cases of English text re-use (cross language plagiarism) observed in non-English languages. Detecting text re-use in non-English languages is a challenging task due to complexity of the language used. Complexity further increases for less resource languages like Arabic and Indian languages. In this paper, we address the problem proposed in FIRE[1] CL!TR 2011[2] task of detecting plagiarized documents in Hindi language which was reused from English language source documents. We proposed three approaches using classification and key-phrase retrieval techniques. Our winning approach attained 0.792 F-measure.

1 Introduction

Growing information on the web in different languages provide many options for people searching for content. Sometimes similarity between content is observed due to myriad reasons. But reuse or plagiarism of text can be biggest concern for the original publishers. Lot of content on web in the form of text is sometimes re-written in to different languages from source English documents without any acknowledgment. Text reuse can be at various granularities. It can be a direct copying of phrases, paragraphs or complete document.

Challenge here is to identify granularity at which document was plagiarized. Imagining the size of web, it is a tedious task to identify the plagiarized text manually. Thus, systems which can detect text reuse automatically come to rescue. We submitted three runs in FIRE CL!TR 2011 task to solve above mentioned problems. In this paper, we present approaches used for runs which can automatically detect the plagiarized documents in Hindi language that are copied from English language source documents. Our third approach which secured first rank in the task differ from other approaches [2] on the following points.

* This work is partially funded by the European Commission as part of the WIQEI IRSES project (grant no. 269180) within the FP 7 Marie Curie People Framework.

[1] http://www.isical.ac.in/~fire/2011/index.html

[2] http://users.dsic.upv.es/grupos/nle/fire-workshop-clitr.html

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 71–78, 2013.

(1) It uses key-phrases instead of n-grams. Key-phrases are considered to be topics that captures the essence of a document [1].
(2) It uses less feature space, therefore less complex.

Our other two approaches which attained high precision are based on classification [3] techniques which maps cross-language documents to possible plagiarized documents of originals.

Remainder of this paper is organized into following sections. Related work Section 2 mention about earlier works similar to the task. Next Section 3 discuss about our approaches using retrieval and classification techniques. Experimental setup Section 4 give information about the collection used for experiments and evaluation metrics used to evaluate the system. Experimental results are explained in Section 5, while result analysis is done in Section 6. Conclusions and future work is discussed in Section 7.

2 Related Work

Plagiarism detection has been of interest for a long time. In this context, cross-language plagiarism detection also plays an important role as most of reused text is found in languages other than English. Martin-Potthast.et.al [4] approach use a comprehensive retrieval process and large-scale evaluation models to measure cross-language text similarity. Similar approach in [5] represented common text with a set of features that denotes its relevance and fragmentation with conjunction to supervised learning algorithms. Another approach [6] used a moving window of four word sequence and chunk ratio for identifying plagiarism passages. This approach work at finer grain level of a document than aforementioned approaches. Alberto-Barron-Cedeno.et.al [7] detected plagiarism across distant language pairs using machine translation and monolingual similarity measure. Some other approaches achieved external plagiarism detection [8] by comparing different similarity measures.

In this paper, we proposed a new approach which use automatically extracted key-phrases from documents [1] to identify the plagiarized documents in Hindi language having source language as English.

3 Approach

Most of the earlier approaches concentrated on detecting plagiarized documents where source and target documents share common language. Task proposed in FIRE CL!TR 2011 aim in detecting plagiarized documents in cross-language where suspicious documents are present in Hindi, while source documents are in English. Major challenge here is projection of source language documents to target language. This inherently creates a dependency on translation systems which can efficiently project. But, resource scarce languages like Hindi lack state-of-art machine translation systems which can project entire document to English effectively.

In-order to solve above mentioned problems, we designed different approaches mentioned below.

3.1 Classification Based Approach with Stemming (Approach 1)

Our first approach treats plagiarism detection task as a classification problem. Word features are extracted from translated Hindi documents to build a training model. Before finalizing the word features, they are stemmed to reduce the redundant words and also to reduce the dimensionality of feature vector. Features are weighed using cosine similarity between the translated Hindi documents and source English documents. Final features are then used to build a model using J48 Decision tree classifier [9]. Built model is applied on the test set of Hindi documents to detect the possible plagiarized documents. If a document gives a classification error below a certain threshold, it is considered as plagiarized document.

3.2 Classification Based Approach without Stemming (Approach 2)

Our second approach also treats plagiarism detection task as a classification problem. Approach is similar to method mentioned in Section 3.1, but without stemming of words. Also, it differs from earlier approach in weighing features. It calculates the relevance score [5], length and quantity of the word sequences of unigram features.

3.3 Cross-Lingual Key-Phrase Mapping (Approach 3)

Key-phrases in documents are used for various tasks such as document classification [10], clustering [11] and summarization [12]. In our third approach, we use key-phrases to detect cross-language plagiarized documents. Initially, pre-processed is done on English documents to eliminate junk characters and stop words. Remaining words are converted to lowercasing to minimize the redundancy. Now to extract key-phrases, n-gram filtration and term weighing scheme [1] is used.

N-gram filtration technique extracts n-grams using data compression techniques which uses simple refinements and pattern filtration algorithms. To formulate n-gram list, LZ78 [13,14] a data compression technique is used with minor modifications. Words are used in place of characters and spaces are used as delimiter between words. But all n-grams extracted cannot be key-phrases like regular verbs [3]. Such n-grams are removed to form new n-grams list. In-order to calculate the weight of possible key-phrases from new n-gram lists, we borrow an idea from earlier approaches which states that position of phrase in document can influence its importance. So, for weighing a phrase importance is given to phrase position in the sentence and document. Apparently, count of n-grams differ as we vary value of "n". In-order to reduce the bias of n-gram counts on weighing, we employ a strategy which treats each n-gram differently for key-phrase extraction.

[3] `http://englishclub.com`

Key-phrases are extracted from suspicious Hindi documents after translation into English language using Google Translation API[4]. Key-phrases are then used to query using pre-created index of source English documents using Nutch[5]. Each key-phrase of translated and plagiarized Hindi document return source English documents. For all the key-phrases in a translated Hindi document, we get N possible unique documents of source English document. All translated Hindi documents mapping to the retrieved source English documents are termed plagiarized. Figure 1 depicts the approach.

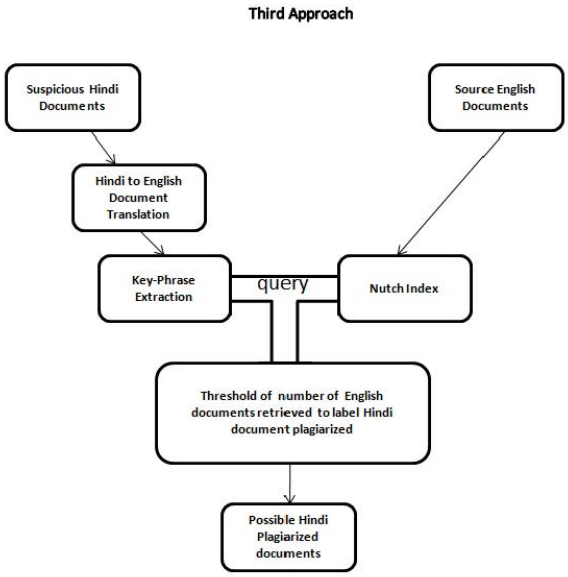


Fig. 1. Cross-lingual key-phrase mapping

4 Experimental Setup

For cross-language plagiarism task, we need a set of potential source language documents and set of suspicious target language documents. Section 4.1 show the collection used, while section 4.2 explains the evaluation metrics followed to evaluate the system.

4.1 Collection

Table 1 and Table 2 show the train and test collection used for experiments. Collection contains a plain text files encoded in UTF-8. The source documents are taken from English Wikipedia[6] which also include Wiki-mark up.

[4] https://developers.google.com/translate/
[5] http://nutch.apache.org/
[6] http://www.wikipedia.org/

Table 1. Training Data

Training Data		
	Source	**Suspicious**
English	5032	-
Hindi	-	198

Table 2. Testing Data

Testing Data		
	Source	**Suspicious**
English	5032	-
Hindi	-	190

4.2 Evaluation

Evaluation of the system which detects cross-language text-reuse is measured in terms of Precision (P), Recall (R), and F-measure (F). Identifying the correct re-used documents with its corresponding source document give the performance of the system. Some of the parameters on which evaluation is done is listed below.

(1) total detected to be the set of suspicious-source pairs detected by the system.
(2) correctly detected to be the subset of pairs detected by the system which actually compose cases of re-use.
(3) total re-used to be the gold standard, which includes all those pairs which compose actual re-used cases.

So P, R and F are defined following equations.

$$P = \frac{correctly - detected}{total - deteted} \tag{1}$$

$$R = \frac{correctly - detected}{total - re - used} \tag{2}$$

$$F - measure = \frac{2 * R * P}{P + R} \tag{3}$$

5 Experiments

Experiments were conducted using the collection mentioned in Section 4.1. Our aim is to identify the set of suspicious documents in Hindi from English source documents. Before applying our approaches, target Hindi documents of training and testing are translated to English.

5.1 Approach 1

To find the plagiarized documents, we built a model on both training and testing collection using J48 classifier with Weka[7] as mentioned in section 3.1. When a

[7] `http://www.cs.waikato.ac.nz/ml/weka/`

trained model of training data applied on translated Hindi collection of training data, we found 130 documents are classified as positive cases of text re-use and 68 documents as negative cases of text re-use. Similar experiment was performed on trained model of test data. It was found that classifier correctly identified 117 documents as re-used out of 190 in Hindi collection of test data. This approach ranked sixth in the task.

5.2 Approach 2

To find the plagiarized documents, we built a model on both training and testing collection using J48 classifier with Weka as mentioned in section 3.2. When a trained model of training data applied on translated Hindi collection of training data, we found 130 documents are classified as positive cases of text re-use and 68 documents as negative cases of text re-use. Similar experiment was performed on trained model of test data. It was found that classifier correctly identified 125 documents as re-used out of 190 in Hindi collection of test data. This approach ranked third in the task.

5.3 Approach 3

To find the plagiarized documents, we followed an approach mentioned in section 3.3 on testing data. Initially indexed documents of English collection of testing data is used to query key-phrases extracted from translated Hindi documents. We kept a minimum threshold frequency score that dictates number of times a document appears for key-phrases taken from single translated Hindi document to label the document plagiarized. During experimentation it was found that frequency score of 31 holds good to achieve better accuracies. Out of possible 190 suspicious translated Hindi documents, our approach identified 147 documents as re-used. This approach ranked first in the task.

Below we compare Precision, Recall and F-measure of our approaches with other teams participated in the task. Table 3 show the results.

6 Result Analysis

From the table 3 it can be observed that even though our approach in **Run 3** have high F-measure, it falls behind 26.7% and 7.6% on precision and recall on best results respectively. This can be attributed to the algorithm of our approach which maintained equal balance of recall and precision. While other best performing approaches concentrated on either recall or precision. When we compare our three runs, it is observed that **Run 1** attained high precision and beats **Run 3** by 3.92%. **Run 1** was able to achieve best precision due to efficiency of classification algorithm in plagiarized documents with less error. But, it lacks recall as it was not able to map all possible results in the collection. **Run 3** was able to achieve good recall due to better ability of key-phrases in distinguishing documents than normal n-grams. Approach 3 was tested with different frequency score thresholds to obtain best threshold. Table 4 lists the observations made.

Table 3. Results Comparison among Teams for CL!TR Task

Comparison among Teams for CL!TR Task					
Team	**Run**	**Precision**	**Recall**	**F-measure**	Rank
IIIT Hyderabad	1	0.820	0.657	0.730	6
IIIT Hyderabad	2	0.816	0.699	0.753	3
IIIT Hyderabad	3	0.789	0.795	**0.792**	**1**
Zhytomyr State University / SkyLine Inc.	1	0.907	0.664	0.767	2
Zhytomyr State University / SkyLine Inc.	2	**1.000**	0.575	0.730	7
Zhytomyr State University / SkyLine Inc.	3	0.853	0.596	0.702	9
DERI Galway and UPM Madrid	1	0.658	**0.856**	0.744	4
DERI Galway and UPM Madrid	2	0.642	0.836	0.726	8
UPV & DA-IICT	1	0.521	0.678	0.589	11
UPV & DA-IICT	2	0.653	0.849	0.738	5
UPV & DA-IICT	3	0.652	0.692	0.671	10
Jadavpur University	1	0.145	0.171	0.157	14
Jadavpur University	2	0.434	0.315	0.365	12
Jadavpur University	3	0.434	0.315	0.365	13
Hong Kong University of science and technology	1	0.000	0.000	0.000	15

Table 4. Comparison of Frequency Scores

Comparison of Frequency Scores			
Threshold	**Precision**	**Recall**	**F-measure**
28	0.761	0.808	0.784
29	0.776	0.808	0.792
30	0.785	0.801	0.793
31	0.789	0.795	0.792
32	0.791	0.780	0.786
33	0.850	0.780	**0.814**
34	0.856	0.773	0.813
35	0.850	0.739	0.791

7 Conclusion and Future Work

In this paper, we present different approaches to detect suspicious documents in cross-language which are plagiarized from source language English documents. Our winning approach differed from other proposed approaches in using key-phrases instead of n-grams. Other approaches used by us are based on text classification algorithms which ranked third and sixth. Future work involves semantic analysis of the text to further improve F-measure.

Acknowledgement. We would like to thank Srikanth Reddy Vaddepally and Aditya Mogadala for their constant help and discussions in improving the quality and presentation of the paper.

References

1. Kumar, N., Srinathan, K.: Automatic Keyphrase Extraction from Scientific Documents Using N-gram Filtartion Technique. In: ACM DocEng. (2008)
2. Stamatatos, E.: Intrinsic plagiarism detection using character n-gram profiles. Threshold 2, 1–500 (2009)
3. Lukashenko, R., Graudina, V., Grundspenkis, J.: Computer-based plagiarism detection methods and tools: an overview. In: International Conference on Computer Systems and Technologies (2007)
4. Potthast, M., Barron-Cedeno, A., Stein, B., Rosso, P.: Cross-language Plagiarism Detection. Springer Science+Business Media B.V. (2010)
5. Sánchez-Vega, F., Villaseñor-Pineda, L., Montes-y-Gómez, M., Rosso, P.: Towards Document Plagiarism Detection based on the Relevance and Fragmentation of the Reused Text. In: Sidorov, G., Hernández Aguirre, A., Reyes García, C.A. (eds.) MICAI 2010, Part I. LNCS, vol. 6437, pp. 24–31. Springer, Heidelberg (2010)
6. Devi, S.L., Rao, P.R.K., Ram, V.S., Akilandeshwari, A.: External Plagiarism Detection. Lab Report for PAN at CLEF (2010)
7. Cedeno, A.B., Rosso, P., Agirre, E., Labaka, G.: Plagiarism Detection across Distant Language Pairs. In: Proceedings of the 23rd International Conference on Computational Linguistics (2010)
8. Hoad, T.C., Zobel, J.: Methods for identifying versioned and plagiarized documents. Journal of the American Society for Information Science and Technology (JASIST) 54(3), 203–215 (2003)
9. Rozinat, A., van der Aalst, W.M.P.: Decision mining in proM. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 420–425. Springer, Heidelberg (2006)
10. Karanikolas, N.N., Skourlas, C.: Key-phrase extraction for classification. In: Medicon and Health Telematics: X Mediterranean Conference on Medical and Biological Engineering (2004)
11. Zhang, D., Dong, Y.: Semantic, hierarchical, online clustering of web search results. In: Yu, J.X., Lin, X., Lu, H., Zhang, Y. (eds.) APWeb 2004. LNCS, vol. 3007, pp. 69–78. Springer, Heidelberg (2004)
12. D'Avanzo, E., Magnini, B., Vallin, A.: Keyphrase extraction for summarization purposes: The LAKE system at DUC-2004. In: Document Understanding Conference (2004)
13. Sayood, K.: Introduction to Data Compression, 2nd edn. Elsevier (2000)
14. Pu, I.M.: Fundamental data Compression, 1st edn. Elsevier (2006)

Mapping Hindi-English Text Re-use Document Pairs

Parth Gupta[1] and Khushboo Singhal[2]

[1] Natural Language Engineering Lab - ELiRF
Department of Information Systems and Computation
Universidad Politécnica de Valencia, Spain
http://users.dsic.upv.es/grupos/nle
pgupta@dsic.upv.es
[2] IR-Lab, DA-IICT, India
http://irlab.daiict.ac.in
khushboo_singhal@daiict.ac.in

Abstract. An approach to find the most probable English source document for the given Hindi suspicious document is presented. The approach does not involve complex method of Machine Translation as a language normalization step, rather relies on standard cross-language resources available between Hindi-English and calculates the similarity using the Okapi BM25 model. We also present the further improvements in the system after the analysis and discuss the challenges involved. The system is developed as a part of CLiTR competition and uses the CLiTR-Dataset for the experimentation. The approach achieves the recall of 0.90 - the highest and F-measure of 0.79 - the 2^{nd} highest reported on the Dataset.

1 Introduction

Text re-use, here, tries to project the phenomena of 'Plagiarism' where it refers to "the unauthorized use or close imitation of the language and thoughts of another author and the representation of them as one's own original work, as by not crediting the author"[1]. Easy access to the information with prolific World Wide Web makes it essential to check the authenticity of the work in certain texts like research papers, dissertations, student reports and so on. Cross-language text re-use is a special case where the information is taken from the the source which is in different language.

Recently, text re-use detection has attracted the attention of information retrieval (IR) and natural language processing (NLP) communities and the state-of-the-art is being advanced with evaluation campaigns like PAN[2] at cross-language evaluation forum (CLEF)[3]. Text re-use system identifies the re-used text fragments in the given suspicious documents, if any, from the source documents available. The text re-use detection systems are broadly comprised of 4 steps [Potthast et al., 2009]:

1. pre-processing, which consists of the normalization of text, language identification and/or translation of documents;

[1] http://dictionary.reference.com/browse/plagiarism
[2] http://pan.webis.de
[3] http://clef-campaign.org/

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 79–85, 2013.

2. selection of candidate documents, i.e., the selection of a small subset of a large collection as potential source of text re-use;
3. detailed analysis, which implies the investigation of suspicious and source documents in detail to identify the re-used text sections; and
4. post-processing, which consists of merging the detected parts of a single re-use case, removing detected cases which are properly cited.

Recently, a few approachs to address cross-language text re-use (CLTR) and plagiarism detection (CLPD) are reported. In [Ceska et al., 2008], the cross-language similarity between texts is calculated using multilingual thesaurus like EuroWordnet and language specific lemmatizer is used as a pre-processing step. In [Steinberger et al., 2002], cross-lingual conceptual thesaurus is used to map a list of concepts to the text in question, and later the similarity is measured in terms of number of matching concepts. The approach presented in [Potthast et al., 2008], exploits the vocabulary correlations in the comparable corpus. [Pinto et al., 2009] measures the cross-language similarity based on the statistical bilingual dictionary generated from the parallel corpus - similar to one used for machine translation. Most popular approach, noticed in the recent editions of PAN , is to translate the documents into the language of comparison using any of the available machine translation (MT) service and then carry monolingual analysis. Some of the popular approaches used for cross-language information retrieval (CLIR) are also tested to detect CLTR like character n-gram model [McNamee and Mayfield, 2004] for the languages which share the same script and can be found in [Potthast et al., 2011].

In our approach, we transform the Hindi documents in English comparable space by the means of available resources like bilingual dictionary, wordnet and transliteration engine. Thereafter, we calculate the similarity based on the probabilistic model Okapi BM25 [Robertson and Spärck Jones, 1994]. From our experiments and analysis in [Rao et al., 2011, Gupta et al., 2011], we believe that similarity based on words has an edge over that based on word n-grams because the in the cross-language environment the sequential information of terms is generally not preserved and hence the latter does not produce the best results. We also notice that bilingual dictionary and transliteration engine produces the best results.

The problem statement, of CLiTR track, is to identify the most potential source document for the text re-use, if any, in the given suspicious document. The source documents are in the English while the suspicious documents are in Hindi. The system, developed to address the aforementioned problem, is described in Section 2. We report the results in Section 3 while in Section 4 we present the analysis of the results. Finally in Section 5 we conclude the work and talk about future activities.

2 Approach

In the standard setting of candidate retrieval, for each suspicious document d_i, a set of source documents S', from entire collection of source documents S, is retrieved, where $|S'| << |S|$. Each candidate document s_i in S' is further compared in detail with the d_i in the following stage to find the fragment level text re-use. In contrast to this, the problem statement in CLiTR limits itself to find the source document of text re-use, if

any, rather finding the re-used fragments. Therefore, we consider it a candidate retrieval step with aim to maximize Recall@1.

The strategy adopted has its roots in cross-language information retrieval (CLIR). The approach is described in Figure 1. The resources involved are described below:

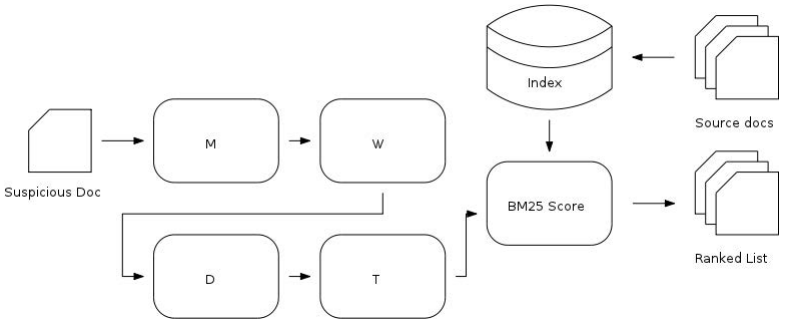


Fig. 1. Block diagram of the approach

It is a two phase process where, first phase tries to bring the document in Hindi and English to comparable space by the means of available cross-language resources, while in phase-2 the similarity is calculated to find the source document of text re-use. These phases are described below

2.1 Resources

In order to compare the Hindi suspicious documents with English source documents, we use different natural language resources like morphological analyser, bilingual dictionary, wordnet and transliteration system. We have tested the impact of these resources on the detection results are reported in Section 3.

Morphological Analyser (M): In order to find the root of each term t_i in suspicious document d, we incorporate morphological analyser. The intension behind this step is to maximize the probability of getting an entry in the bilingual dictionary for term t_i. We used Stemmer proposed by [Ramanathan and Rao, 2003] which largely handles inflectional morphology. It does not account for most of the derivational morphology of Hindi.

Wordnet (W): For each term t_i in suspicious document d, we retrieve all its senses and synonyms from the Hindi Wordnet[Narayan et al., 2002] if it has an entry in wordnet.

Bilingual Dictionary (D): We substitute each term t_i of suspicious document d by its corresponding English dictionary entry if any. We use The Hindi Universal Word (UW) dictionary[4], which contains total 134968 words.

[4] `http://www.cfilt.iitb.ac.in/hdict/webinterface_user/index.php`

Transliteration (T): If the term t_i of the suspicious document q has no entry in the dictuonary then we transliterate t_i using Google Transliterate API[5].

2.2 Similarity Score

We index all the English source documents in S. Each suspicious document d_i in D is fired as a query on this index and the ranklist is retrieved. The s_i with the highest BM25 score for given d_i is considered to be the potential source of re-use for d_i.

We also introduce a similarity threshold θ. If a suspicious document does not have any source document with similarity score above θ, we consider it free from text re-use. Though we intend to introduce this mechanism in hope to take care of false positives, it is not the best strategy in the present form as discussed in Section 4.

3 Results

We tested the above mentioned strategies on the training & test data. Table 1 contains the results on training data.

Table 1. Results on training data

Method	Precision	Recall	F-Measure
D	0.455	0.692	0.549
W+D	0.172	0.262	0.207
D+T	0.505	0.769	0.610

After looking at the high value of recall we worked on improving the precision. In order to reduce the false positives, we introduced a threshold on similarity score. Figure 3 describes the evaluation performance with different threshold values on training data.

It can be seen that setting the θ below 9.0 will hurt the precision without gaining in terms of recall, similarly, setting it above 20.0 will hurt recall greatly. So we set the θ = 15.0, between 9.0 and 20.0 based on empirical tuning, which achieves the maximum F-Measure on training data.

Table 2 show the results achieved on the test data.

Table 2. Results on test data

Method	Precision	Recall	F-Measure
D (Run-1)	0.342	0.580	0.430
W+D	0.263	0.343	0.298
D+T (Run-2)	0.474	0.804	0.596
D+T+θ (Run-3)	0.439	0.607	0.509

[5] `www.google.com/transliterate`

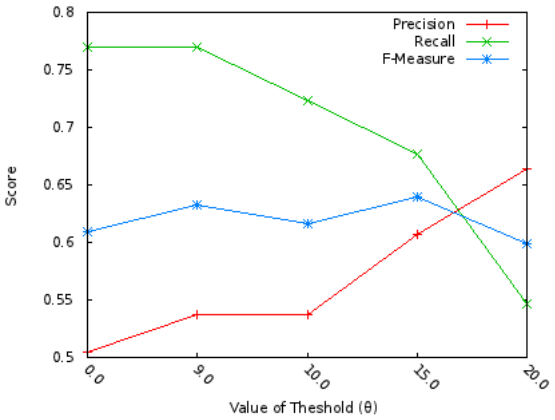


Fig. 2. Behaviour of evaluation measures with threshold values

We further carried the tuning of threshold according to the test corpus and realised $\theta = 64$ produces best results. We also incorporate the morphological analyser and the results are depicted in Table 3, which involved the highest recall achieved and 2^{nd} highest F-measure score in the competition.

Table 3. Improved results on test data

Method	Precision	Recall	F-Measure
D+T+θ	0.850	0.739	0.783
M+D+T	0.695	0.904	0.786

4 Analysis

From the results depicted in Table 2, it is noticeable that bilingual dictionary was itself capable to help in identifying 58% of re-used documents. The average length of suspicious documents is 358 and the average hit in dictionary look up is 104 terms. This signifies that there are lot of terms which does not participate in similarity, can improve the detection.

We also considered a fact that, if the word itself is not in the dictionary then we can find its all possible senses and synonyms to look in dictionary with the hope to increase the similarity. Surprisingly the performance deteriorates as can be seen in Table 2, we believe incorporation of all the senses triggered topic drift. Wordnet incorporation in the present state is not useful and there has to be a logical criterion for inclusion of a particular sense for similarity, which we aim to investigate in future.

With our experience with named entities (NEs) in [Gupta et al., 2010], we considered to transliterate all the words, which could not be found in dictionary. These words have high probability to be a NE. The results in Table 2, achieved with transliteration, confirms this hypothesis and the recall is increased to 85%.

The similarity threshold in order to abandon false positives, improves the performance as shown in Table 3. This threshold selection strategy is not comparable across corpora and hence is unreliable. As we discussed in Section 1, the main aim of our approach is to maximize the recall and provide the candidate list to the later stages, where the documents are compared in detail at sentence or fragment levels. These later stages take care of false positives. We also intend to further investigate a robust document level threshold in future.

In further analysis of results we noticed that some of the Hindi terms are in their morphological form while the root term exists in the dictionary. Hence we incorporate the morphology analyser which further improves the performance as depicted in Table 3 and achieves the recall as high as 90%.

Table 4 presents the performance evaluation of the best run (M+D+T) for the different types of re-use cases (exact, heavy and light).

Table 4. Performance evaluation based on amount of re-use

Type	Exact	Heavy	Light
F-measure	1.000	0.907	0.855

5 Conclusion and Future Work

The obtained results suggest that available resources are capable enough in finding the text re-use document pairs for Hindi-English. Transliteration helps in identifying the named entities and contribute to obtain a higher recall. Morphological analyzer is a better option to perform look-up in dictionary. In future, we wish to work on the precision of the system. Moreover, we would also like to investigate the better way to incorporate wordnet.

Acknowledgment. The work of the first author has been partially funded by the European Commission as part of the WIQ-EI IRSES project (grant no. 269180) within the FP 7 Marie Curie People Framework.

References

[Ceska et al., 2008] Ceska, Z., Toman, M., Jezek, K.: Multilingual plagiarism detection. In: Dochev, D., Pistore, M., Traverso, P. (eds.) AIMSA 2008. LNCS (LNAI), vol. 5253, pp. 83–92. Springer, Heidelberg (2008)

[Gupta et al., 2010] Gupta, P., Rao, S., Majumder, P.: External plagiarism detection: N-gram approach using named entity recognizer - lab report for pan at clef 2010. In: CLEF (Notebook Papers/LABs/Workshops) (2010)

[Gupta et al., 2011] Gupta, P., Singhal, K., Majumder, P., Rosso, P.: Detection of Paraphrastic Cases of Mono-lingual and Cross-lingual Plagiarism. In: ICON 2011. Macmillan Publishers, Chennai (2011)

[McNamee and Mayfield, 2004] McNamee, P., Mayfield, J.: Character n-gram tokenization for european language text retrieval. Inf. Retr. 7(1-2), 73–97 (2004)

[Narayan et al., 2002] Narayan, D., Chakrabarti, D., Pande, P., Bhattacharyya, P.: An experience in building the indo wordnet - a wordnet for hindi. In: First International Conference on Global WordNet, Mysore, India (2002)

[Pinto et al., 2009] Pinto, D., Civera, J., Barrón-Cedeño, A., Juan, A., Rosso, P.: A statistical approach to crosslingual natural language tasks. J. Algorithms 64(1), 51–60 (2009)

[Potthast et al., 2011] Potthast, M., Barrón-Cedeño, A., Stein, B., Rosso, P.: Cross-Language Plagiarism Detection. Language Resources and Evaluation, Special Issue on Plagiarism and Authorship Analysis 45(1) (2011)

[Potthast et al., 2008] Potthast, M., Stein, B., Anderka, M.: A wikipedia-based multilingual retrieval model. In: Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., White, R.W. (eds.) ECIR 2008. LNCS, vol. 4956, pp. 522–530. Springer, Heidelberg (2008)

[Potthast et al., 2009] Potthast, M., Stein, B., Eiselt, A., Barrón-Cedeño, A., Rosso, P.: Overview of the 1st International Competition on Plagiarism Detection. In: Stein, B., Rosso, P., Stamatatos, E., Koppel, M., Agirre, E. (eds.) SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 2009), pp. 1–9. CEUR-WS.org (2009)

[Ramanathan and Rao, 2003] Ramanathan, A., Rao, D.D.: A lightweight stemmer for Hindi. In: Computatinal Linguistics for South Asian Languages, Budapest (April 2003)

[Rao et al., 2011] Rao, S., Gupta, P., Singhal, K., Majumder, P.: External & intrinsic plagiarism detection: Vsm & discourse markers based approach - notebook for pan at clef. In: CLEF (Notebook Papers/Labs/Workshop) (2011)

[Robertson and Spärck Jones, 1994] Robertson, S., Spärck Jones, K.: Simple, proven approaches to text retrieval. Technical Report UCAM-CL-TR-356, University of Cambridge, Computer Laboratory (1994)

[Steinberger et al., 2002] Steinberger, R., Pouliquen, B., Hagman, J.: Cross-lingual document similarity calculation using the multilingual thesaurus EUROVOC. In: Gelbukh, A. (ed.) CICLing 2002. LNCS, vol. 2276, pp. 415–424. Springer, Heidelberg (2002)

Text Retrieval Using SMS Queries: Datasets and Overview of FIRE 2011 Track on SMS-Based FAQ Retrieval

Danish Contractor[1], L. Venkata Subramaniam[1], Deepak P.[1], and Ankush Mittal[2]

[1] IBM Research - India, New Delhi and Bangalore, India
[2] Graphic Era University, Dehradun, India
{dcontrac,lvsubram,deepak.s.p}@in.ibm.com,
director5research@gmail.com

1 Introduction

Analytics for noisy text has recently been of much interest[1]. Short text snippets sent using the short messaging service (SMS) has been one of the popular sources of noisy text. Due to a multitude of factors such as inconvenience in using the small mobile keyboard and the inherent carelessness while typing on-the-go, SMS authors often tend to shorten messages using compression techniques such as dropping vowels in words, replacing words by their shorter phonetic substitutions and dropping entire words. Due to the non-standard nature of such shortening, it becomes awkward to process SMSes electronically. For example, the word *thanks* may be shortened to *thx* or *tnx*; any text analytics technique that cannot identify that these three vairants are used to indicate the same concept is bound to be less accurate. Off late, there has been a wide interest in cleansing such data in an unsupervised [3] or supervised fashion [4] so that they may be better processed electronically. Hidden Markov models have been found to be reasonably effective in cleansing SMS text [2].

In the SMS-based FAQ retrieval track at FIRE 2011, we focus on the retrieval of text from Frequently Asked Questions repositories when the queries that are posed to a retrieval system are in the form of SMS messages. Systems for retrieving information (e.g., examination results, travel ticket status etc) using SMS queries typically enforce a strict format restriction on the SMS messages[2] [1]. Some recent research looks into addressing challenges in handling free-form text messages as queries in retrieval systems [6]. Through this task on SMS-based FAQ retrieval, we hope to elevate the importance of the problem and release datasets to spark more interest in text retrieval using SMS queries.

2 Tasks

Our SMS-based FAQ retrieval contains three major sub-tasks. These are described in separate sections herein. We then describe some terminologies that we will use in the rest of the paper.

[1] http://en.wikipedia.org/wiki/Noisy_text_analytics
[2] e.g., http://www.examresults.net/sms.htm

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 86–99, 2013.

2.1 Mono-Lingual FAQ Retrieval

This task presents a simple retrieval problem. A large set of FAQs, each containing a question and a corresponding answer, *all in the same natural language*, is made available to the retrieval engine. Each query is represented by an SMS *in the same language*, typically, a single SMS not containing more than 160 characters. Being SMSes, the queries would contain typical SMS-type noise such as shortened and non-grammatical text. A simple exemplary retrieval solution that maintains an FAQ repository $\mathcal{F} = \{[q_1, a_1], \ldots, [q_n, a_n]\}$, comprising n FAQs (with the i^{th} FAQ having the question part q_i and the answer part a_i) could perform the following steps:

1. Normalize the q to remove noise according to a noise-removal model such as [2]; let the normalized query be q'
2. Find the top-k similar questions to q' from $\mathcal{F}$ as $\{q_{r1}, q_{r2}, \ldots, q_{rk}\}$; k is the number of results solicited by the user or determined by the physical limit of the device (e.g., display screen size, resolution)
3. Output the corresponding answers $\{a_{r1}, a_{r2}, \ldots, a_{rk}\}$

Alternatively, Steps 1 and 2 could be interleaved to exploit any synergy between the tasks they consider. Steps 2 and 3 are similar to analogous steps in a case-based reasoning system [5]. Though retrieving useful answers to the SMS query would be the target of a real system, since our task is focussed on developing techniques that can be robust to SMS-type noise, we only evaluate whether the retrieval solution is able to identify the correct question corresponding to the SMS query (i.e., steps 1 and 2 in the exemplary solution outlined above). This leads to the following problem definition:

Definition 1. *Consider a set of FAQs, $\mathcal{F} = \{[q_1, a_1], \ldots, [q_n, a_n]\}$, written in the natural language $\mathcal{L}1$. For each SMS query q also in the natural language $\mathcal{L}1$, retrieve a set of k FAQs from $\mathcal{F}$ such that the SMS query corresponds well to the question part of those FAQs in terms of semantics. The retrieved FAQs may be scored to indicate the match between them and the SMS query q.*

To simplify evaluation, we consistently use $k = 5$ in this task.

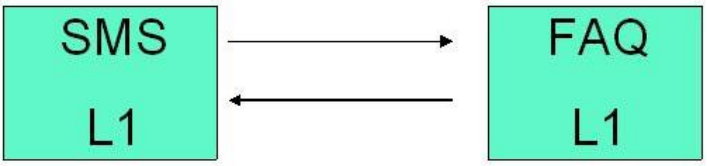


Fig. 1. Mono-lingual Retrieval Task

2.2 Cross-Lingual FAQ Retrieval

This task is similar in all respects to the mono-lingual retrieval task in Section 2.1 except that the natural language in which the FAQs are written and those of the SMS queries are different. We put down the problem definition as follows:

Definition 2. *Consider a set of FAQs, $\mathcal{F} = \{[q_1, a_1], \ldots, [q_n, a_n]\}$, written in the natural language $\mathcal{L}2$. For each SMS query q also in the natural language $\mathcal{L}1$, retrieve a set of k FAQs from $\mathcal{F}$ such that the SMS query corresponds well to the question part of those FAQs in terms of semantics. The retrieved FAQs may be scored to indicate the match between them and the SMS query q.*

An exemplary solution for this task may learn a translation model between the languages $\mathcal{L}1$ and $\mathcal{L}2$, and use it to translate the SMS query from its source language to the language of the FAQs, as an additional step between Steps 1 and 2 in the solution outlined in Section 2.1.

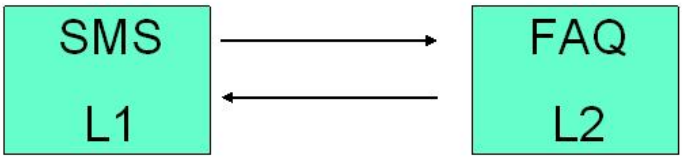


Fig. 2. Cross-lingual Retrieval Task

2.3 Multi-Lingual FAQ Retrieval

In this task, we provide an FAQ set that would contain FAQs across different natural languages. In particular, one FAQ in the set may be in a language $\mathcal{L}1$, and another may be in a language $\mathcal{L}2$; however, each FAQ have a question part and an answer part, both written in the same language. Analogously, the SMS query may also be in one of the many langauges. The framework of the retrieval task still remains the same as in Sections 2.1 and 2.2.

Definition 3. *Consider a set of FAQs, $\mathcal{F} = \{[q_1, a_1], \ldots, [q_n, a_n]\}$, each written in one of the natural languages $\{\mathcal{L}1, \mathcal{L}2, \mathcal{L}3\}$. For each SMS query q in one of the the natural languages from $\{\mathcal{L}1, \mathcal{L}2, \mathcal{L}3\}$, retrieve a set of k FAQs from $\mathcal{F}$ such that the SMS query corresponds well to the question part of those FAQs in terms of semantics. The retrieved FAQs may be scored to indicate the match between them and the SMS query q.*

A simple solution for this task could translate the cleansed SMS query q into each of the languages, leading to three versions $q1, q2, q3$ in languages $\mathcal{L}1, \mathcal{L}2, \mathcal{L}3$ respectively. These may then be fired separately against the subsets of $\mathcal{F}$ comprising of FAQs only from each of the languages. The results from such queries on subsets of $\mathcal{F}$ may then be combined to form a single set of k FAQs, for each SMS query q.

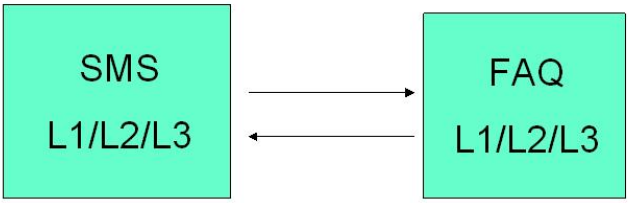


Fig. 3. Multi-lingual Retrieval Task

2.4 Other Details

We now descibe some of the terminologies that we will use in the rest of the paper to aid easy understanding of the tasks.

Training Dataset: The training dataset comprises of two parts; the FAQ part and the SMS query part.

The FAQ part contains FAQs across three languages $\{\mathcal{L}1, \mathcal{L}2, \mathcal{L}3\}$; we refer to the subset of FAQs that are in those languages as $\mathcal{F}1$, $\mathcal{F}2$ and $\mathcal{F}3$ respectively. Each of these are used for the separate runs in the monolingual task and the cross-lingual task, whereas all these put together form the FAQ dataset for the multi-lingual task.

The set of SMS queries $\mathcal{Q}$ is also similarly segregated language-wise into $\mathcal{Q}1$, $\mathcal{Q}2$ and $\mathcal{Q}3$. Each query q is labeled with a set of matching FAQs $m(q)$, which could come from across languages. While we will use the entire set $m(q)$ for the multi-lingual task, we will just use the subset of $m(q)$ that pertains to the language of the appropriate FAQ dataset, for the mono and cross lingual tasks.

These and other data could be used to learn models to correct SMSes, create translation models to translate SMSes and do other kinds of learning that could be used to retrieve FAQs for queries from the test dataset.

Test Dataset: The test dataset comprises of just SMS queries (e.g., q) and their corresponding answer sets (i.e., $m(q)$); the answer sets point to FAQs from the training FAQ dataset. The techniques proposed for each of the tasks would be evaluated on the corresponding language-specific subset of queries from the test dataset.

Out-of-domain Queries: For a task which operates on a FAQ dataset $\mathcal{F}'$, we may have SMS queries in both the training and test datasets that have no matching FAQs in $\mathcal{F}'$. These queries are called out-of-domain queries since no correct matching FAQ exists in the corresponding FAQ datasets. Such queries are common in realistic scenarios since the FAQ dataset may be incomplete, and users who are either unaware of the FAQ dataset details may pose queries that may not be handled by any FAQ in the dataset.

3 Datasets

The dataset[3] comprises FAQs and SMSes in English, Hindi and Malayalam. The details about the dataset are given in Figure 4.

[3] The data can be downloaded from:
`http://www.isical.ac.in/~fire/faq-retrieval/2011/data.html`

The English FAQs numbering 7251 are collected from online sites of banks, railway services, government departments and others. They are categorized into 15 domains that include career, agriculture, general knowledge, health, insurance, sports, tourism, bank, loan, personality development, recipes, visa, web, telecommunication and railways.

The 1994 Hindi FAQs are from 10 domains that include agriculture, Bharat, business, constitution, general knowledge, health, railways, Rajya Sabha, telecommunication and videsh. These are collected from online sites and some are generated by manually translating the English FAQ into Hindi.

The 681 Malayalam FAQs are from 3 domains that include railways, general knowledge, telecommunication. They were all generated by manually translating the English FAQs in these domains into Malayalam.

The SMSes in English were generated by a several people who were told the domains and also shown the FAQs in these domains and asked to write an SMS query. These volunteers were mostly engineering college students who were instructed to write the SMS query in their usual texting language.

The Hindi and Malayalam SMSes were generated by taking the FAQ questions and dropping diaclitic marks and some non-content words.

The SMS queries in all the languages were both in-domain and out-domain. In most cases the number of in-domain queries exceeded the out-domain. Except in the English test data the number of out-domain queries far exceeded the number of in-domain queries and in the Malayalam test data there were no out-domain queries.

FAQ	Language	No. of SMS Queries (In-domain, Out-of-domain)		
		Monolingual Task	Cross-lingual task	Multilingual Task
7251	English	(701,370)	(291,181)	(290,170)
		(728,2677)	(37,3368)	(724,2681)
1994	Hindi	(181,49)		(183,47)
		(200,124)		(200,124)
681	Malayalam	(120,20)		(60,20)
		(50,0)		(50,0)

Fig. 4. Details of the data set released for the task

4 Submissions Overview

A total of 13 teams participated and submitted a total of 72 runs across the 9 sub tasks, see Figure 5.

Most of the teams that participated in the monolingual tasks had the following two steps, correction step on the SMSes to correct spelling errors, abbreviations and non-standard writings and the retrieval step to match the SMS to the correct FAQ.

For the correction step the DCU team created a manual training data by manually correcting the SMS queries and then learning correction rules from them. The DAIICT team used hand crafted rules to correct the SMS queries. The BUAP team modeled the correction step as a machine translation step and used the IBM-4 model for constructing a statistical correction dictionary. The IIITH team used manual and semi-automated approaches to create correction dictionaries. The RVCE team effected the correction by removing all vowels in all the words. The DCE team converted all words to their metaphone equivalents. The Iowa team used the Google search spelling suggestions to correct the SMS queries. They also used a list of abbreviations. Some of the teams did not do much cleaning of the Hindi and Malayalam queries as they found the noise levels in these to be much lower.

For retrieval DCU used several retrieval methods separately and in combination using a weighted sum. The DAIICT, DCE, Iowa teams also used retrieval scores for matching. The BUAP team matched the SMS query to FAQ by minimizing the translation score using IBM-1 model. The DTU team skipped the correction step and de-

Team (# of runs submitted)	English Mono	Hindi Mono	Mal Mono	Cross	English Multi	Hindi Multi	Mal Multi
Iowa (19)	✔	✔	✔	✔	✔	✔	✔
BUAP (11)	✔	✔	✔	✔	✔	✔	✔
DCE (5)	✔	✔		✔	✔	✔	
IIITH (12)	✔	✔	✔	✔			
DAIICT (6)	✔	✔	✔				
Jadhavpur-IPN	✔	✔		✔			
DTU (4)	✔	✔					
DCU (3)	✔						
MSRIT (3)	✔						
TCS (2)	✔						
SASTRA (1)	✔						
RVCE (1)	✔						
IIITD (1)	✔						

Fig. 5. Participating teams and the runs submitted : The 13 teams submitted a total of 72 runs across 9 sub tasks

termined a match score by taking into consideration intra-token matches. They also considered word position information. The RVCE team computed match scores by considering a number of features to compute the word and sentence similarity between the SMS query and FAQ question. The IIITH team used a language modeling approach to score the matches. The TCS team used Latent Semantic Indexing to do the matching.

Most teams empirically determined a threshold to remove the the out-of-domain queries. The DCU team went a step futher and explicitly removed out-of-domain queries by using a combination of retrieval score, classification score and number of matching terms between the query and FAQ question.

Fewer teams participated in the cross-lingual and multi-lingual tasks. The IIITH team created cross-langauge dictionaries using manual and semi-automated techniques for the cross-lingual task. The Iowa team translated all the FAQs into English using Google and Microsoft APIs and crowdsourcing before doing the matches as if they were doing monolingual tasks. The BUAP team used IBM-4 model to create statistical bi-lingual dictionaries and then used IBM-1 model to do the matching.

5 Results

5.1 Evaluation Measure

As described in section 3 the queries contain both in-domain (queries that contain matches in the FAQ corpus) as well as out-of-domain queries (queries that don't contain matches in the FAQ corpus). A practical FAQ retrieval system should be able to identify queries that it cannot answer. The task was, therefore, evaluated using two accuracy measures : the in-domain accuracy, i.e. the number of queries correctly answered and the out-of-domain accuracy, i.e. the number of out-of-domain queries correctly identified.

If N is the total number of queries released in the test set, P be the number of in-domain queries and Q be the number of out-of-domain queries.

$$N = P + Q \tag{1}$$

If the number of in-domain queries identified by a submission is p then the in-domain accuracy A_P can be defined as :

$$A_P = \frac{p}{P} \tag{2}$$

Similarly, if the number of out-of-domain queries dentified by a submission is r, then the out-of-domain accuracy A_R can be defined as :

$$A_R = \frac{r}{R} \tag{3}$$

The total score, S , of a run is computed as :

$$S = \frac{p + r}{N} \tag{4}$$

In addition, we also report the MRR (Mean reciprocal rank) on the in-domain queries for the best runs from each team. Mean reciprocal rank is used to evaluate a system by producing a list of possible responses to a query, ordered by probability of correctness. The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct match. The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of queries Q.

$$MRR = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{rank_i} \tag{5}$$

The best match in-domain accuracy can be considered as a special case of MRR where the size of the ranked list is 1.

5.2 Mono-Lingual Retrieval Task

The sections below present the results for the mono-lingual retrieval task for each of the three languages - English, Hindi and Malayalam.

English: Figure 6 shows the runs the results of the runs submitted for the English mono-lingual retrieval task. The number of in-domain and out-of-domain queries correctly matched have been shown on the bar plots.

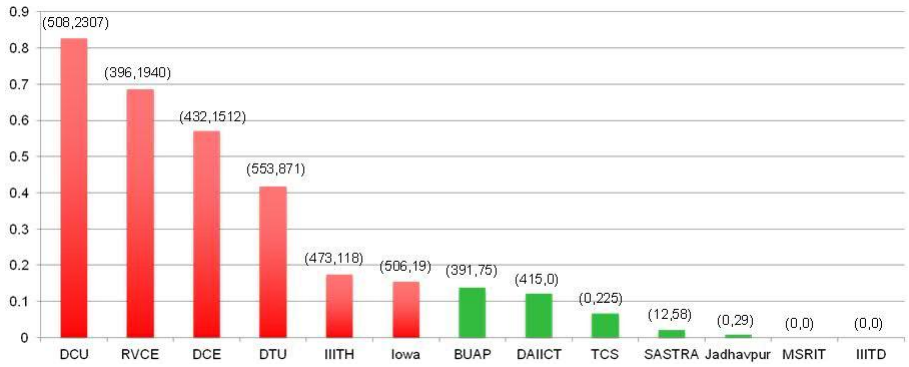


Fig. 6. Results of the English Mono-lingual retrieval task.Values in brackets indicate number of in-domain, out-of-domain queries that were correctly identified.

Table 1 shows the accuracy and MRR scores for the teams. The median score was 0.14 while the higest score was 0.83 for a run submitted by team DCU.

Hindi: Figure 7 shows the runs the results of the runs submitted for the Hindi mono-lingual retrieval task. The number of in-domain and out-of-domain queries correctly matched have been shown on the bar plots.

Table 2 shows the accuracy and MRR scores for the teams. The median score was 0.53 while the highest score was 0.62 for a run submitted by team DTU.

Table 1. Results of English mono-lingual retrieval task

Team Name	In-domain Accuracy	Out-of-domain Accuracy	Mean Reciprocal Rank	Score
Iowa	0.6951	0.0071	0.7353	0.1541
BUAP	0.5371	0.0280	0.5919	0.1369
DCE	0.5934	0.5648	0.7844	0.5709
IIITH	0.6497	0.0441	0.7031	0.1735
DAIICT	0.5701	0.0000	0.6321	0.1219
Jadhavpur-IPN	0.0398	0.0000	0.0522	0.0085
DTU	0.7596	0.3254	0.8251	0.4182
DCU	0.6978	0.8618	0.8908	**0.8267**
MSRIT	0.0000	0.0000	0.0000	0.0000
TCS	0.0000	0.0840	0.0000	0.0661
SASTRA	0.0165	0.0217	0.0251	0.0205
RVCE	0.5439	0.7247	0.8630	0.6860
IIITD	0.0000	0.0000	0.0000	0.0000

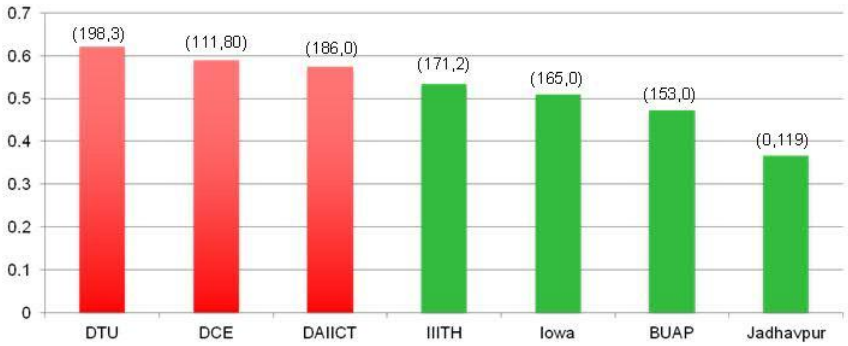


Fig. 7. Results of the Hindi Mono-lingual retrieval task. Values in brackets indicate number of in-domain, out-of-domain queries that were correctly identified.

Table 2. Results of Hindi mono-lingual retrieval task

Team Name	In-domain Accuracy	Out-of-domain Accuracy	Mean Reciprocal Rank	Score
Iowa	0.8250	0.0000	0.8604	0.5092
BUAP	0.7650	0.0000	0.8071	0.4722
DCE	0.5550	0.6452	0.7222	0.5895
IIITH	0.8550	0.0161	0.8858	0.5339
DAIICT	0.9300	0.0000	0.9450	0.5740
Jadhavpur-IPN	0.0000	0.9596	0.0000	0.3673
DTU	0.9900	0.0242	0.9900	**0.6203**

Malayalam: Figure 8 shows the runs the results of the runs submitted for the Malayalam mono-lingual retrieval task. The number of in-domain and out-of-domain queries correctly matched have been shown on the bar plots.

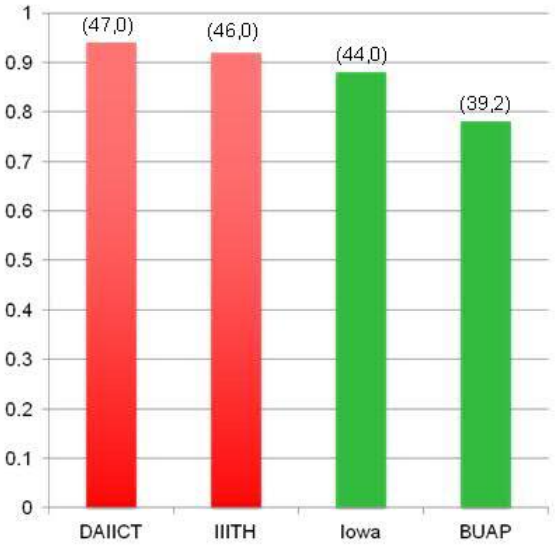


Fig. 8. Results of the Malayalam Mono-lingual retrieval task.Values in brackets indicate number of in-domain, out-of-domain queries that were correctly identified.

Table 3 shows the accuracy and MRR scores for the teams. The median score was 0.90 while the highest score was 0.94 for a run submitted by team DAIICT.

Table 3. Results of Malayalam mono-lingual retrieval task

Team Name	In-domain Accuracy	Out-of-domain Accuracy	Mean Reciprocal Rank	Score
Iowa	0.8800	N.A.	0.8935	0.8800
BUAP	0.7800	N.A.	0.8304	0.7800
IIITH	0.9200	N.A.	0.9402	0.9200
DAIICT	0.9400	N.A.	0.9562	**0.9400**

5.3 Cross-Lingual Retrieval Task

Figure 9 shows the runs the results of the runs submitted for the cross lingual retrieval. The number of in-domain and out-of-domain queries correctly matched have been shown on the bar plots.

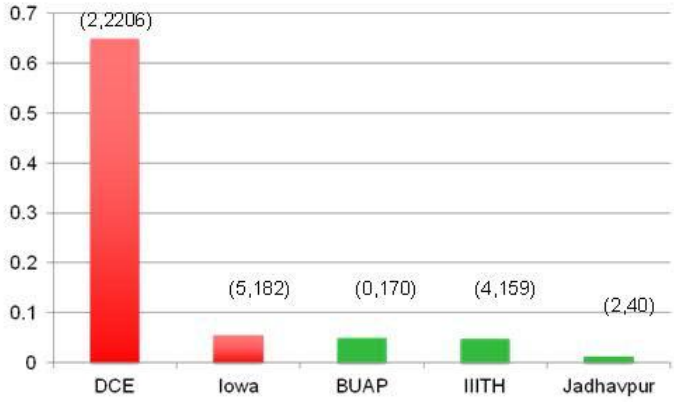


Fig. 9. Results of the cross lingual retrieval task. Values in brackets indicate number of in-domain, out-of-domain queries that were correctly identified.

Table 4 shows the accuracy and MRR scores for the teams. The median score was 0.0499 while the highest score was 0.65 for a run submitted by team DCE. Team DCE did substantially better than the rest of the teams participating in the task.

Table 4. Results of cross lingual retrieval task

Team Name	In-domain Accuracy	Out-of-domain Accuracy	Mean Reciprocal Rank	Score
DCE	0.0540	0.6549	0.1296	**0.6484**
Iowa	0.1351	0.0540	0.1604	0.0549
BUAP	0.0000	0.0505	0.0000	0.0499
IITH	0.1081	0.0472	0.1279	0.0478
Jadhavpur-IPN	0.0540	0.0119	0.0540	0.0123

5.4 Multi-lingual Retrieval Task

English: Figure 10 shows the runs the results of the runs submitted for the multi-lingual retrieval using English queries. The number of in-domain and out-of-domain queries correctly matched have been shown on the bar plots.

Table 5 shows the accuracy and MRR scores for the teams. The median score was 0.15 while the highest score was 0.52 for a run submitted by team DCE.

Hindi: Figure 11 shows the runs the results of the runs submitted for the multi-lingual retrieval using Hindi queries. The number of in-domain and out-of-domain queries correctly matched have been shown on the bar plots.

Table 6 shows the accuracy and MRR scores for the teams. The median accuracy (in-domain) was 0.51 while the highest in-domain accuracy was 0.57 for a run submitted by team DCE.

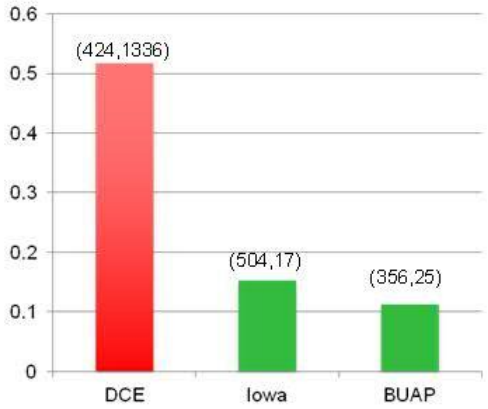


Fig. 10. Results of the multi-lingual retrieval task using English SMS queries. Values in brackets indicate number of in-domain, out-of-domain queries that were correctly identified.

Table 5. Results of multi-lingual retrieval task using English SMS queries

Team Name	In-domain Accuracy	Out-of-domain Accuracy	Mean Reciprocal Rank	Score
DCE	0.5856	0.4983	0.7546	**0.5169**
Iowa	0.6961	0.0063	0.7101	0.1530
BUAP	0.4917	0.0093	0.5270	0.1119

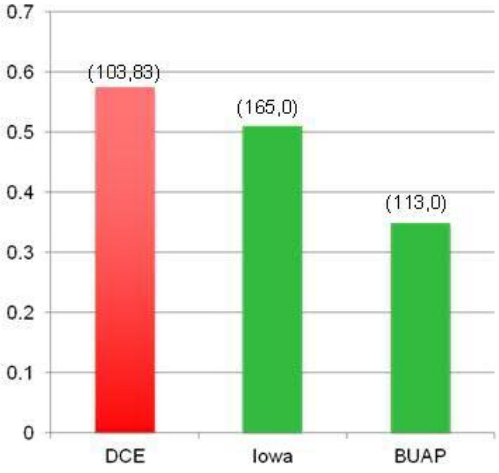


Fig. 11. Results of the multi-lingual retrieval task using Hindi SMS queries.Values in brackets indicate number of in-domain, out-of-domain queries that were correctly identified.

Table 6. Results of multi-lingual retrieval task using Hindi SMS queries

Team Name	In-domain Accuracy	Out-of-domain Accuracy	Mean Reciprocal Rank	Score
DCE	0.5150	0.6694	0.6196	**0.5741**
Iowa	0.8250	0.0000	0.8391	0.5093
BUAP	0.5650	0.000	0.6163	0.3487

Malayalam: Figure 12 shows the runs the results of the runs submitted for the multi-lingual retrieval using Malayalam queries. The number of in-domain and out-of-domain queries correctly matched have been shown on the bar plots.

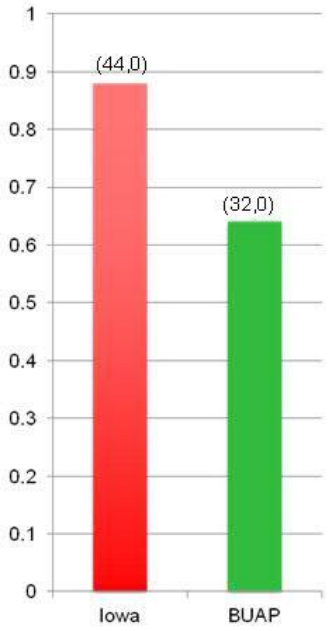


Fig. 12. Results of the multi-lingual retrieval task using Malayalam SMS queries. Values in brackets indicate number of in-domain, out-of-domain queries that were correctly identified.

Table 7 shows the accuracy and MRR scores for the teams. There were only two submissions for this subtask and Team Iowa performed better than Team BUAP with a score of 0.88.

6 Summary

The first edition of the SMS based FAQ retrieval task has been a success. The participation has been encouraging and we plan to continue the task in subsequent FIRE conferences.

Table 7. Results of multi-lingual retrieval task using Malayalam SMS queries

Team Name	In-domain Accuracy	Out-of-domain Accuracy	Mean Reciprocal Rank	Score
Iowa	0.8800	N.A.	0.8895	**0.8800**
BUAP	0.6400	N.A.	0.7037	0.6400

We received a total of 72 run submissions from 13 different teams across the world. Most teams chose to participate in the mono-lingual retrieval tasks (the English mono-lingual task had participation from all teams) but there was significant interest in the cross lingual and multi-lingual tasks as well.

Multi-lingual retrieval is still a challenging problem and as seen from the results, there is huge scope for improvement in the techniques used for this task. While, the highest score for the English mono-lingual task was 0.83, it was only 0.51 in case of the English multi-lingual retrieval task.

The performance scores in the Malayalam tasks were surprisingly high. The lesser amount of noise and the smaller size of the data enabled teams to score higher in these tasks.

References

1. Ahmad, R., Sarlan, A., Maulod, K.A.A., Mazlan, E.M., Kasbon, R.: Sms-based final exam retrieval system on mobile phones. In: International Symposium on Information Technology, ITSim (2010)
2. Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S., Basu, A.: Investigation and modeling of the structure of texting language. IJDAR 10(3-4), 157–174 (2007)
3. Contractor, D., Faruquie, T.A., Subramaniam, L.V.: Unsupervised cleansing of noisy text. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters, COLING 2010, pp. 189–196 (2010)
4. Gadde, P., Goutam, R., Shah, R., Bayyarapu, H.S., Subramaniam, L.V.: Experiments with artificially generated noise for cleansing noisy text. In: Proceedings of the 2011 Joint Workshop on Multilingual OCR and Analytics for Noisy Unstructured Text Data, MOCR AND 2011, pp. 4:1–4:8 (2011)
5. Kolodner, J.L.: An introduction to case-based reasoning. Artificial Intelligence Review 6(1), 3–34 (1992)
6. Kothari, G., Negi, S., Faruquie, T.A., Chakaravarthy, V.T., Subramaniam, L.V.: Sms based interface for faq retrieval. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, ACL 2009, vol. 2, pp. 852–860 (2009)

SMS Based FAQ Retrieval Using Latent Semantic Indexing

Arijit De

TCS Innovation Labs-Mumbai, Tata Consultancy Services,
Thane (W), Mumbai 400601
arijit6.d@tcs.com

Abstract. In this approach note we describe two Latent Semantic Indexing approaches to SMS based FAQ retrieval. The first approach Naïve LSI is based on simply applying Latent Semantic Indexing to compute the vector distance between the SMS and Questions within the FAQ set. The second approach uses the Levenshtein distance translate the SMS language in one word token strings present within the FAQ set. Then the LSI algorithm is used to construct the LSI matrix and compute the vector distance with various questions within the FAQ database. For in domain queries the second approach outperforms the first approach with a accuracy of 26%. The first approach however does better that the second approach for cross domain with an accuracy of 8%.

1 Introduction

To meet the information needs of mobile users, SMS-based Question Answering (QA) and Information Retrieval (IR) systems have gained popularity. SMS-based question answering systems accept a user's information need from a mobile phone using an SMS. This SMS is then interpreted and the information needs of the users are evaluated and the question asked through SMS is answered. A rudimentary way to build a automated SMS based Question Answering (QA) platform is to have a standard set of questions and answers. When an SMS comes in the SMS is matched to the nearest question and then the corresponding answer is returned back to the user. In this approach note mention two algorithms to match SMS text to a specific question for answer retrieval. Both our systems are based on Latent Semantic Indexing (LSI). The first is a naïve application of LSI and the second is a more advances approach. This approach note is organizes as follows. In the next section 2, we formulate the question / SMS matching. In section 3 we discuss our naïve LSI approached. In section 4 we describe our more refined LSI approach. In section 5 we describe our experiments and results and then in section 6 we summarize our algorithms in a conclusion.

2 Problem Definition: Defining the Task

This task is about matching a user SMS s to a set of Question Answer set already available. Let say we have a set of QA = $\{(q_i, a_i), 1 \leq i \leq N\}$ where n is the number of

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 100–103, 2013.

question answer pairs (q_i, a_i) available and q_i is the question and a_i is the answer in the pair. We are also provided with a set of training SMS $T_{SMS} = \{s_i, 1 \leq i \leq M\}$ and set of possible matches $QA_S = \{qas_i, 1 \leq i \leq m\}$. For each SMS s_i in T_{SMS} there is a set of possible matches to questions that match $qas_i = \{(q_k, sc_k), 1 \leq k \leq P\}$. Each contains top P matching queries. Here sc_k is a score representing the degree or extent to which the SMS s_i matches the query q_k. The task is to learn a pattern from this training set, which can be used to interpret and match incoming user SMS's to possible question answers.

3 First Approach: Using Naïve LSI

Our first approach is called the Naïve LSI and is based on Latent Semantic Indexing (LSI) technique proposed by Deerwester [1].

Input
We start with the set of questions QA, a set of training T_{SMS} and the set of top P questions that match with each SMS s_i in T_{SMS} , QA_S as defined earlier.

Step 1: Tokenization
We begin by tokenizing all SMS s_i in T_{SMS} and all questions in QA to form a set of one-gram tokens (words) $T = \{ t_j, 1 \leq j \leq R\}$, where R is the total number of one-gram tokens or words.

Step 2: Constructing LSI Matrix
The LSI matrix is similar to Term-Frequency matrix M mentioned in Salton [2]. Here each token is similar to a term and represented by a row and each question is similar to a document in [2] and represented by a column. Each element in the $M(t_j, q_k)$ is either the number of times the token t_j is in an SMS s_i for which question q_k is returned as a top p match or in case t_j is a token derived from a question itself, $M(t_j, q_k)$ the number of times the token occurs in the question. The matrix M is therefore R X N.

Step 3: Latent Semantic Indexing
We apply Latent Semantic Indexing on M. The number of factors for Singular Value Decomposition is set at F. After training and decomposition over a pre specified number of epochs we obtain, decomposed vectors U, D, and V. The vectors, U (R X F) is a vector representing tokens, D is a diagonal vector of singular values (F X 1) and V is a vector (N X F) representing questions after final decomposition.

Step 4: Handling SMS to form Query Vector
When an SMS query comes in it is similarly tokenized into one-gram tokens. A query vector of dimensions (R X F) is formed. For each token in the SMS query that is in T we sum up the corresponding columns (representing the common token) of the token vector U to form a query vector of the dimension (1 X F).

Step 5: Scoring Questions
Each of the N rows in the V vector represents a question. The dot-product of the row, the singular values and the query vector give the score for the specific question. Questions are sorted in descending order and P-top scores are returned.

4 Second Approach: Using Refined LSI

The second LSI based approach is similar to the LSI approach discussed in Section 3. The main difference is that we construct the LSI Matrix by only tokenizing the questions in the set QA into one gram tokens. So in this approach each unique one-gram token is represented in the LSI matrix M in a row and each question is represented by a column. For each SMS s_i and each corresponding top-P matching questions Q we obtain token set T_i and TQ_i respectively. For each token $t_i \in T_i$ we compute the edit distance (Levenshtein [3]) to each token $tq_j \in TQ_j$ for all i and j The tq_j that has the least distance is mapped to t_i. Thus for every token from every SMS in the training set T_{SMS} we can obtain a set of possible matching words of the question set QA with corresponding edit distance as a measure of the likelihood of a good match. We end with a set of SMS tokens T-SMS = { $t\text{-}sms_i$ } and a set of corresponding question tokens T-QA = {tq_j} for all i and j.

When a testing SMS comes in we tokenize the SMS into one-gram tokens. For each SMS tokens are translated into the closest token in T-SMS and based on the match the SMS token is translated into a question token in T-QA. With incoming SMS tokens translated into question tokens, the LSI algorithm in section 3 is used to determine the P-most suitable questions with corresponding scores.

5 Experiments, Results and Conclusions

We experiment with the FIRE 2011 SMS-Based English Frequently Asked Questions (FAQ) Retrieval dataset. The SMS queries were mono-lingual dataset, cross-lingual dataset and multi-lingual in nature. The FAQ were in English, Hindi and Malayalam languages. For simplicity our experiments were on mono-lingual English SMS queries which were matched to Questions and Answers in English. Our results are tabulated below. Number of In-Domain Queries were 708 and Number of Out of Domain Queries was 2701. Table 1 shows the results of experiments. The first approach however does better that the second approach for cross domain with an accuracy of 8%.

Table 1. Results

	Naïve LSI	Refined LSI
In Domain correct	0/704	142/704
Out of Domain correct	225/2701	19/2701

References

[1] Deerwester, S., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the Society for Information Science 41(6) (1990)
[2] Salton, G., Wong, A., Yang, C.S.: A Vector Space Model for Automatic Indexing. Communications of the ACM 18(11), 613–620 (1975)
[3] Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10, 707–710 (1966)

Data-Driven Methods for SMS-Based FAQ Retrieval

Sanmitra Bhattacharya, Hung Tran, and Padmini Srinivasan

Department of Computer Science,
The University of Iowa,
14 MacLean Hall, Iowa City, IA 52246
{sanmitra-bhattacharya,hung-trv,padmini-srinivasan}@uiowa.edu

Abstract. SMS text messaging is one of the most popular data applications on mobile phones these days. Other than personal communication, text messaging can also be used for various purposes like bill payment, banking, inquiry, etc. However these messages are extremely noisy and contain misspellings, abbreviations, transliterations, etc. Keeping this in mind, FIRE 2011 introduced a new retrieval task called SMS-based FAQ retrieval in English, Hindi and Malayalam. Within-language and cross-language tasks were designed for this retrieval problem. As solutions we propose various data-driven retrieval techniques that includes noise reduction in the SMS queries and the FAQ corpora. Overall, we find that our methods work well for the retrieval experiments in the different languages. For English, the use of Google Spelling Suggestions and term expansion strategies improve retrieval scores. For Hindi and Malayalam retrieval experiments, we find that translation of queries and corpus to English increases retrieval accuracy.

1 Introduction

The mobile phone market has seen exponential growth since its inception in the mid-1970's. This spark in growth can be attributed primarily to advances in cellular technology making mobile phones inexpensive as well as easy to use. Over 75% of the world population has access to mobile phones with China and India being the top contributors in its recent growth. The total number of mobile subscribers in India in May 2011 was over 840 million, up 233 million from May 2010. The most popular data applications on mobile phones is SMS text messaging. The number of SMS messages sent in 2010 was 6.9 trillion and for 2011 this number should reach over 8 trillion[1]. The wave of modernization and globalization in India has also resulted in an increased use of text messaging in various domains. SMS messaging is now used not only for personal communication but also for inquiry, advertising, marketing, polling, bill payment, banking, etc.

In line with these developments, FIRE 2011, in its third year, introduced a new retrieval problem called SMS-based FAQ retrieval. This problem addresses

[1] http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 104–118, 2013.

finding question entries from a corpora of Frequently Asked Questions (FAQs) that best match a given SMS query. In a traditional question-answering (QA) retrieval task such as the TREC QA track (1999-2007), retrieval systems were required to retrieve answers (i.e. entries containing answers) rather than documents in response to questions. For example, the main task of the TREC 2007 QA track required participants to define a target by answering a series of factoid and list questions about the target. Participants also had to return any information not covered by the question series. For example, for a target such as "House of Chanel" participants are required to find answers to questions like "Who founded the House of Chanel?" (factoid) or "What museums have displayed Chanel clothing?" (list). The source of questions for the QA track were participants, assessors, FAQFinder system logs, MSNSearch logs, query logs of Microsoft and AOL, etc. The target corpus consisted of newswire documents, AQUAINT disks and blog data.

In contrast, the FIRE 2011 FAQ corpora consists of well-formed questions and answers in various domains like agriculture, career, general knowledge (GK), Indian Railways, telecommunication, etc. Here the queries are SMS messages inquiring about different topics of the FAQ corpus. In essence, participants were required to find FAQ entries that answer or match the SMS queries. One of the major challenges in this task is the structure of the SMS queries, or the lack thereof. SMS messages are generally noisy with misspellings, non-standard abbreviations, omission of words, etc. From a linguistic point of view SMS messages generally do not conform to proper grammatical usage and hence comprehension of such messages depend largely on context and intellectual effort. For example, the SMS message "wht is career counclng" highlights a few of these problems. Words like "wht" (for "what") or "counclng" (for "counseling") are non-standard abbreviations that require the expertise of experienced texters to decipher their meaning. Matching such ill-formed SMS messages to reasonably well-formed FAQs entries is the challenge.

2 Description of the Task

As already mentioned, the overall goal of the task is to match SMS queries to FAQs. Participants are permitted to use the information in both the questions and the answers in FAQs for matching the SMS queries.

A sample (abbreviated) XML-formatted FAQ topic is given below:

```
<FAQ>
<FAQID>ENG_CAREER_1</FAQID>
<DOMAIN>CAREER</DOMAIN>
<QUESTION>What is career counseling?</QUESTION>
<ANSWER> Career counseling is a process ...
</ANSWER>
</FAQ>
```

In this example, ENG_CAREER_1 refers to a unique identifier of a particular FAQ in English, DOMAIN refers to the broader domain of the question and QUESTION and ANSWER refers to the core FAQ component.

A sample XML-formatted SMS query is shown below:

```
<SMS>
<SMS_QUERY_ID>ENG_405</SMS_QUERY_ID>
<SMS_TEXT>wht is career counclng</SMS_TEXT>
<MATCHES>
<ENGLISH>ENG_CAREER_1</ENGLISH>
<MALAYALAM>NONE</MALAYALAM>
<HINDI>NONE</HINDI>
</MATCHES>
</SMS>
```

In the above example, SMS_QUERY_ID refers to a unique identifier of the SMS message and SMS_TEXT refers to the actual content of the text message. The MATCHES field contains three tags which refer to the three languages pertaining to this task (namely, English, Malayalam and Hindi). The ENGLISH tag contains the identifier of the FAQ which is relevant to this SMS query. There are no relevant Malayalam or Hindi FAQ entries.

The dataset of the SMS queries is split into two parts, one for training and the other for testing. The sample SMS query in the above example belongs to the training corpus. In the test set, the values in the MATCHES fields are masked. The same target FAQ corpus is used for both training and testing.

The overall research problem of SMS query-based FAQ retrieval consists of the following three sub-tasks:

2.1 Mono-Lingual FAQ Retrieval

In this sub-task, participants are required to find the best matching FAQ entries for a given SMS query where the FAQ corpus and SMS queries are expressed in the same language. For instance, participants are required to find FAQs in English for SMS queries written in English (similarly for Hindi and Malayalam).

2.2 Cross-Lingual FAQ Retrieval

For this sub-task, participants are required to find matching FAQs in a language different from the SMS query language. For instance, participants are required to find Hindi FAQs for SMS queries written in English. Other permutations using other languages were not considered in this task by track organizers.

2.3 Multi-lingual FAQ Retrieval

In this sub-task, participants are required to find matching FAQs for SMS queries where both FAQs and messages can be in any language (English, Malayalam

or Hindi). For instance, SMS queries written in English (or Hindi/Malayalam) should retrieve relevant English, Hindi and Malayalam FAQs.

For each task, a ranked list (ordered by degree of match or relevance) of 5 retrieved FAQs per SMS query has to be returned.

3 Data and Preprocessing

In this section we outline our strategies for all three sub-tasks of the SMS-based FAQ retrieval problem. In all our retrieval methods we use only the information content in the `QUESTION` tag of the FAQs. Although answers to the FAQ questions are present in the `ANSWER` tags, these were not used. The answers, which are quite elaborate at times, may contain valuable information that facilitates retrieval. But the nature of the SMS queries, in terms of length or language, tally better with the FAQ `QUESTION` entries rather than the `ANSWER` entries. Furthermore, the matches for SMS queries in the gold standard training set are themselves based on the content of the `QUESTION` fields, i.e. "text of the questions" and not the "answers" of the FAQs. We leave the exploration of usage of `ANSWER` tag contents for this retrieval problem to future work.

3.1 Dataset

The FAQ corpus consists of 7251 FAQs in English, 1994 in Hindi and 681 in Malayalam. The number of SMS queries per sub-task for both training and testing is shown in Table 1. For the cross-lingual task (Section 2.2) the number of SMS queries is shown only for English because only these queries are run on the Hindi FAQ dataset. The rest of the table corresponds to the task descriptions in Sections 2.1 (mono-lingual) and 2.3 (multi-lingual) respectively. In the training set, the cross-lingual task had the least number (61.6%) of relevance judgments while the Malayalam monolingual task had the highest number (85.7%) of relevance judgments.

Table 1. Number of SMS queries for each sub-task in the training and test sets. Percentage of SMS queries having relevance judgment is indicated in parenthesis

Sub-task	**Training**			**Testing**		
	English	Hindi	Malayalam	English	Hindi	Malayalam
Mono-lingual	1071(65.4%)	230(78.6%)	140(85.7%)	3405	324	50
Cross-lingual	472(61.6%)	-	-	3405	-	-
Multi-lingual	460(63%)	230(79.5%)	80(75%)	3405	324	50

3.2 Indexing

We index the FAQ datasets for the various languages using the INDRI information retrieval system[6]. As explained before, we use only the contents of the

QUESTION field for our retrieval experiments. Hence we index only the QUESTION fields of the FAQs. For Indian languages, Hindi and Malayalam, we create two types of indexes. For *Type 1*, we index the FAQ datasets in their native UTF-8 encoding. For *Type 2*, we index the English-translated versions of the FAQ datasets. Details of the translation mechanism are given next.

3.3 Translation Mechanism

Hindi Translation. We translate the FAQs in Hindi to English. We first use the Google Translate API v2[2] for translating Hindi to English. For some longer FAQ questions the Google Translate API did not return any results. For such cases, we use the Microsoft Bing Translator[3]. Manual comparison of outputs of Google Translate and Microsoft Bing Translator shows that Google Translate performs noticeably better. While both translation APIs take into account the context in which a particular word appears in a sentence, Google Translate seems to do a better job at interpreting the context. Hence Google Translate's renditions seem to be more natural than literal.

A sample Hindi-English translation using both the APIs is shown below. We can see that Google Translate's output, though not perfect, is closer to the natural English translation of the Hindi FAQ. The Microsoft Bing Translator output is not only worse in terms of sentence structure but it also does transliteration instead on translation for the last but one word.

Hindi FAQ: धान भण्डारण करते समय क्या-क्या सावधानियां बरतनी हैं?

Google Translate Output: When grain storage - what are the precautions?

Microsoft Bing Translator Output: What-if, when Paddy cold storage savdhaniyan be?

Malayalam Translation. For Malayalam-English translation we did not find any freely available API. Google Translate and Microsoft Bing Translator do not support Malayalam-English translation. Hence for this task we used crowdsourcing techniques [4]. Crowdsourcing has seen a lot of traction in recent years for tasks that are amenable to human intelligence compared to computational processing. Hence tasks like translation, relevance judgment, data labeling, etc. are well suited for crowdsourcing. For the Malayalam-English translation we used the crowdsourcing platform oDesk[4]. Compared to other popular crowdsourcing platforms like Amazon's Mechanical Turk[5], oDesk has been found to provide better quality of service[1]. It also allows recruiters to screen employees based on skill-level required for a particular task. For our translation task, we selected

[2] http://code.google.com/apis/language/translate/overview.html
[3] http://www.microsofttranslator.com/dev/
[4] https://www.odesk.com/
[5] https://www.mturk.com/

participants whose native language is Malayalam and who have excellent comprehension skills in English. Section 4.3 discusses our use of crowdsourcing in more detail.

4 Mono-Lingual FAQ Retrieval

4.1 English

As described in Section 1, the SMS queries, particularly the English ones are notoriously noisy. In order to alleviate this problem we follow a multi-stage approach (see Fig. 1).

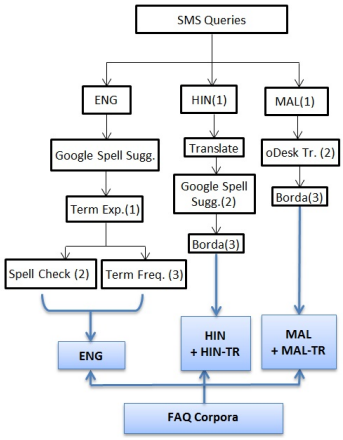


Fig. 1. Flowchart of methods adopted for mono-lingual SMS-based FAQ retrieval (ENG : English, HIN : Hindi, MAL:Malayalam, HIN-TR : Hindi translated to English and MAL-TR : Malayalam translated to English). Run numbers are shown in parenthesis.

Google Spelling Suggestions. In the first step, we use Google Web Search's spelling suggestion feature for capturing spelling and grammatical mistakes in the SMS queries. Google Web Search, equipped with autocomplete feature also provides spell-checking and auto-correction functions. These feature can automatically fix typos and suggest the most suitable keywords that should appear with the terms of a partially typed query. These functions rely primarily on the information learned from user-provided queries (user-specific, if signed into ones Google Account). It also depends on the query logs from Google searches which can track co-occurring terms. For example, if a user types in *infornation retrieva* (note the typos) in the search box of Google Web Search, Google suggests *information retrieval* as the correct form of query. It also find results based on this corrected form, with an option to retrieve results for the original query. Since

Google does not provide a standard API for accessing this feature, we used a web-page scraping technique for accomplishing this step.

For each SMS query of our test corpus for English mono-lingual retrieval we executed the technique described above. An example of using this technique for a given SMS query is shown below. It may be observed that the transformation is correct.

Original query: wht is career counclng
Transformed query: what is career counselling

Term Expansion. Google spelling suggestions depend largely on the context in which the misspelled terms appear. Since it relies on user's querying habits and query logs for spelling suggestions, novel associations of words that seldom or never appeared before in queries will be missed. Misspelled terms in such queries will not be corrected using Google's spell-check. For example, for the SMS query "how 2 update resi add", Google's spell-check does not perform any transformations. For this reason, we employ another method for spell check and term expansion. In this case, we extract all 1–4 character terms present in the SMS queries of the training set. These terms are then expanded to their original unabridged form to construct a lookup table. We select only 1–4 character long terms because most non-standard abbreviations fit within this window of term length. This lookup table is then augmented with a list of commonly used shorthand notations and their expansions (in the context of text messaging). This resulted in a list of 766 abbreviated terms with their corresponding unabridged versions. This lookup table is later used for expansion of non-standard abbreviations in the test queries.

It is important to note here that we did not use popular chat acronyms and text message shorthands in our term expansion list. We found that the nature of abbreviations in the training set were clearly molded to the Indian context. Hence commonly used Western abbreviations (e.g. '411' or information) used for text messaging are generally absent in our dataset.

Traditional Spell-Check. All the SMS queries processed in the previous two stages are now checked for further spelling errors. We use Aspell[6], a free and open-source spell checker, for this purpose. Each out-of-vocabulary word identified using this program is replaced with the first suggested correction. This process helps us in correcting misspelled words that do not fit in the 1–4 character window and terms that are unseen in the training dataset. Words like 'rsrch' and 'soldeirs' are corrected to 'research' and 'soldiers' respectively using this method. However, this method is prone to errors for commonly used abbreviations ('cdma' transformed to 'cd ma') and proper nouns (e.g. 'Ghaziabad' transformed to 'Gasbag').

[6] `http://aspell.net/`

Retrieval Strategies. Based on the techniques described above we submitted 3 runs which are outlined below.

Run 1. For this run, we execute the *Google Spelling Suggestions* and *Term Expansion* strategies on the test set of SMS queries. We then use INDRI's belief operator (#combine) for retrieving FAQ entries that match the English SMS queries. Unlike Boolean operators (e.g. AND, OR, etc.) which returns only binary values, the #combine operator weighs each term equally and prioritizes documents (FAQ questions entries in this case) containing more query terms. The queries are executed on the index of the English FAQ corpus (Section 3.2). The matches are limited to the content of the `QUESTION` tags in the FAQs.

Run 2. For this run, following the *Google Spelling Suggestions* and *Term Expansion* strategies (as in Run 1), we use the *Traditional Spell-check* approach.

Run 3. For this run, we try to emphasize the importance of the rarer terms in the SMS queries while trying to diminish the influence of the more frequently occurring terms. First, we calculate the term frequency of each word in the SMS query training set. Then, for each SMS query of the test set we keep at most six[7] least frequent terms while discarding the rest of the terms. In this process we aim to filter out the terms which do not contribute any significant information towards the retrieval process. The rest of the process is identical to the *Run 1* strategy.

4.2 Hindi

For the Hindi mono-lingual FAQ retrieval we find that the noise in SMS queries is substantially less compared its English counterparts. Not only are there fewer misspellings, but also use of non-standard abbreviations is very limited. These observations were based on manual analysis of a small sample of SMS queries from the training set. Further, the *Google Spelling Suggestions* technique used in Section 4.1 is unavailable for Hindi. The three strategies (Fig. 1) used for run submission in this task are described below.

Retrieval Strategies

Run 1. For this run, we execute the SMS queries in their native UTF-8 format on the index of UTF-8 encoded Hindi FAQs (Section 3.2). Similar to retrieval strategies followed in Section 4.1 we use INDRI's belief operator #combine for finding relevant FAQ question entries.

[7] This number was selected based on manual evaluation of retrieval accuracy on the training set.

Run 2. For this run, we first translate the Hindi FAQ corpus (only questions) and the Hindi SMS queries of the test set using the mechanism described under *Hindi Translation* in Section 3.3. The translated SMS queries are then processed using the *Google Spelling Suggestions* mechanism described in Section 4.1. This step helps in fixing minor problems that might have occurred during the translation process. The rest of the run execution is identical to the process explained in *Run 1.*

Run 3. For this run, we introduce a method for merging the results of two runs, *Run 1* and *Run 2.* Since these two runs are performed with completely disjoint strategies, a merging of results from these two runs might give us some interesting results. With this intuition, we explore Straight Borda Count[3], a consensus-based method for merging result sets which are ranked by some score. In Straight Borda Count, each retrieved FAQ entry gets a score, which is a sum of its ranks in all returned result sets (i.e. Runs 1 and 2). The resulting set is ranked using this score.

4.3 Malayalam

Similar to Hindi, we do not use most of the techniques used in Section 4.1. We perform three retrieval runs (Fig. 1) which are outlined below.

Run 1. Similar to *Run 1* of Hindi FAQ retrieval (Section 4.2), we use the native UTF-8 encoded Malayalam SMS queries on the index of UTF-8 encoded Malayalam FAQs (Section 3.2). Here too we use INDRI's #combine operator for retrieving the relevant FAQ question entries.

Run 2. As mentioned in Section 3.3, we use the oDesk crowdsourcing platform for Malayalam-English translation. We translate both the Malayalam FAQ corpus (questions only) and the SMS queries using this approach. A total number of 731 questions had to be translated from Malayalam to English. We recruited two individuals for performing this task. The accuracy of translation of a given sentence is based on the comparison of translations done by both translators. We manually evaluate a set of 20 sentences translated by both translators. All the translations were found to express the same idea, while expressions and sentence structure varied. The set of translations with better overall structure was selected for use in the retrieval experiments. The complete translation process took us around 2 days and cost us 40 USD.

Unlike *Run 2* from Section 4.2 for Hindi, we do not use *Google Spelling Suggestions* as the translated set do not contain such errors.

Run 3. This run is identical to *Run 3* of Section 4.2. Here also we use Straight Borda Count for merging results from *Runs 1* and *2.*

4.4 Mono-Lingual FAQ Retrieval Results

Table 2 shows the Mean Reciprocal Rank(MRR) scores along with in-domain(ID) and out-of-domain(OOD) scores for the mono-lingual SMS-based FAQ retrieval task. The Reciprocal Rank score for an individual query is the reciprocal of the rank at which the first relevant entry is returned (or 0 of no relevant entry is returned). The MRR is the average of the reciprocal ranks of all the queries. ID and OOD scores are simply the accuracy scores for SMS queries for which there are either corresponding FAQ entries or not. Table 2 shows these scores for each run submission per language. For English, we get the highest MRR and ID scores for Run 1 (Section 4.1). Run 2, which uses traditional spell-checker gives us the lowest MRR, ID and OOD scores. For both Hindi and Malayalam, we get the highest MRR scores of 0.860 and 0.893 using the English-translated versions (Run 2) of the SMS queries and the FAQ corpus. For these two languages we

Table 2. MRR scores for mono-lingual retrieval. In-domain (ID) and out-of-domain (OOD) scores are in the parenthesis. In case all queries are in-domain, OOD scores are blanked out (-).

	English MRR (ID/OOD)	Hindi MRR (ID/OOD)	Malayalam MRR (ID/OOD)
Run 1	0.736 (0.695/0.007)	0.746 (0.715/0.0)	0.838 (0.800/-)
Run 2	0.686 (0.644/0.006)	0.860 (0.825/0.0)	0.893 (0.880/-)
Run 3	0.709 (0.664/0.011)	0.819 (0.780/0.0)	0.881 (0.840/-)

notice that the native UTF-8 retrieval (Run 1) performs worse compared to the translated versions. Since Run 3 for these two languages take Run 1 into account, the scores for Run 3 are also negatively affected. OOD scores for the various systems are typically low since our methods assume every SMS query to have at least one corresponding FAQ.

5 Cross-Lingual FAQ Retrieval

Details of the cross-lingual FAQ retrieval task has been given in Fig. 2 and Section 2.2.

Retrieval Strategies. We execute three runs for this task. These three runs employ the same retrieval strategies as *Runs 1* to *3* described in Section 4.1 (mono-lingual English). Both tasks use the same set of English SMS queries. The only difference is that, the index used for retrieving the FAQ entries is built from Hindi-English translated questions (Section 3.2). Hence retrieved FAQ entries correspond to the Hindi FAQs for this task.

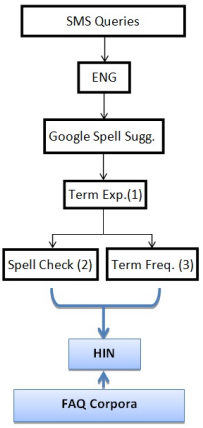


Fig. 2. Flowchart of methods adopted for cross-lingual SMS-based FAQ retrieval (ENG : English, HIN : Hindi). Run numbers are shown in parenthesis.

5.1 Cross-Lingual FAQ Retrieval Results

Table 3 shows the MRR, ID and OOD scores for the cross-lingual SMS-based FAQ retrieval task. The overall scores for the cross-lingual task are noticeably lower compared to the scores of the mono-lingual retrieval. The highest MRR score of 0.540 is obtained using Run 1. The highest ID score is obtained from Runs 1 and 3, while the highest OOD score is obtained using Run 3.

Table 3. MRR scores for cross-lingual retrieval. In-domain (ID) and out-of-domain (OOD) scores are in the parenthesis.

	MRR Scores (ID/OOD)
Run 1	0.540 (0.466/0.012)
Run 2	0.496 (0.400/0.011)
Run 3	0.525 (0.466/0.053)

6 Multi-lingual FAQ Retrieval

Details of the multi-lingual FAQ retrieval task has been given in Fig. 3 and Section 2.3. This task consists of three sub-tasks which are outlined below.

6.1 English

In this task, SMS queries given in English are executed on index of FAQs of all three languages – English, Hindi and Malayalam. Three runs were submitted for this task.

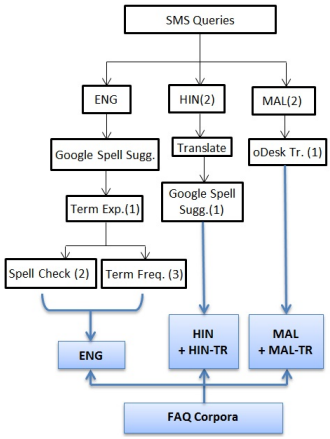


Fig. 3. Flowchart of methods adopted for multi-lingual SMS-based FAQ retrieval (ENG : English, HIN : Hindi, MAL:Malayalam, HIN-TR : Hindi translated to English and MAL-TR : Malayalam translated to English). Run numbers are shown in parenthesis.

Retrieval Strategies. For *Run 1* we perform the same steps as in *Run 1* of Section 4.1 for processing the SMS queries. These transformed queries are then executed on 3 indexes – the English FAQ index, the Hindi-English translated FAQ index and Malayalam-English translated FAQ index (Section 3.2). Similarly, for *Runs 2* and *3* we perform the same steps as in *Runs 2* and *3* of Section 4.1 for processing the SMS queries. These transformed queries are then executed on the 3 indexes described previously.

6.2 Hindi

In this task, SMS queries given in Hindi are executed on index of FAQs of all three languages – English, Hindi and Malayalam. Two runs were submitted for this task.

Retrieval Strategies. For *Run 1* we perform the same steps as in *Run 2* of Section 4.2 for processing the Hindi SMS queries. These transformed queries are then executed on 3 indexes – the English FAQ index, the Hindi-English translated FAQ index and the Malayalam-English translated FAQ index (Section 3.2). For *Run 2* we perform the same step as in *Runs 1* of Section 4.2 for retrieving the Hindi FAQ entries (from Hindi FAQ index). For the other two languages, English and Malayalam, we use the same FAQ entries retrieved in the previous run.

6.3 Malayalam

In this task, SMS queries given in Malayalam are executed on index of FAQs of all three languages – English, Hindi and Malayalam. Two runs were submitted for this task.

Retrieval Strategies. For *Run 1* we perform the same steps as in *Run 2* of Section 4.3 for processing the Malayalam SMS queries. These transformed queries are then executed on 3 indexes – the English FAQ index, the Hindi-English translated FAQ index and the Malayalam-English translated FAQ index (Section 3.2). For *Run 2* we perform the same step as in *Run 1* of Section 4.3 for retrieving the Malayalam FAQ entries (from Malayalam FAQ index). For the other two languages, English and Hindi, we use the same FAQ entries retrieved in the previous run.

6.4 Multi-lingual FAQ Retrieval Results

Table 4 shows the MRR, ID and OOD scores for multi-lingual SMS-based FAQ retrieval. For English, Run 1 performs the best with MRR and ID scores of 0.710 and 0.696 respectively. For Hindi, the best MRR and ID scores (0.839 and 0.825) are achieved for Run 2. For Malayalam, the highest MRR and ID score of 0.889 is obtained for Run 1. Similar to the mono-lingual retrieval results, we find

Table 4. MRR scores for multi-lingual retrieval. In-domain (ID) and out-of-domain (OOD) scores are in the parenthesis. In case all queries are in-domain, OOD scores are blanked out (-).

	English MRR (ID/OOD)	Hindi MRR (ID/OOD)	Malayalam MRR (ID/OOD)
Run 1	0.710 (0.696/0.006)	0.727 (0.715/0.000)	0.889 (0.889/-)
Run 2	0.680 (0.665/0.010)	0.839 (0.825/0.000)	0.829 (0.800/-)
Run 3	0.660 (0.645/0.005)	-	-

that Google Spelling Suggestions and subsequent term expansion strategy work well. For Hindi, quite surprisingly, involving the native UTF-8 retrieval gives better score in Run 2. However, for Malayalam, the English-translated retrieval experiment (Run 1) performs better than involving the corresponding UTF-8 retrieval. For Hindi and Malayalam, we had submitted only two runs.

7 Comparison of Best MRR Results for Mono-, Cross- and Multi-lingual Tasks

In Table 5 we compare the best runs for the mono- and multi-lingual tasks and find their MRR scores to be very similar, with the mono-lingual runs performing

marginally better than the multi-lingual runs. The overall performance of our system for both these tasks is significantly better compared to the cross-lingual task (Table 3).

Table 5. Best MRR scores for mono- and multi-lingual retrieval

	English	Hindi	Malayalam
Mono	0.736	0.860	0.893
Multi	0.710	0.839	0.889

However, since the cross-lingual task (English SMS queries on Hindi FAQs) is in fact a subset of the English multi-lingual subtask (English SMS queries on English/Malayalam/Hindi FAQs) we expect the errors from cross-lingual task to be transposed to the multi-lingual task. Quite surprisingly, these errors had minimal effect on the MRR scores of English multi-lingual runs. An analysis of the relevance judgments for the English multi-lingual subtask reveals that cross-lingual retrieval comprises of a smaller subset of the the English multi-lingual run.

Table 6. Distribution of languages in 724 relevance judgments

	Count (%)
English	724 (100%)
Hindi	37 (5.1%)
Malayalam	84 (11.6%)

Out of the 3405 English SMS queries, only 724 (20.6%) queries have relevance judgments. The distribution of various languages in the 724 relevance judgments is shown in Table 6. Since the number of Hindi relevance judgments (corresponding to cross-lingual task) is only 5.1% of all relevance judgments it has a minimal effect in the MRR scoring for the English multi-lingual task. Thus the scores for English multi-lingual runs are less affected compared to the cross-lingual runs.

8 Conclusion

In this paper we describe our approaches towards mono-, cross- and multi-lingual SMS-based FAQ retrieval. We follow a data-driven approach for our retrieval experiments. Google spelling suggestions generally proved useful both in native English and translated-to-English retrieval runs. Term expansion strategies perform well for the native English retrieval runs. With the exception of Hindi multi-lingual retrieval, UTF-8 encoded retrieval runs perform worse compared to the English-translated versions. For Hindi to English translations Google Translate performs better than Microsoft Bing Translator. The use of crowdsourcing

for Malayalam to English translations helped in improving the retrieval results. In fact our best multi-lingual result is rather close to the best mono-lingual run for Malayalam. As expected, we find multi-lingual retrieval to be more challenging compared to mono-lingual retrieval. Across the different tasks, we find that the retrieval performances of Malayalam and Hindi are better than English. In case of English, retrieval results are negatively affected by using standard spell-checker. Users make far fewer spelling errors in Hindi and Malayalam. The retrieval experiments in English and Indian languages helped us in understanding of the nature of complexities relevant to the various tasks. In future work, we would like to explore other techniques for normalizing text messages[7], especially, methods for handling non-standard abbreviations in the Indian text messaging context[5,2]. Our current implementation considers all SMS queries to be having at least one relevant FAQ entry. In future work we would like to incorporate filtering of SMS queries not having any possible relevance judgments.

References

1. Caraway, B.: Online labour markets: an inquiry into odesk providers. Work Organisation, Labour and Globalisation (2010)
2. Contractor, D., Kothari, G., Faruquie, T.A., Subramaniam, L.V., Negi, S.: Handling noisy queries in cross language faq retrieval. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, pp. 87–96. Association for Computational Linguistics, Stroudsburg (2010)
3. Fox, E.A., Shaw, J.A.: Combination of Multiple Searches, vol. 500-215, pp. 243–252. National Institute for Standards and Technology, NIST Special Publication 500215 (1994)
4. Howe, J.: The rise of crowdsourcing. Wired Magazine 14(14), 1–5 (2006)
5. Kothari, G., Negi, S., Faruquie, T.A., Chakaravarthy, V.T., Subramaniam, L.V.: Sms based interface for faq retrieval. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, ACL 2009, vol. 2, pp. 852–860. Association for Computational Linguistics, Stroudsburg (2009)
6. Strohman, T., Metzler, D., Turtle, H., Croft, W.B.: Indri: a language-model based search engine for complex queries. In: Proceedings of the International Conference on Intelligence Analysis (2005)
7. Whitelaw, C., Hutchinson, B., Chung, G.Y., Ellis, G.: Using the web for language independent spellchecking and autocorrection. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, vol. 2, pp. 890–899. Association for Computational Linguistics, Stroudsburg (2009)

Language Modeling Approach to Retrieval for SMS and FAQ Matching

Aditya Mogadala, Rambhoopal Kothwal, and Vasudeva Varma

International Institute of Information Technology, Hyderabad
{aditya.m,bhupal_iiit}@research.iiit.ac.in, vv@iiit.ac.in

Abstract. Short Messaging service popularly known as "SMS" has seen growth due to the growth in Mobile phone users. A mobile phone is considered as a cheap and easy device for communication. It is also used as a source to acquire and spread information. SMS based FAQ Retrieval task proposed in FIRE 2011 aims to provide the required information from frequently asked questions (FAQs). Challenge is to find a question from corpora of FAQs that best answers/matches with the SMS query. But, SMS queries are noisy as users tend to compress text by omitting letters, using slang, etc. This is observed due to a cap on the length of messages (160 characters constitute one SMS), lack of screen space (which makes reading large amounts of text difficult). In this paper, we propose a method using language modeling approach to match noisy SMS text with right FAQ. We extended this framework to match SMS queries with Cross-language FAQs. Results are promising for monolingual retrieval applied on English, Hindi and Malayalam languages.

1 Introduction

Growth in mobile phones and telecommunication operators made exchanging and seeking of information very easy. Sharing information in mobile phones is done through short texts known as SMS. But, text used in SMS is quite noisy and very difficult to interpret. SMS sometimes contain requests from users about information present in websites in the form of FAQs. For example, a user can ask about a train arrival and departure timings using his mobile phone to a customer service. This information may be already available on railways website in the form of FAQs. Information linkage between FAQs and SMS will avoid lot of customer service calls. Our main goal is to provide such a solution by linking SMS to FAQs.

Our team took part in 4 different tasks from two major categories of Monolingual and Cross-lingual SMS and FAQ matching in FIRE 2011. Mainly, English SMS - English FAQ, Hindi SMS - Hindi FAQ, Malayalam SMS - Malayalam FAQ and English SMS - Hindi FAQ. Challenge was observed in matching noisy SMS queries to FAQs. Finding exact match of SMS queries with FAQs require prior cleansing of SMS queries.

In this paper, we present dictionary based approaches to clean the SMS text. Also, we present techniques to create these dictionaries. After cleansing, SMS

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 119–130, 2013.

text is matched with FAQs using language model based retrieval approach. In this approach, SMS is treated as query while FAQ is treated as document. Based on the query likelihood calculated between the SMS query and FAQ document top 5 possible matches are retrieved. In-order to keep the approach language independent, no language specific tools or resources are used for matching SMS and FAQs. Same system was used across languages used for experimentation.

Remainder of this paper is organized into following sections. Related work is mentioned in the Section 2. Next Section 3 discuss about noise removal from the SMS queries. In the Section 4 information is given about language model approach for matching FAQs with SMS queries. While the experimental setup Section 5 give information about the collection. Experiments are explained in Section 6 while result analysis in Section 7. Conclusions and future work is discussed in Section 8 followed by References.

2 Related Work

SMS text and Twitter[1] has been of interest over the past few years to understand the users information need. Due to noisy characteristics of SMS text, analyzing SMS information is cumbersome and difficult [1]. Some of the earlier attempted approaches to solve the problem used web as a resource. These approaches not only help in understanding the SMS text but also reduce time and effort of creating new resources. Using frequently asked questions (FAQs) present in different websites is one such approach. Matching SMS queries [2] with FAQs help understanding the user information need. Other related works which are similar to this approach identifies the best matching question to retrieve the relevant answer [3]. Here, SMS queries were used to provide the best possible FAQs based on similarity between them. But similarity between two different texts or sentences is not a novel task. It has been worked upon over the years [4,5,6,7].

But when we deal with noisy queries like SMS in different languages we need novel approaches to match FAQs. In this paper, we presented SMS match with different monolingual and cross-lingual FAQs using language model based retrieval.

3 Noise Removal from SMS Query

The noise in a SMS corpus can be typically categorized into five different categories [1]. Following observations are made for English SMS corpus collected from FIRE SMS Task[2].

1. The commonly observed patterns include deletion of vowels, addition of repeated character and truncation. For example, “abt” is written for “about“ after removing vowels, “sooo” for “so” and truncating days like “Thursday“ to “Thurs”.

[1] http://twitter.com

[2] http://www.isical.ac.in/~clia/faq-retrieval/data.html

2. The SMS data belongs to different domains like telecommunication, railways, insurance etc. Some of the frequently used abbreviations in these areas are IRCTC in the area of railways, BSNL in the area of telecommunication etc.
3. Substitution of spoken words with the actual spellings of the words popularly known as phonetic substitution. For example, usage of "bookin" for "booking" in tourism domain etc.
4. Informal usage of different words is common in SMS text. Often multiple words are combined into a single token following certain dialectal conventions. For example, "wr2" for "where to", "hw2" for "how to" etc.
5. Missing words in the sentences due to the limitation of text message. SMS query sometimes just provide keywords and miss conjunctions, prepositions and other words which connect the keywords. For example, "SMS packs" used instead of "SMS packs available for recharge" etc.
Above problems make SMS text very noisy. Cleansing of SMS text is required to do further processing and analysis. For cleansing, different dictionary based strategies are employed. Dictionaries are created using manual and automatic efforts to match the noisy terms in the SMS with their corresponding actual terms.

3.1 Dictionary Creation

Dictionaries are created to provide easy access to the misspelled or cross-language words. But, creation of dictionaries can be time consuming and expensive. To minimize the effort, we created dictionaries using automated and manual techniques. These dictionaries contain misspelled words and their corresponding correct words for English. Bi-lingual dictionaries are also created for cross-lingual task. Below, we will present approaches and list of resources generated for monolingual and cross-lingual task.

Dictionary Created from SMS for English (Manual)

SMS queries provided as training data for English monolingual and cross-lingual task is used to create the dictionary. Unique words are extracted from SMS queries to create a noisy words list. Each SMS noisy query word is manually annotated with corresponding correct word. Total 4432 word list is created.

Dictionary Created from FAQs for English (Automatic)

FAQs provided as a training data is used to create a unique words list. Noisy SMS words extracted from SMS queries is matched with most probable word in the FAQs unique words list using a spell checker[3]. For a given noisy word, we try to choose the most likely spelling correction for the word. Using these approach 3262 words with correct spellings are created.

[3] `http://norvig.com/spell-correct.html`

English to Hindi Bi-lingual Dictionary Using Wikipedia (Automatic)

Bilingual dictionaries are specialized dictionaries used to translate words or phrases from one language to another. Bi-lingual English to Hindi dictionary is an important resource for translation of English SMS to Hindi SMS. Manual development of them is expensive. So, we used Wikipedia[4] as an external resource to develop the dictionaries. Wikipedia is a semi-structured resource which can be very helpful in extracting resources of different languages. Here, we exploit the Wikipedia link structure. We followed the approach [8] to use Wikipedia title information of English and Hindi word.

Each article in the English Wikipedia is considered to get cross lingual link to Hindi article. All such articles are collected. The same is followed by considering Hindi Wikipedia and the collection is updated with new title pairs. We created a list of 3027 words bi-lingual dictionary using this approach.

English to Hindi Bi-lingual Dictionary (Manual)

Dictionaries created as a part of CLIA consortium[5] for named entities and bi-lingual dictionaries are used for replacement of English with Hindi words. The words which are out of this list are translated using the Google Translate API[6]. There are around 4686 bi-lingual dictionary words and 24112 named entities in the dictionaries.

3.2 Noise Removal

Resources that are created for noise removal as mentioned in earlier sections are used to remove the noise from SMS text. Each SMS query is treated as a separate sentence. Unique noisy words are extracted from these sentences. Each noisy word from unique words is searched for the corresponding correct word in the dictionaries. If the word is matched from any of the dictionaries, the word is replaced with the correct word. But, sometimes a word can be found in manually created dictionary and automatically created dictionaries. In that case manually created dictionary is given higher priority for replacement due to high accuracy of manual dictionaries. **Algorithm 1** mentioned below explains the approach used.

4 Language Modeling Approach to Retrieval for FAQ and SMS Matching

Language modeling approach to retrieval [9] is used to match the SMS with FAQs. Each FAQ is treated as a document and each SMS as a query. Top ranked

[4] http://www.wikipedia.org/
[5] http://clia.forumotion.com/
[6] http://code.google.com/apis/language/translate/overview.html

Algorithm 1. Noise Removal from SMS Query

Require: User SMS U,Dictionaries D^e
  for u^k in U **do**
    Find Unique words UW in u^k
    for uw_i in UW **do**
      if uw_i in D^e **then**
        Replace(uw_i,Correct uw_i)
      end if
    end for
    return Corrected u^k
  end for

FAQ documents are retrieved using each SMS query. But, FAQs are combination of questions and answers. So, FAQ documents are segregated into three different collections. One that contains only questions, second that contains only answers and third containing both questions and answers.

Each of these document collections are used to create different language models. Since this approach is completely natural language independent. Different language models are created for English, Hindi and Malayalam. Below, we will see the approach for monolingual and Cross-lingual SMS and FAQ match.

4.1 Monolingual

Each collection of monolingual documents is used to create their respective language models (LM). A Language model (LM) approach for SMS retrieval is probability of SMS query Q_{sms} is being generated by a probabilistic model based on a FAQ document D_{faq} denoted as $p(Q_{sms}|D_{faq})$. In order to create LM for the documents collection, sentences are selected. Then to rank the FAQ documents based on sentences, posterior probability needs to be estimated using Bayes formula given by Equation 1.

$$p(D_{faq_i}|Q_{sms}) \propto p(Q_{sms}|D_{faq_i})p(D_{faq_i}) \tag{1}$$

where $p(D_{faq_i})$ is a prior belief that is relevant to any of the SMS queries and $p(Q_{sms}|D_{faq_i})$ is the query likelihood given the document D_{faq_i}, which captures the particular SMS query information.

Assumption of $p(D_{faq_i})$ distribution is considered multinomial distribution as opposed to the existing work [10]. This assumption helps in choosing better smoothing techniques.

So, for each document D_{faq_i} in the collection, its language model defines the probability of $p(SMS_{w_1}, SMS_{w_2}, ..., SMS_{w_n}|D_{faq_i})$ of a sequence of n SMS query terms and the documents are ranked by that probability.

The probabilities of the SMS words SMS_{w_i} in document $p(D_{faq_i})$ is enhanced by their presence in the document collection. Equation 2 gives the probability

scores for entire document collection C_{faq} while Equation 3 gives only for a document D_{faq}.

$$p(SMS_{w_i}|C_{faq}) = \frac{cfreq(SMS_{w_i}, C_{faq})}{\Sigma_{SMS_w} cfreq(SMS_w, C_{faq})} \tag{2}$$

$$p(SMS_{w_i}|D_{faq}) = \frac{tfreq(SMS_{w_i}, D_{faq})}{\Sigma_{SMS_w} tfreq(SMS_w, D_{faq})} \tag{3}$$

Here, $cfreq(SMS_{w_i}, C_{faq})$ represents collection frequency of the term SMS_{w_i} in the collection C_{faq} and $tfreq(SMS_{w_i}, D_{faq})$ is term frequency of the SMS_{w_i} in a document D_{faq}.

The non smoothed model gives a maximum likelihood estimate of the relative counts given by $p_{ml}(SMS_w|D_{faq})$. But, if the word is unseen in the document then it results in the zero probability. So, smoothing is helpful to assign a non-zero probability to the unseen words and improve the accuracy of word probability estimation in general. We used Dirichlet smoothing to assign non-zero probabilities to unseen words in the documents and collection as mentioned in [11]. So after smoothing, the Equation 3 is converted into Equation 4.

$$p_\mu(SMS_w|D_{faq}) = \frac{tfreq(SMS_{w_i}, D_{faq}) + \mu p(SMS_w|C_{faq})}{\Sigma_{SMS_w} tfreq(SMS_w, D_{faq}) + \mu} \tag{4}$$

where μ is Dirichlet parameter.

4.2 Cross-Lingual

A traditional technique that matches SMS with FAQs for monolingual documents identifies relevant FAQ documents in the same language as the SMS query. It is similar to the monolingual IR technique. For Cross-language SMS and FAQ match we need to identify relevant documents in a language different from that of the SMS query. This problem is similar to Cross-language Information retrieval (CLIR) [12] where it identifies documents in other language for a query given in one language. This is done due to truly multilingual environment of the web.

Similar approach is followed to achieve Cross-lingual SMS match with FAQs. Initially, SMS queries needs to be translated from source language to target language. But, Translation of SMS query from source (English) to target (Hindi) language requires identification and replacement of noisy words with correct words. This task is executed using the noise removal approaches mentioned in earlier sections. After cleaning the SMS query, it is translated from source language to target language using dictionary based translation approaches. Then language model created for target language is used to rank the FAQ documents for the translated SMS queries using the language model based retrieval approach mentioned in previous **section 4.1**.

5 Experimental Setup

Experiments were conducted using the data collected from FIRE SMS TASK[7] containing monolingual, cross-lingual and multilingual XML files. Each of these files are parsed and segregated for monolingual, cross-lingual and multilingual tasks. The files in the dataset contained FAQs and In-domain, Out-domain SMS queries. The dataset is also further divided into training and test data. Table 1 and Table 2 show the statistics of the In-domain and Out-of-domain SMS queries of training data. While, Table 3 and Table 4 show the statistics of the In-domain and Out-of-domain of SMS queries of test data. Table 5 show the figures of FAQs present in the training data.

Table 1. SMS Queries Count in Training Data (In-Domain)

SMS Training Data (In-Domain)		
	Monolingual	**Cross-lingual**
English	701	291
Hindi	181	-
Malayalam	120	-

Table 2. SMS Queries Count in Training Data (Out-Domain)

SMS Training Data (Out-of-Domain)		
	Monolingual	**Cross-lingual**
English	370	181
Hindi	49	-
Malayalam	20	-

Table 3. SMS Queries Count in Testing Data (In-Domain)

SMS Testing Data (In-Domain)		
	Monolingual	**Cross-lingual**
English	728	37
Hindi	200	-
Malayalam	50	-

6 Experiments

Experiments were conducted using SMS Test data and FAQ training data mentioned in Section 5. FAQ training data is further divided into three different collections for three different runs. Below, we see description of each run performed for monolingual and cross-lingual setting.

[7] `http://www.isical.ac.in/~clia/faq-retrieval/data.html`

Table 4. SMS Queries Count in Testing Data (Out-Domain)

SMS Testing Data (Out-of-Domain)		
	Monolingual	**Cross-lingual**
English	2677	3368
Hindi	124	-
Malayalam	0	-

Table 5. FAQs Count in Training Data

FAQs Training Data	
English	7251
Hindi	1994
Malayalam	681

6.1 Monolingual Run 1

This run constitutes an approach which uses FAQ questions as documents and SMS as queries for monolingual SMS and FAQ match. Each FAQ question is written into a document and indexed using lemur[8] toolkit. Cleansed SMS is then used as a query to match 5 best documents containing FAQ question using language model approach to retrieval. This approach is executed on English, Hindi and Malayalam languages.

6.2 Monolingual Run 2

This run constitutes an approach which uses FAQ answers as documents and SMS as queries for monolingual SMS and FAQ match. Each FAQ answer is written into a document and indexed using the lemur toolkit. Cleansed SMS is then used as a query to match 5 best documents containing FAQ answer using language model approach to retrieval. This approach is executed on English, Hindi and Malayalam languages.

6.3 Monolingual Run 3

This run constitutes an approach which uses both FAQ questions and answers as documents and SMS as queries for monolingual SMS and FAQ match. Each FAQ question and its corresponding answer is written into a document and indexed using the lemur toolkit. Cleansed SMS is then used as a query to match 5 best documents containing FAQ question and answer using language model approach to retrieval. This approach is executed for English, Hindi and Malayalam languages.

[8] `http://www.lemurproject.org/`

6.4 Cross-Lingual Run 1

Hindi FAQs and English SMS are used for Cross-lingual SMS and FAQ match. Each SMS is translated from English to Hindi using dictionary based translation approaches. For Cross-lingual Run 1, FAQ questions in Hindi are selected to form FAQ question documents. These documents are indexed using lemur toolkit. Each translated SMS query is then used to match 5 best FAQ documents using language model approach to retrieval.

6.5 Cross-Lingual Run 2

For Cross-lingual Run 2, FAQ answers in Hindi are selected to form FAQ answer documents. These documents are indexed using lemur toolkit. Each translated SMS query is then used to match 5 best FAQ documents using language model approach to retrieval.

6.6 Cross-Lingual Run 3

For Cross-lingual Run 3, FAQ questions and answers in Hindi are selected to form FAQ question and answer documents. These documents are indexed using lemur toolkit. Each translated SMS query is then used to match 5 best FAQ documents using language model approach to retrieval.

7 Result Analysis

Comparison between different runs are done using Mean Reciprocal Rank(MRR) and total matches for In-domain and Out-of-domain SMS queries. **Table 6** shows the results obtained for monolingual task for three different languages. It is observed from the table that for English **Run 1** which considers only FAQ questions performs better than other Runs **2 & 3** which uses answers and combination of questions and answers on MRR for SMS match. **Run 1** outperforms **Run 2 and Run 3** by 226.69% and 3.4% respectively.

Similarly, Hindi **Run 1** which considers only FAQ questions performs better than other Runs **2 & 3** which uses answers and combination of questions and answers on MRR for SMS match. **Run 1** outperforms **Run 2 and Run 3** by 263.9% and 17.1% respectively.

Malayalam has been seen less fluctuations between top two best performing runs which consider questions and combination of questions and answers. But contrastingly Malayalam **Run 3** which considers FAQ questions and answers performs better than other Runs **1 & 2** which uses answers and questions respectively on MRR for SMS match. **Run 3** outperforms **Run 1 and Run 2** by 1.5% and 241.4% respectively.

However, out-of-domain results are low for all languages for monolingual and cross-lingual SMS and FAQ matching. It is observed due to $P(D_{faq}|SMS)$ counts are pretty low for out-of-domain as compared to those for in-domain

queries. Reason can be accounted to common vocabulary between out-of-domain queries and in-domain queries which failed to identify FAQs of out-of-domain.

Cross-lingual task results are shown in **Table 7**. It can be observed from table that **Run 1** which considers only FAQ questions performs better than other Runs **2 & 3** which uses answers and combination of questions and answers on MRR for SMS match. **Run 1** outperforms **Run 2 and Run 3** by 833.3% and 9.8% respectively. In this task, difference between top two runs is not much observed. Also, due to the translation errors low scores were observed. This can be inherently attributed to the dictionary based approaches followed for cross-lingual porting of SMS text. Statistical Machine Translation based approaches would have provided more possibilities for the word replacements of improper or wrongly spelled words.

Table 8 shows the comparison of our team performance with rest of the teams participated in different tasks. Our team surpassed median score by 28.5% and 2.2% in English Monolingual and Malayalam Monolingual tasks respectively. In Hindi Monolingual and Cross-lingual our team narrowly missed median score by 1.9% and 2.0% respectively. Overall our team was able to achieve 5th position in English Monolingual task out of 13 teams participated, 4th out of 7 teams participated in Hindi Monolingual task, 2nd out of 4 teams participated in Malayalam Monolingual task and 4th out of 5 teams participated in Cross-lingual task.

Table 6. Monolingual Task Results

Monolingual Task				
Language	**RunId**	**In-Domain**	**Out-of-Domain**	**MRR**
English	1	**458/728**	118/2677	**0.70375574**
English	2	121/728	95/2677	0.21540777
English	3	396/728	91/2677	0.62051535
Hindi	1	**171/200**	2/124	**0.88581**
Hindi	2	39/200	0/124	0.24342921
Hindi	3	139/200	0/124	0.7560981
Malayalam	1	45/50	0/0	0.9257142
Malayalam	2	11/50	0/0	0.2753623
Malayalam	3	**46/50**	0/0	**0.94019043**

Table 7. Cross-lingual Task Results

Cross-lingual Task				
Language	**RunId**	**In-Domain**	**Out-of-Domain**	**MRR**
English-Hindi	1	3/37	159/3368	**0.1009009**
English-Hindi	2	0/37	64/3368	0.010810811
English-Hindi	3	3/37	61/3368	0.09189189

Table 8. Results Comparison among Teams for different Tasks

Comparison among Teams for different Tasks				
Team	**English-Mono**	**Hindi-Mono**	**Malayalam-Mono**	**Cross-lingual**
Median-Scores	0.140	0.530	0.900	0.049
Iowa	0.140	0.510	0.880	0.051
BUAP	0.130	0.480	0.780	0.048
DCE	0.570	0.590	-	**0.650**
IIIT-H	0.180	0.520	0.920	0.048
DAIICT	0.110	0.580	**0.940**	-
Jadhavpur-IPN	0.01	0.380	-	0.021
DTU	0.420	**0.620**	-	-
DCU	**0.830**	-	-	-
MSRIT	0.000	-	-	-
TCS	0.07	-	-	-
SASTRA	0.02	-	-	-
RVCE	0.780	-	-	-
IIITD	0.000	-	-	-

8 Conclusion and Future Work

In this paper, we matched English, Hindi and Malayalam SMS with FAQs for monolingual task and FAQs in Hindi with SMS in English for cross-lingual task. Language modeling approach for retrieval is used after noise reduction from SMS queries to match SMS queries with FAQs. Initially, FAQs were divided into three different collections using combination of questions and answers. Different language models are formed constituting 9 different sets for three languages. FAQs division was based on only questions, only answers and combination of questions and answers for each of the three languages. It is observed from the results that questions language model outperformed both answers and combination of questions and answers for matching SMS queries.

In future, this approach can provide beter results using more robust noise removal techniques. Also, language specific dependencies like bi-lingual dictionaries can be reduced by using hybrid approaches which uses statistical machine translation techniques etc.

References

1. Contractor, D., Faruquie, T., Subramaniam, L.: Unsupervised cleansing of noisy text. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters, pp. 189–196 (2010)
2. Kothari, G., Negi, S., Faruquie, T., Chakravarthy, V., Subramaniam, L.V.: SMS based Interface for FAQ Retrieval. In: Annual Meeting of the Association for Computation Linguistics (2009)
3. Sneiders, E.: Automated FAQ Answering: Continued Experience with Shallow Language Understanding Question Answering Systems. In: AAAI Fall Symposium. Technical Report FS-99-02, pp. 97–107. AAAI Press (1999)

4. Mihalcea, R., Corley, C., Strapparava, C.: Corpus-based and knowledge-based measures of text semantic similarity. In: AAAI (2006)
5. Sahami, M., Heilman, T.: A web-based kernel function for measuring the similarity of short text snippets. In: World Wide Web. ACM Press (2006)
6. Pedersen, T.: Computational approaches to measuring the similarity of short contexts: A review of applications and methods. CoRR, abs/0806.3787 (2008)
7. Shrestha, P.: Corpus-based methods for short text similarity. In: 15th Rencontre des Etudiants Chercheurs en Informatique pour le Traitement Automatique des Langues, vol. 2, pp. 297–302 (2011)
8. Bharadwaj, R., Tandon, N., Varma, V.: An Iterative approach to extract dictionaries from Wikipedia for under-resourced languages. In: 8th International Conference on Natural Language Processing, ICON (2010)
9. Ponte, J.M., Bruce Croft, W.: A language modeling approach to information retrieval. In: 21st ACM SIGIR, pp. 275–281 (1998)
10. Berger, A., Lafferty, J.: Information retrieval as statistical translation. In: ACM SIGIR, pp. 222–229 (1999)
11. Zhai, C., Lafferty, J.: A Study of Smoothing Methods for Language Models Applied to Information Retrieval. ACM Transactions on Information Systems 22(2), 179–214 (2004)
12. Ballesteros, L., Croft, B.: Dictionary Methods for Cross-Lingual Information Retrieval. In: Thoma, H., Wagner, R.R. (eds.) DEXA 1996. LNCS, vol. 1134, pp. 791–801. Springer, Heidelberg (1996)

SMS Based FAQ Retrieval

Nishit Shivhre

Department of Computer Science, R.V. College of Engineering,
Mysore Road, Bangalore-560059, India
ns5491@gmail.com

Abstract. Short Messaging Service (SMS) is popularly used to provide information access to people on the move. An Automated FAQ answering system is used to answer people's queries through an automated system. This system is designed in a way that it is able to answer the queries asked in SMS(Short Messaging Service) language which is more convinient to users but also contains a lot of noise like abbreviations, slangs, etc. The effectiveness of this system is demonstrated on the FIRE Test Data set.

Keywords: Information Retrieval, Levenshtein Distance, Longest Common Subsequence Ratio, Inverse Document Frequency.

1 Introduction

The number of mobile users is growing at an amazing rate. In India alone a few million subscribers are added each month with the total subscriber base now crossing 370 million[1]. The anytime anywhere access provided by mobile networks and portability of handsets coupled with the strong human urge to quickly find answers has fueled the growth of information based services on mobile devices. This FAQ retrieval system is designed to find a match from the given set of Frequently Asked Questions for a query written in SMS language. The problem with questions asked in SMS language is that the SMS text has a lot of noise present in it which might be due to lack of a proper keypad in low cost mobile phones, less screen space and also cause of convenience to the users. Understanding user questions in natural languages requires Natural Language Processing (NLP). This QA system can provide a convenient and effective way of giving answers to such questions.

The nature of texting language, which often as a rule rather than exception, has misspellings, non-standard abbreviations, transliterations, phonetic substitutions and omissions, makes it difficult to build automated question answering systems around SMS technology. This is true even for questions whose answers are well documented like a FAQ database. Unlike other automatic question answering systems that focus on generating or searching answers, in a FAQ database the question and answers are already provided by an expert. The task is then to identify the best matching question-answer pair for a given query. The approach we

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 131–141, 2013.

have adopted in this project is an automated FAQ (Frequently Asked Question) answering system that gives the best matching questions, from a pre-stored set of questions and answers that have been provided, to questions asked in ordinary SMS text. This is achieved using sequence matching techniques, disemvoweling, etc.

The rest of the paper is organized as follows. In Section 2 the problem statement is given. Section 3 provides system implementation details and describes the Algorithm which finds the best matching question for a given SMS query. Section 4 provides details about the experiments conducted on the system. Finally I conclude in Section 5.

2 Problem Statement

In this task, we have a corpus of frequently asked questions and answers from various domains that have been provided. The corpora of questions in the database are represented by Q. The query is in SMS language which may or may not contain noise. The goal of the task is to find a question Q* from the corpora of FAQ's Q, that is the best possible match for the SMS query S. In order to achieve this, we have made use of techniques like disemvoweling, removal of stop words, Longest Common Subsequence (LCS), etc.. We remove all the stop words from the SMS query. In disemvoweling, we remove all the vowels from the user's query and from the corpus of questions and search for keywords of the disemvoweled query in the disemvoweled set of questions. The question that has maximum number of matching keywords gets the highest score (keyword score) Also, we find the best possible match for each word in the query from all the words occurring in all the questions in Q. For this we use techniques like Longest Common Subsequence. The question which has words that are best possible matches for the words in the query gets the highest score (similarity score).

Therefore, we have two parameters for calculating the score of a question, keyword score and similarity score. The methods for calculating the keyword score, like disemvoweling, are based on the general observations made about the language and slangs used by people while typing SMS text. On the other hand, the similarity score is calculated using dynamic programming techniques for string comparison and pattern matching algorithms, like Longest Common Subsequence and Gestalt Pattern Matching.

We combine the two scores obtained to get the total score for the questions and the question having the maximum total score is returned as the best possible match Q* for the SMS query S.

3 System Implementation

3.1 Preprocessing

In this section we describe the prior work required before we start finding the match for the SMS query.

We obtained a set of words W that contains all the words occurring in all the questions in Q. These words have been stored in a hash table in which the keys are characters from a-z and numbers 0-9. Thus the words get stored in alphabetical order. For example, a key 'i' contains all the words in the set Q that start with the letter 'i', like 'insurance', 'improve', and so on. This is done so that we can find the lists of matching words for each of the words in the SMS query S more efficiently. This is described in detail later.

A list of stop words is also prepared and disemvoweled.

Digits occurring in SMS tokens (e.g. 'w8', '4get') are replaced by a string based on a manually designed digit-to- string mapping ('8' →'eight').

3.2 Calculation of Weight of Each Word in W

For each token of the SMS query (not disemvoweled), we calculate its similarity with every word w in the corpus W. The weight of a word is given by the equation:

$$Weight(w,s) = \frac{LCSR(w,s) * SMRatio(w,s) * IDF(w)}{LevDistance(w,s)} \ . \tag{1}$$

Where,
LCSR(w,s) - Longest Common Subsequence Ratio of the SMS query token s and the word w in W.
SMRatio(w,s) - Similarity ratio using Ratcliff/Obershelp algorithm.
LevDistance(w,s) - Levenshtein Distance between disemvoweled w and s.
IDF(w) - Inverse Document Frequency of w.

Longest Common Subsequence Ratio(LCSR). The longest common subsequence (LCS) problem is to find the longest subsequence common to all sequences in a set of sequences (often just two). A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements. The Longest Common Subsequence Ratio between a word w in W and a token s in SMS query S is the ratio of their LCS to the maximum of the lengths the two.

$$LCSR(w,s) = \frac{LCS(w,s)}{maxlength(w,s)} \ . \tag{2}$$

Example: LCSR of 'thru' and 'through' is 3/7=0.4285.

For a word to have high weight, its LCSR should be high.

Similarity Ratio. The similarity ratio(SMRatio) of two words is calculated using Ratcliff/Obershelp algorithm[2] for 'gestalt pattern matching'. Gestalt is a word that describes how people can recognize a pattern as a functional unit that has properties not derivable by summation of its parts. For example, a person can recognize a picture in a connect-the-dots puzzle before finishing or

Table 1. An example of calculating Similarity Ratio between two strings using the Ratcliff Obershelp Algorithm

Word1	Word2	Common substring	Length
Pennsylvania	Pencilvaneya	lvan	8
Pennsy ia	Penci eya	Pen	6
nsy ia	ci ey	a	2
nsy i	ci ey	(none)	0
Subtotal			16
Length of original strings			24
SMRatio=16/24			0.67

even beginning it. This process of filling in the missing parts by comparing what is known to previous observations is called gestalt. The Ratcliff/Obershelp pattern-matching algorithm uses this same process to decide how similar two strings are.

The algorithm works by examining two strings passed to it and locating the largest group of characters in common. The algorithm uses this group of characters as an anchor between the two strings. The algorithm then places any group of characters found to the left or the right of this anchor on a stack for further examination. This procedure is repeated for all substrings on the stack until there is nothing left to examine. The algorithm calculates the score returned as twice the number of characters found in common divided by the total number of characters in the two strings

For example, suppose you want to compare the similarity between the word 'Pennsylvania' and a mangled spelling as 'Pencilvaneya'. The largest common group of characters that the algorithm would find is 'lvan'. The two sub-groups remaining to the left are 'Pennsy' and 'Penci', and to the right are 'ia' and 'eya'. The algorithm places both of these string sections on the stack to be examined, and advances the current score to eight, two times the number of characters found in common. The substrings 'ia' and 'eya' are next to come off of the stack and are then examined. The algorithm finds one character in common: a. The score is advanced to ten. The substrings to the left—'i' and 'ey'—are placed on the stack, but then are immediately removed and determined to contain no character in common. Next, the algorithm pulls 'Pennsy' and 'Penci' off of the stack. The largest common substring found is 'Pen.' The algorithm advances the score by 6 so that it is now 16. There is nothing to the left of 'Pen', but to the right are the substrings 'nsy' and 'ci', which are pushed onto the stack. When the algorithm pulls off 'nsy' and 'ci' next, it finds no characters in common. The stack is now empty and the algorith ready to return the similarity value found. There was a score of 16 out of a total of 24.

$$SMRatio(w, s) = 2 * \frac{No.\ of\ common\ characters}{No.\ of\ characters\ in\ the\ two\ strings}\ . \qquad (3)$$

Levenshtein Distance. Levenshtein distance is a 'distance' (string metric) between two strings, i.e., finite sequence of symbols, given by counting the minimum number of operations needed to transform one string into the other, where an operation is defined as an insertion, deletion, or substitution of a single character, or a transposition of two characters. To find the Levenshtein Distance between w and s, we first disemvowel them and then calculate the Levenshtein distance.

For example, the Levenshtein distance between "kitten" and "sitting" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

kitten → sitten (substitution of 's' for 'k').

sitten → sittin (substitution of 'i' for 'e').

sittin → sitting (insertion of 'g' at the end).

Words which have a low Levenshtein Distance are the ones which require less number of operations to get transformed into the SMS token. Thus for the weight of a word to be high, its Levenshtein Distance must be low.

Inverse Document Frequency (IDF)[1]. If f number of documents in corpus Q contain a term w and the total number of documents in Q is N, the Inverse Document Frequency (IDF) of w in W is

$$IDF(w) = log\frac{N}{f} \ . \tag{4}$$

This means that a word which occurs less number of times in the corpus Q will have a high IDF. The reason behind this logic is that queries are composed of more informative words.

We have used a hashtable to store IDF for each word in W.

3.3 Creation of Variant Lists

In order to calculate the similarity score of each question in Q, we first create a variant list for each SMS token. This is done by calculating weight of each word w in W with respect to each SMS token s using equation (2). This list is then sorted in descending order. A word is said to be a variant of the SMS token if it starts with the same character and if

Length (LCS (w, s)) ≥ 1

To get a more accurate and quick result we limit the list of variants for each SMS token to five. Thus the final variant list for an SMS token s will contain its top five variants, i.e., those words that have the highest weight with respect to s.

3.4 Creation of Candidate List *Q-Poss*

A search is performed on the corpus Q for the questions that contain the variants for an SMS token and all these question are added to a candidate list called Q-Poss. Thus, Q-Poss will contain all the questions that could possibly be the matching question Q* for the SMS query S.

Algorithm 1. Creation of candidate list Q-Poss

Input : *SMS tokens* $s_1, s_2, \ldots, s_n$
Output: *Candidate list Q-Poss*

begin
Construct variant lists L_1, L_2, ..., L_n ;
i ← 1
k ← *No. of questions in Q*
Q-Poss ← ∅
while $i \neq k$ **do**
for $j \leftarrow 1$ **to** 5 **do**
$t = L_i[j]$;
Query the index and fetch Q_t;
foreach $Q \in Q_t$ **do**
add Q to *Q-Poss*;
endfch
endfor
$i \leftarrow i + 1$
endw
end

3.5 Calculation of Similarity Score

The similarity score is calculated for all the questions in Q-POSS. In order to calculate similarity score for a question q in Q-POSS, we create a list,Q-words, of the words occurring in q.

In an iterative manner, we select a word from the question q which has the maximum weight with respect to an SMS token s and add its weight is added to the similarity score for q. That word is then removed from the list Q-words. This process is repeated till the word for each SMS token is searched for. Thus,for each token s_i, the scoring function chooses the term from q having the maximum weight; then the weight of the n chosen terms are summed up to get the similarity score.

$$SimilarityScore(q) = \sum_{i=1}^{n} Weight(w^*, s_i) \ . \tag{5}$$

Where,
w^* = word in the question q with max $Weight$ w.r.t. SMS token s_i

3.6 Keyword Matching

In this section, we describe the methods used to calculate the second parameter used for calculating the score of a question, keyword score. For this, vowels and stop words are removed from each question in the list Q-POSS and these processed questions sre kept in a separate list.

Algorithm 2. Finding best matching question Q^*

Input : *Candidate list Q-Poss*, *SMS tokens* $s_1, s_2, \ldots, s_n$
Output: *Best matching question* Q^*

```
begin
    Q-words ← ∅
    foreach q ∈ Q−Poss do
        SimilarityScore(q) ← 0
        Create list Q-words;
        for i ← 1 to n do
            foreach qw ∈ Q-words do
                w* = qw with max Weight(qw,s_i) ;
                Delete w* from q-words ;
                Compute SimilarityScore(q) using equation 6 ;
            endfch
        endfor
        Compute TotalScore(q) using equation 7 ;
    endfch
    Q* = q ∈ Q-Poss with max TotalScore(q) ;
    Output Q* and exit.;
end
```

Disemvoweling[3]. We describe the process of removing vowels from a string as disemvoweling and the string from which vowels have been removed is said to be disemvoweled. We apply this process of disemvoweling to the SMS query. The reason behind using this technique is that while entering text in an SMS, the user tries to compress the text by using slangs and omitting letters and we have observed that in general, it is done by omitting some vowels from the text or difference in the usage of vowels. Vowels can also account for most of the spelling mistakes made by users.

Example: 'transaction' → 'trnsctn'.

Removal of Stop Words. In computing, stop words are words which are filtered out prior to, or after, processing of natural language data (text). It is controlled by human input. There is not one definite list of stop words. The list of stop words that we have used includes the most common short function words such as the, is, at, which, on, etc. and common lexical words as well. The list of stop words is disemvoweled and words occurring in the disemvoweled SMS query that are present in the list of stop words are removed from the query. This is done to increase the performance and effectiveness of the system by saving time and disk space and it also improves the process of keyword matching.

The SMS query obtained after disemvoweling and removing stop words is called processed SMS query S_p.

Calculation of the Keyword Score. In order to calculate the keyword score of a question q in Q-POSS, we find the number of words of the SMS query it

Algorithm 3. Generation of processed SMS query

Input : *Disemvoweled SMS tokens* $sv_1, \ldots, sv_n$,
Disemvoweled list of stop words L
Output: *Processed SMS Query* S_p

begin
  disemvowel(S);
  for $i \leftarrow 1$ **to** n **do**
    if $sv_i \in L$ **then**
      Remove sv_i from disemvowled SMS query;
    endif
  endfor
end

contains. We call the tokens (words) of the SMS query, keywords. The keyword score is a ratio of the number of keywords matched for each question to the number of keywords in the processed SMS query. Thus for a question q in Q,

$$KeywordScore(q) = \frac{No.\ of\ keywords\ in\ q}{No.\ of\ keywords\ in\ S_p} \ . \tag{6}$$

Thus, a disemvoweled question that contains all the keywords has a keyword score of 1.

3.7 Total Score

The total score for a question q in Q-POSS is calculated by adding its keyword score and similarity score and is kept along with q. Finally, the score(s) of the matching question(s) is converted to a ratio by dividing it with the score of Q*. Thus, if there is only one match its score is 1. Otherwise the scores are less than or equal to 1.

$$TotalScore(q) = KeywordScore(q) + SimilarityScore(q) \ . \tag{7}$$

The question with the maximum total score is returned as the match Q*, for the SMS query S.

4 Experiments and Results

This system has been tried on the given SMS queries and the provided FAQ dataset and has proved to be quite efficient. The system returns up to top 5 matching questions from the FAQ set with Q* as the best match for the SMS query S. The score of these matches are between 0 and 1 with the score of Q* being 1. Thus if only one match is found, its score is 1.

As many of the given queries were irrelevant and had no matching FAQ question, a minimum threshold total score was defined. If all the matching questions for an SMS query had a total score lower than this threshold score, then the

Table 2. Results for the system

Resuts	
In-domain Correct	396/728
Out-domain Correct	1940/2677
Total Score	69%
MRR	0.863

query was considered to be irrelevant and "NULL" was given as the output. This threshold score also helps in determining the number of matches for an SMS query if there are any.

4.1 Performance of the Various Components of the System

It was found that using the technique of calculating Inverse Document Frequency (IDF) for each word gave more accurate results than what we got without using it. This has been because of the fact that the queries consist of more informative words and thus words occurring in fewer questions should have a higher weight in comparison to common words.

The contribution of each component is given below.

Table 3. Performance of the system using the various components separately

	In-domain Correct(728)	Out-domain Correct(2677)
With only IDF	180	853
With only R/O	204	976
With only Levenshtein Distance	198	907
With only LCSR	235	1013

The performance of the system when one of the components for calculating the similarity were excluded one at a time are tabulated below.

Table 4. Performance of the system when one of the components were excluded

	In-domain Correct(728)	Out-domain Correct(2677)
Without IDF	214	1096
Without R/O	358	1720
Without Levenshtein Distance	291	1504
Without LCSR	347	1705

The method of storing the words occurring in all FAQ questions in a hash table arranged in alphabetical order proved to be much more time efficient than storing the words in a list. This was because by using this method, weights for less number of FAQ words w.r.t. to SMS tokens are calculated and also because a hash table is much more efficient than a list.

4.2 Comparision against Other Systems

This system was also tested against Python's fuzzy match and it proved to be much more efficient.

Table 5. Comparision with Python's fuzzy match

	In-domain Correct	Out-domain Correct
System	396/728	1940/2677
Pyhton's fuzzy match	152/728	12/2677

The system also fared well against other systems presented at FIRE 2011 and had the second best scores among all other systems presented by the participating teams.

The system scored 0.69 which was well above the median score for the task and the second highest score for the English monolingual task.

Table 6. Comparision with the median and high score

Median score for the task	System's score	High score for the task
0.14	0.69	0.83

The Mean Reciprocal Rank(MRR) of the system was 0.86 with the highest for the task being 0.89.

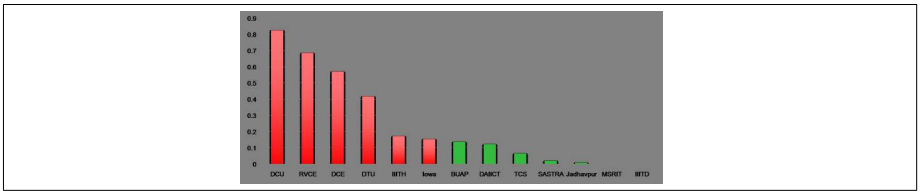


Fig. 1. Results for english monolingual task at FIRE 2011

5 Conclusion and Future Work

Thus, developing such an automated systems has been a challenge but this system gives a smart and efficient algorithm for answering FAQ's asked in SMS language. The results obtained for this system have been good.

As future work I would like to address the following issues:

- Using a synonym dictionary that can add similar meaning words to the variant list for an SMS token.
- Improving the accuracy of the system with respect to in-domain queries.

References

1. Kothari, G., Negi, S., Faruquie, T.A., Chakaravarthy, V.T., Subramaniam, L.V.: Sms based interface for faq retrieval. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, ACL 2009, Stroudsburg, PA, USA, vol. 2, pp. 852–860 (2009)
2. Ratcliff, J.W., Metzener, D.: Pattern Matching: The Gestalt Approach. Dr. Dobb's Journal, 46 (1988)
3. Gunawardena, T., Lokuhetti, M., Pathirana, N., Ragel, R., Deegalla, S.: An Automatic Answering System with Template Matching for Natural Language Questions. In: Proceedings of the 5th International Conference on Information and Automation for Sustainability (ICIAfS 2010), Colombo, Sri Lanka, pp. 353–358 (December 2010)
4. Contractor, D., Kothari, G., Faruquie, T.A., Subramaniam, L.V., Negi, S.: Handling noisy queries in cross language faq retrieval. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, Stroudsburg, PA, USA, pp. 87–96 (2010)
5. Kopparapu, S.K., Srivastava, A., Pande, A.: SMS based Natural Language Interface to Yellow Pages Directory. In: Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology, Singapore (2007)
6. Sneiders, E.: Automated FAQ Answering: Continued Experience with Shallow Language Understanding. In: Question Answering Systems. Papers from the 1999 AAAI Fall Symposium, North Falmouth, Massachusetts, USA, November 5-7, pp. 97–107. Technical Report FS-99-02. AAAI Press (1999)

Improving Accuracy of SMS Based FAQ Retrieval System

Anwar D. Shaikh*, Mukul Jain, Mukul Rawat, Rajiv Ratn Shah, and Manoj Kumar

Computer Engineering Department, Delhi Technological University, India
{anwardshaikh,mukuljain.dce,mukulrawat18869,
rajivratn}@gmail.com, mkg1109@rediffmail.com

Abstract. In the present scenario, we are looking for a better way to access information. Short Messaging Service (SMS) is one of the popularly used services that provide information access to the people having mobile phones. However, there are several challenges in order to process a SMS query automatically. Humans have the tendency to use abbreviations and shortcuts in their SMS. We call these inconsistencies as noise in the SMS. In this paper we present an improved version of SMS based FAQ retrieval system. We have mainly added three improvements to the previous system. They are (i) proximity score, (ii) length score and (iii) an answer matching system. Our experiments show that the accuracy of our system outperforms the accuracy of current state-of-the-art system. We demonstrate the effectiveness of our approach by considering many real-life FAQ-datasets from different domains (e.g. Agriculture, Bank, Health, Insurance and Telecom etc.).

Keywords: FAQ retrieval, FAQ system, Similarity score, Proximity score, Length score, SMS Query, SMS processing, IDF.

1 Introduction

Due to increased penetration of the internet, now information can be accessed at any place, any time from any device connected with internet. Huge amount of information is spread over the internet which require a good information retrieval technique to make information access anytime and anywhere to everyone. Therefore, making an information retrieval system convenient has become an interesting area of research. Nowadays, there are several resources through which users can access information such as internet, telephone lines, mobile phones, etc. With the rapid growth in mobile communication, mobile phone has become a common mode of communication for most of the people. The numbers of mobile users are growing at a very fast rate. In India alone, there are around 893 million mobile subscribers[1]. The popularity of mobile phones is due to its unmatched portability. This encourages different businesses or information providers to think upon implementing information services

* Corresponding author.

[1] http://www.trai.gov.in/annualreport/English_Front_Page.pdf

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 142–156, 2013.

based on mobile phones. SMS information service is one of the examples of mobile based information services. Existing SMS services such as service to access CBSE Exam Result requires user to type the message in some specific format. For example, to get the result of a particular student in CBSE examination, the user has to send a message CBSE-HS-XXXX (Where XXXX is the Roll number of the student)[2]. These are constraints to the users who generally feel it easy and intuitive to type a query in a "texting" language (i.e. abbreviations and the shortcuts).

Some businesses such as "ChaCha"[3] allow their users to make query through the SMSs without using any specific format. These queries, on the other hand, are handled by the human experts. However this approach provides users a kind of independence in writing the query yet this is not an efficient way because the system is limited to handle a small number of queries proportional to the number of human experts on the business side. This approach can be efficient if we have any system which can automatically handle query at the business side. A similar system based on SMS question answering system over a SMS interface was proposed in [1]. This system enabled user to type his/her question in SMS texting language. Such questions might contain short forms, abbreviations, spelling mistakes, phonetic spellings, transliterations etc. The system handled the noise by formulating the SMS query similarity over the FAQ database. This FAQ database was already provided to the system in the pre-processing stage. In this paper we present our approach based on proximity score, length score and an answer matching system. We have implemented this system for English & Hindi language, where the language of SMS and the language of FAQ are same. This system was developed as part of the event organized by Forum for Information Retrieval Evaluation (FIRE).

The rest of the paper is organized as follows. Section 2 describes the prior work done in this area. In Section 3 we describe our contributions, explaining in detail the various changes we made in the original system. In Section 4, we provide details about our implementations, experiments and the results. Finally we conclude the paper in Section 5.

2 Prior Work

An automated question answering system was designed by authors of [3]. Authors of [1] proposed an approach named SMS based FAQ retrieval. The proposed system was a SMS based question answering system in which user is allowed to enter the question in the SMS texting language. System was given a FAQ corpus containing all possible frequently asked questions. Noise in the SMS query was handled by formulating the query similarity over the FAQ database as a combinatorial search problem. System views the SMS as a sequence of tokens and each question in the FAQ corpus was viewed as a list of terms. The goal was to find a question from the FAQ corpus that matches best with the SMS query and return the answer of the selected question as a response of the input query. SMS string is bound to have

[2] SMS service- `http://results.icbse.com/cbse-result-class-10/`
[3] `http://www.chacha.com/`

misspellings and other distortions, which needed to be taken care of while performing the match. There is a pre-processing stage in which the system develops a domain dictionary and a synonym dictionary containing all the terms that are present in the FAQ corpus. For each term t in the dictionary and each token s_i in the SMS query, they defined a similarity measure α (t, s_i) that measures how closely the term t matches the SMS token s_i. They said the term t was a variant of s_i, if $\alpha(t, s_i) > 0$. They defined a weight function ω (t, s_i) by combining the similarity measure and the inverse document frequency (idf) of t in the corpus. Based on the weight function, they defined a scoring function for assigning a score to each question in the corpus Q. The score measures how closely the question matches the SMS string S.

$$Similarity\_Score(Q) = \sum_{i=1}^{n} \max_{\text{t}\in\text{Q and t}\sim s_i} \omega\,(\text{t}, s_i) \tag{1}$$

$$\text{Where} \qquad \omega\,(\text{t}, s_i) = \alpha\,(\text{t}, \text{s}_\text{i}) * \text{idf(t)} \tag{2}$$

3 Our Contribution

Our work is extension of the system described in [1]. Significant differences between these two systems are described below

1. FAQ Scoring function is modified to include Length score and Proximity score. Similarity Measure is used as mentioned in [1].
2. While pre-processing Answers of the FAQs are also considered for the creation of domain dictionary and synonym dictionary.
3. There are no changes in the process of list creation [1] and candidate set generation [1].
4. If there are many FAQs with similar score then we find the similarity between Answers of FAQs and the SMS query to break the tie.
5. Also, if there is no matching FAQ found then we try to match the Answers with the SMS query to get the result.

In order to increase the accuracy of SMS based FAQ retrieval we proposed few enhancements in evaluating the score of FAQ from candidate set. We proposed that accuracy can be improved by considering proximity of SMS query and FAQ tokens as well as by considering length of the matched tokens from the SMS query to the FAQ question under consideration. We have formalized that:

$$\begin{aligned}\text{Score(Q)} = {} & \text{W1} * \text{Similarity\_Score(Q,S)} \\ & + \text{W2} * \text{Proximity\_Score(Q,S)} \\ & - \text{W3} * \text{Length\_Score(Q,S)}\end{aligned} \tag{3}$$

Where Q is the FAQ question under consideration and S = {s_1, s_2, …, s_n} is the SMS query. W_1, W_2 and W_3 are real valued weights. Their values determine the portion of Similarity Score, Proximity Score and Length Score from the overall score of the

FAQ question. W_1 and W_2 are adjusted such that their sum is 1.0 (or 100%). We have given more than half portion to Similarity score. W_3 is assigned comparatively less value, as it tries to reduce the overall score if there are variations in the length of SMS and FAQ text. To calculate Similarity Score we have employed the methods proposed in [1]. Figure 1 shows various steps involved in our SMS based FAQ System.

Step-1: Pre-processing on SMS query.
Step-2: for each *token* in SMS
- Find ranked list of dictionary variants of *token.*

Step-3: Find the Candidate Set *C* using technique in [1].
Step-4: for each Q_i in *C*
-Find Similarity_Score using (1).
-Find Proximity_Score using (4).
-Find Length_Score using (6).
-Find total score using (3).
Step-5: Return FAQs having highest score as a result.

Fig. 1. Algorithms for SMS based FAQ System

Working of the system is depicted by an example in figure 2. After performing pre-processing on SMS query, Domain dictionary and Synonym dictionary are looked up to find the dictionary terms matching with the SMS token using Similarity measure described in [1], and a list of such terms in maintained, this step is referred as list creation. Based on the list, FAQs containing the terms present in the list are retrieved, all such FAQs are called Candidate set. For question in the candidate set, score of the FAQ is calculated using (3). Questions with the highest score are returned.

Where C_1, C_2, C_3, C_4 and C_5 are Candidate set of FAQ's having different dictionary variants of tokens of SMS.

C is the final Candidate set derived from C_1, C_2, C_3, C_4 and C_5

$Q1$, $Q2$ … Q_{n-1} and Q_n are the set of FAQ's from Candidate set *C*.

fun() is the module responsible for calculating the total score using (1), (4), (6) and (3).

e.g. *C* having the following FAQ's for given SMS query

C = {
Q_1: Which country won the most medals in Athens Olympics?
Q_2: Which is the first country who hosted modern Olympics?
Q_3: Which country will host 2016 Olympics?
Q_4: Which country won most medals in swimming in Olympics?
Q_5: Which country won gold medal in hockey in Beijing Olympics?
…….
}

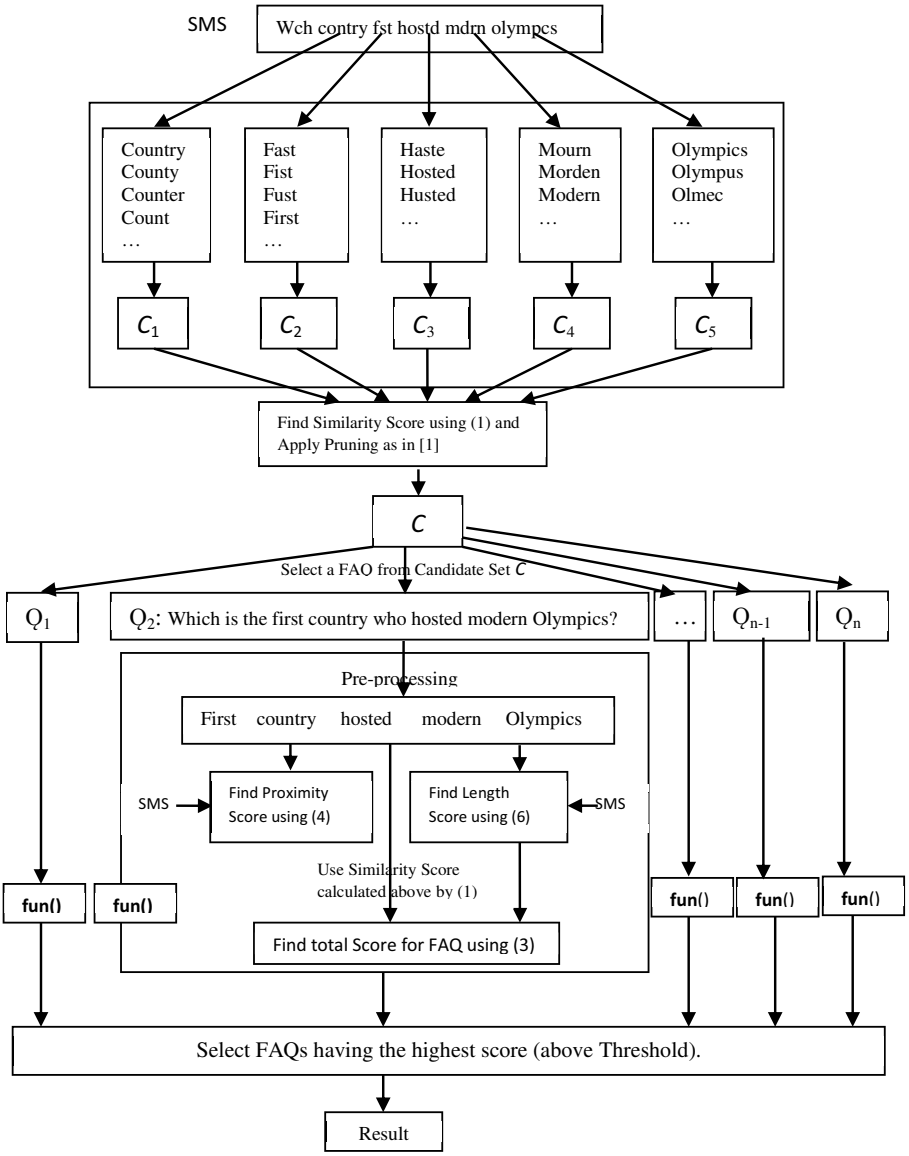


Fig. 2. Working of SMS based FAQ System

3.1 Proximity Score

In further steps of improving the accuracy of the system, we have introduced the concept of proximity search. The working of our proximity search technique is depicted with an example in figure 3 and figure 4.

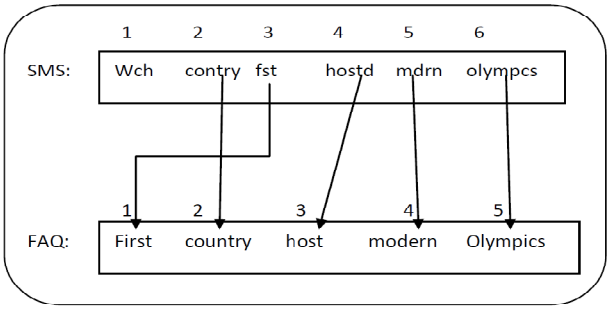


Fig. 3. Mapping of SMS tokens with FAQ

Relative position of words in a sentence plays an important role; it allows us to differentiate this sentence with various other possible sentences – which have same words but in different order. So while finding a best match, we must consider the proximity of words. In the proposed solution we do not check proximity of a token with all remaining tokens, but we only consider two consequent words. In proximity search process, we save the positions of the matched SMS tokens and FAQ tokens, stop words are removed before saving position of tokens. Based on the distance between two consecutive tokens in SMS text and FAQ the calculation of Proximity Score is done. The proximity score can be calculated by (4):

$$Proximity\_Score = \frac{matchedToken}{((distance\ +1) * totalFAQTokens)} \tag{4}$$

Where *totalFAQTokens* = number of tokens in FAQ and *matchedToken* = number of matched token of SMS in FAQ

$$distance = \sum_{k=0}^{n} \text{absolute difference between adjacent token pairs in SMS and corresponding pair in FAQ} \tag{5}$$

Where n = number of matched adjacent pairs in SMS

Figure 4 describes the calculation of Proximity Score with an example SMS and FAQ question. For calculating the value of distance we have taken only absolute value of distance as we believe that if two tokens were swapped their positions than in most of the cases the meaning of the SMS and FAQ question is unchanged. Unlike the Length Score, Proximity Score is always positive.

The algorithms to calculate Proximity_Score in depicted in figure 5. Input to the function is the position of the matched token in the SMS and the FAQ. The function first calculates the distance by (5), which is the absolute difference between the consecutive SMS and FAQ token positions. Based on the distance final proximity score is calculated as per (4).

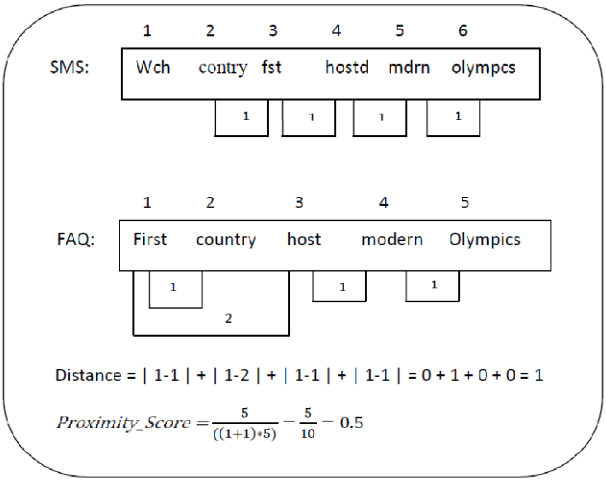


Fig. 4. Calculation of Proximity Score

```
/*
Input:
SMStokenPosition[] = Position of the SMS tokens which have matching FAQ token.
FAQtokenPosition[] = FAQ token position corresponding to SMS token.

eg. SMStokenPosition[] = {2,3,4,5,6};
    FAQtokenPosition[] = {2,1,3,4,5};

Output:
Proximity score.
*/
double calculateProximity(int SMStokenPosition[],int FAQtokenPosition[]) {

    int distance=0;
    for(int i=0;i<matchedToken-1;i++)
    {
        int currentDistance=Math.abs(
                                (SMStokenPosition[i+1] - SMStokenPosition[i])
                              - (FAQtokenPosition[i+1] - FAQtokenPosition[i])
                            );
        distance = distance + currentDistance;
    }

    proximityScore =1.0* matchedToken / ((distance +1)* totalFaqTokens ) ;

    return proximityScore;
};
```

Fig. 5. Function for Calculating Proximity_Score

3.2 Length Score

We further improved the accuracy of the system by considering the length of the unmatched SMS tokens in the FAQ under consideration. Length Score is defined as follows:

$$Length\_Score = \frac{totalFAQToken - matchedToken}{1 + totalSMSToken - matchedToken} \quad (6)$$

Where totalFAQToken = total number of Tokens in FAQ question, totalSMSToken = total number of Tokens in SMS, matchedToken =number of SMS which matched from tokens of FAQ question.

Since the Length Score is negative score (i.e. this score is subtracted from the overall score), so best Length Score is achieved when all the tokens of the FAQ question were matched from the all tokens in the SMS query. So in the best case Length Score is Zero (i.e nothing to be subtracted from the overall score).

For e.g. In figure (4) we can see that for question Q_2 all tokens matched with tokens in SMS. So this is the case of perfect matching and Length_Score can be calculated as follows.

totalFAQToken = 5, matchedToken = 5, totalSMSToken = 6

Length_Score = $\frac{5-5}{1+6-5} = \frac{0}{2} = 0$

Though we have used (6) only for calculations Length_Score in our system but we have identified a drawback of using this Length Score in case of a question having more number of tokens than SMS e.g. if there are 40 tokens in the FAQ and only 5 tokens in the SMS, in such cases the result would be always negative, even if there is match in FAQ for all SMS tokens.

We think that there can be two possible solutions to the above problem. The first solution is applicable when very few FAQ questions in FAQ database have more number of tokens. Solution to this problem is that rewrite the big FAQ question into the FAQ question having less number of tokens. For e.g.

Original FAQ Question: *"DTU offers various Tech courses. What are the Internship opportunities for M Tech students at DTU? Do all M Tech students get the Internship offer?"*

Corresponding Small Question: *"What are Internship opportunities for M Tech students at DTU?"*

The second solution is applicable when there are many big questions in FAQ database and rewriting them is not possible. In this case instead of subtracting the Length_Score (7), we add the Length Score in the overall score (8). In this particular case we think that modified Length Score (7) with modified total Score (8) can be used. A transition function can be designed in future for smooth transition between (6) and (7) to calculate the Length_Score based on the condition stated above.

$$Length\_Score = \frac{1 + totalSMSToken - matchedToken}{1 + totalFAQToken - matchedToken} \quad (7)$$

$$Score(Q) = W_1 * Similarity_{Score(Q,S)} + W_2 * Proximity_{Score(Q,S)} + W_3 * Length\_Score(Q,S) \quad (8)$$

Where totalFAQToken = total number of Tokens in FAQ question, totalSMSToken = total number of Tokens in SMS, matchedToken = number of SMS which matched from tokens of FAQ question.

In the best case Length Score would be 1 when all the tokens in FAQ were matched by all tokens in SMS.

3.3 Matching with Answers

In further steps of improving the accuracy of the system, we have introduced the idea that along with matching of SMS query with the FAQ-Question - we can match the SMS with the FAQ-Answer also, because some of the words in the SMS might be present in the FAQ-Answer but not in the FAQ-Question. Matching with answers is considered in both the cases mentioned below:

- There is more than one FAQ-question having the closest matching with the SMS query.
- There is no matching FAQ-question found.

Note: In pre-processing step we have also considered FAQ answers for creation of Domain Dictionary. For e.g. Let there is a question-answer in the FAQ database as follows:

FAQ: "*What are the different insurance schemes?"*

Answer: "*LIC, LIC Jivan Saral, LIC Jivan Tarang, LIC Plus, Bajaj Allianz, ICICI Lombard etc. are different insurance schemes."*

SMS: *"wht r difrnt LIC scems?"*

Suppose "LIC" word is not present in any other FAQ question then the earlier technique will not able to answer this question correctly but our technique will able to answer this question correctly as in this case we will search for "LIC" token in FAQ answer too and will get the correct result.

Also, if there are more than one FAQ's are having same score, then we find the similarity between the answer and the SMS query. The FAQ Answers having more matching SMS tokens will be considered as the best match.

4 Implementation and Experiments

4.1 Implementation

4.1.1 FAQ Pre-processing

As described in [1] we do the pre-processing of the FAQ corpus. In pre-processing a domain dictionary and synonym dictionary is created based on FAQ corpus, we have considered questions as well as answers for the creation of domain and synonym

dictionaries. All questions and answers in FAQ corpus are indexed for fast lookup during FAQ retrieval process. While creating domain dictionary stop words are removed from the FAQ, as the stop words are not important and are generally not used by SMS users.

4.1.2 SMS Pre-processing

Stop words are also removed from the SMS input if present. Also if there are numbers present in the SMS then they are converted into their corresponding string format and these strings are used for calculating similarity over the token.

e.g. **2**day is converted to **two**day.

Occurrences of single characters in the SMS are also removed, because generally single characters are not much important in deciding the meaning of the SMS.

4.1.3 Tools Used for Implementation

We have used Lucene[4] for indexing the tokens of FAQ. Wordnet[5] English was used to find synonyms of different English words while creating synonym dictionary.

4.1.4 Language Specific Changes

There were some changes done to make the system applicable for Hindi language. We have used Hindi Wordnet[6] API 1.2 while creation of synonym dictionary. The list of stop words for Hindi language was created and used in the pre-processing of the SMS and FAQs. The similarity threshold for list creation and score threshold for selecting the correct FAQ were changed and made suitable for Hindi language.

4.2 Experiments

4.2.1 SMS Based FAQ Retrieval Task

Experiments were conducted for the fulfilment of the tasks organized by Forum for Information Retrieval Evaluation (FIRE[7]) in year 2011. There were various subtasks of SMS based FAQ Retrieval Task, out of which we have participated in Mono-Lingual FAQ Retrieval (same language FAQ Retrieval) for English and Hindi language. In this subtask the language of input SMS and the FAQ corpus was same. So, the goal was to find best matching questions from the mono-lingual collection of FAQs for a given SMS[8].

4.2.2 Dataset

The FAQ and SMS dataset was provided by FIRE. FAQs were collected from online resources and from government and private sector. This dataset contained data from

[4] http://www.lucene.apache.org
[5] http://www.wordnet.princeton.edu
[6] http://www.cfilt.iitb.ac.in/wordnet/webhwn
[7] http://www.isical.ac.in/~clia/
[8] http://www.isical.ac.in/~fire/faq-retrieval/faq-retrieval.html

various domains – Agriculture, Banking, Career, General Knowledge, Health, Insurance, Online railway reservation, Sports, Telecom and Tourism.

Table 1 show the number of FAQs and the *In-domain* and *Out-domain* SMS queries used in the experiments. SMS queries for which there is a matching question in the FAQ corpus then its In-Domain SMS query, other queries are called Out-Domain SMS queries.

Table 1. Number of FAQs and SMS Queries

Language	*FAQs*	*In-domain SMS*	*Out-domain SMS*	*Total SMS*
Hindi	1994	200	124	324
English	7251	728	2677	3405

4.2.3 FIRE 2011- SMS Based FAQ Retrieval Task Results

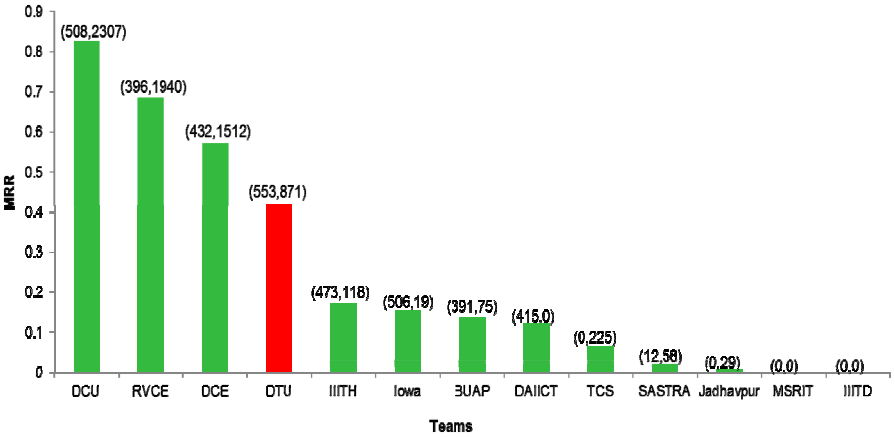


Fig. 6. Results of English-monolingual task

13 teams from various universities participated in the English Monolingual task. The results are shown in the figure-6; Vertical axis represents Mean Reciprocal Rank (MRR). Performance of our team is marked in red colour in the graph. Our result for this task is shown in table 2 –

Table 2. English-Monolingual task result

Task	*In-domain Correct*	*Out-domain Correct*	*MRR*
English-Monolingual	553 / 704	871 / 2701	0.830

7 teams participated in Hindi Monolingual task; the results are shown in the figure-7. Performance of our team is marked in red colour in the graph. Table 3 shows our result in detail-

Table 3. Hindi-Monolingual task result

Task	*In-domain Correct*	*Out-domain Correct*	*MRR*
Hindi-Monolingual	198 / 200	3 / 124	0.99

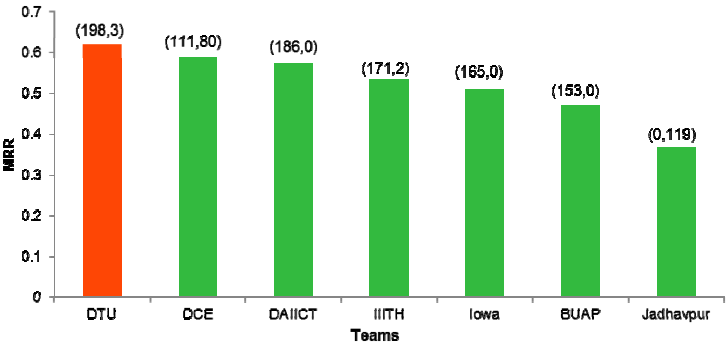


Fig. 7. Results of Hindi-monolingual task

As you can observe the out domain results in case of English and Hindi task is very low because the FAQ score (3) threshold was not properly selected. So, we have repeated this experiment with different threshold. The improved results are shown in section 4.2.4.

4.2.4 Effect of Various Proposed Techniques on the Result

As explained above, the matching between FAQ and SMS is performed based on three factors- Similarity, proximity and length. We have conducted experiments to evaluate the correctness of the systems based on these three factors in four different possible ways. As the similarity score is the base of the matching process, we have considered similarity score in all experiments. In first experiments only Similarity alone is considered for matching, in second – Similarity along with Proximity is considered, in third experiment Similarity and Length are considered and in fourth experiment all three factors are considered for matching. Table 4 and table 5 shows the result of experiments conducted for Hindi language and English language respectively. MRR indicates mean reciprocal rank. Those questions, which had score greater than a particular threshold, were considered for matching. The experiments were conducted with the same threshold for all experiments. The threshold used for these experiments was different and more accurate than the threshold used in FIRE 2011 task.

Table 4. Results of Hindi FAQ retrieval experiments

	In-domain Correct	*Out-domain Correct*	*MRR*
Similarity	197	6	0.9905
Similarity & Proximity	197	22	0.9905
Similarity & Length	197	97	0.9916
Similarity & Length & Proximity	**198**	**118**	**0.9949**

We can see from our experimental results in table 4 and 5 that the accuracy of our system is much better than the accuracy of current state-of-the-art system. It also shows that we are achieving the best accuracy when we are considering similarity, length and proximity score in calculation of overall score of a FAQ.

Table 5. Results of English FAQ retrieval experiments

	In-domain Correct	*Out-domain Correct*	*MRR*
Similarity	519	234	0.7529
Similarity & Proximity	520	393	0.7568
Similarity & Length	538	1981	0.8877
Similarity & Length & Proximity	**521**	**2281**	**0.9041**

These results are shown in the form of graph in Figure 8 and 9:

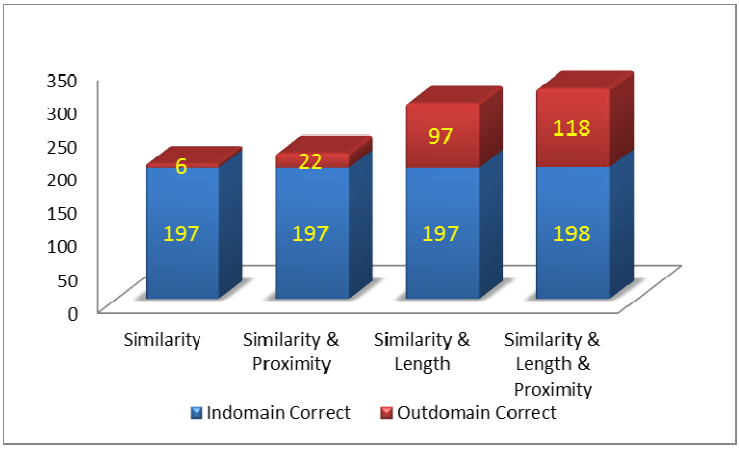


Fig. 8. Result of Hindi FAQ retrieval task

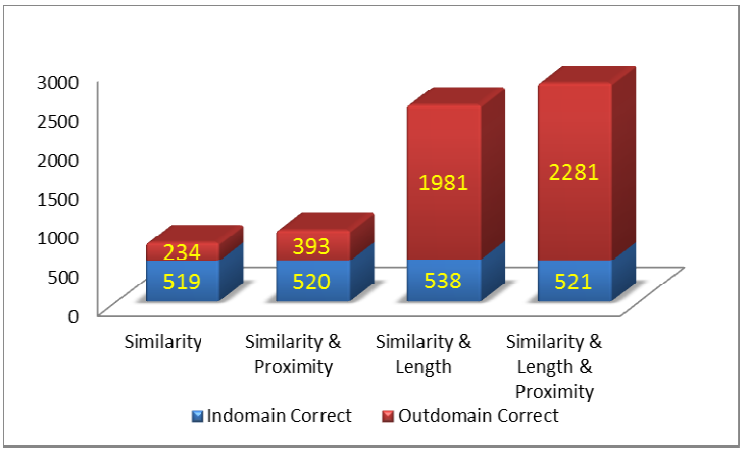


Fig. 9. Result of English FAQ retrieval task

The results show that the Proximity factor does not improve the in domain result too much but improved the out domain result. As per our observations, the reason behind the improvement of the out domain result is that – in some FAQs there are tokens much more similar to those in SMS- compared to the other FAQs, but they are located at positions far different that the expected/original SMS token position. In such cases, the proximity score tries to eliminate these FAQs from the result.

Length factor improved the result by a large extent as compared to the result obtained by considering Proximity factor. And after combining the effect of Similarity, Length and proximity the results are better as compared to the previous experiments.

5 Conclusion and Future Work

SMS based Question Answering systems may become one of the efficient, convenient and cheapest way to extract information. SMS based FAQ retrieval system proposed by [1] is an automatic question answering system that handles the noise in the SMS query by formulating the query similarity over the FAQ database. In this paper, we present three techniques (i) Proximity Score, (ii) Length Score and (iii) an answer matching system in order to improve accuracy the SMS based FAQ retrieval system. We have demonstrated with experiments that after applying our proposed techniques, the accuracy of our system outperforms the accuracy of the current state-of-the-art system.

In future this system can be extended for FAQ retrieval using spoken queries, instead of SMS queries. One such approach is described in [4].

Acknowledgement. We provide our sincere thanks to Dr. L. Venkata Subramaniam for his continuous support and encouragement to complete this SMS based Question Answering systems.

References

1. Govind, K., Sumit, N., Tanveer, A.F., Venkatesan, T.C., Subramaniam, L.V.: SMS based interface for FAQ retrieval. In: Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, Singapore, pp. 852–860 (2009)
2. Contractor, D., Kothari, G., Faruquie, T.A., Subramaniam, L.V., Negi, S.: Handling Noisy Queries in Cross Language FAQ Retrieval. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, October 9-11, pp. 87–96. MIT, Massachusetts (2010)
3. Sneiders, E.: Automated question answering using question templates that cover the conceptual model of the database. In: Andersson, B., Bergholtz, M., Johannesson, P. (eds.) NLDB 2002. LNCS, vol. 2553, pp. 235–239. Springer, Heidelberg (2002)
4. Chang, E., Seide, F., Meng, H.M., Chen, Z., Shi, Y., Li, Y.-C.: A system for spoken query information retrieval on mobile devices. Proceedings of the IEEE Transactions on Speech and Audio Processing, 531–541 (November 2002)

Mapping SMSes to Plain Text FAQs

Arpit Gupta

Delhi College of Engineering, New Delhi, India
Arpitg1991@gmail.com

Abstract. The present day users of the public information systems such as passenger query systems and patient query systems in a hospital prefer to query the system by way of SMS. In this paper, we have addressed the problem of mapping the user queries on government portals in the form of SMSes to their equivalent plain text frequently asked questions (FAQs) stored in the database. Lucene indexer has been used to index the FAQs. The score for a query SMS is determined by counting the words in the SMS at hand that have high similarity score. Experiments show high success rate on the unseen **SMSes.**

1 Introduction

Noisy text is widely used in SMS communication, tweets, chat on the Internet, posts to user groups, and email communication. The present day users of the public information systems such as passenger query systems and patient query systems in a hospital prefer to query the system by way of SMS. The problem of mapping the SMSes to their plain text equivalent forms falls in the realm of noisy text translation. Conventionally, statistical machine translation techniques have been used for noisy text translation. Normalization of noisy text has recently attracted the attention of researchers[4] [5].

In this paper we have addressed the problem of mapping the SMSes to their equivalent plain text FAQs on government portals. These messages relate to different subjects such as health, employment, and train schedule. Given an SMS, we have attempted to solve the aforementioned problem of mapping the SMSes to their equivalent plain text FAQs in two steps:

1) Text normalization comprising of removal of stop words and generation of phonetic equivalents of SMSes and FAQs.
2) Extracting a list of plain text FAQs that contain words similar to it: We compute a similarity score between the SMS at hand and the plain text FAQs, shortlisted above, and report the best matched FAQ.

As the category under which the query is posted is often known, we focused on search within a category.

2 Related Work

SMS language has been studied by several researchers, for example, Lorna Mphahlele and Mashamaite [2005], relate the use of SMS to learning skills and point out that

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 157–162, 2013.

certain abbreviations that may be quite acceptable in a domain, may be entirely incomprehensible to those outside of that domain. In a study Fairon and Paumier presented a corpus of 30,000 messages they collected through a project in Belgium named "Faites don de vos SMS à la science" (Give your SMS to Science). They found wide variations in word spellings and forms. There were intentional and unintentional deviations in spellings and punctuations. For example, while some messages did not include spaces, others mixed lower and uppercase letters in unconventional ways.

Kobus et al. [2010] attribute the variability in SMS text to several factors such as phonetic type of writing (e.g., rite for write), consonantal writing (dropping of vowels), un-conventional use of numbers and letters to encode phonetic values for their spellings (e.g., 4mula, c, and bcas for formula, see, and because respectively), unconventional abbreviations (e.g., affair for as far as I can recall and lol for laugh out loud), and disregard for case distinction, tense agreement, and punctuation; notwithstanding unintentional typo errors. They study three metaphors, namely spell checking, translation, and speech recognition. The spell checking metaphor considers each token as noisy version of some correctly formed word and attempts to correct it. In the translation metaphor the text normalization is viewed as purely machine translation task. In the speech recognition metaphor, the SMS is viewed as alphabetic approximation of phonetic forms. Using statistical machine translation for SMS normalization, followed by decoding using ASR like system, they achieved 11% error rate on a test set of about 3000 messages. Kaufman [2010] uses the aforementioned two phase approach to normalize twitter messages. Contractor et al. [5] adopt a two- step approach for cleansing of noisy text. Given a noisy sentence, a weighted list of possible clean tokens for each noisy token are obtained. Subsequently, the clean sentence is obtained by maximizing the product of the weighted lists and the language model scores.

3 Our Approach

As a first step we indexed all FAQs using Lucene. Lucene is a public domain search utility that builds an inverted index on the given document set. Subsequently, when a search is made on some key words, it searches this index. We considered two schemes to index the documents (index only the FAQs, and index both FAQs and their answers). Preliminary tests showed that accuracy reduced when answers were also indexed along with FAQs. Therefore in further experiments, we indexed only the FAQs.

3.1 Text Normalization

The database consisted of 7251 FAQs and their answers, and 1071 SMS queries. Out of 1071 queries 371 were out of domain i.e., not relating to any category. Stop word list [1] originally contained 429 words. It was used to remove stop words from FAQs as well as SMSes. It was noted that SMS text still contained possible candidates for

stop words as the original list [1] does not take into account noisy stop words. We give below some examples of stop-words in noisy form and the corresponding words in plain text:

abt (about), wer (were), wich (which), wh (who), wht (what), alng (along), wat (what), shw (show)

In order to remove stop words both from FAQs and SMSes, noisy words such as those listed above were appended to list [1].

In the next step, Metaphone library was used to generate phonetic equivalents. Each FAQ document was broken into tokens and phonetic key was generated for each token using Metaphone library with length of key, set to MAX_POSSIBLE. Similarly, SMS queries were also tokenized and converted to phonetic equivalent tokens. Metaphone codes use the 16 consonant symbols 0BFHJKLMNPRSTWXY. The '0' represents "th" (as an ASCII approximation of Θ), 'X' represents "sh" or "ch", and the others represent their usual English pronunciations. A set of rules as described in [6] are applied.

3.2 FAQ Retrieval

```
Algorithm Build_Index

1. Transform input FAQ into sequence of tokens
2. Remove stop words from token stream obtained in
   step 1
3. Use output of step 2 for parsing to generate pho-
   netic equivalents
4. Add FAQ to index

Algorithm Search_Index

1. Transform input FAQ into sequence of tokens
2. Remove stop words from token stream obtained in
   step 1
3. Use output of step 2 for parsing to generate pho-
   netic equivalents
4. Search for each token in Index let score for token_j
   in doc_i be x
5. Select top 5 documents, taking care if the score <
   No of vdocuments*c, do not report the document
```

Given SMS text was divided into tokens. Each token was then queried separately in index. Top 1000 matching documents were extracted and given the score (tf-idf) as provided by Lucene. Score for a document (FAQ) was computed by summing over scores obtained for all tokens. Top 5 documents were then selected. The documents

that had score less than a threshold value *No. of Tokens * C* (where *C* was determined experimentally) were discarded. Given an SMS query, if all the documents had score less than *No. of Tokens * C*, it was marked *out of domain* (no corresponding FAQ exists in any category). Applying this approach with various values of C (in the range 0-8) to all the 1071 queries, we computed the precision and recall values. Table 1 summarizes this information.

Table 1. Precision and recall for different values of C

C	Precision	Recall
0	0.65	1
1	0.74	0.904
2	0.85	0.366
3	0.81	0.089
4	0.82	0.032
5	0.85	0.008
6	0.66	0.002
7	0.5	0.001
8	1	0.001

Further experimentation was carried out with refinement in values of C and 1.15 was selected as the best value considering results for both *in domain* and *out of domain* queries.

Motivated by the observation that when an SMS query is matched against the specific category only, the accuracy shoots up to about 90%, we considered the following heuristic to boost the performance: multiply the score of each document by a factor depending on the number of documents belonging to its category that were returned in top *n* matches. However, the experimentation with different values of *n* only worsened the results and was therefore dropped. Subsequent attempts at multi-class classification also failed.

3.3 Experimental Results

Table 2 shows the performance of the proposed method when SMS was queried against FAQs of the same category.

Table 2. Percentage of correctly mapped SMS queries

Category	No of FAQs	No of SMSes	Correctly mapped	%age Correct
Agriculture	230	40	32	80
Career	939	56	46	82.14

Table 2. (Continued)

GK	716	83	75	90.36
Health	433	38	31	81.57
Insurance	991	57	45	78.94
Indian	268	64	53	82.81
Sports	684	51	43	84.31
Telecom-munication	103	79	69	87.34
Tourism	134	43	40	93.02
Bank	565	35	22	62.85
Loan	514	31	26	83.87
Internal	56	25	18	72
Personality	122	10	5	50
Recipes	876	41	39	95.12
VISA	511	19	13	68.42
WEB	109	29	19	65.51

4 Conclusion and Scope of Future Work

Given the problem of mapping an SMS to its equivalent plain text FAQ, the proposed approach yielded reasonable accuracy when SMS was queried against FAQs of the same category. Although our attempt to apply the standard multi-class classification methods directly to this problem failed, but one can consider modifying classification techniques to suit the specific nature of this problem.

Acknowledgement. The author would like to thank Dr. L. V. Subramaniam of IBM, IRL for constant encouragement and Prof. Sharma Chakravarthy of the University of Texas at Arlington and Ms. Shikha Gupta of University of Delhi for useful discussions.

References

1. Mphahlele, M.L., Mashamaite, K.: The impact of short message service (SMS) language on language proficiency of learners and the SMS dictionaries: a Challenge for educators and lexicographers. In: IADIS International Conference on Mobile Learning (2005)

2. Kobus, C., Yvon, F., Damnati, G.: Normalizing SMS: are two metaphors better than one? In: COLING 2008: Proceedings of the 22nd International Conference on Computational Linguistics, pp. 441–448. Association for Computational Linguistics, Morristown (2008)
3. Fairon, C., Paumier, S.: A translated corpus of 30,000 French SMS. In: Proceedings of LREC 2006, Genoa, Italy (2006)
4. Weiser, S., Cougnon, L.-A., Watrin, P.: Temporal Expressions Extraction in SMS messages. In: Proceedings of the Workshop on Information Extraction and Knowledge Acquisition, Hissar, Bulgaria, September 16, pp. 41–44 (2011)
5. Contractor, D., Fauquie, T.A., Subramaniam, L.V.: Unsupervised Cleansing of Noisy Text. In: COLING 2010: Proceedings of the 23rd International Conference on Computational Linguistics. Association of Computational Linguistics, Stroudsburg (2010)
6. http://www.morfoedro.it/doc.php?n=222&lang=en

SMS Normalization for FAQ Retrieval

Khushboo Singhal, Gaurav Arora, Smita Kumari, and Prasenjit Majumder

IR-Lab, DA-IICT, India
khushboo_singhal,gaurav_arora@daiict.ac.in,
smita_kumari,p_majumder@daiict.ac.in
http://irlab.daiict.ac.in

Abstract. This paper reports a system for retrieving similar Frequently-Asked-Questions (FAQ) when queries are through Short-message-Service (SMS). The system was developed to participate in FIRE 2011 SMS-based FAQ Retrieval track (Monolingual). SMS contains various User Improvisations and Typographical errors. Proposed approach use approximate string matching (ASM) techniques to normalize SMS query with minimum linguistic resources. MRR obtained for English, Hindi and Malayalam are 0.85, 0.93 and 0.92 respectively.

1 Introduction

In Indian subcontinent, mobiles are more used than laptops. Due to Rapid increase in mobile penetration, accessing information through SMS, have attracted attention of research communities. SMS can be used as a user interaction mode in many applications, like password verification, voting, SMS remote control, SMS alert, banking, etc. SMS being faster, cheap and convenient are very popular. With a diversified language base in India, the usage is multilingual and multi-script as well. More than half of the SMS use Roman scripts to represent Indian languages in electronic format. Information access with the help of SMS with low linguistic resources is a challenging task.

SMS-based FAQ Retrieval aims to retrieve relevant answers from FAQ documents. SMS queries vary a lot from traditional information retrieval queries. One major challenge in this task is to handle noisiness in the query. A related discussion on this is presented later in this paper.

Researchers projected SMS as a new language or spoken language to solve the problem. Aw et al., 2006 [1] used statistical phrase based machine translation and Kobus et al., 2008 [4] and Gouws et al., 2011 [3] proposed to look at the problem as automatic speech recognition (ASR). It is shown that machine translation helps in forming understandable sentences if learning data contain all the possible variations, and ASR helps in finding best word boundary and phonetic variations.

Also, Kobus et al., 2008 [4] concluded that machine translation in a better, but resource hungry measure and ASR gives very less improvement in result.

Contractor et al., 2010 [2] concentrated on learning techniques over structural analysis, they tried statistical machine translation technique based on source paradigm of communication language model, which is less resource hungry by generating model on clean text from newspaper and web. Since such text is outside domain of FAQ, thereby unnecessary terms from general corpus have affected significantly the performance.

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 163–174, 2013.

Subramanian et al., and Kothari et al., 2009 [10,5] classified different kinds of noise, based on factors like syntactic and structural details of word. Techniques to handle structural noise are also discussed, which leads to basic foundation of our approach.

Projecting this as traditional information retrieval problem, we considered SMS as a query S_q and each FAQ as document D_f. A rank list is generated and top N documents D_r are retrieved. This study mainly focuses on normalizing SMS S_n (noisy) to S_q, which is a clean version.

Variations in SMS can be of three types, first and most important being improvisations done by users. Each user preferance to represent a word may be different and that leads to large number of variants (improvisations) making the problem more challenging. Variation for tomorrow can be "tomm", "2mro", "tomro", etc. Other being typos by users due to difficulty in typing on keypads. In this category errors are due to the input interface like qwerty keypad, touchscreen, normal mobile keypad. As for example "2nro" while writing "2mro" for "tomorrow" as "mno" are on same key in multi-tap mobile keyboards. In case of touchscreen one may slip the actual key to be touched and end up with one of the adjacent keys causing error. Typically these errors are wrong substitution, deletion or insertion of characters. Typographical errors can be removed by modeling the input method. The most challenging error is the combination of both the categories. Table 1 describes different user improvisations in detail and figure 1 shows SMS noise taxonomy.

Table 1. User Improvisation with example

Type	**Description**	**Example**
User Improvisation		
Abbreviations	Shortened form of word or phrase	"2mrw" "2day" "et. al" "lol" "brb"
Deletion of Characters	removal of alphabets, so that word remains same phonetically and easy to understand	"rng" for "range" "nse" for "noise"
Deletion of words	Words which don't change the meaning of sentence.	"Reached college" for "I have reached college" "Doing gud" for "I am doing good"
Truncation	Cut-off trailing characters from a long word under assumption "Context is enough to understand full word"	"Rising of sun is a univ truth" -"univ" for "universal" "He is topper of local univ" - "univ" for "university"

User improvisation are the most prevalent and have large room for improvement, hence the focus of our study is to normalize the user improvement in a language independent way. In this paper, we have used minimal linguistic resources and achieved total score of 0.84 in English, 0.929 in Hindi and 0.92 in Malayalam.

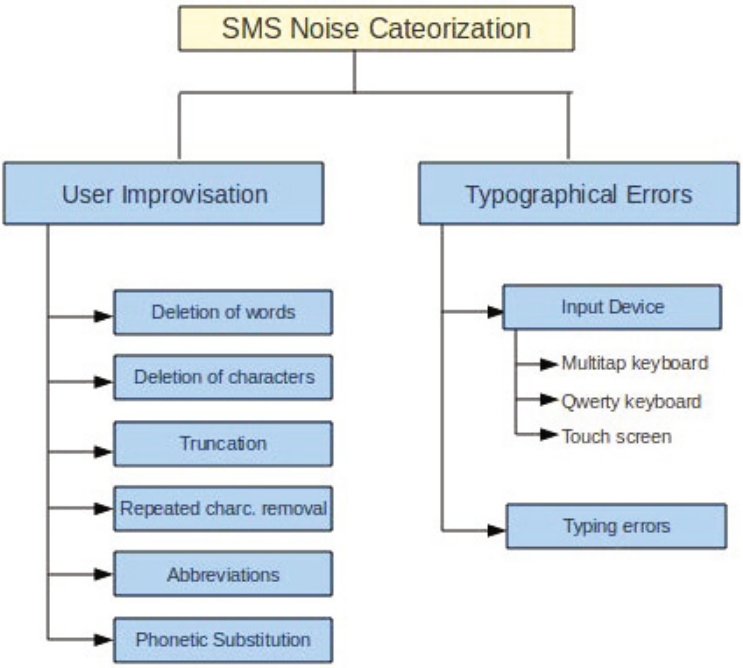


Fig. 1. SMS noise taxonomy

The rest of paper is organized as follows. Section 2 explains the Dataset. Section 3 and 4 explains proposed approach and Implementation details. Section 5 contains experiments and results tried on FIRE 2011 SMS-based FAQ Retrieval Dataset.

2 Data Set Details

FIRE 2011 SMS-based FAQ Retrieval Track consists of three tasks:

- Mono-Lingual FAQ Retrieval
- Cross-lingual FAQ Retrieval
- Multi-lingual FAQ Retrieval

Out of which we participated only in Mono-Lingual FAQ Retrieval task. This task further was divided into three subtask which consist of English monolingual, Hindi monolingual and Malayalam monolingual Retrieval. We Submitted runs for each of these sub tasks.

Statistics of FIRE 2011 SMS-based FAQ Retrieval Track corpus statistics can be obtained from track overview paper. Figure 2 and 3 is an example FAQ document in English and Malayalam respectively, and figure 4 shows example SMS queries.

```
<FAQS>
<FAQ>
<FAQID>ENG_AGRICULTURE_232</FAQID>
<DOMAIN>ENG_AGRICULTURE</DOMAIN>
<QUESTION>Will the be making its historical agriculture materials available to the
private sector If a private entity is given the material, will it then become their
property, exclusively How long will the be keeping its old records/forecasts</QUESTION>
<ANSWER> The plans to archive and make available all agricultural records to the public
and private sector. Also, forecasting techniques and software will be made available to
the private sector. Because these materials were developed with taxpayer dollars, no
private entity can "own" them.
Environmental Assessments and Agriculture:
Key Questions and Answers
</ANSWER>
</FAQ>
```

Fig. 2. English FAQ document

```
<FAQS>
<FAQ>
<FAQID>MALAYALAM_GK_1</FAQID>
<DOMAIN>GK</DOMAIN>
<QUESTION>ആരായിരുന്നു പാകിസ്ഥാന്റെ ആദ്യത്തെ പ്രധാന മന്ത്രി?</QUESTION>
<ANSWER>
മുഹമ്മദ് അലി ജിന്നയാൽ നിയമിക്കപ്പെട്ട ലിയാഖത്ത് അലി ഖാൻ ആയിരുന്നു ആദ്യത്തെ പാക്കിസ്ഥാൻ പ്രധാന മന്ത്രി.
</ANSWER>
</FAQ>
```

Fig. 3. Malayalam FAQ document

```
<SMS>
<SMS_QUERY_ID>HINDI_SMS_QUERY_I44</SMS_QUERY_ID>
<SMS_TEXT>कन सा देश 2008 के ओलिंपिक खेल की मेजबनी देश है?</SMS_TEXT>
</SMS>
<SMS>
<SMS_QUERY_ID>ENG_SMS_QUERY_I1</SMS_QUERY_ID>
<SMS_TEXT>whats nece 2 change name on pport after a marriage</SMS_TEXT>
</SMS>
<SMS>
<SMS_QUERY_ID>ENG_SMS_QUERY_I16</SMS_QUERY_ID>
<SMS_TEXT>1st Cheif Justce ov Sup Court</SMS_TEXT>
</SMS>
<SMS>
<SMS_QUERY_ID>MALAYALAM_202</SMS_QUERY_ID>
<SMS_TEXT>വേറെ ആളുകൾക്കായി ടിക്കറ്റ് ബുക്ക് ചെയ്യാമോ</SMS_TEXT>
</SMS>
<SMS>
<SMS_QUERY_ID>MALAYALAM_242</SMS_QUERY_ID>
<SMS_TEXT>വിമാനം പറത്തിയ ആദ്യ വ്യക്തികൾ</SMS_TEXT>
</SMS>
```

Fig. 4. SMS Query Example

3 Proposed Approach

An SMS $S_n = \{t_1, t_2, t_3, ...t_k\}$ is tokenized. A lexicon $L = \{l_1, l_2, l_3, ..., l_n\}$ consisting of words is FAQ Data is created. Each SMS token is then compared with all lexicon words $(t_i, l_1), (t_i, l_2)..(t_i, l_n)$ and a score is assigned $\eta(t_i, l_j)$, $\forall t_i \in S_n$ and $l_j \in L$. For each token t_i a set of lexicon words L_{ti}, is selected, where $L_{ti} \subset L$ and $\forall l_j \in L_{ti}$, $\eta(t_i, l_j) > \eta_{threshold}$. This gives a set of possible normalizations of a particular word. This forms normalized SMS $S_q = L_{t1} \bigcup L_{t2} \bigcup, ..., \bigcup L_{tk}$. This SMS is considered as a query and is fired on a traditional information retrieval system to retrieve relevant FAQs.

Figure 5 is an overview of the whole system. We designed our experiments using FAQ lexicons as correct words list to test our hypothesis, general corpus introduce domain unrelated term and thereby hinder precision and recall [2].

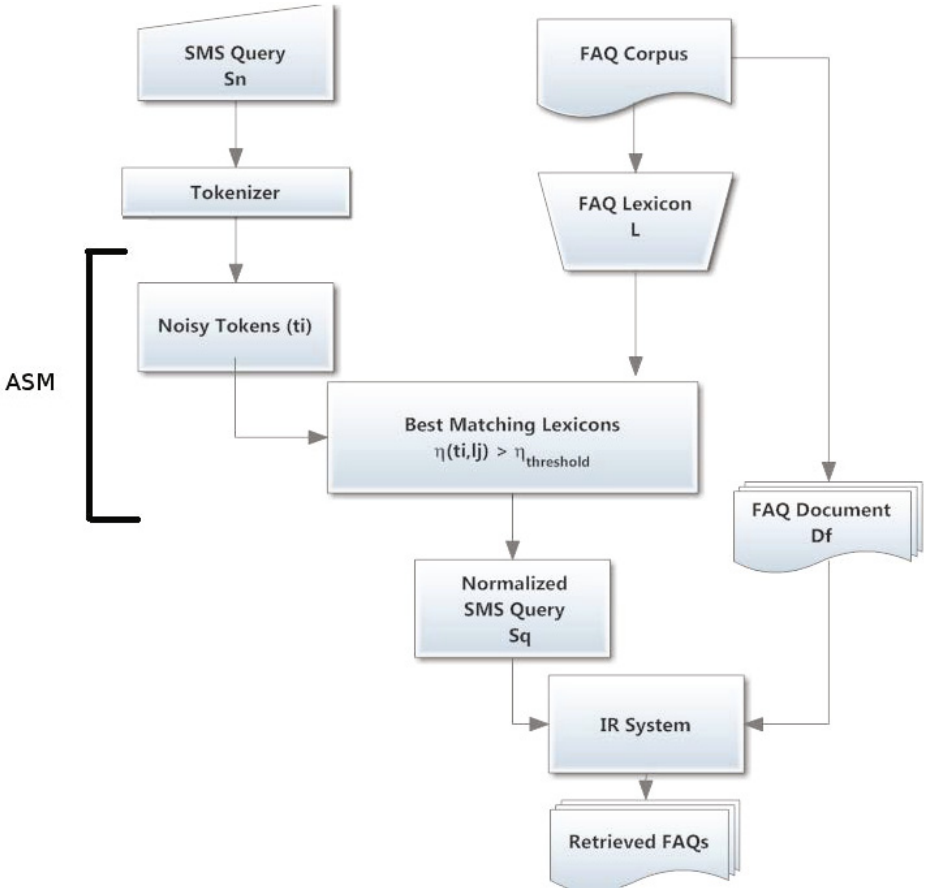


Fig. 5. Block diagram of the approach

4 Implementation Details

4.1 Preprocessing

Preprocessing step includes SMS tokenization, removal of stop-words, removal of single characters and replacement of abbreviations in SMS query S_n. This is handled by "Tokenizer" block as shown in figure 5

Removal of stop words from SMS queries can handle "deletion of word", where we normally delete the stop words from sentence to make it short. Removal of single character words as they are mostly stop-words, for example, 'y' for 'why', 'u' for 'you', 'r' for 'are' and '4' for 'for'. This style of writing is called Rebus style [8]. Unabridged form of abbreviation is added to SMS query along with abbreviation as FAQs may contain either abbreviated word or its full version.

Few stop-words in SMS may also present in their short form but, we did not try to remove them as they can be an important candidates to some important words like "gd" can become "good" or "god" where "god" cannot be ignored while searching related FAQs.

4.2 Approximate String Matching (ASM)

In order to find best matches for a noisy SMS token, we combined approximate string matching (ASM) techniques.

Consonant Skeleton. Most of the improvisations are formed by removing vowels and vowel signs in a word as suggested by Prochasson et al.,2007 [8]. Vowels are removed from both SMS word and lexicon and a score is assigned if their consonant skeletons match.

$$CSR(w1, w2) = w1.length/w2.length \tag{1}$$

The intention behind giving weightage according to length ratio between noisy token and lexicon is that more weightage is given to words that is more eligible match as can be seen from figure 6

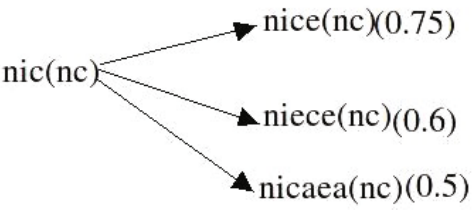


Fig. 6. CSR ratio example

Truncation Ratio. One important form of user improvisation is truncation. Truncation is handled as described in Algorithm 1 given below. Table 2, explains the heuristic behind selecting ratio of token and lexicon length. It is clear from table 2 that "probability" gets more similarity score for "probab" than that for "pro".

Algorithm *1 Truncation Ratio*(t_i, l_j)
1. $TR \leftarrow 0$
2. $len \leftarrow length(t_i)$
3. **if** $len \geq 3$ and $len \leq length(l_j)$ **then**
4. **if** substring$(i_j, 0, len) == t_i$ **then**
5. $TR \leftarrow len/length(l_j)$
6. **return** TR

Table 2. Truncation Ratio Examples

Noisy Token	Lexicon word	TSR
probab	probability	0.54(6/11)
pro	probability	0.27(3/11)

Levenshtein Distance (Edit Distance). Edit distance is used to handle typographical errors, phonetic substitutions, repeated character removal errors. Edit distance of SMS token is calculated with each lexicon token. Less the distance between two words, more probable are they to match.

We combined these three approximate string matching techniques as shown in equation 2.

$$\begin{aligned} \eta(t_i, l_j) &= \alpha(UI) + (1-\alpha)(ER) \text{ , } \textit{if } t_i \textit{ and } l_i \textit{ share first character} \\ &= 0 \qquad\qquad\qquad\qquad \textit{otherwise} \end{aligned} \tag{2}$$

$$UI = (CSR(t_i, l_j) + TR(t_i, l_j)) \tag{3}$$

$$ER = \tfrac{1}{LD(t_i, l_j)}) \tag{4}$$

where, $t_i \in S_n$ and $l_j \in L$
UI stands for user improvisation
ER stands for other errors
CSR is Consonant-Skeleton-Match-Ratio
TR is Truncation-Match-Ratio
$(0 \leq \alpha \leq 1)$
LD is Levenshtein Distance[6].

α is introduced in above equation to provide weightage factor between user improvisation (which mostly are detected by consonant skeleton and truncation errors) and other

errors (mainly typographical errors). Other errors equation will also score user improvisation problems like "deletion of characters" and "phonetic substitution". Zero score is assigned if first character of t_i and l_i don't match. Assumption is user will always type first character correctly. This assumption is not proper, but save lots of processing time.

Hypothesis is SMS language is more prone to user improvisation rather typographical errors so given weightage accordingly can improve the system performance. User improvisations are detected using CSR and TR and Typographical errors using LD.

4.3 FAQ Retrieval and Identifying Out-of-Domain (OOD) Queries

Once Normalized SMS S_q is formed after pre-processing and ASM step, Terrier[1] [7] is used to retrieve FAQ. We have used BM25 [9] retrieval model. Here, each FAQ Question is considered as a Document D_f and similarity is calculated between S_q and D_f, $\forall D_f \in$ FAQ Data and best matching FAQs D_r are retrieved. Two indexes were created, first Q where only questions of FAQ (i.e. Only questions were considered as a part of FAQ document D_f) were indexed and second Q+A where both question and answers of FAQ were indexed.

To select OOD SMS queries (declaration of queries that cannot be answered by FAQ data) a threshold is set on BM25 similarity score $\theta_{threshold}$, this threshold is calculated empirically as can be seen in figure 7.

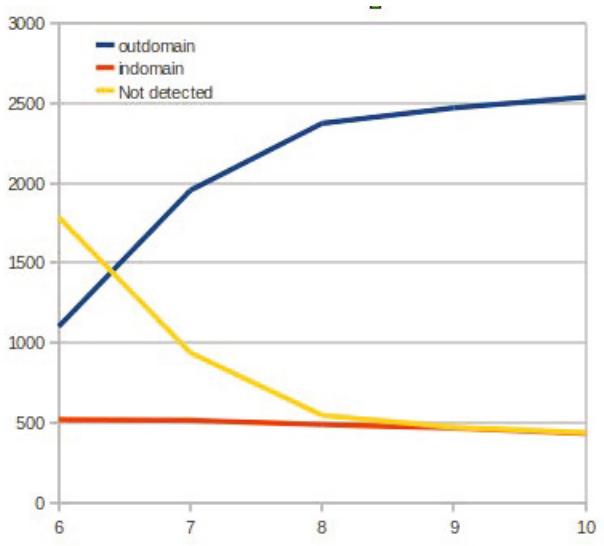


Fig. 7. BM25 threshold for maximizing in-domain and minimizing out-domain retrieval

[1] http://terrier.org/

5 Experiments and Results

We evaluated the system performance for English, Hindi and Malayalam. The only difference between these systems is the vowel list (Vowel list used in ASM for CSR Ratio is language dependent).

5.1 Selection of $\eta_{threshold}$ and α

$\eta_{threshold}$ is inversely proportional to number of lexicon words added to SMS query. $\eta_{threshold}$ required to be set such that it maximizes the selection of chunks with right peer token from lexicon L. For the same we tested our system on different values of η from 0.1 to 0.9 and selected the minimum value of η which gives maximum result. As can be seen from figure 8(a), the least η at which system attains maximum performance is 0.3.

Further tuning was required to decide weightage between user improvisation and typographical errors. This is done by changing α in equation 2. Performance of system was evaluated by changing values of α from 0.3 to 0.8 as shown in figure 8(b).

It is clear from figure 8(b) that increasing weightage to user improvisation gives better results, but saturates starting with 0.7 and starts dripping down after 0.8. For example, if SMS query contains "trans" matching words could be "trains" , "trams" and "transfer" but more weightage would be given to transfer as it is caused due to truncation error.

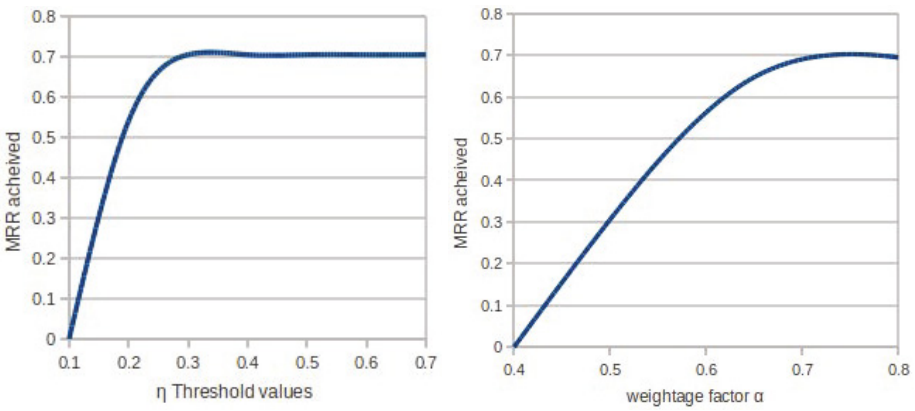


Fig. 8. Maximum performance at different $\eta_{threshold}$ and Performance with change in α

Figure 9 gives the performance of the system with 0.3 $\eta_{threshold}$ and 0.7 α on English SMS queries when documents are considered consisting of Q[2] and Q+A[3].

[2] Q signifies FAQ documents D_f consists of only Questions.
[3] Q+A signifies FAQ documents D_f consists of both Questions and Answers.

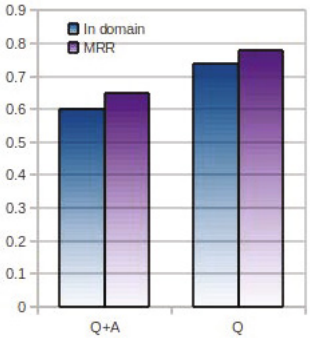


Fig. 9. Results retrieved using Q and Q+A index

It is observed that system performs better when we consider only Questions in FAQ. As while processing SMS queries to normalize user improvisation few general words matching or closer to user improvisation are added to final query token. For example, for user improvisation "gd" words like "good" and "god" were inserted in query.

Our hypothesis is, answers contain a lot of general conversation words and a wide range of concept not necessarily representing the context of questions (out-of-context words), for example answer to question related to "nation" or "honesty" may have words like "good peoples" and "good followers" of nation, which can get matched with question having "god", so searching only questions boosted the result.

5.2 Selection of OOD Queries

As explained in section 4.3 to select OOD queries we set $\theta_{threshold}$ on BM25 Similarity score of SMS queries and retrieved FAQs.

Table 3 shows the results achieved on test collection for English, Hindi and Malayalam Queries with $\eta_{threshold}$ = 0.3 and α = 0.7 and $\theta_{threshold}$ = 0.8. This performance achieved is more than the highest reported in SMS-FAQ Competition in FIRE Conference 2011 for English and Hindi as can be seen from table 4.

Table 3. Performance of our approach on SMS-based FAQ Retrieval Test Collection

Language	Total SMS	In domain	Out of domain	MRR	Total Score
English	3405	0.706	0.723	0.725	0.849
Hindi	324	0.93	0.927	0.985	0.929
Malayalam	50	0.92	NA	0.937	0.92

Table 4. Comparison of our approach to best and median result obtained across all the systems

Language	Best Score	Our Score	Median	Our Score
English	0.83	0.849 (+2.29%)	0.14	0.849 (+506.42%)
Hindi	0.62	0.929 (+47.46%)	0.53	0.929(+75.28%)
Malayalam	0.94	0.92 (-2.12%)	0.90	0.92(+2.22%)

5.3 Failure Analysis

We did some analysis to find out the reason for missing SMS. For this we checked manually all the SMS which we were not able to retrieve. We found that our algorithm missed few noise types. Table 5 gives details about these with example and corresponding query.

Table 5. Various other problems in SMS Retrieval - which our approach cant solve

Type	Example	Description	Query
Ambiguous Abbreviations	"wc", "hw"	this system replaced "welcome" for "wc" and "how" for "hw" which caused to miss wordcup and homework	QNO:23 Who was the winner in 83 cricket wc
Word boundary detection	"interlib" , "united-states"	system cant divide the words in 2 and matched it with internal	QNO:94 whos not allwed to use t Interlib Loan?
FAQ Lexicon error	"amt"	"amt" being present in corpus got treated as correct word	QNO:79 hau much amt is the rt one for health insur
too general queries	such queries need some context or domain information to get good results	-	QNO:214 when my stuff comes how will i no?
first character wrong	"dorver"	"driver" gets substituted instead "forever"	-

6 Conclusion and Future Work

The results obtained imply that the technique employed here is simple, but gives better results compare to other techniques used. This technique normalize improvisations in SMS queries with minimum linguistic resource (list of vowels). The study of various values of α suggest that user improvisations are more prone than typographical errors in SMS queries.

Future scope of this study may include study of typographical errors that are related with different kind of mobile keyboards. This will help to differentiate more clearly between typographical errors and user improvisations. As in example "comrse" where

'mno' being on same keypad cause substitution of 'm' in place of 'o'. So this is more prone to typographical error than User improvisation of word 'commerce'.

Also, currently we added all the possible matches directly to SMS query in place of noisy token just restricted by a threshold. Strategy to find the single best match among these is required. As we saw in example of truncation error in table 1 the truncated form "univ" has different match in different contexts. Therefore, digging out context information of SMS query for normalization can be helpful.

References

1. Aw, A., Zhang, M., Xiao, J., Su, J.: A phrase-based statistical model for SMS text normalization. In: Proceedings of the COLING/ACL on Main Conference Poster Sessions, pp. 33–40. Association for Computational Linguistics (July 2006)
2. Contractor, D., Faruquie, T.A., Subramaniam, L.V.: Unsupervised cleansing of noisy text. In: Huang, C.R., Jurafsky, D. (eds.) COLING (Posters), pp. 189–196. Chinese Information Processing Society of China (2010)
3. Gouws, S., Hovy, D., Metzler, D., Rey, M.: Unsupervised Mining of Lexical Variants from Noisy Text. English, pp. 82–90 (2011)
4. Kobus, C., Marzin, P.: F-Lannion: Normalizing SMS: are two metaphors better than one? Computational Linguistics, 441–448 (August 2008)
5. Kothari, G., Negi, S., Faruquie, T.A., Chakaravarthy, V.T., Subramaniam, L.V.: Sms based interface for faq retrieval. In: Su, K.Y., Su, J., Wiebe, J. (eds.) ACL/AFNLP, pp. 852–860. The Association for Computer Linguistics (2009)
6. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10(8), 707–710 (1966)
7. Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., Johnson, D.: Terrier information retrieval platform. In: Losada, D.E., Fernández-Luna, J.M. (eds.) ECIR 2005. LNCS, vol. 3408, pp. 517–519. Springer, Heidelberg (2005)
8. Prochasson, E., Viard-Gaudin, C., Morin, E.: Language models for handwritten short message services. In: ICDAR, pp. 83–87. IEEE Computer Society (2007)
9. Robertson, S., Spärck Jones, K.: Simple, proven approaches to text retrieval. Tech. Rep. UCAM-CL-TR-356, University of Cambridge, Computer Laboratory (December 1994)
10. Subramaniam, L.V., Roy, S., Faruquie, T.A., Negi, S.: A survey of types of text noise and techniques to handle noisy text. In: Lopresti, D.P., Roy, S., Schulz, K.U., Subramaniam, L.V. (eds.) AND, pp. 115–122. ACM (2009)

Two Models for the SMS-Based FAQ Retrieval Task of FIRE 2011*

Darnes Vilariño, David Pinto, Saul León, Esteban Castillo, and Mireya Tovar

Faculty of Computer Science
Benemérita Universidad Autónoma de Puebla, Mexico
{darnes,dpinto,mtovar}@cs.buap.mx, saul.ls@live.com, ecjbuap@gmail.com

Abstract. In this paper we propose a normalization model in order to standardize the terms used in SMS. For this purpose, we use a statistical bilingual dictionary calculated on the basis of the IBM-4 model for determining the best translation for a given SMS term. In order to compare our proposal with another method of document retrieval, we have submitted to the FIRE 2011 competition forum a second run which was obtained by using a probabilistic information retrieval model which employes the same statistical dictionaries used by our normalization method.

The obtained results show that the normalization model greatly improves the performance of the probabilistic one. An interesting finding indicates that the Malayalam language is the one that seems to be better written in the SMS context, in comparison with the English and Hindi languages which were also evaluated in the framework of the monolingual, crosslingual and multilingual environments.

1 Introduction

In recent years there has been a tremendous growth of services based on mobile devices which has been a consequence of the overcrowding of devices of this kind. One of the communication medium used in this type of devices is the SMS, which basically is a short text message containing no more than 160 characters. The cost for sending SMS and the easy access to the purchase of mobile devices have made instant messaging emerge as the preferred communication medium just after spoken media and e-mails.

From the point of view of information retrieval systems it is important to exploit the potential number of users using such kind of devices and communication services. In this paper, we aim to face the problem of searching answers of FAQs (Frequently Asked Questions) when one SMS is used as query for the information retrieval system. It is well known that millions of Instant Messaging (IM) users, including SMS, generate e-content in a language that adheres to conventional grammar, or punctuation rules and usual pronunciation. The words are intentionally compressed, using abbreviations, acronyms, phonetic transliterations and even neologisms. Additionally, it is important to note that the reduced

* This project has been partially supported by projects CONACYT #106625, VIEP #VIAD-ING12-I y #PIAD-ING12-I.

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 175–183, 2013.

space of the mobile device screen and the size of the keyboard leads very often messages to contain inadvertent typographical errors and spelling mistakes. This fact, emphasizes the complexity of constructing an information retrieval system which considers SMS queries, even when the corpus is conformed by questions whose answers are well documented (FAQs).

In this research work we present one information retrieval system which considers the monolingual, crosslingual and multilingual approaches for three different languages: English, Hindi and Malayalam. The dataset was obtained from FIRE, a well-known evaluation forum for information retrieval systems[1]. In this paper, we evaluate two different approaches (employing the same bilingual statistical dictionaries in both cases) in the task of SMS-based FAQ retrieval by considering the aforementioned dataset for the experiments. In the first case, we have proposed to normalize the queries (SMS) by using the most frequent translation calculated from a training corpus, whereas the second approach considers a straightforward probabilistic search engine which integrates in a single step the process of searching and translating the query. It is important to note, that even in the case of the monolingual task, we are considering to solve the problem by means of machine translation techniques, assuming that both, the queries and the target documents are written in a different language. Therefore, the statistical bilingual dictionaries are in fact, a kind of association thesaurus which probabilistically determines which sentence is the "translation" (correct way of writing) for a given SMS word.

The obtained results show that we may significantly improve the retrieval results by normalizing the queries (SMS) before applying the typical information retrieval procedures. We consider that given the great success of instant messaging in the world, these findings would be of high benefit for all users of mobile devices, in particular in India because we have studied three languages widely used by close of 400 millions of indian mobile device users.

The rest of this paper is structured as follows. In Section 2 we outline the state of the art of FAQs retrieval. Section 3 introduce the normalization approach evaluated in this paper. Section 4 describes the bilingual probabilistic model used as a reference for comparison with the normalization approach. In Section 5 the experimental results are presented and discussed. Finally, in Section 6 the conclusions and further work are given.

2 Previous Works

Harksoo Kim has studied the problem of FAQ retrieval as it may be seen in [1], where he presented a trustly way of recovering FAQs by using a clustering of previous query logs. In fact, he also improved this first approach in [2] by employing latent semantic analysis, and also by using latent term weights [3]. In [4] it is proposed the use of machine translation techniques for the alignment of questions and answers of a FAQ corpus, with the aim of constructing a bilingual statistical dictionary which is further used for expanding the queries

[1] http://www.isical.ac.in/~clia/

introduced in an information retrieval system. The experiments were performed with the English and Chinese languages, and we consider interesting the idea of using machine translation techniques for generating a terminological association model between the question and answer vocabulary, which may be further used in order to estimate the probability of an answer given a query (set of terms). We have extended this approach by considering the same type of alignment, but in this case between the SMS and the FAQ questions. In [5] it is presented an approach for domain specific FAQ retrieval based on a concept named "independent aspects". This concept basically consists of extracting terms and relationships by employing WordNet and Hownet which are then used in a mixture-based probabilistic model with the aim of analyzing queries and query-answer pairs by means of independent aspects. It is worth noting that any of the presented approaches use SMS as an input query, but they use queries written in normal text, which is a very important difference with respect to the experiments carried out in this research paper.

An excelent work for SMS normalization may be found in [6]. They prepared a training corpus of 5000 SMS aligned with reference messages manually prepared by two persons which are then introduced to a phrase-based statistical method to normalize SMS messages. The obtained results are very interesting despite they do not apply their method to any particular task such as information retrieval. Their findings include the observation of a great difference between SMS and normal written texts due to the particular written style of SMS writers and the high frequency of non-standarized terms which very often occur in short versions, shortened, truncated or phonetically transliterated.

Even so, there exist some works facing the problem of FAQ retrieval based on SMS queries. In [7] and [8], for instance, a web service for retrieving Hindi FAQs considering SMS as queries. This propossal consists on formulate the similarity criterion of the search process as a combinatorial problem in which the search space is conformed of all the different combinations for the vocabulary of the query terms and their N best translations. Unfortunately, the corpus used in these experiments is not available and, therefore, it is not possible to use it for comparison with our approach.

3 Normalization of Short Texts

In this paper we propose a normalization approach to be applied in the framework of the SMS-based FAQ retrieval (monolingual, crosslingual and multilingual). As we have previously mentioned, this problem is highly relevant due to the particular terminology used in the SMS which necessarily requires of a normalization process. For the experiments carried out in the monolingual task, we have considered that both, the SMS and the FAQs are written in the same language (English, Hindi or Malayalam). In the case of the crosslingual task, the SMS is written in English, whereas the FAQs are written in Hindi. Finally, in the case of the multilingual task, both the SMS and the FAQs may be written in any of the three different languages.

Formally, the problem faced in this paper may be formulated as follows: Let ζ be the set of questions in the FAQ corpus, and $S = s_1 s_2 \ldots s_n$ be an SMS. Both, the SMS and each question $q \in \zeta$, are seen as a sequence of terms. The aim is to find the question q^* belonging to the corpus ζ that obtains the maximum degree of similarity with respect to the SMS S. For this purpose, we have used a single model of information retrieval based on set theory, in particular, by using intersection of term sets. We have proposed a normalization model for queries (SMS) which is described in the following paragraphs.

The SMS messages contain a high number of terms that do not appear in regular dictionaries. In SMS written context, it is common to introduce new terminology that very often is associated with contractions of the original words or with an ortographical representation of their corresponding phonetic representation. It is worth noting that such terminology changes according to the different age ranges of the users, since young people (teenagers) used to employ phonetic representations which may be derived from the fact that they do not domain enough vocabulary of their native language. Some examples of the terms used and their corresponding interpretation are shown in Table 1.

Table 1. Examples of terminology used in SMS

SMS message	Interpretation
10q	Thank you
tna	temporarily not available
afk	away from keyboard
L8	late
LOL	Laugh Out Loud
26Y4U	Too sexy for you
@WRK	at work
⋮	⋮

For the problem we are interested in, we must consider that SMS may contain not only those terms that occur very frequently, but also those that are not so frequent. Let us take for example the following phrase taken from the test corpus: "whr can i find info abt pesticide estb reg and rep", which may be interpreted as "where can i find information about pesticide establishment registration and reporting". Therefore, the idea of mantaining a generic dictionary of frequently used terms on the SMS context may be unuseful on narrow domains.

With the aim of determining the correct "meaning" of terms appearing in an SMS query, we propose to substitute each query term with the closest translation offered by a bilingual statistical dictionary. In order to construct this dictionary, we have used the Giza++ tool[2] which allowed us to calculate the IBM-4 model by using a training corpus conformed of a set of aligned phrases (one SMS with its corresponding FAQ). In total, we have constructed 13 different statistical bilingual dictionaries as it is shown in Table 2.

[2] `http://code.google.com/p/giza-pp/`

Table 2. Distribution of the different statistical bilingual dictionaries

Task	Source language (SMS)	Target language (Question-FAQ)
Monolingual	English	English
	Hindi	Hindi
	Malayalam	Malayalam
Crosslingual	English	Hindi
Multilingual	English	English
	English	Hindi
	English	Malayalam
	Hindi	Hindi
	Hindi	English
	Hindi	Malayalam
	Malayalam	Malayalam
	Malayalam	English
	Malayalam	Hindi

In order to calculate the similarity among the SMS terms and each one of the FAQ questions (P_{FAQ}), we used the Jaccard similarity coefficient as it is shown in Eq. (1).

$$Similarity(SMS, P_{FAQ}) = \frac{\mid SMS \cap P_{FAQ} \mid}{\mid SMS \cup P_{FAQ} \mid} \quad (1)$$

The pseudocode associated to the FAQ retrieval is shown in the Algorithm 1. This algorithm receives as input the set of SMS (topics o search queries), the target dataset or FAQs (ζ) and the statistical bilingual dictionary (ϕ) used in the SMS normalization process. Each topic (SMS) is normalized according to the criterion explained in Section 3. The normalized message is then compared with each FAQ question by means of the Jaccard similarity measure. All those values greater than the minimum of the N best similarity values (which we will known as *threshold*), are returned in the answer set (P_{FAQ}).

4 Probabilistic Model

We have used a probabilistic model which considers both, the translation and the search process in a single step. It basically uses a statistical bilingual dictionary for calculating the probability of each topic (search query) to be associated to a target document. The training phase is done by applying the IBM-4 model to a set of pairs of query vs. relevant documents. The obtained statistical dictionary is used in conjunction with the set of target documents in order to show the most relevant ones given a query which is written in a different language of that of the target documents (see [9]).

Formally, let x be a query text in a certain (source) language, and let y_1, $y_2, \ldots, y_n$ be a collection n of documents written in a different (target) language. Given a number $k < n$, we are interested in finding the k most relevant

Algorithm 1. SMS-based FAQ retrieval

Input: Topics: $SMS = \{sms_1, sms_2, \ldots, sms_n\}$
Input: FAQs: $\zeta = \{q_1, \cdots, q_n\}$
Input: Statistical bilingual dictionary: $\phi = p(t_{SMS}, t_q)$
Output: N best answers for each SMS: Q

```
foreach sms_i ∈ SMS do
    smsN_i = Normalize(sms_i, φ);
    Q[i] ← {∅};
    foreach P_FAQ ∈ ζ do
        if Similarity(smsN_i, P_FAQ) > Threshold then
            Q[i] = Q[i] ∪ {P_FAQ, Similarity(smsN_i, P_FAQ)};
        end
    end
    if |Q[i]| > N then
        Q[i] = NBestValues(Q[i]);
    end
end
return Q
```

documents with respect to the source query x. For this purpose, it is employed a probabilistic approach in which the k most relevant documents are computed as those most probable given x, i.e.,

$$\hat{S}_k(x) = \underset{\substack{S \subset \{y_1,\ldots,y_n\} \\ |S|=k}}{\arg\max} \min_{y \in S} p(y \mid x) \tag{2}$$

Actually, $p(y \mid x)$ is modelled by using the IBM-4 model.

5 Experimental Results

In this section we present and discuss the results obtained for each task evaluated. First, we present a general description of the training and test corpora used in the experiments. Secondly, we provide a discussion attempting to find evidence of a relationship between the values obtained in the evaluation and the peculiarity of each language considered in the experiments.

5.1 Training Dataset

The training corpora are conformed by aligned sentences (SMS with its corresponding FAQ). The different FAQ corpora is divided according to the language. In Table 3 we may observe the number of FAQs associated to each language in the training corpora.

The SMS used as topics are distributed according to their language and task, as may be seen in Table 4.

Table 3. Distribution of FAQs in the training corpora

Language	# de FAQs
English	7,251
Hindi	1,994
Malayalam	681

Table 4. Distribution of SMS in the training corpora

Task	Language	# of SMS
Monolingual	English	1,071
	Hindi	230
	Malayalam	140
Crosslingual	English	472
Multilingual	English	460
	Hindi	230
	Malayalam	80

5.2 Test Dataset

In the test dataset, we use the same number of FAQs (see Table 3), however, the number of SMS changes. The number of SMS used in the test phase (evaluation) are distributed according to their language and task, and may be seen in Table 5.

Table 5. Distribution of SMS in the test corpora

Task	Language	# of SMS
Monolingual	English	3,405
	Hindi	324
	Malayalam	50
Crosslingual	English	3,405
Multilingual	English	3,405
	Hindi	324
	Malayalam	50

5.3 Evaluation Results

In Table 6 we may see the Mean Reciprocal Rank (MRR) obtained for each run submitted to the SMS-based FAQ retrieval task. The normalization model is identified by the "NORM" tag, whereas the probabilistic one is identified by the "PROB" tag. As it may be seen, in the multilingual task, we did not send a probabilistic run because we observed a very low performance when we used the training dataset.

The most interesting result is that we have greatly outperformed the probabilistic model by using the normalization one. After a simple analysis of the

Table 6. Results obtained at the SMS-based FAQ retrieval task

Task	Run	In Domain correct	Out of Domain correct	Mean Reciprocal Rank (MRR)
Crosslingual	NORM	1/37 (0.027)	163/3368 (0.048)	0.0398
Crosslingual	PROB	0/37 (0.000)	170/3368 (0.050)	0
Monolingual-English	NORM	385/704 (0.546)	75/2701 (0.027)	0.6006
Monolingual-English	PROB	1/704 (0.001)	140/2701 (0.051)	0.0025
Monolingual-Hindi	NORM	153/200 (0.765)	0/124 (0.000)	0.8070
Monolingual-Hindi	PROB	0/200 (0.000)	5/124 (0.040)	0.0051
Monolingual-Malayalam	NORM	39/50 (0.780)	0/0 (NaN)	0.8304
Monolingual-Malayalam	PROB	1/50 (0.020)	0/0 (NaN)	0.0714
Multlingual-English	NORM	353/704 (0.501)	25/2701 (0.009)	0.5360
Multilingual-Hindi	NORM	113/200 (0.565)	0/124 (0.000)	0.6163
Multilingual-Malayalam	NORM	32/50 (0.640)	0/0 (NaN)	0.7037

obtained results we may conclude that the use of the maximum probability of translation instead of the product of the probable translation is the best solution for this particular task.

A quite interesting finding is that the best MRR obtained is when the Malayalam language is used. We consider that this result comes from the fact that the people is using a very low number of terms out of the vocabulary, even when they are writing SMS-type messages. It seems that people talking in Malayalam consistently use standard vocabulary of that language. This fact should be an expected result, because this language may be considered as a variation of the Tamil which is used in education and administration. Actually, Malayalam has borrowed thousands of nouns, hundreds of verbs and some indeclinable part of speechs from sanscrit, which is language assumed to be used by aristocracy and academics, as it happened with latin in the European countries.

The obtained results may be considered competitive in the case monolingual and multilingual, but in the case of the crosslingual evaluation we obtained a very low performance. We are still analyzing the reason of the above mentioned behaviour.

6 Conclusions and Further Work

We have presented two different models of information retrieval for the SMS-based FAQ retrieval task of FIRE 2011. On the one hand we attempted a probabilistic information retrieval model which tackles the multilingual task by means of bilingual statistical dictionaries constructed on the basis of pairs SMS and their original text. On the other hand, we normalized the SMS terms by means of the best translations of the same bilingual statistical dictionaries. The difference between both approaches is basically the way in which we associate a SMS term, i.e., in the first case we use a summatory of all the possible translations, whereas the second case only uses the best translation.

It is interesting to note that the normalization based model is seen to be more effective than the other one. This indicates that the translation model

works better in translating SMS to normal forms than in translating SMS to the documents to be retrieved.

As future work we would like to improve the information retrieval model used. We should replace the Jaccard similarity measure by other one that take into account the frequency of the terms among the documents to be compared. Finally, we are considering to improve the performance of the monolingual task by using a phonetic codification of both, the SMS and FAQs.

References

1. Kim, H., Seo, J.: High-performance faq retrieval using an automatic clustering method of query logs. Inf. Process. Manage. 42, 650–661 (2006)
2. Kim, H., Lee, H., Seo, J.: A reliable faq retrieval system using a query log classification technique based on latent semantic analysis. Inf. Process. Manage. 43, 420–430 (2007)
3. Kim, H., Seo, J.: Cluster-based faq retrieval using latent term weights. IEEE Intelligent Systems 23, 58–65 (2008)
4. Riezler, S., Vasserman, A., Tsochantaridis, I., Mittal, V., Liu, Y.: Statistical machine translation for query expansion in answer retrieval. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 464–471. Association for Computational Linguistics, Prague (2007)
5. Wu, C.H., Yeh, J.F., Chen, M.J.: Domain-specific faq retrieval using independent aspects. ACM Transactions on Asian Language Information Processing (TALIP) 4, 1–17 (2005)
6. Aw, A., Zhang, M., Xiao, J., Su, J.: A phrase-based statistical model for sms text normalization. In: Proceedings of the COLING/ACL on Main Conference Poster Sessions, COLING-ACL 2006, pp. 33–40. Association for Computational Linguistics, Stroudsburg (2006)
7. Kothari, G., Negi, S., Faruquie, T.A., Chakaravarthy, V.T., Subramaniam, L.V.: SMS based interface for FAQ retrieval. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, ACL-IJCNLP 2009, vol. 2, pp. 852–860. Association for Computational Linguistics, Morristown (2009)
8. Contractor, D., Kothari, G., Faruquie, T.A., Subramaniam, L.V., Negi, S.: Handling noisy queries in cross language faq retrieval. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, pp. 87–96. Association for Computational Linguistics, Stroudsburg (2010)
9. Pinto, D., Civera, J., Barrón-Cedeño, A., Juan, A., Rosso, P.: A statistical approach to crosslingual natural language tasks. J. Algorithms 64, 51–60 (2009)

SMS Normalisation, Retrieval and Out-of-Domain Detection Approaches for SMS-Based FAQ Retrieval

Deirdre Hogan[1], Johannes Leveling[1], Hongyi Wang[1], Paul Ferguson[2], and Cathal Gurrin[2]

[1] Centre for Next Generation Localisation (CNGL)
School of Computing
Dublin City University (DCU)
Dublin 9, Ireland
[2] CLARITY Research Centre
School of Computing
Dublin City University (DCU)
Dublin 9, Ireland
{dhogan,jleveling,pferguson,hwang,cgurrin}@computing.dcu.ie

Abstract. This paper gives an overview of DCU's participation in the SMS-based FAQ Retrieval task at FIRE 2011. DCU submitted three runs for monolingual English experiments. The approach consisted of first transforming the noisy SMS queries into a normalised, corrected form. The normalised queries were then used to retrieve a ranked list of FAQs by combining the results from three different retrieval methods. Finally, using information from the retrieval results, out-of-domain (OOD) queries were identified and tagged. The results of our best run on the final test set were the highest of all 13 participating teams. Our FIRE submission retrieved 70.2% in-domain query answers correctly and 85.6% identified out-of-domain queries correctly.

1 Introduction

This paper describes the participation of Dublin City University (DCU) in the FIRE 2011 evaluation for the SMS-based FAQ Retrieval Task. The task consisted of retrieving the correct answer to an incoming SMS question from an English FAQ consisting of questions and answers on a variety of different topics from career advice to popular Indian recipes. The incoming queries were written in noisy SMS *"text speak"* and contained many misspellings, abbreviations and grammatical errors. Some SMS queries were out-of-domain and had no corresponding FAQ answer in the collection. Such queries needed to be identified and flagged as an out-of-domain (OOD) result before returning *"NONE"* as an answer string.

Figure 1 gives an overview of the DCU system, which can be broken down into three distinct steps: SMS normalisation, retrieval of ranked results, and identifying out of domain query results.

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 184–196, 2013.

The first step involves normalising words in the SMS text so that they more closely resemble the text in the FAQ data set (e.g. with correct and standardised spelling). This is achieved by generating a set of candidate corrections for SMS tokens using rules extracted from a mixture of annotated and unannotated corpora. The most likely token substitution, given the context, is then selected from the set of candidates. This step is detailed in Section 2.

For the second step in the process we experimented with different retrieval engines and approaches (i.e. Lucene, Solr and a simple word overlap metric) to retrieve ranked lists of candidate answers from the FAQ, given the normalised query. The retrieval results were combined to produce a single ranked list of question answer pairs. This step is described in more detail in Section 3.

In a final step, outlined in Section 4, we identified likely out-of-domain (OOD) questions using a filtering mechanism based on a combination of evidence from the results of the retrieval engines. For in-domain (ID) questions, the top answers from the combined list were returned; for OOD questions, *"NONE"* was returned. We present test set results in Section 5, before concluding and giving an outlook on planned future work in Section 6.

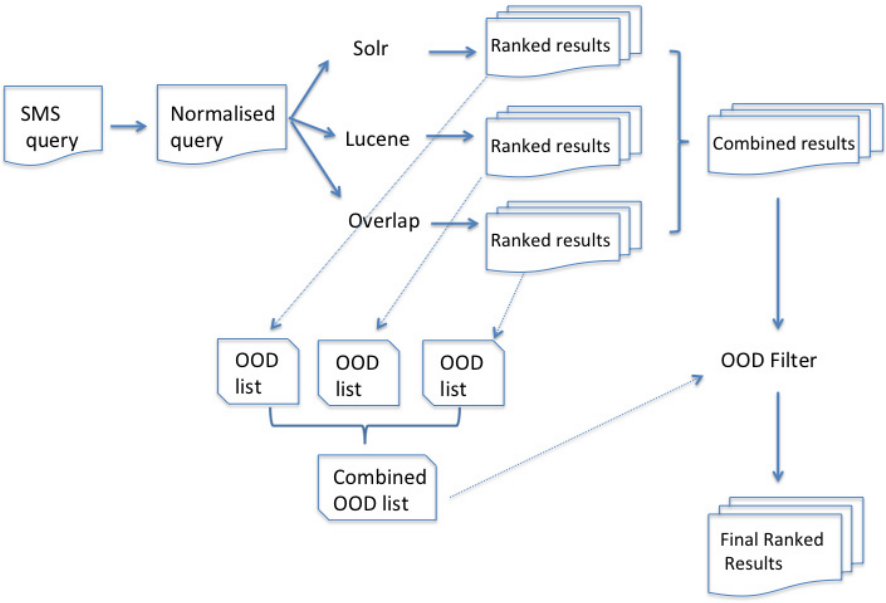


Fig. 1. Data flow diagram of DCU system

2 SMS Normalisation and Correction

The irregular spelling and abbreviations in SMS questions leads to poor retrieval performance due to mismatches between terms in the SMS and terms in the FAQ

text. An SMS normalisation and correction step will thus increase the chance of finding correct answers. Our initial idea to correct SMS messages was to train a statistical MT system, similar to the approach described in [1]. However, *"text speak"* or *"textese"* is productive and often generates new ill-formed or non-standard words which increase the out-of-vocabulary problem of statistical machine translation. Furthermore, training data in large enough quantities to train an accurate machine translation system for SMS correction does not exist and it is time-consuming to construct such data manually. Due to the lack of training data we decided against this approach and implemented a heuristic token substitution approach to correct SMS tokens.

The approach we took involved first carrying out some basic normalisation steps on both SMS queries and FAQ documents (described in Section 2.1). The SMS data then went through a correction step, where candidate corrections for SMS tokens were generated and then the candidate correction with the highest score was chosen. This process is outlined in detail in Section 2.2.

Table 1. Preprocessing steps

Token type	Example	Action	Description
contraction	*"I'll"* → *"I_will"*	expand	rules extracted from PoS-tagged Brown Corpus and annotated SMS corpus
interjection	*"Eeh"* → *"_"*	remove	rules extracted from manual annotation and PoS-tagged Brown Corpus
spelling variant	*"center"* → *"centre"*	normalise	dictionary-based spelling normalisation from AE to BE
acronyms	*"M.Sc."* → *"MSc"*	normalise	words of more than 50% uppercase characters and full stops, extracted from FIRE FAQ data and EN-1M corpus
spelling error	*"Austrailia"* → *"Australia"*	correct	most frequent spelling errors extracted from English Wikipedia
concatenation	*"12ft"* → *"12_ft"*	split	monetary values and measurements

2.1 Pre-processing for Documents and Queries

FAQ documents and SMS questions underwent the same preprocessing steps, which consisted of text normalisation as shown in Table 1. As SMS text can contain non-standard tokens, we adapted tokenisation to allow for digits in a word (e.g. *"2gether"*), and split character sequences of words (typically measurement units) and numbers (e.g. *"12ft"*).

2.2 SMS Correction

We employed three different techniques to generate candidate corrections for SMS tokens. These are described in Section 2.2. One of these techniques involved using correction rules extracted from corpora of hand annotated microtext data. The generation of these corpora is described in the next section.

Generating Training Data for SMS Correction and Normalisation. We created training data to use for automatically correcting SMS queries by manually annotating different microtext corpora (SMS and Tweets). The original text messages and the annotation were aligned on the level of tokens, so that there is a one-to-one correspondence between original token and corrected token. In order to preserve this one-to-one alignment, if necessary one or more tokens were joined together by underscore (e.g. *"I'll"* → *"I_will"*)

Table 2. Twenty most frequent corrections in FIRE preview and training questions

Rank	Word	Correction	Frequency	Rank	Word	Correction	Frequency
1	*"d"*	*"the"*	194	11	*"n"*	*"and"*	37
2	*"2"*	*"to"*	147	12	*"gt"*	*"get"*	32
3	*"hw"*	*"how"*	146	13	*"whch"*	*"which"*	29
4	*"r"*	*"are"*	108	14	*"bst"*	*"best"*	24
5	*"wht"*	*"what"*	101	15	*"fr"*	*"for"*	22
6	*"4"*	*"for"*	82	16	*"frm"*	*"from"*	22
7	*"f"*	*"of"*	71	17	*"wt"*	*"what"*	22
8	*"cn"*	*"can"*	63	18	*"wrld"*	*"world"*	21
9	*"wat"*	*"what"*	50	19	*"s"*	*"is"*	20
10	*"whr"*	*"where"*	38	20	*"watz"*	*"what_is"*	19

We manually annotated the following corpora with the corrected, normalised forms:

1. FIRE SMS training questions (1071 questions)
2. FIRE SMS preview questions (456 questions)
3. All SMS messages containing a question mark extracted from the NUS SMS Corpus [2] (3786 questions); the corpus was created at the National University of Singapore and consists of about 10000 SMS messages collected by students[1].
4. Tweets (549 messages) from CAW 2.0 - Content Analysis for the WEB 2.0[2]

Table 2 shows the top twenty corrections in the FIRE SMS QA training data.

[1] `http://www.comp.nus.edu.sg/~rpnlpir/downloads/corpora/smsCorpus/`
[2] `http://caw2.barcelonamedia.org/node/`

Generation and Selection of Correct Tokens. Our SMS correction approach is token-based. First all tokens are pre-processed as outlined in Table 1. Then each token in the SMS query is examined in turn to decide if it remains unchanged. Stopwords, punctuation, numerals, and acronyms are not modified. For each remaining token, a set of correction candidates is generated and the best candidate in the context is selected as a correction. The candidate corrections are generated by combining lists of candidates obtained from the following methods.

Correction rules: The manually corrected SMS questions were employed to extract correction rules and their corresponding frequencies, which are then used to generate the first list of candidate corrections. If applicable correction rules are found for a token, the frequencies of the rules in the annotated data are used to calculate normalised weights for each correction. For example, the token *"2"* can be corrected into *"two"*, *"too"*, or *"to"*, with *"to"* being the most frequent (see Table 2).

Consonant skeletons: An additional set of candidate corrections was created using consonant skeletons. We used two background corpora, the English 1M sentence corpus[3] from the Leipzig Corpora Collection (EN-1M), and the FIRE FAQ corpus used in the SMS QA track (43,871 sentences). Each token in the background corpora is processed to obtain its consonant skeleton [3], a shorter form of the word with all vowels removed (for example, *"rsdnt"* is the consonant skeleton for *"resident"*). The mapping between consonant skeletons and words is used to obtain additional correction candidates for question words that match a consonant skeleton.

Clippings: Finally, candidates are generated by looking up all words in the background corpora that have the same prefix as the question word to identify truncated or clipped words, e.g. *"exam"* → *"examination"* or *"lab"* → *"laboratory"*.

These methods yield lists of replacement candidates, which are merged by adding up their weights (derived from their term frequency in the background corpora). For each of the top twenty candidates, a token score (similar to a probability) is computed based on the so-called 'stupid' backoff [4] for 3-grams with $\alpha = 0.4$ (see Equation 1). We used the background corpora to collect n-gram statistics. The candidate with the maximum product of weight and n-gram score is selected as the token correction. Equation 1 shows the n-gram score where w_a^b is the n-gram ($n = b - a$) of tokens between position a and b and $f(w_a^b)$ is the frequency of the n-gram in the corpus.

$$S(w_i|w_{i-k+1}^{i-1}) = \begin{cases} \frac{f(w_{i-k+1}^{i})}{f(w_{i-k+2}^{i-1})} & \text{if } f(w_{i-k+1}^{i}) > 0 \\ \alpha S(w_i|w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases} \quad (1)$$

[3] `http://corpora.uni-leipzig.de/`

2.3 Evaluation of SMS Correction

Table 3 shows results for our SMS correction approach applied to the FIRE SMS QA training and preview questions. When testing on preview and training sets, we excluded the correction rules generated from corresponding annotated SMS data. Note that the results for the training data are actually much lower than expected because many of the token correction rules were missing. In contrast, the results for the preview data might be too high because there is an overlap in training and preview data. Note also that correcting to a wrong word form, i.e. an incorrect surface form with the same stem as the correct token, is counted as an error (e.g. correcting to *"resident"* instead of *"residents"*).

Table 3. Performance of SMS normalisation on FIRE preview and training data

	Count	Correct	Incorrect
Training sentences	1071	156 (15%)	915 (85%)
Training tokens	8432	6246 (74%)	2186 (26%)
Preview sentences	456	152 (33%)	304 (67%)
Preview tokens	5087	4546 (89%)	541 (21%)

In these tests, stopwords were among the most frequent errors in the SMS normalisation. For example, *"r"* was often replaced with *"are"* instead of *"or"*. However, these errors will not affect retrieval performance when stopwords are removed from the IR query.

3 Retrieval Engines

Before conducting our retrieval experiments, both SMS queries and FAQ documents were preprocessed as described in Section 2.1. The SMS queries then went through a further correction step (Section 2.2).

We experimented with three different retrieval methods, Lucene, Solr and a simple similarity metric (Term overlap) based on the number of overlapping words between query and document, and achieved the best performance by combining the outputs from the different systems.

We report results for experiments based on indexing FAQ questions, FAQ answers, and both questions and answers. The metric used is the in-domain score, calculated as:

$$\frac{count(\text{correct results in first ranked position})}{count(\text{queries with corresponding FAQ answers})} \quad (2)$$

3.1 Experimental Details

Search Engines. We experimented with two full-text search engines, Lucene and Solr, initially because we wanted a comparison of the ease of use of the two engines for this new task. Although Solr uses Lucene as its underlying search engine, we found it difficult to exactly replicate the results from both engines and found that Lucene consistently gave us better results on the training set.

We adapted both Lucene and Solr to use the BM25 ranking function [5] (with parameters $b = 0.75$, $k_1 = 1.2$, and $k_3 = 7$) and experimented with different stopword lists but otherwise used the default settings.

In our submission to the FIRE challenge, the SMART stopwords[4] were used for the Lucene experiments. We have since found much better results on the training set using Lucene's (much smaller) default stopword list and, indeed, get the best results on the training set by using no stopword list at all for Lucene. However, these improvements did not carry over to the Test set.

Table 4 and Table 5 show the results achieved by Solr and Lucene respectively on the in-domain queries of the training set. We display results for three different indexes (questions only, answers only and questions and answers). The numbers in the tables denote the accuracy (fraction of correct in all correct answers) and the absolute number of correct results (in brackets). Indexing the questions gives the best results. This is unsurprising given that the text of the corrected SMS queries is often very similar to the matching FAQ question. In Table 5, for point of comparison, an additional row displays the Lucene score on the question index when the SMART stopword list was used. For all other rows in Table 5 no stopword list was used.

In both tables, results are given for the original, unaltered *"textese"* SMS queries (raw), the automatically corrected queries (auto-correct) and the hand-corrected version of the SMS queries (gold).

Table 4. Training set comparison of **Solr-BM25** in-domain results when indexing questions only, answers only and both questions and answers. Total number of in-domain queries is 701.

	SMS question type		
Indexing	raw	auto-correct	gold
Questions	38.37 (269)	**72.04 (505)**	**72.46 (508)**
Answers	39.08 (274)	66.48 (466)	66.76 (468)
Questions & Answers	**39.66 (278)**	71.89 (504)	72.03 (505)

Term Overlap (Overlap). In addition to Solr and Lucene, we used a simple overlap metric as a baseline, The term overlap uses a text similarity score to rank results based on the number of matching terms in the query and each FAQ question.

[4] `ftp://ftp.cs.cornell.edu/pub/smart/`

Table 5. Training set comparison of **Lucene-BM25** in-domain results when indexing questions only, answers only and both questions and answers. Total number of in-domain queries is 701.

	SMS question type		
Indexing	raw	auto-correct	gold
Questions (SMART)	50.07 (351)	77.46 (543)	78.17 (548)
Questions	**57.06 (400)**	**80.88 (567)**	**80.03 (561)**
Answers	15.41 (108)	22.97 (161)	22.82 (160)
Questions & Answers	43.08 (302)	72.33 (507)	72.75 (510)

Table 6. Training set comparison of **Term overlap** in-domain results when indexing questions only, answers only and both questions and answers. Total number of in-domain queries is 701.

	SMS question type		
Indexing	raw	auto-correct	gold
Questions	**46.22 (324)**	**72.18 (506)**	**76.61 (509)**
Answers	5.56 (39)	7.99 (56)	8.13 (57)
Questions & Answers	2.11 (148)	3.22 (226)	3.30 (231)

The similarity score between two texts is calculated as a normalised score based on the number of words the two texts have in common. The (F1) score (Equation 3) is between 0 and 1 and is scaled based on the length of the strings.

$$\text{F1} = Sim(text_1, text_2) = \frac{2 * precision * recall}{precision + recall} \tag{3}$$

where

$$precision = \frac{count(\text{overlapping terms})}{count(\text{terms in } text_1)} \tag{4}$$

and

$$recall = \frac{count(\text{overlapping terms})}{count(\text{terms in } text_2)} \tag{5}$$

Software for finding the overlaps between two strings and calculating this text similarity score can be downloaded from CPAN[5]. Table 6 displays in-domain results for the overlap metric on the training set.

[5] `http://search.cpan.org/~tpederse/Text-Similarity/`

Combining Results. The best overall results on the training set (and subsequently on the test set) were achieved by combining the results from the three different retrieval methods. The result sets were combined using a mechanism whereby each search result x is associated with a score $S_{combined}(x)$ which is a weighted sum of the individual normalised scores from each retrieval engine (see Equation 6).

$$S_{combined}(x) = w_s * S_{Solr}(x) + w_l * S_{Lucene}(x) + w_o * S_{Overlap}(x) \quad (6)$$

The weights were determined by manual fine-tuning on the training set. The final weight settings chosen were: $w_l = 0.6$ (Lucene), $w_o = 0.3$ (Overlap), $w_s = 0.1$ (Solr).

Training set in-domain scores for the combined results, indexing questions only, are displayed in Table 7. For ease of comparison we repeat in Table 7 the results achieved by the individual retrieval mechanisms. Combining results gives a marginal increase in score for the automatically corrected queries, and leads to a slight drop for raw and gold queries. Although we used the combined results in our FIRE submission, we now conclude that the benefits of combining are small and using Lucene as the sole retrieval mechanism would simplify the system considerably while still delivering comparably good performance.

Table 7. Training set comparison of in-domain results for three different retrieval engines, indexing questions only. Total number of in-domain queries is 701. The numbers in brackets correspond to the number of correct results.

	SMS question type		
Retrieval method	raw	auto-correct	gold
Lucene	57.06 (400)	80.88 (567)	80.03 (561)
Solr	38.37 (269)	72.04 (505)	72.46 (508)
Overlap	46.22 (324)	72.18 (506)	76.61 (509)
Combined	**57.20 (401)**	**81.46 (571)**	**80.88 (567)**

4 Filtering Out-of-Domain Queries

For each retrieval method we produced a list of SMS queries which were predicted to be out-of-domain.

Solr. In order to generate the list of OOD queries from Solr, we used the same approach that was used by [6] for determining the number of relevant documents to use for query expansion. This approach produces a score based on the inverse document frequency (*idf*) component of BM25 for each query. This essentially disregards the term frequency and document length components which, since the queries are reasonably short, tend to be less important:

$$score(q,d) = \sum_{t \epsilon q} log\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right) \quad (7)$$

Using this approach we can calculate the maximum possible score for any document as the sum of the *idf* scores for all of the query terms: any document containing all the query terms will have this maximum score. We then use a threshold to determine if a query should be considered as OOD. Here we choose to add a query to the OOD list if its score is below 70% of the maximum score.

Lucene. TiMBL [7] implements a memory-based learning approach and supports different machine learning algorithms. For the experiments described in this paper, the IB1 approach (similar to k-nearest-neighbours approach) was employed to train a classifier distinguishing between OOD queries and ID questions. The features for the training instances include query performance estimates, result set size, and document score. The query performance estimates used are: Average Inverse Collection Term Frequency (AvICTF) [8], Simplified Query Clarity Score (SCS) [9], and an estimate derived from the similarity score between collection and query (SumSCQ, AvSCQ, MaxSCQ) [10]. In addition the (unnormalised) BM25 document scores [5] for the top five documents were employed as features.

This classifier achieved 78% accuracy (835 out of 1071 correctly classified instances) on the FAQ SMS training data. Table 8 shows true positives (TP), false negatives (FN) etc. per class, using leave-one-out validation.

Table 8. Scores per question class

Class	TP	FP	TN	FN	F-Score
ID	459	111	376	125	0.80
OOD	376	125	459	111	0.76

Term Overlap. In this approach, the list of OOD queries was predicted based on the number of terms in each incoming query and the number of overlapping terms between incoming query and the highest ranked question from the FAQ (For example, if the incoming query consists of more than one term and has only one term in common with the highest ranked FAQ question, then classify the query as out-of-domain). The heuristic algorithm was fine-tuned on the training set and optimised to maximise both out-of-domain and in-domain accuracy.

Combining OOD Results. Based on experiments on the training set data, we found that combining the OOD lists through simple majority voting led to the best results. Table 9 displays the best in-domain and out-of-domain results achieved on the training set. The OOD score is calculated as:

$$\frac{count(\text{OOD queries correctly identified})}{count(\text{all OOD queries})} \quad (8)$$

Table 9. Training set - 3 final run configurations. ID and OOD results after applying OOD filtering for auto-corrected queries.

		SMS query type								
		raw			auto			gold		
Retrieval	Index	ID	OOD	all	ID	OOD	all	ID	OOD	all
Combined	Q	52.06	**69.18**	**57.98**	**73.32**	**69.19**	**71.90**	**73.03**	**69.19**	**71.71**
Lucene	Q	**52.07**	61.62	55.37	72.61	**69.19**	71.43	71.90	**69.19**	70.96
Lucene	Q+A+QA	**52.07**	62.43	55.65	72.61	69.19	71.43	71.89	**69.19**	70.96

The weights used for combining different retrieval results are as follows: Weights for *Combined* (combining Lucene, Solr and Overlap results): $w_l = 0.5$ (Lucene), $w_s = 0.2$ (Solr), $w_o = 0.3$ (Overlap). Weights for *Lucene Q/A/QA* (combining Lucene results on different indexes): Q: 0.7, A: 0.1, QA: 0.2.

5 Test Set Retrieval Experiments and Results

Table 10. Results on the test set for the normalised queries, as submitted to the FIRE evaluation

Run	Index	Retrieval	ID Correct	OOD Correct	all	MRR
1	Q	Combined	**0.70 (494/704)**	**0.86 (2311/2701)**	**82.38**	**0.896**
2	Q	Lucene	0.67 (472/704)	0.86 (2310/2701)	81.70	0.865
3	Q+A+QA	Lucene	0.68 (477/704)	**0.86 (2311/2701)**	81.88	0.873

Table 11. Latest results on the test set. ID and OOD Results after applying the OOD filtering.

		SMS query type								
		raw			auto			gold		
Retrieval	Index	ID	OOD	all	ID	OOD	all	ID	OOD	all
Combined	Q	**63.35**	**84.93**	**80.47**	**70.57**	84.80	**81.94**	**70.31**	**84.86**	**81.85**
Lucene	Q	62.78	**84.93**	80.35	70.31	**84.89**	81.88	70.03	**84.86**	81.79
Lucene	Q+A+QA	62.78	**84.93**	80.35	70.31	**84.89**	81.79	70.03	**84.86**	81.79

The DCU team submitted three runs for the English monolingual task. Table 10 details the results for the three DCU runs. For the first run, only the question text from the FAQ was indexed. The final ranked results list was produced by combining the individual ranked lists from the three different retrieval approaches as described in Section 3. For the second and third run, only the

Lucene search engine was used. In the second run, FAQ question text was indexed whereas in the final run, both question and answer text from the FAQ was indexed.

Table 11 gives our results when no stopword list was used with the Lucene retrieval. Unlike on the training set, where results improved considerably, the combined results for the test set are slightly lower than previously. However, results for using only the Lucene search engine are much the same whether or not stoplists are used.

6 Conclusion and Future Work

Our submission achieved the best performance in the official results of the FIRE 2011 SMS-based FAQ monolingual English retrieval task.

We found that the best retrieval results for an individual method are obtained when using Lucene-BM25 scoring. While the combination of approaches for retrieval and OOD detection increases the number of correct results marginally, we conclude that the benefits of combining are small and using Lucene as the sole retrieval mechanism simplifies the system considerably while still delivering comparably good performance.

As part of future work, we want to simplify the system further by using a single OOD detection approach, rather than combining OOD lists generated from different retrieval methods.

Acknowledgments. This research is supported in part by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (CNGL) project.

References

1. Aw, A., Zhang, M., Xiao, J., Su, J.: A phrase-based statistical model for SMS text normalization. In: COLING/ACL 2006, pp. 33–40 (2006)
2. How, Y., Kan, M.Y.: Optimizing predictive text entry for short message service on mobile phones. In: Human Computer Interfaces International (HCII 2005). Lawrence Erlbaum (2005)
3. Kothari, G., Negi, S., Faruquie, T.A., Chakaravarthy, V.T., Subramaniam, L.V.: SMS based interface for FAQ retrieval. In: ACL/IJNLP 2009, pp. 852–860 (2009)
4. Brants, T., Popat, A.C., Xu, P., Och, F.J., Dean, J.: Large language models in machine translation. In: EMNLP-CoNLL, pp. 858–867. ACL (2007)
5. Robertson, S.E., Walker, S., Jones, S., Beaulieu, M.M.H., Gatford, M.: Okapi at TREC-3. In: Harman, D.K. (ed.) TREC-3, pp. 109–126. NIST, Gaithersburg (1995)
6. Ferguson, P., O'Hare, N., Lanagan, J., Smeaton, A.F., Phelan, O., McCarthy, K., Smyth, B.: CLARITY at the TREC 2011 Microblog Track. In: Text Retrieval Conference, TREC (2011)
7. Daelemans, W., Zavrel, J., van der Sloot, K., van den Bosch, A.: TiMBL: Tilburg memory based learner, version 6.2, reference guide. Technical Report 09-01, ILK (2004)

8. Kwok, K.L.: A new method of weighting query terms for ad-hoc retrieval. In: SIGIR 1996, pp. 187–195. ACM, New York (1996)
9. He, B., Ounis, I.: Inferring query performance using pre-retrieval predictors. In: Apostolico, A., Melucci, M. (eds.) SPIRE 2004. LNCS, vol. 3246, pp. 43–54. Springer, Heidelberg (2004)
10. Zhao, Y., Scholer, F., Tsegay, Y.: Effective pre-retrieval query performance prediction using similarity and variability evidence. In: Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., White, R.W. (eds.) ECIR 2008. LNCS, vol. 4956, pp. 52–64. Springer, Heidelberg (2008)

Overview of the FIRE 2011 RISOT Task

Utpal Garain[1,*], Jiaul H. Paik[1,*], Tamaltaru Pal[1],
Prasenjit Majumder[2], David S. Doermann[3], and Douglas W. Oard[3,4]

[1] Indian Statistical Institute, Kolkota, India
[2] DAIICT, Gandhinagar, India
[3] University of Maryland Institute for Advanced Computer Studies (UMIACS)
[4] College of Information Studies, University of Maryland, College Park, MD USA
{utpal,tamal}@isical.ac.in, jia.paik@gmail.com,
Prasenjit.majumdar@gmail.com, {doermann,oard}@umd.edu

Abstract. RISOT was a pilot task in FIRE 2011 which focused on the retrieval of automatically recognized text from machine printed sources. The collection used for search was a subset of the FIRE 2008 and 2010 Bengali test collections that contained 92 topics and 62,825 documents. Two teams participated, submitting a total of 12 monolingual runs.

1 Introduction

The first Retrieval of Indic Script OCR'd Text (RISOT) task was one of seven tasks at the Third Forum for Information Retrieval Evaluation (FIRE), which was held in Mumbai, India in December, 2011. The focus of the task was on evaluation of Information Retrieval (IR) effectiveness for errorful text generated from machine printed documents in an Indic script using Optical Character Recognition (OCR). Substantial effort has been invested in developing OCR for Indic scripts[1], but RISOT is the first effort to formally characterize the utility of such systems as part of an information retrieval application. The track has three primary goals: (1) supporting experimentation of retrieval from printed documents, (2) evaluating IR effectiveness for retrieval based on Indic script OCR, and (3) providing a venue through which IR and OCR researchers can work together on a challenge that requires perspectives drawn from both communities. RISOT was included in FIRE 2011 as a pilot task to begin the development of test collections and to provide an opportunity for multidisciplinary research teams to come together and collaborate.

This paper presents an overview of activities in this first year of the RISOT task. Section 2 briefly reviews prior work in evaluation of IR from printed documents, Section 3 describes the test collection and evaluation method, Section 4 introduces the

* The authors to whom correspondence should be directed.

[1] The Ministry of Information Technology, Govt. of India has been funding a nationwide consortium for developing robust OCR system for ten Indic scripts/languages: http://tdil.mit.gov.in/Research_Effort.aspx

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 197–204, 2013.

participating teams and presents aggregate results, and Section 5 concludes the paper with a brief discussion of the future of the RISOT task.

2 Background

The design of the RISOT task was influenced by two previous TREC (Text Retrieval Conference) evaluations that had similar goals: the Confusion Track and the Legal Track. The TREC Confusion track was part of TREC-4 in 1995 [1] and TREC-5 in 1996 [2]. In the TREC-4 Confusion Track, random character insertions, deletions and substitutions were used to model degradations (with electronic text as the starting point). The collection to be searched included about 260,000 English electronic text documents from multiple sources, and distortion modeling was applied to either 10% or 20% of the characters. This use of character distortion models for collection development was useful as a way of quickly gaining some experience with the task, but such an evaluation design raises fidelity concerns, particularly when error models are also used in the retrieval process. The concern arises from the potential for unmodeled phenomena (e.g., correlated errors) yielding evaluation results that might not be representative of actual applications. For the TREC-5 Confusion Track, about 55,000 government announcements that had been printed, scanned, and then OCR'd (with a roughly 5% or a roughly 20% character error rate) were used instead. Electronic text for the same documents was available for comparison. Relevance judgment costs were minimized in the TREC-5 Confusion Track by using a known-item evaluation design in which each query was designed to retrieve a single item from the collection. All experiments in both years of the TREC Confusion Track were run in automatic mode (i.e., with no human intervention). Participants experimented with techniques that used error models in various ways and with techniques that sought to accommodate OCR errors by using relatively short overlapping character n-grams.

TREC returned to evaluating retrieval of printed documents in the Legal Track each year between 2006 and 2009 [3,4,5,6]. A collection of about 7 million scanned English business documents (e.g., memoranda, reports, and printed email) was searched, with the same collection being used in each of the four years. These documents were made available as part of the so-called "Tobacco Lawsuits" which took place in the USA between 1999 and 2004. Access to the printed and scanned documents provided a more natural range of variability than the documents of the TREC-5 Confusion Track, although no corresponding electronic text was available. A notable feature of the TREC Legal Track was the use of rich topic statements that were representative of those commonly used to request evidence in the legal process of "e-discovery." The TREC-2006 Legal Track included only automated experiments. This same collection (with new topics) was used in the TREC-2007 and TREC-2008 Legal Tracks with the involvement of real users being incorporated in one of two ways. The first was the use of relevance feedback experiments in which some pre-existing relevance judgments were provided with the query (2007-2009). The second was the use of fully interactive experiments in which users could work as

a part of a human-machine system to obtain optimal results for a smaller number of topics (2007 and 2008).

Experiments with retrieval from printed documents were, of course, also conducted outside of community-based evaluation venues such as TREC. Most notably, early experiments with OCR-based retrieval on small collections were reported by Taghva and his colleagues at the University of Nevada, Las Vegas (UNLV) as early as 1993 [7]. Perhaps the best known among the early work was that of Singhal and his colleagues using larger collections (simulated with character corruption modes, as in TREC-5), which showed that linear document length normalization models were better suited to collections containing OCR errors than the quadratic (cosine normalization) models that were widely used at the time [8]. OCR-based retrieval is now widely used in many applications, most notably Google Books [9]. To date, however, none of this work has focused on Indic languages.

3 Test Collections

FIRE 2008 and 2010 were the first information retrieval community evaluation venues to create large-scale IR text collections for Indic languages. The RISOT 2011 data set is a subset of the existing FIRE Bengali test collection, which contains articles from a leading Bengali newspaper that were published between 2004 and 2006. The subset contains 62,825 documents, about one fourth of the FIRE Bengali collection. We refer to the electronic text from that collection as the "clean" text collection or simply the "TEXT" collection.

For RISOT, each document in the clean text collection was rendered as a document image using standard next rendering software at a resolution of 300 dots per inch (dpi). Some documents generated multiple pages, and on average 2.8 images were generated per document. The correspondence between a text page and its corresponding image(s) was maintained using a file naming convention. The resulting images are of high quality, free from the kinds of skew, distortion and spurious marks that might be found in scanned images of actual newspaper pages.

3.1 OCR Collection

A Bengali OCR system was used to convert these images into electronic text using a feature-based template matching approach [10]. Automatic evaluation [11] found the Unicode glyph accuracy to be about 92.5%. A single Bengali character might be represented using two or more Unicode glyphs, so glyph accuracy somewhat understates character accuracy. For example, if <প><ু> were misrecognized as <এ>, two Unicode glyph errors would be counted. Similarly, if <স><্><ন> were misrecognized as <ল>, three Unicode glyph errors would be counted. In each case, only one Bengali character substitution would actually have occurred.

The principal causes of OCR errors are segmentation errors (specifically, errors in the division of words into characters) and character misclassification. The Bengali alphabet contains about 250 characters, counting both basic characters and conjuncts.

There are also some vowel modifiers which can attach to consonants, forming yet more new shapes. Our current OCR system treats all of these shapes as separate classes, resulting in about 700 shapes that character classification must distinguish. Thus, the character recognition problem in Bengali is nearly an order of magnitude more challenging than is the case for English.

When a single document generated multiple images, the OCR outputs for each of those images are reassembled to produce a single OCR'd document. There are therefore 62,825 OCR'd documents, and this collection is referred to simply as the "OCR" collection.

3.2 Topics

The 92 RISOT Bengali topics were taken from FIRE 2008 (topics 26-50) and FIRE 2010 (topics 51-125). Each topic consists of three parts – a Title (T), a Description (D) and a Narrative (N) – and a unique query number. The title represents what a searcher might initially type into a Web search engine, the title and description together (which we call TD) represents what the searcher might say to a human intermediary who has offered to help them with their search, and the title, description and narrative together (which we call TDN) represents what that intermediary might understand the information need to be after some discussion with the searcher. The machine's task is then to take a T or TD query and to return results that would be judged to be relevant on the basis of the full TDN topic description. The topic statements are available in several languages, but only Bengali queries were used in the 2011 RISOT pilot task. A sample topic is shown in Bengali and English below.

<top>
<num>26</num>
<title>সিঙ্গুরে জমি অধিগ্রহণ সমস্যা</title>
<desc>সিঙ্গুরে বামফ্রন্ট সরকারের জমি অধিগ্রহণ কর্মসূচি এবং ভূমি উচ্ছেদ প্রতিরোধ কমিটির বিক্ষোভ সংক্রান্ত নথি খুঁজে বার করো। </desc>
<narr>শিল্পোন্নয়নের জন্য সিঙ্গুরে কৃষি জমি অধিগ্রহণ, বামপন্থী ও বিরোধী দলের মধ্যে সঙ্ঘর্ষ, সাধারণ মানুষকে নিষ্ঠুর ভাবে হত্যা, সমাজের বিভিন্ন স্তরের মানুষের প্রতিবাদ ও সমালোচনা প্রাসঙ্গিক নথিতে থাকা উচিৎ। </narr>
</top>

<top>
<num>26</num>
<title>Singur land dispute</title>
<desc>The land acquisition policies of the Left Parties in Singur and the protest of Bhumi Ucched Protirodh Committee against this policy.</desc>
<narr>Relevant documents should contain information regarding the acquisition of agricultural land for industrial growth in Singur, the territorial battle

```
between the Left Parties and the opposition parties, the
brutal killing of the innocent people and the protests
and the criticism by people from different sections of
society.</narr>
  </top>
```

3.3 Relevance Judgments

Relevance judgments had been created for these topics in 2008 or 2010 as part of the FIRE ad hoc task [12]. The existing FIRE relevance judgments have been limited to the documents in the RISOT 2011 collection and we reused those relevance judgments for the 2011 RISOT pilot task. Only a subset of the documents was judged (generally, those that were highly ranked by some participating system in the 2008 or 2010 ad hoc task); unjudged document were treated as not relevant.

3.4 Evaluation

RISOT 2011 participants were asked to evaluate their runs using the relevance judgments provided by the organizers and version 9.0 of the trec-eval package.[2] Participants were asked to report MAP and P@10 for both the TEXT and the OCR conditions, and to explain how they had formed their queries (e.g., as T, TD or TDN).

4 Results

Two teams participated in RISOT 2011, one from the Indian Statistical Institute, Kolkata, India (ISI) and one from the University of Maryland, College Park, USA (UMD). Both teams submitted TEXT and OCR runs with no special processing as baseline conditions. The ISI team also experimented with rule-based error correction and with query expansion. The UMD team also experimented with stemming and with statistical accommodation of likely errors. Table 1 shows the reported results for the 12 submitted runs.

As Table 1 illustrates, the best results (by P@10) were obtained using TD queries on clean text. Stemming yielded apparent improvements for each condition in which it was tried (TD TEXT, T TEXT, TD OCR) and these observed differences are statistically significant. Error modeling yielded apparent improvements for the OCR condition in all three cases in which it was tried (TD unstemmed, TD stemmed, T stemmed). Among these improvements error modeling on TD unstemmed and TD stemmed produced statistically significant improvements but improvement for T stemmed is observed to be statistically not significant. Notably, ISI and UMD used rather different error modeling techniques. The best results for the OCR condition achieved 88% of the P@10 (and 90% of the MAP) achieved by the same team's TEXT condition. These results suggest that practical search applications for printed

[2] http://trec.nist.gov/trec_eval/

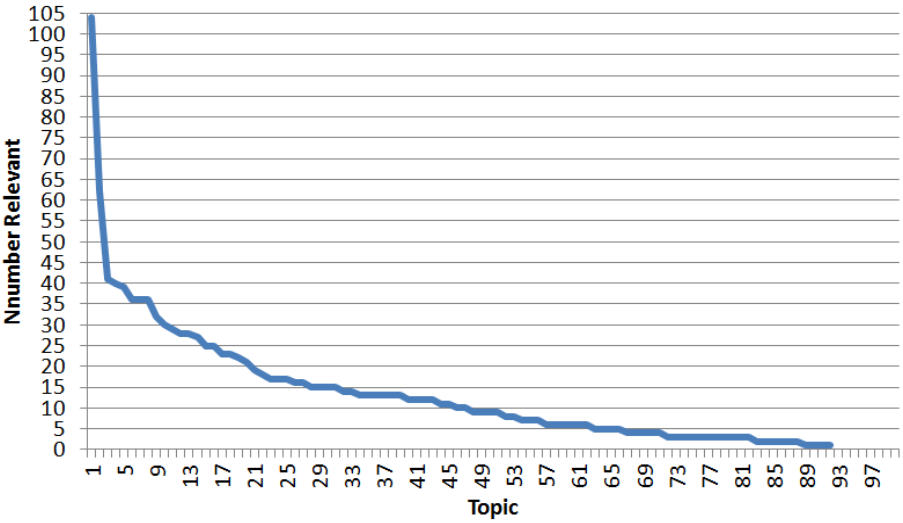


Fig. 1. Quantization noise risk

Table 1. RISOT 2011 results

Run	Query	Docs	Processing	P@10	MAP
umdT2	TD	TEXT	Stemming	0.3554	0.4229
isiT1	TD	TEXT	None	0.3239	0.3540
umdE5	TD	OCR	Stemming + OCR single-error model	0.3008	0.3521
umdT1	T	TEXT	Stemming	0.2967	0.3487
isiE1	TD	OCR	OCR multiple-error model	0.2859	0.3193
umdE2	T	OCR	Stemming + OCR single-error model	0.2686	0.2967
umdE1	T	OCR	OCR single-error model	0.2583	0.2588
umdO4	TD	OCR	Stemming	0.2489	0.2915
isiO1	TD	OCR	None	0.2293	0.2318
umdO3	TD	OCR	None	0.2217	0.2293
umdO2	T	OCR	Stemming	0.2187	0.2349
umdO1	T	OCR	None	0.1901	0.1922

Bengali documents could be constructed now. Moreover, in view of the fact that this year's relevance judgments could serve as training data for next year's RISOT task, continued research using more highly tuned approaches to error modeling and to stemming for Bengali OCR results might reasonably be expected to yield further improvements. Topic wise analysis of results shows that for many topics RISOT data does not contain sufficient number of relevant document. Fig.1 plot shows this quantization noise risk. Out of 92 topics, for only 66 topics the number of relevant documents is five or more. Next year data preparation will take care of this aspect so that accuracy of statistical significance or topic wise analysis is not affected by the quantization noise.

5 The Future

In subsequent years, we anticipate conducting an extended version of RISOT. Future evaluations may consider a number of changes:

- For the 2011 pilot task we asked participants to compute their own results using existing relevance judgments; in future years we expect to conduct blind evaluations using new relevance judgments.
- For this year's task we generated clean images. In future years, image degradation models could be applied before running the OCR. Alternatively, we could model the actual application with even higher fidelity by printing and then re-scanning at least a part of the collection. Indeed, even higher fidelity might be achieved by finding a subset of the documents that have actually been printed in the newspaper and scanning those newspaper clippings. With these approaches we could generate as many as four versions of an OCR collection.
- Some participants in future years might wish to contribute additional OCR results, or to perform retrieval tasks using image domain techniques. For such cases, the participants would need to be provided with an image collection along with the clean text collection.
- Documents in other Indic scripts such as Devanagari may also be added in future years.
- Additional evaluation measures such as Normalized Discounted Cumulative Gain (NDCG) or inferred Average Precision (infAP) may also be considered in future years.

The specific design of the task in future years will, of course, be discussed among the potential participants. We therefore encourage the broadest possible participation in the forthcoming RISOT task in order to provide a rich basis for those discussions.

Acknowledgement. The part of this work was conducted with support from the Indo-US Research Fellowship 2011.

References

1. Harman, D.: Overview of the Fourth Text Retrieval Conference. In: The Fourth Text Retrieval Conference, Gaithersburg, MD, USA, pp. 1–24 (1995)
2. Kantor, P., Voorhees, E.: Report on the TREC-5 Confusion Track. In: The Fifth Text Retrieval Conference, Gaithersburg, MD, USA, pp. 65–74 (1996)
3. Baron, J., Lewis, D., Oard, D.: The TREC-2006 Legal Track. In: The Fifteenth Text Retrieval Conference, Gaithersburg, MD, USA (2006)
4. Tomlinson, S., Oard, D., Baron, J., Thompson, P.: Overview of the TREC 2007 Legal Track. In: The Sixteenth Text Retrieval Conference, Gaithersburg, MD, USA (2007)
5. Oard, D., Hedin, B., Tomlinson, S., Baron, J.: Overview of the TREC 2008 Legal Track. In: The Seventeenth Text Retrieval Conference, Gaithersburg, MD, USA (2008)
6. Hedin, B., Tomlinson, S., Baron, J., Oard, D.: Overview of the TREC 2009 Legal Track. In: The Eighteenth Text Retrieval Conference, Gaithersburg, MD, USA (2009)
7. Taghva, K., Borsack, J., Condit, A.: Results of Applying probabilistic IR to OCR Text. In: The Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, pp. 202–211 (1994)
8. Singhal, A., Salton, G., Buckley, C.: Length Normalization in Degraded Text Collections. In: Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, USA, pp. 149–162 (1996)
9. Vincent, L.: Google Book Search: Document Understanding on a Massive Scale. In: Ninth International Conference on Document Analysis and Recognition, Curitiba, Brazil, pp. 819–823 (2007)
10. Garain, U., Chaudhuri, B.: Compound character recognition by run number based metric distance. In: Proceedings of the IS&T/SPIE 10th International Symposium on Electronic Imaging: Science & Technology, SPIE, San Jose, CA, USA, vol. 3305, pp. 90–97 (1998)
11. Sauvola, J., Kauniskangas, H., Doermann, D., Pietikainen, M.: Techniques for automated testing of document analysis algorithms. In: Brazilian Symposium on Document Image Analysis, Curitaba, Brazil, pp. 201–212 (1997)
12. Majumder, P., Mitra, M., Pal, D., Bandyopadhyay, A., Maiti, S., Pal, S., Modak, D., Sanyal, S.: The FIRE 2008 Evaluation Exercise. ACM Transactions on Asian Language Information Processing 9(3), 10:1–10:24 (2010)

Maryland at FIRE 2011: Retrieval of OCR'd Bengali

Utpal Garain[1,*], David S. Doermann[2], and Douglas W. Oard[2,3]

[1] Indian Statistical Institute, Kolkota, India
[2] University of Maryland Institute for Advanced Computer Studies (UMIACS)
[3] College of Information Studies, University of Maryland, College Park, MD USA
utpal@isical.ac.in, {doermann,oard}@umd.edu

Abstract. In this year's Forum for Information Retrieval Evaluation (FIRE), the University of Maryland participated in the Retrieval of Indic Script OCRed Text (RISOT) task to experiment with the retrieval of Bengali script OCR'd documents. The experiments focused on evaluating a retrieval strategy motivated by recent work on Cross-Language Information Retrieval (CLIR), but which makes use of OCR error modeling rather than parallel text alignment. The approach obtains a probability distribution over substitutions for the actual query terms that possibly correspond to terms in the document representation. The results reported indicate that this is a promising way of using OCR error modeling to improve CLIR.

1 Introduction

The key problem that distinguishes retrieval of OCR'd documents from other information retrieval problems is that character distortions that result from the OCR process result in mismatches between the (undistorted) query representation and the (distorted) document representation. Two broad classes of techniques have emerged to mitigate the effects of those distortions on retrieval effectiveness: (1) techniques based on the use of short overlapping character n-grams and (2) techniques based on error modeling. The basic goal of the n-gram techniques is to make n large enough to capture some meaning, while keeping it small enough to have a good chance of obtaining a sequence of n undistorted characters. Using character n-grams has the advantage of simplicity (at some cost in efficiency), but for some applications that simplicity limits its effectiveness. Error modeling, by contrast, seeks to model regularities in the types of errors made by OCR systems, thus gaining access to evidence that is unavailable when character n-grams are used alone (i.e., without error modeling). Error modeling thus offers the potential to achieve improved retrieval effectiveness over those that character n-gram techniques could obtain alone. Although these techniques could be used in combination, in this paper we focus on the use of error modeling with ordinary space-delimited tokens. For a language such as Bengali (the language used in RISOT 2011) in which words are visibly delimited

* This work was conducted at the University of Maryland with support from the Indo-US Research Fellowship 2011.

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 205–213, 2013.

in a way that word segmentation can recognize fairly accurately, this seems to be a reasonable choice that limits the complexity of the experiments that we need to run.

Three ways of using error models have been investigated. The first approach is to use the error model to generate a one-best correction for each document term, thus (hopefully!) generating a closer approximation to the actual content of each document. Indeed, OCR systems that employ correction techniques based on lexicons or language models implicitly employ this technique. By forcing the choice of a single representation for each term, this approach leaves some evidence unexploited at retrieval time.

The second possibility is to embrace uncertainty and expand the query with several plausible distortions of each meaning-bearing query term (i.e., each non-stopword) in the hope that some of those distortions will better match the (distorted) document representations. This approach raises the question of how the resulting query should be structured, since it does not seem reasonable that different alternatives for one query term should have the same effect on the retrieval results as different query terms would have (as a simple English example, we don't want "white house" to retrieve "whito whlte". Building on work by Pirkola and Kwok, Darwish and Oard proposed a query structuring technique that can incorporate distortion probabilities derived from an OCR error model [1]. We use that same technique in this paper, but with a more principled derivation of the distortion probabilities than was used by Darwish and Oard.

The third way of using a distortion model would be to embrace uncertainty by generating multiple plausible corrections for each term in each (distorted) document. Wang and Oard have analytically shown that a principled implementation of this technique in which partial counts for plausible distortions can be used in the term weight computation would yield results identical to Darwish and Oard's query-time technique. The choice is then one of efficiency (which would favor the indexing-time implementation in query-intensive applications) vs. flexibility (which favors using Darwish and Oard's query-time implementation in experimental settings) [2]. Darwish and Oard initially proposed their query structuring approach (which they call Probabilistic Structure Queries (PSQ)) for both OCR-based retrieval and Cross-Language Information Retrieval (CLIR). Interestingly, all of the subsequent experimentation that we are aware of with that technique has been applied only in the CLIR context. This may be simply because OCR-based information retrieval evaluations have not been as common over that period.

Our goals in our RISOT experiments are: (1) to build error models for Bengali OCR, and (2) to use those error models with the RISOT test collection to evaluate Darwish and Oard's approach on a second test collection. Section 2 of this paper introduces our approach, Section 3 presents our results, and Section 4 concludes the paper with a brief discussion of future plans.

2 Modeling OCR Errors

Consider a query word "cat". Because of OCR errors "cat" may sometimes appear as "cot" (if 'a' is misrecognized as 'o') in the OCR'd documents. Therefore, documents containing "cat" or "cot" or both should be retrieved against the query word "cat".

One way of doing this is to expand the query (e.g., to include the word "cot" as well whenever the query word "cat" is seen). In order to do this one should have an idea how characters are affected by OCR errors.

2.1 OCR Error Probabilities

In our study, we gather this knowledge by comparing 20,000 pages containing 37 million characters of clean TEXT with the electronic text generated from OCR. The OCR text was obtained by rendering the original TEXT, and performing OCR. We used a dynamic programming approach to compare each pair of documents and to generate summary statistics for character errors (more specifically, for Unicode glyph errors). This summary reports which Unicode glyphs are observed to have been inserted into, deleted from, or substituted in the OCR text, and with what frequency each error is observed. The summaries for these 20,000 pages are combined and global statistics are computed. Several characteristics are observed from these error statistics including: (i) a few character translation errors are content (i.e. in some cases, a given character is always translated to the same character or set of characters), (ii) in most of the cases, a character is mapped into multiple characters with varying frequencies, (iii) along with isolated characters, sometimes a group of two or more adjacent characters is mapped into one or more characters [e.g. when "ri" is translated to "n") , and (iv) the character translation list has a long tail of characters being mapped into other characters with very low frequencies.

From this knowledge we build a table of triplets $<t_i, o_i, p_i>$ where t_i is translated to o_i with probability p_i. We call this probability the corruption probability. Note that both t_i and o_i refer to a single character or a group of characters. Our further investigation reveals that though the table contains more than 200 such triplets, the 75 top frequent entries cover 80% error cases. In our method, we consider these 75 triplets. E_t denotes the table of these 75 entries.

2.2 Query Expansion

Let $W_t = w_1w_2\ldots w_n$ be an n-letter query word. Each of these n letters has its corruption probability (the letters that do not appear in the left most column of E_t have zero corruption probability). Assuming that the letters of W_t are corrupted (by OCR engine) independent of each other, there may be many corrupted versions of the word W_t. All these corrupted words are considered as synonyms and the set of such words replace W_t in the query.

2.2.1 Single Error Model

This model allows only one corruption to generate a synonym while expanding a query. For example, if m letters out n ($m \leq n$) have non-zero corruption probability then m synonyms (let $W_t^1, W_t^2, \ldots, W_t^m$ denote this set of corrupt words) are generated from the query word W_t. The probability of generating a corrupt word or synonym

(say, W_s^i) is P($W_t \rightarrow W_s^i$) = P($w_i \rightarrow o_j$), where the letter w_i is translated to o_j. This probability can be found from the error table E_t.

2.2.2 Multiple Error Model

This model assumes multiple character level errors may take place to form a synonym. So following this model, the query word W_t may generate up to 2^m synonyms, where m letters ($m \leq n$) have non-zero corruption probability. Since this number could be quite large for many query words, only the 32 top choices are considered for query expansion. These choices are made according to the probabilities by which synonyms are generated. If two corruptions ($w_i \rightarrow o_j$ and $w_j \rightarrow o_k$) are considered to form a synonym, W_s^i then P($W_t \rightarrow W_s^i$) = P($w_i \rightarrow o_j$) * P($w_j \rightarrow o_k$). Both P($w_i \rightarrow o_j$) and P($w_j \rightarrow o_k$) are found from the table E_t.

2.2.3 Weighting the Synonyms

While weighting a synonym in the query, we consider the following fact. Say, we know that "cot" is a synonym of the query word "cat". So whenever we encounter a "cot" in the documents, we would like to know the probability that this "cot" is actually "cat". More specifically, if we know P ("cat" | "cot") then this probability may serve as the weight of the term "cot" in the expansion of "cat" as a query term. Note that the term "cot" occurs in the OCR'd document while the term "cat" appears as a query term. Let W_{ocr} be a synonymous term of a query term W_{text}. The weight by which W_{ocr} appears in the query is set to P ($W_{text} \mid W_{ocr}$). This probability is in turn computed as

$$P(W_{text} \mid W_{ocr}) = \frac{P(W_{ocr} \mid W_{text}) P(W_{text})}{P(W_{ocr})} \quad (1)$$

P(W_{text}) can be computed from a standard language corpus, P(W_{ocr}) can be computed from the OCR'd corpus. The third component, i.e. P($W_{ocr}|W_{text}$) is basically P($W_t \rightarrow W_s^i$) which as shown before can be computed from the error table E_t.

3 RISOT 2011 Experiments

We participated in the first Retrieval of Indic Script OCRed Text (RISOT) task, one of seven tasks at the Third Forum for Information Retrieval Evaluation (FIRE) held in Mumbai, India in December 2011. The focus of the task was on the evaluation of Information Retrieval (IR) effectiveness of errorful text generated from machine printed documents in an Indic script using Optical Character Recognition (OCR). The test collections, our submitted runs and results are described below.

3.1 Test Collections

The RISOT 2011 data set is a subset of the existing FIRE Bengali test collection. The RISOT collection contains two directories: TEXT and OCR. Each document in TEXT

was converted into images and then OCR'd. Therefore, each document in the OCR collection corresponds to a clean document in the TEXT collection. Each collection contains 62,825 documents. The OCR system achieved about 93% character accuracy. The RISOT topics are also taken from the existing FIRE topics for Bengali. There are 92 TREC-like topics, each consisting of three parts: a title (T), a description (D) and a narrative (N). RISOT reuses the relevance judgments from the FIRE ad hoc monolingual Bengali retrieval task.

3.2 Experiments

We indexed the collections (TEXT as well OCR) separately using the Lemur Toolkit[1]. Two types of queries were formed: one from the title (T queries) and the other from the title and description (TD queries). These queries were executed separately for the Text and OCR'd collections and retrieval efficiencies were noted. Next, we applied our proposed OCR error modeling and expanded the queries. Both single error and multiple error models were investigated. The effect of OCR error modeling based query expansion was studied for both the T and TD queries.

We conducted one more experiment using a generative stemmer [3] for query expansion. Given a query word, say, "retrieval", the stemmer generates possible word forms of this word like "retrieve", "retrieved", and "retrieving". All these words are treated as synonyms of "retrieval" and used in its expansion. These synonyms are combined with equal weights in the query. The effect of this stemming on the T queries and the TD queries are studied separately for the TEXT and OCR'd collection. Initially, no OCR error modeling is used and just the effect of query time stemming is investigated. Finally, queries expanded by stemming are further expanded by following OCR error modeling approach.

3.3 Results

We submitted 9 official runs to RISOT 2011, and we scored an additional six unofficial runs after the submission deadline to explore additional conditions. Table 1 presents the reported results of our officially submitted runs. As the table illustrates, the best results (by MAP, which we focus on in this section) were obtained using TD queries on clean text. Stemming yielded statistically significant improvements for each condition in which it was tried alone (runs 1:4, 3:7, 8:11, 10:14; $p<0.01$ by a two-tail t-test). Modeling single errors (1-Error in Table 1) yielded statistically significant improvements over not modeling errors for TD queries (runs 2:3, 6:7; $p<0.05$) the similarly sized apparent improvement for T queries was not statistically significant (runs 9:10, 12:14: $p>0.05$). Shorter queries are known to exhibit greater cross-topic variability, so the failure to observe statistical significance with T queries is not surprising. Modeling single errors with stemming yielded improvements but not statistically significant improvements for either TD queries (runs 2:6; $p>0.05$) or for T queries (runs 9:12; $p>0.05$). Modeling multiple errors (M-Error in Table 1) yielded meaningful improvement over modeling single errors for T queries (run

[1] http://www.lemurproject.org/

13:12; $p<0.01$) but not for TD queries (run 5:6; $p>0.05$); that comparison is only available in our results for the unstemmed condition. Our best absolute results for the OCR condition resulted from combining modeling of single errors with stemming; this achieved between 83% and 85% of the MAP achieved for the corresponding TEXT condition (for TD and T queries, respectively).

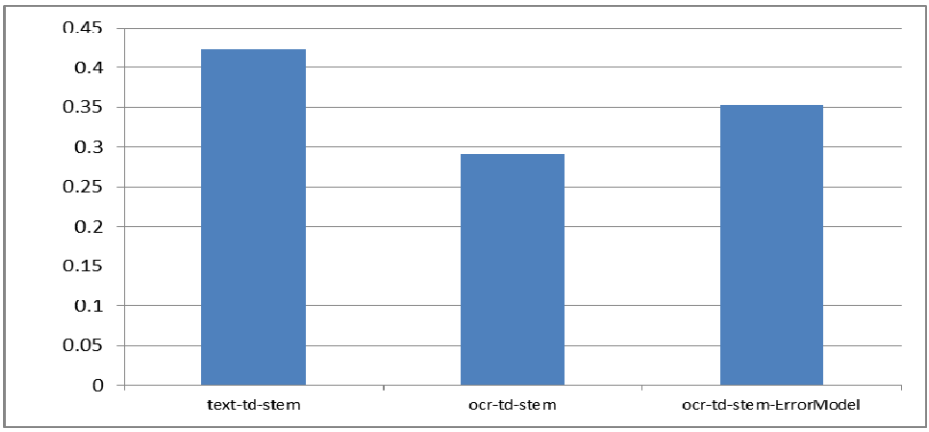


Fig. 1. Comparison MAPs for retrieval from Text and OCR collections

Figure 1 compares the MAP values for three comparable conditions: TEXT-TD-Stemmed; OCR-TD-Stemmed; and OCR-TD-Stemmed-1-Error model. Figure 2 breaks the comparison between OCR-TD-Stemmed and OCR-TD-Stemmed-1-Error-Model down by query. The statistically significant improvement in MAP from 0.2915 to 0.3521 results from more topics showing increases in average precision than declines, but that the average precision does indeed decline for a substantial number of topics. This suggests that continued work on the optimal use of error modeling might be productive.

4 Conclusions and Future Research

Our RISOT 2011 experiments provide clear evidence that our way of using OCR error modeling results in improved retrieval effectiveness for OCR'd Bengali documents, but much remains to be done. Most directly, we need to study our modeling of multiple errors over a broader range of conditions (e.g., in conjunction with stemming), and we need to look at those results in detail in order to see why we are not getting measurable gains over modeling single errors in the conditions that we have tried to date. Second, it might be worthwhile to try error modeling in conjunction with overlapping n-grams. Third, because different techniques may make different types of errors it could prove useful to try combination of evidence techniques. Finally, we have already started to experiment with Cross Language Information Retrieval (CLIR), taking advantage of the English translations of the

Table 1. Results for official and unofficial runs

Run	Official	Q	Doc	Stem	1-Error	M-Error	MAP	MAP%	P@5	P@10	Rprec
1	umdT2	TD	TEXT	Yes			0.4229	100%	0.4413	0.3554	0.3940
2	umdE5	TD	OCR	Yes	Yes		0.3521	83%	0.3858	0.3008	0.3294
3	umdO4	TD	OCR	Yes			0.2915	69%	0.3109	0.2489	0.2832
Run	**Official**	**Q**	**Doc**	**Stem**	**1-Error**	**M-Error**	**MAP**	**MAP%**	**P@5**	**P@10**	**Rprec**
4		TD	TEXT				0.3449	100%	0.3826	0.3152	0.3250
5		TD	OCR			Yes	0.3434	100%	0.3577	0.2962	0.3131
6		TD	OCR		Yes		0.3251	94%	0.3388	0.2694	0.3068
7	umdO3	TD	OCR				0.2293	66%	0.2717	0.2217	0.2336
Run	**Official**	**Q**	**Doc**	**Stem**	**1-Error**	**M-Error**	**MAP**	**MAP%**	**P@5**	**P@10**	**Rprec**
8	umdT1	T	TEXT	Yes			0.3487	100%	0.3457	0.2967	0.3234
9	umdE2	T	OCR	Yes	Yes		0.2967	85%	0.3200	0.2686	0.2794
10	umdO2	T	OCR	Yes			0.2349	67%	0.2505	0.2187	0.2364
Run	**Official**	**Q**	**Doc**	**Stem**	**1-Error**	**M-Error**	**MAP**	**MAP%**	**P@5**	**P@10**	**Rprec**
11		T	TEXT				0.2649	100%	0.3109	0.2630	0.2548
12	umdE1	T	OCR		Yes		0.2588	98%	0.3000	0.2583	0.2489
13		T	OCR			Yes	0.2418	91%	0.2804	0.2424	0.2305
14	umdO1	T	OCR				0.1922	73%	0.2440	0.1901	0.1978

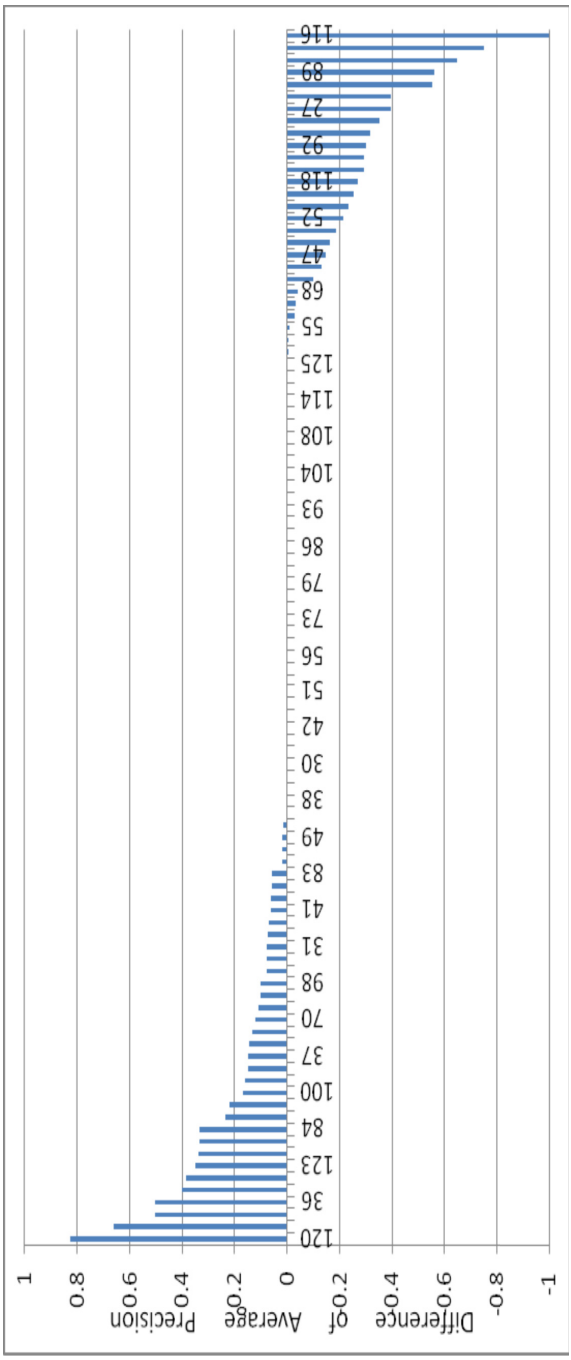


Fig. 2. Topicwise comparison between OCR-TD-Stem-1Error and OCR-TD-Stem

FIRE topics that are already available. A small English-Bengali dictionary consisting of about 35,000 words is being used for translating the queries. Techniques like Blind Relevance Judgment Feedback, bidirectional translation model are being investigated for this purpose.

Acknowledgment. Help from Jiaul Paik of CVPRU, Indian Statistical Institute, Kolkata, India for implementing some parts of the present experiment is sincerely acknowledged.

References

1. Darwish, K., Oard, D.: Probabilistic Structured Query Methods. In: Twenty-Sixth International ACM-SIGIR Conference on Research and Development in Information Retrieval, Toronto, Canada, pp. 338–344 (2003)
2. Wang, J., Oard, D.: Matching Meaning for Cross-Language Information Retrieval. In: Information Processing and Management (2011) (in press)
3. Paik, J.H., Mitra, M., Parui, S.K., Järvelin, K.: GRAS: An Effective and Efficient Stemming Algorithm for Information Retrieval. ACM Transactions on Information Systems 29(4) (2011) (to appear)

Retrieval from OCR Text: RISOT Track

Kripabandhu Ghosh and Swapan Kumar Parui

Indian Statistical Institute, Kolkata, West Bengal, India

Abstract. In this paper, we present our work in the RISOT track of FIRE 2011. Here, we describe an error modeling technique for OCR errors in an Indic script. Based on the error model, we apply a two-fold error correction method on the OCRed corpus. First, we correct the corpus by *correction with full confidence* and *correction without full confidence* approaches. Finally, we use query expansion for error correction. We have achieved retrieval results which are *significantly* better than the baseline and the difference between our best result and the original text run is *not significant*.

1 Introduction

Retrieval from Indic Script OCR'd Text or RISOT was introduced as a pilot task in the third Forum for Information Retrieval Evaluation (FIRE) which was held in Mumbai, India in December, 2011 [1]. The objective of the task was to test Information Retrieval (IR) effectiveness on OCRed documents generated from machine printed text in Indic script. The track was aimed at presenting a common platform for researchers of two different communities - IR and OCR - where they can collaborate to solve a problem which requires expertise from both these communities.

In this task, the participants were provided with a relevance judged collection of 62,825 articles of a leading Bangla (an Indic script) newspaper, Anandabazar Patrika (2004-2006). For each article, both the original digital text and corresponding OCR results were given. Relevance judgments were available for 92 topics. The OCR output was obtained by rendering each digital document as a document image, which was then processed by a Bangla OCR system. The dataset was available at www.isical.ac.in/∼clia/data_2011.html. The participants in the 2011 RISOT pilot task had to develop IR techniques to retrieve documents from these collections and report the MAP and Precision@10 separately for the digital text collection and for the OCR collection.

The OCRed documents naturally have erroneous words due to OCR misclassification. So, we first try to model the OCR error pattern at the symbol level. On the basis of that, we intend to correct the OCRed corpus and also to develop a query expansion method. For indexing and retrieval, we used Terrier 3.0[2] using DFR [2]-BM25 [12] model.

[1] `www.isical.ac.in/∼fire`
[2] `http://terrier.org/`

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 214–226, 2013.

The rest of the paper is organised as follows:

We discuss the related works in Section 2. Then we describe the character classes of the Bangla script in Section 3. Next, we describe our approach in Section 4. The results are presented in Section 5. Finally, we conclude in Section 6.

2 Related Work

The RISOT task finds its roots in two previous TREC (Text Retrieval Conference) tasks : the Confusion Track and the Legal Track. The TREC Confusion track was part of TREC-4 (1995) [6] and TREC-5 (1996) [9]. In the TREC-4 Confusion Track, random character insertions, deletions and substitutions were used to model degradations. Such distortions were done on 260,000 English electronic text documents from multiple sources. For the TREC-5 Confusion Track, 55,000 government announcement documents were printed, scanned, OCRed and then were used instead. Electronic text for the same documents was available for comparison. Something quite similar was done in RISOT except that in RISOT the original documents were in an Indic script. In TREC legal track [7][13][5][3] the IIT CDIP 1.0 collection was prepared by OCRing 7 million scanned English business documents for Ad Hoc, Relevance Feedback and Batch tasks from 2006 to 2009. But the original documents were not available for reference. The Legal track was an attempt to model the real life "e-discovery" paradigm. But the OCRed errors in the corpus posed a separate challenge altogether, to go with the core IR issues, for the researchers who participated in these tasks.

In addition to TREC, some work has been done on IR from OCRed collections. Among the earliest works, Taghva et al. [8] applied probabilistic IR on OCRed text. A. Singhal et al. [1] showed that linear document length normalization models were better suited to collections containing OCR errors than the quadratic (cosine normalization) models. But nothing much has been done on OCRed Indic texts. Very recently, Language and Media Processing Laboratory[3] (LAMP), Maryland, USA is studying the aspects of retrieval from Bangla OCRed texts. Two of the researchers of LAMP happen to be among the co-ordinators of the RISOT track.

However, one can find substantial work in the literature on OCR error modeling and correction. Kolak and Resnik [10] applied a pattern recognition approach in detecting OCR errors. Walid and Kareem [11] used Character Segment Correction, Language Modeling, and Shallow Morphology techniques in error correction on OCRed Arabic texts. On error detection and correction of Indic scripts, B.B. Chaudhuri and U. Pal produced the very first report in 1996 [4]. This paper used morphological parsing to detect and correct OCR errors. Separate lexicons of root-words and suffixes were used. Our approach uses far lesser language-specific knowledge in detecting and correcting OCR errors. This paper proposes a new and simple method of error modelling that compares words in parallel corpora based on error frequency.

[3] `http://lampsrv02.umiacs.umd.edu/projdb/project.php?id=70`

Table 1. A consonant with the vowel modifiers

Modifier	া	ি	ী	ু	ূ	ৃ	ে	ৈ	ো	ৌ
Modified form	কা	কি	কী	কু	কূ	কৃ	কে	কৈ	কো	কৌ

3 Character Classes

In this section, we describe the different character classes in Bangla alphabet set. Bangla is highly inflectional, containing basic characters, modifiers and compound characters.

3.1 Basic Characters

There are 11 Bangla basic vowels which are অ, আ, ই, ঈ, উ, ঊ, ঋ, এ, ঐ, ও, ঔ (Hex code range : 0985-0994) and 38 Bangla basic consonants which are ক, খ, গ, ঘ, ঙ, চ, ছ, জ, ঝ, ঞ, ট, ঠ, ড, ঢ, ণ, ত, থ, দ, ধ, ন, প, ফ, ব, ভ, ম, য, র, ল, শ, ষ, স, হ, ড়, ঢ়, য়, ৎ, ং, ঃ (Hex code range : 0995-09B9, 0981, 0982).

3.2 Character Modifiers

Bangla script has both vowel and consonant modifiers.

Vowel Modifiers: The first basic vowel shown above does not have a modified version. But the other 10 vowels have modifiers which are shown in the first row of Table 1 in the same order. In other words, when a vowel occurs with a consonant or compound character, its shape gets modified. The second row of the table shows how these vowel modifiers occur with the first consonant ক.

Consonant Modifier: ঁ features as the only consonant modifier. Examples are ফাঁ, চাঁ, কঁ, ক্বাঁ . Note that a single basic consonant or a compound character can have both ঁ and a vowel modifier. Basic vowels do not normally have a modifier. However, in a few cases, ঁ sign acts as a modifier for basic vowels, for example, হ্যাঁ, আঁ etc.

3.3 Compound Characters with/without Vowel/Consonant Modifiers

A compound character consists of two or more basic consonants, for example, শ্র, ন্ত etc. শ্র is the combination of basic consonants শ and র joined by a ্ (hasanta) symbol. In Unicode শ্র is encoded as < শ > <্ > < র >. Instances of more than two basic consonants being present in the formation of a compound character are also frequent in Indic scripts. One such example is ন্দ্র which is encoded as < ন > <্ > < দ > <্ > < র >. A compound character can have both vowel and consonant modifiers as indicated above. Some examples are: শ্রু, ন্তু .

– By a *symbol*, we will mean a basic character or a compound character with or without a character modifier.

4 Our Approach

Our primary goal in error modeling here is to see how the OCR error at the symbol level behaves. In other words, if $symbol_1$ is OCRed as $symbol_2$ such that $symbol_1 \neq symbol_2$, then we consider the frequencies of such ordered error pairs of symbols. But to identify such error pairs of symbols, we first need to find the error pairs of words. Once the error pairs of symbols and their statistics are obtained, they are used for (i) symbol level correction and (ii) expansion of query terms with an aim to better the retrieval performance. Details are provided below.

4.1 Error Modeling

Determination of Original - OCRed Word Pairs: In the RISOT task, we are provided with parallel corpora - one, the original text corpus and the other, its OCRed version. So, we intend to find the original correct words and their spelling variants caused due to OCR errors.

To find such pairs, we first consider a file (that is a document), say, *original* in the original collection and its OCRed version, say, *ocred*. Then we form a list of word pairs from the two files by joining the word lists of these files side-by-side in the order they appear in the files. The resulting list contains (original word, OCRed word) pairs where the first word of the tuple comes from the file *original* and the other from the file *ocred*. Due to OCR errors several words are split into two or more words. So, even if we have the original file and the OCRed version of the same file, one-to-one mapping of an original word and its OCRed version is not trivial to obtain. We look for word matches between the original and OCRed documents sequentially. Between two consecutive matches we store the mismatched pairs. We fix a window of some size (we empirically choose the size as 5 words). For every mismatched original word we look for the match in the OCRed file within the window. The algorithm is illustrated in Table 2. The first match is the pair (রাম, রাম). The next match is found in the pair (ভজন, ভজন). The mismatched pairs (পুরাণ, থরাণ) and (কণকবাবু, কণকবান্ন) are stored in the file $MismatchedWordPairs$.

Then we merge the $MismatchedWordPairs$ for all the documents and their corresponding OCRed version. Thus, we get the suspected correct - erroneous word pairs for the whole collection, say,
$MismatchedWordPairsAll$. We call these pairs "suspected" bacause, we assume that the number of correct words is the same as that of wrong versions of the same words in the parallel corpora between two consecutive word matches. For example, this number is 2 in Table 2. But this may not always be true, since a word may be split into multiple words in the OCRed corpus. However, since we are looking for the pairs over the whole corpora, we expect to capture the

Table 2. Illustration of Find Correct - Erroneous Word Pairs Algorithm

word from original file	word from corresponding OCRed file	match status
.	.	MATCH
রাম	রাম	MATCH
পুরাণ	থরাণ	MISMATCH
কণকবাবু	কণকবান্ব	MISMATCH
ভজন	ভজন	MATCH
.	.	MATCH

Table 3. Sample original - OCRed word pairs produced by OCR errors

original word	OCRed word
মূক	মক
পুরাণ	থরাণ
ওঠার	ওঠ।র
কণকবাবু	কণকবান্ব
কণ্ঠে	কভে

highly frequent error patterns by our algorithm. Some of these pairs are shown in Table 3.

Determination of Original - OCRed Symbol Pairs: Now our next task is to find the correct - erroneous symbol pairs from $MismatchedWordPairsAll$. With the suspected original - OCRed word pairs in hand, we split both the words into the constituent *symbols*. Let us consider the (পুরাণ, থরাণ) word pair (second entry in Table 3). The original word পুরাণ is broken down into পু, রা and ণ. Similarly, the OCRed word থরাণ gives থ, রা and ণ. Next, we compare the corresponding *symbols*. So, the symbols পু, রা and ণ are compared with the symbols থ, রা and ণ respectively. The unmatched *symbols* are the suspected original - OCRed wrongly mapped pairs. In our example, (পু, থ) is the resulting pair.

Table 4. Sample original - OCRed symbol pairs produced by OCR errors

original symbol	OCRed symbol
মূ	ম
পু	থ
বু	ন্ব
ণ্ঠে	ভে

Table 5. Original modifier symbols and compound characters with their frequencies and percentages of correct/incorrect mappings

Symbol	Total no.	Correct mapping	Modifier missing	Other incorrect mappings
hrossou-kaar(ু)	97476	9.54%	12.68%	77.78%
dirghou-kaar(ূ)	18770	0%	59.49%	40.51%
hri-kaar(ৃ)	8402	0.26%	75.12%	24.62%
chandrabindu(ঁ)	15237	0%	0%	100%
okaar(া)	9110	0%	0.36%	99.64%
oukaar(ৌ)	751	0%	0%	100%
aa-kaar(া)	213319	73.76%	2.23%	24.01%
hrossoe-kaar(ি)	106941	67.33%	17.03%	15.64%
dirghoe-kaar(ী)	23819	67.66%	1.15%	31.19%
ae-kaar(ে)	108496	87.47%	1.28%	11.25%
oi-kaar(ৈ)	1062	64.03%	0.47%	35.5%
compound characters	135659	53.79%	5.12%	41.09%

We take care of the cases where a compound character is split into its constituent symbols. Consider the word লঙ্ঘনin which the middle symbol is a compound character. Suppose the compound character ঙ্ঘ (i.e., <ঙ > < ্ > < ঘ >) is broken down into ঙ and ঘ, so that লঙ্ঘন is mapped to লঙঘন. Here after ল is matched correctly with ল (in the OCRed word), the algorithm detects ঙ and ঘ as the broken pieces of ঙ্ঘ. So, in this case ঙ্ঘ is mapped to ঙঘ. Finally, ন is matched with ন in the OCRed word.

Thus, we obtain the suspected pairs of correct and erroneous symbols of the whole corpus in
$MismatchedSymbolPairsAll$. Some of these pairs are shown in Table 4.

4.2 Candidate Symbol Selection

For error modeling, we consider only the symbols involving character modifiers and the compound characters since these types of symbols are usually prone to OCR errors. Table 5 shows such candidates with the rates of correct OCR recognition. The figures are computed on the RISOT dataset. For example, a symbol containing the modifier hrossou-kaar (ু) is recognized correctly in 9.54% of the cases, only the modifier is lost in 12.68% of the cases and in 77.78% cases the symbol is recognized as some other symbol (first entry in Table 5). From the table we can see that for hrossou-kaar (ু), dirghou-kaar (ূ), hri-kaar (ৃ), chandrabindu (ঁ), okaar (া) and oukaar (ৌ) the recognition rate is very poor (<= 9.54%). On the other hand, for aa-kaar (া), hrossoe-kaar (ি), dirghoe-kaar (ী), ae-kaar (ে) and oi-kaar (ৈ), the recognition rate is much higher (>= 64.03%). So, we consider the symbols containing one of the first six modifiers for error modeling. In addition, we consider the symbols containing the compound characters although more than 50% of compound characters are correctly recognized. This is because the compound characters involve the juxtaposition of multiple characters and this may lead to complex error patterns.

4.3 Corpus Correction

In this step, we want to correct some unique errors in the corpus by substituting erroneous symbols in the corpus with their suspected correct forms. Since this method comprises in eliminating a particular error pattern from the corpus, it has the risk of distorting potential key terms in the corpus. So, we usually intend to replace the symbols which are uniquely mapped from the original symbols. That is, a unique original symbol is mapped to an OCRed symbol. But, if the OCRed symbol is a valid symbol, we have a problem. While correcting the wrongly formed valid symbols in the OCRed text, we may substitute all the other correctly mapped instances of those valid symbols in the corpus. For example, suppose "son" is uniquely mapped to "sun" due to OCR error. Since we are looking to correct errors at symbol level, the situation suggests that "u" in the OCRed text should be replaced with "o" to get back the original word. But suppose "turn" is correctly OCRed. Then, our attempted error correction strategy will convert all "turn"s to "torn"s! Even if a symbol is uniquely mapped to some valid symbol, it is not necessary that all the instances of that symbol in the OCRed text are wrongly mapped. To do this sort of substitutions, we must ensure that, in our example, "o" is never correctly mapped. We have treated such cases in Section 4.4 (Query expansion) below. However, if "son" is mapped to "s)n", since ")" (in a word) is an invalid symbol and the mapping is unique, ")" can be safely replaced by "o". So, we have replaced an OCRed symbol only if it is invalid.

Table 6. Original - OCRed symbol pairs : correction with full confidence

original symbol	OCRed symbol	freq
রা	রা ।	153
লা	লা ।	126
শা	। ।	113
ভা	ভা ।	70
রু	রু	61
শী	7া ।	51
রো	রো ।	47
হ্নি	কৃি	42
মো	মো ।	38
র্গি	গিা	37
টা	টা ।	35
ক্যা	ক্যা ।	32
র্দো	)ো	32
অ্যা	অ্যা ।	30

Table 7. Original - OCRed symbol pairs : correction without full confidence

original symbol	OCRed symbol	freq	original symbol	OCRed symbol	freq
মা	0া	735	গু	গ্া	493
সা	0া	264	ঁঝা	গ্া	39
যা	0া	50	ফাঁ	গ্া	11
0	0া	33	স্টা	গ্া	8
পাঁ	0া	20	গুঁ	গ্া	2
শ্রা	0া	9	ং	গ্া	1
ম	0া	5	কো	গ্া	1
স	0া	5	ভা	গ্া	1
স্না	0া	5	ম	গ্া	1
ধা	0া	4	মে	গ্া	1
স্মা	0া	3	য	গ্া	1
আ	0া	2	রু	গ্া	1
য	0া	2	স	গ্া	1
শ্র	0া	2			
অ	0া	1			
কা	কা ।	265	তা	তা ।	138
কে	কা ।	1	প	তা ।	1
			ভা	তা ।	1
			ভো	তা ।	1
জা	জা ।	116	ধী	(া	12
র্জা	জা ।	1	1	(া	1
সো	জা ।	1	তা	(া	1
			নি	(া	1
			য়	(া	1

Algorithm 1. Choose Correct - Erroneous Symbol Pairs for query expansion

```
For each group with the same Correct Symbol do the following steps, one
after another:
if OCRed Symbol is Invalid then
  IGNORE the pair
end if
if Highest Frequency for the group >= 50 then
  Take the pair with the highest frequency
else
  if for the group, Correct Symbol is NEVER CORRECTLY MAPPED
  then
    Take the pair with the highest frequency
  end if
end if
```

Correction with Full Confidence. The OCR process can be thought of as a mapping from the set S_1 of the original symbols to the set S_2 of the OCRed symbols. Now, some symbols in S_2 are invalid in the sense that they do not belong to S_1. An OCR error at the symbol level means a symbol x in S_1 is mapped to y in S_2 and $x \neq y$. Note that for such an error, y may or may not belong to S_1. If x is mapped to y such that y is not in S_1 and there is no other $x^{'}$ (in S_1) that is mapped to y, then all the occurrences of y in the OCRed documents are corrected as x. This is called *correction with full confidence*. We made a list of original - OCRed symbols where the OCRed symbol is invalid and is mapped uniquely. There were 306 such pairs in the corpora. Some of the highest occurring pairs are listed in Table 6. It may be noted that y may represent one or more symbols.

Correction without Full Confidence. Suppose $x_1, x_2, ..., x_n$ $(n > 1)$ are mapped to y that is not in S_1. We computed the number of occurrences (or frequency) of the event that x_i is mapped to y (denoted as $(x_i \rightarrow y)$) for all i and sort them in descending order. Suppose these sorted frequencies are $f_1 \geq f_2 \geq ... \geq f_n$. Table 7 shows six such instances of y. If $(x_k \rightarrow y)$ has f_1, the largest frequency, all occurrences of y in the OCRed documents are replaced as x_k. Clearly, in the process, this replacement strategy leads to f_1 valid corrections and $f_2 + ... + f_n$ invalid corrections in the sense that when y is supposed to be corrected as $x_j (j \neq k)$, it is actually corrected as x_k. We call this strategy *correction without full confidence*. However, with this strategy we do not lose anything but make f_1 valid corrections. The only risk here is that if a word having the symbol $x_j (j \neq k)$ in an original document becomes another valid word w after x_j gets replaced by x_k, then it may lead to a wrong document being retrieved in response to a query having w as a term. For a single pair, the error-free correction would account for the correction of maximum 153 invalid symbols (i.e., for রা→ রা ৷ pair). The number is quite larger for erroneous correction. For

মা→ 0া pair, the maximum frequency is 735 and for গু→ গ্া the maximum is 493. So, our proposed strategy has succeeded in recovering more erroneous OCRed symbols.

4.4 Query Expansion

We use query expansion to take care of the mappings of valid symbols in the original corpus to valid symbols in the OCRed one. We choose original - OCRed symbol pairs by Algorithm 1. Since we handle the invalid OCRed symbols in our error correction section, this algorithm ignores them (lines 2-4). Next, we consider the highly occurring original - OCRed symbol pairs found over the entire parallel corpora (lines 4-6). These pairs represent the most frequently occurring error patterns in the OCRed corpus. The threshold for frequency is empirically chosen as 50. The remaining part of the algorithm (lines 8-10) is aimed at recovering those original symbols which are never correctly mapped to the OCRed corpus and whose occurrences are rare in the corpus (mostly, rare compound characters). A few such symbols are shown in Table 8. By expansion, we mean that for each word containing at least one symbol that has been mapped to some other symbol in the OCRed corpus, we keep the original word and add the variant of the word. For example, suppose *o* is mapped to *u* and the query contains the word *torn*. Then the word *turn* will be addded to the query. Some examples of expanded queries are shown in Table 9. For topic 26, সিঙ্গারে is added as a variant of সিঙ্গুরে since ঙ্গু is wrongly recognised as ঙ্গ. Similarly, for query 31, কাশমীরে, বিভির্ঘ and আতঙ্কাবাদী are variants of কাশ্মীরে, বিভিন্ন and আতঙ্কবাদী respectively.

Table 8. Original - OCRed symbol pairs : symbols never correctly mapped and frequency < 50

original symbol	OCRed symbol	highest freq	words in topics containing the symbol
শ্মী	শমী	2	কাশ্মীরে
ঙ্ঘ	ঙঘ	4	সিঙ্ঘল, সঙ্ঘর্ষ, লঙ্ঘন
গ্ু	শ্ু	37	গ্ুপের
ন্ধ্র	ন্ধ	26	অন্ধ্র
ব্দী	ফী	22	শতাব্দীতে

Table 9. Examples of query expansion

Topic no.	Query before expansion	Query after expansion
26	সিঙ্গুরে জমি অধিগ্রহণ সমস্যা	সিঙ্গুরে সিঙ্গরে জমি অধিগ্রহণ সমস্যা
31	কাশ্মীরে বিভিন্ন জায়গায় আতঙ্কবাদী হামলা	কাশ্মীরে কাশমীরে বিভিন্ন বিভির্ঘ জায়গায় আতঙ্কবাদী আতঙ্কাবাদী হামলা

Table 10. Comparative performance - text, baseline and improved runs

Run description	MAP	RPrec	P@10
Text (original)	0.3540	0.3307	0.3239
OCRed (baseline)	0.2318	0.2257	0.2293
Run submitted in FIRE 2011	0.3193	0.3087	0.2859
Correction with full confidence	0.2617	0.2619	0.2554
Correction with/without full confidence	0.2781	0.2644	0.2696
Query expansion with all symbols in topic set	0.2824	0.2612	0.2565
Query expansion using Algorithm 1	0.2888	0.2648	0.2663
Query expansion using Algorithm 1 but removing symbols of freq < 50	0.2752	0.2501	0.2609
Correction with/without full confidence + Query expansion with all symbols in topic set	0.3361	0.3179	0.2946
Correction with/without full confidence + Query expansion using algorithm 1	**0.3424**	**0.3229**	**0.3098**

5 Results and Analysis

We have started with the OCRed version of the original corpus as the baseline run. We then incrementally apply the error correction strategies described so far on the OCRed corpus. We apply a two-pronged error correction strategy - corpus cleaning on one hand and query expansion on the other. Initially, we apply only one of the two methods at a time. First we use only corpus correction and only query expansion. Then we combine them. The run submitted in FIRE 2011 is also reported. In this run, we applied the discussed algorithms at a pre-mature stage. It also included some manual efforts in choosing error patterns. All the remaining runs shown in the table are fully automatic.

We attempt to clean the corpus by error-free strategy (correction with full confidence) and follow it by the erroneous one (correction without full confidence). Table 10 shows that erroneous corpus correction produced better performance than its error-free counterpart.

Next, we study the effects of query expansion in error-correction. We apply a few variations of our proposed Algorithm 1 on the erroneous corpus. Instead of applying Algorithm 1, we choose the variations of all the symbols (with vowel modifiers and compound characters) irrespective of the maximum frequency for the original symbol. Algorithm 1 slightly outperforms in this case. But using the rules for all the symbols is computationally more expensive than using them selectively. For the given topic set, there are 161 rules for Algorithm 1 as opposed to using all the symbols for expansion, where the number of rules is 283. We also remove the symbols with frequency < 50. By doing so, we have the risk of losing two types of mappings:

1. original symbols which are never correctly mapped in the corresponding OCRed documents
2. original symbols which are most of the time correctly mapped but are occasionally mis-mapped

The second class is not of much concern. But there are several symbols which have low frequencies in the corpus and the OCR never recognizes them correctly. A few of such original symbols with their OCRed forms and the highest frequencies are shown in Table 8. All of them are compound characters and their frequencies are relatively low in the corpus. On some occasions, the OCR failure resulted in removal of the compound character information (represented by *hasanta* symbol), e.g., শ্মী being mapped to শমী or chopping off modifiers, e.g., ন্ধ্র being mapped to ন্ধ. On other occasions, the original symbols are mapped to totally different symbols, e.g., গ্রু is mapped to শু. But since they were never recognized correctly by the OCR, ignoring them would result in losing vital words in the topic like inflectional variants of কাশ্মীর (Kashmir), সিঙ্ঘল (Singhal), গ্রুপ (group), অন্ধ্র (Andhra), শতাব্দী (century), etc. From the table we can see that removal of these symbols resulted in a fall in the performance.

We see that query expansion alone has produced better results than corpus correction alone. Next, we combine query expansion and corpus correction. In other words, we run the retrieval with expanded queries on the corrected OCRed corpus. Here, we use expanded topics produced by considering all the symbols in the topic and also, those produced by Algorithm 1. The corpus is the OCRed one corrected by erroneous correction scheme. The best results are produced by Algorithm 1.

Our best run was found *significantly* better than our baseline by paired t-test ($p = 0.000009491 < 0.05$). Also, the difference of our best run with the run on original text was *not statistically significant* by paired t-test ($p = 0.07762 > 0.05$).

6 Conclusion and Future Work

In this paper, we have proposed a method to alleviate the effect of OCR errors in retrieval performance by error modeling, query expansion and correction. But, in the process of doing so, we have used knowledge about the Indic scripts (here Bangla). Understanding of different characteristics of Indic scripts was vital at the inception of our method development. We would like to test our approach on other Indic scripts (e.g. Hindi, Marathi, etc.). Also, we plan to design an error modeling method in future that will use less script-dependent knowledge.

References

1. Salton, G., Singhal, A., Buckley, C.: Length normalization in degraded text collections. In: Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval, pp. 149–162 (1996)
2. Amati, G., Van Rijsbergen, C.J.: Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans. Inf. Syst. 20(4), 357–389 (2002)
3. Baron, J., Hedin, B., Tomlinson, S., Oard, D.: Overview of the trec 2009 legal track. In: The Eighteenth Text Retrieval Conference (2009)

4. Chaudhuri, B.B., Pal, U.: Ocr error detection and correction of an inflectional indian language script. Pattern Recognition 3, 245–249 (1996)
5. Tomlinson, S., Oard, D., Hedin, B., Baron, J.: Overview of the trec 2008 legal track. In: The Seventeenth Text Retrieval Conference (2008)
6. Harman, D.: Overview of the fourth text retrieval conference. In: The Fourth Text Retrieval Conference, pp. 1–24 (1995)
7. Lewis, D., Baron, J., Oard, D.: The trec-2006 legal track. In: The Fifteenth Text Retrieval Conference (2006)
8. Borsack, J., Taghva, K., Condit, A.: Results of applying probabilistic ir to ocr text. In: The Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 202–211 (1994)
9. Kantor, P., Voorhees, E.: Report on the trec-5 confusion track. In: The Fifth Text Retrieval Conference, pp. 65–74 (1996)
10. Kolak, O., Resnik, P.: Ocr error correction using a noisy channel model. In: HLT, pp. 149–162 (2002)
11. Magdy, W., Darwish, K.: Arabic ocr error correction using character segment correction, language modeling, and shallow morphology. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, pp. 408–414 (2006)
12. Robertson, S.E., Zaragoza, H.: The probabilistic relevance framework: Bm25 and beyond. Foundations and Trends in Information Retrieval 3(4), 333–389 (2009)
13. Baron, J., Tomlinson, S., Oard, D., Thompson, P.: Overview of the trec 2007 legal track. In: The Sixteenth Text Retrieval Conference (2007)

Overview of the Personalized and Collaborative Information Retrieval (PIR) Track at FIRE-2011

Debasis Ganguly, Johannes Leveling, and Gareth J.F. Jones

CNGL, School of Computing
Dublin City University
Dublin 9, Ireland
{dganguly,jleveling,gjones}@computing.dcu.ie

Abstract. The personalized and collaborative information retrieval (PIR) track at FIRE 2011 was organized with the aim of extending standard information retrieval (IR) ad-hoc test collection design to facilitate research on personalized and collaborative IR by collecting additional meta-information during the topic (query) development process. A controlled query generation process through task-based activities with activity logging was used by each topic developer to construct a final set of topics. The standard ad-hoc collection is thus accompanied by a new set of thematically related topics and the associated log information. This better simulates a real-world search scenario and encourages mining user information from the logs to improve IR effectiveness. A set of 25 TREC formatted topics and the associated metadata of activity logs were released for the participants to use. We illustrate the data construction phase in detail and also outline simple ways of using the additional information from the logs, such as query and document click history, to improve retrieval effectiveness.

1 Introduction

One major challenge in information retrieval (IR) is the potential to adapt retrieval results according to personal choices and preferences. Different users may enter the same query string into a search system, but their information needs can be vastly different. The notion of relevance depends upon factors such as the domain knowledge of the searcher, information gained from reading previous documents in the past, and general search behavior of a searcher, e.g. how many documents he normally reads before reformulating his search [2].

In a conventional laboratory evaluation scenario of ad-hoc IR, participants are given a document collection and a set of queries (topics). The task of the participating systems is then to retrieve documents which satisfy the information need expressed in each query. Such a traditional evaluation framework does not provide enough information to facilitate personalized IR. This information includes: a) closely related topics formulated and assessed by different people, and b) additional meta-information such as the prior queries executed and the documents viewed.

The process of TREC-style topic development is artificial and does not resemble iterative query reformulation in real search activities where typically a user of a search system enters an initial query, reads a few top ranked documents before reformulating

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 227–240, 2013.

the initial query possibly multiple times until his information need is satisfied. The final query, based on the content read thus far, retrieves one or more relevant items which satisfy his information need up to this point. Our main hypothesis is that this iterative process of topic development is more similar to the real-world search than a search based on a single topic. Test data for PIR thus constitutes an ad-hoc IR test collection enriched to include logs of user behaviour leading to a set of *final* TREC type test queries.

To our knowledge, little or no research has been directed towards collecting and providing metadata[1] for the query development process under the framework of an ad-hoc retrieval dataset. Unlike the available ad-hoc IR test data suites, there is no standard test data set for a direct comparison between various personalized retrieval strategies. Existing work on user studies for personalized IR differ in the tasks given to the users and the document collection on which these tasks are to be performed by the users [10,8,7]. Also over the years, researchers proposing novel personalized retrieval methodologies have used their own in-house web log data collected from a variable number of users [15,16,11,14]. It is therefore difficult to compare between these retrieval techniques due to the lack of a common test data suite for personalized IR. Our work attempts to bridge this gap by providing a common evaluation framework to test various personalized IR systems developed in the recent years and encourage development of new ones.

The rest of the paper is organized as follows: Section 2 surveys work on user modelling and personalized IR, Section 3 presents our approach to generating user logs in a controlled environment. Section 4 outlines experiments and results with the test data, and Section 5 concludes the paper with a brief summary and outlook.

2 Related Work

This section provides a brief survey of the previous research in search personalization which can be categorized in two broad areas, first the studies on detecting patterns in user search behaviour through assigned navigational tasks, and second exploiting user search history to improve retrieval. We highlight the two areas of study in personalized search in the next two subsections, which is followed by a brief survey of the work done towards log analysis and user query sessions in evaluation forums. We conclude the section with an outlook on the previous studies, and how our proposed evaluation framework is different to these.

2.1 User Task Studies

Recent works on the study of user search patterns include that of Kellar et al. [7]. They report that users spend most of their time viewing pages and extensively use the browser functions for information gathering tasks. This establishes the necessity for extensive user studies of information gathering tasks. Kelly and Belkin [8] report that there is no direct relationship between the display time of a document and its usefulness, and that

[1] Metadata includes all information from the search history for all query formulations.

display times differ significantly according to a specific task and according to individual users. White and Kelly [17] show that tailoring document display time thresholds for implicit relevance feedback based on task grouping is beneficial for personalization. Liu and Belkin [10] designed a method for decomposing tasks into sets of (in)dependent subtasks and show that the task stage for an independent subtask is helpful in predicting document usefulness. This is attributed to the fact that users gain knowledge across stages regarding the usefulness of documents.

2.2 Personalized Search

Tan et al. show that prior queries and clickthrough data can be used to improve retrieval effectiveness [15]. Their experiments were based on web search logs collected over a period of one month from four users. Relevance assessment was done on the top 20 documents retrieved for each query and standard evaluation metrics such as MAP and P@5 were reported. It was observed that prior queries alone were not effective. However, usage of clicked results yielded the highest increase in search accuracy, suggesting the usefulness of clickthrough as implicit feedback. Matthijs and Radlinski modeled users by their search interests collected from the browsing history and re-ranked the retrieved results based on user profiles [11]. They used re-ranking methods involving usage of long-term search interest terms present in user profiles and also those involving using clicked documents for implicit relevance feedback. For evaluation they used i) an offline method of graded relevance judgments for top 50 documents, and ii) an online method of using an interleaved ranked lists. Teevan et al. report that implicit measure click entropy (the number of different results that different people clicked) is highly correlated with explicit judgments of relevance by individuals [16]. Sontag et al. proposed a probabilistic model for personalized search utilizing prior queries and clicked documents to model a user's search intent [14]. Wei et al. show that the retrieval results for a current user can be improved by the implicit ratings of documents collected from other users with related search interests [9].

2.3 Evaluation Forums

The LogCLEF[2] log analysis initiative provides log data from different providers, but these datasets lack relevance assessments [3].

TREC 2010 introduced the Session track[3], where the motivation is to form and evaluate a session of related queries [6]. This track involves modifying an initial query into a more general query, a more specific query, or one addressing another facet of the information need. Query reformulations were done manually in the session track of 2010. In 2011, queries were compiled from the descriptive questions of the TREC Question Answering track and the narratives from the TREC 2009 Million Query track that had clear sets of subtopics [5]. The queries (topics) thus formulated were presented at random to real users who had to use a customized search interface to search on a given

[2] http://www.promise-noe.eu/mining-user-preference/logclef-2011/
[3] http://ir.cis.udel.edu/sessions/

topic until their information need was satisfied. User actions logged by the search interface including previous queries, clicked document URLs, dwell times etc. were made available to the participants.

Our proposed track is different from Session Track 2010 because firstly, we do not manually form query variants but expect the participants to contribute in generating search data, and provide them with search logs from other participating and volunteering topic developers. Secondly, our track is not primarily concerned with query sessions, but with categorizing users based on their interests and with exploring whether individual searchers can profit from information about similar searches or users.

Although the method of log data collection is somewhat similar to the Session Track 2011 [5], there are some differences. Firstly, the document collection is different in that Session track 2011 uses the ClueWeb09 collection, whereas we use the FIRE-2011 ad-hoc English collection. Secondly, instead of randomly assigning the search topics to the users, we display a list of categories from which the users are free to choose one. Thirdly, in our session data collection approach, we ask the user to fill up a brief summary, the quality of which enables us to know about how seriously have the users been pursuing their search topic of interest. Furthermore, our interface also provides a *bookmarking* feature which is absent in the Session track 2011 interface. The most prominent difference is in the way the retrieval quality is evaluated. The Session track 2011 aims to evaluate faceted search, i.e. it favours retrieval systems which retrieve relevant documents for each aspect (sub-topic) of the given query. The objective of the PIR track however, is to measure the improvement in the retrieval quality of the final query entered into the system.

NTCIR-9[4] organized the Intent task, where topics are formed automatically by random sampling from Chinese web search query logs. A difference between our proposed task and the NTCIR Intent task is that the latter deals with web search and uses a bottom-up approach (starting from existing query logs), whereas we try to address elements of personalization with a top-down approach, aiming to create interaction logs.

2.4 Outlook

Although previous research provides evidence that IR systems can present more relevant documents to individual searchers (hence addressing personalization) by exploiting his browsing information and that of users with similar search interests, these typically involve collecting in-house web search logs over a period of time by installing a browser plug-in. The main problem with this methodology is that it is very difficult to reproduce the experiments without an access to the log files. Difficulty in reproduction of the experimental results and in cross comparison of various retrieval approaches is also due to the dynamic nature of the web collection itself. The related work on user studies motivated us to generate a log of the entire topic creation process to make information about the search process available to the retrieval systems, which can help tuning IR systems to user-specific needs. The lack of availability of user logs greatly limits the research that can be undertaken in this area outside of industrial research laboratories. Our proposed methodology is designed to make a set of query logs freely available and

[4] http://www.thuir.org/intent/ntcir9/

distributable to promote personalized IR research. In summary, there are two important differences compared to previous research:

1. Presence of queries with overlapping information needs is ensured by the fact that queries have to be chosen from a set of pre-defined query categories. This also in turn ensures the presence of users with similar interests which can potentially be utilized in a collaborative setting to improve search performance.
2. A static corpus is used for search and logging the topic development process, because experiments which are based on web search logs are not reproducible due to the dynamic nature of web documents and also do not facilitate cross comparison between retrieval systems.

3 PIR Evaluation Framework

3.1 Data Construction Methodology

To promote our approach to automatic "closed-box" personalized and collaborative IR experiments and encourage researchers to use and contribute to this method, we organized the pilot track named personalized IR (PIR)[5] in the Forum of Information Retrieval and Evaluation[6] (FIRE) 2011. The closed set of documents, on which browsing activities were logged, is the FIRE 2011 English ad-hoc document collection comprising of news from the Indian newspaper *The Telegraph* from 2001 to 2010 and news from Bangladesh, comprising of almost 400K documents in total. A web service[7] was developed and hosted, which was used during the topic development phase to browse through the collection and construct topics. The retrieval engine which the web interface uses at the back end is Lucene.

The sequential steps towards creation of a topic are as follows. A topic developer logs into the system with a registered user ID and is henceforth refered to as a *user* of the web interface. The user then goes through a search phase (selecting the search category, submitting queries and viewing result documents) and a topic formulation and evaluation phase (summarizing the found information, formulating the final topic, and assessing relevance for documents). The system logs all these user actions. The topic development procedure is illustrated in Fig. 1. We explain each step as follows.

Category Selection. The system presents a list of broad search *categories* from which the topic developer has to select one. The categories are listed below.

The categories were intended to represent broad search domains of news articles which a user can freely browse, gain knowledge during the search phase and finally enter his own specific query. Also deriving the topics from a pre-defined list of categories is intended to ensure development of related topics with overlapping information needs for different users. This in turn ensures that the logs can potentially be utilized for collaborative search, where the retrievval results of a current user's query can be improved by search histories of other users for queries of that particular category [9].

[5] http://www.cngl.ie/Fire-PIR/
[6] http://www.isical.ac.in/~clia/
[7] http://www.cngl.ie/Fire-logs/

1. Social impact on land acquisition
2. Honour killing incidents
3. Indian cricketing events
4. Indian tourism
5. Relation of India with its neighboring countries
6. Indian political scams
7. Healthcare facilities
8. Indian paintings and painters
9. Indian traditions and customs
10. Indian armed forces
11. Indian education policy
12. Bollywood movies
13. Adventure sports
14. History of Indian vernaculars
15. Terrorist attacks

The search categories have been selected in accordance to the TREC guidelines of topic development, which involves performing trial retrievals against the document set and choosing topics for which the result set is not too small or too large [4]. Also to ensure that we have roughly a uniform distribution of queries across these broad categories, our system dynamically adapts the list for different users e.g. if enough queries have been formed from category-1 and none from category-2, then the system removes category-1 from the list presented to a new user.

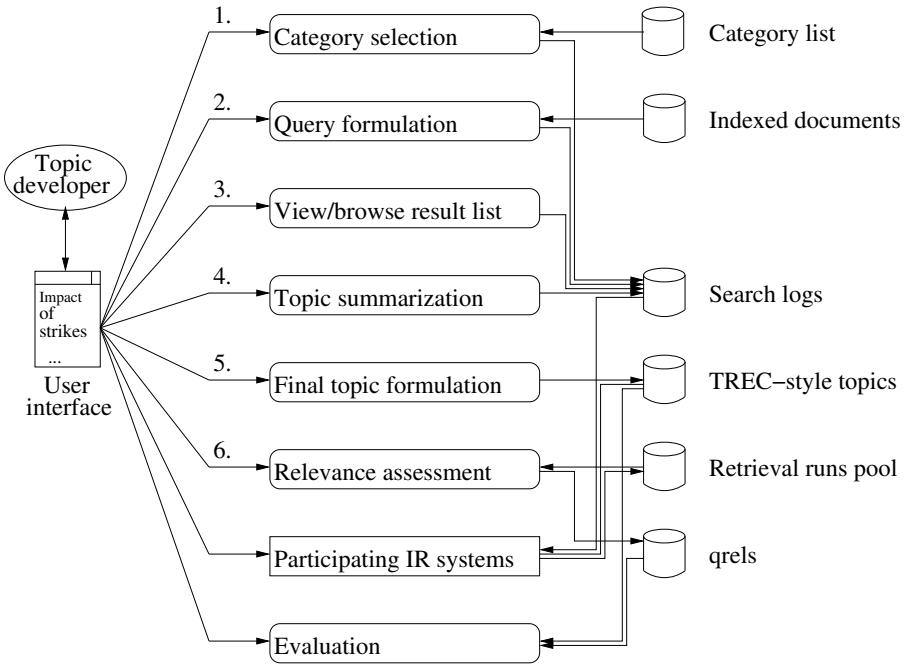


Fig. 1. Data flow diagram of the topic development phase

Query Formulation and Retrieval. After selecting a category, the user iterates through query formulations, retrieving different documents at each iteration. We log the query titles of intermediate queries because prior queries have been used successfully to improve retrieval performance for the final query of a search session [15].

View/Browse Result Documents. The user can read documents retrieved in the previous step by clicking on the result URLs and can also bookmark a document to refer to it later. The user is expected to go through a series of query reformulations before he feels that he has gained sufficient knowledge to enter a topic to the system.

Topic Summarization. The system presents a form where the topic developer has to enter a report/summary on the subject matter of the chosen category. The content of the summary acts as a means to ensure that the topic developer has indeed gained knowledge about the search category and that the final topic indeed is based on information from documents viewed by him. A randomly entered summary on the other hand can indicate an ill-formed test topic unsuitable for the evaluation experimentation.

Final Topic Formulation. As a next step the system asks them to form a TREC-formatted topic based on the knowledge gained thus far. This query aims at one user-specific, "personalizable" aspect of the initial search category, i.e. one particular aspect of the category that the topic developer is especially interested in. We refer to this topic as the *final topic* (because it is the final topic entered by a user for a particular search category) or a *test topic* (because this topic is released in the test topic file to be used for the final evaluation). The topic developers have to fill in the *title*, *description* and *narrative* fields for the query, describing the information need by a phrase, a full sentence, and a description of which documents are relevant and which are not. These TREC-style topics serve as input for the IR runs.

Relevance Assessment. Relevance assessments are based on the pool of submissions. The developer of a topic was assigned the responsibility to mark the relevant documents according to the relevance criteria expressed in the narrative field of the topic provided by him. We aimed to investigate if there exists a *personal* notion of relevance, i.e. how often a document is relevant for two different topics belonging to the same category. Another aspect of research was to see how many of the documents bookmarked or viewed for a long time by topic developers for a category are actually relevant for the test topic entered in that category.

3.2 An Illustrative Example

Let us assume a topic developer selects the example topic "Terrorist attacks" from the pre-defined list of search categories. He then enters a series of queries in the system (e.g. "Terrorist attacks Kashmir", "Terrorist organizations", "Terrorism in Mumbai"), views the documents, bookmarks some of them and starts gaining knowledge about the category given to him. Figure 2 illustrates this topic development process. For the chosen initial search category, the user issues a query Q_1 and gets a ranked list of returned documents $\{D_1^1, \ldots D_m^1\}$. A subset of relevant documents (viewing or bookmarking a document might be an implicit indicator of relevance) is used to reformulate Q_1 into Q_2. The released meta-information corpus would thus contain each intermediate query Q_i, the set of top documents returned for Q_i namely $\{D_1^i, \ldots D_m^i\}$ and the actions of the user.

After going through a few iterations, the user then fills up the topic summary and submits his topic titled "26/11 Mumbai attack" with an appropriate description and narrative. Later on he also has to assess documents for relevance for this topic.

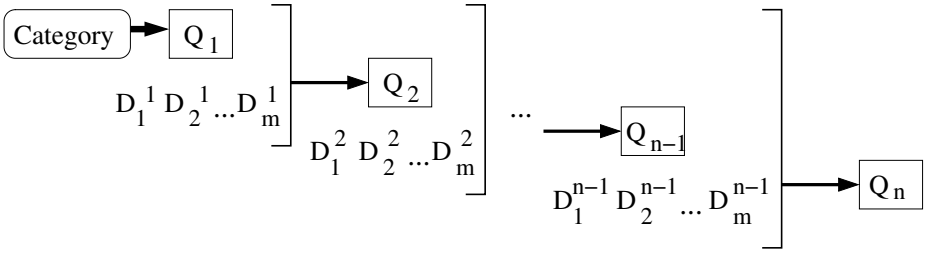


Fig. 2. Structure of the meta-information

Consider another topic developer who selects the same topic. He also browses documents through the system and eventually ends up with a topic titled "Ajmal Kasav trial". Now we see that the topic "Ajmal Kasav trial" is related to the query "26/11 Mumbai attack" since Ajmal Kasav was the person convicted of the murders on the 26th Nov night in Mumbai.

The assumption is that the set of documents that the first user views or bookmarks can also be the potential relevant candidates for the second query. Also the principle of recommender systems can be applied here where we predict the relevance of a document by its popularity (of viewing or bookmarking) among different users executing similar queries. Building up on the hypothesis that information obtained from one user might benefit satisfying the information need(s) of the other, the intended challenge for the participants was to develop ideas of how to increase retrieval effectiveness for all searchers with similar search intentions by exploiting the browsing history of all such searchers.

3.3 Data Details

Test Topics. Twenty-five TREC formatted test topics having two additional tags of *username* and *categoryname* were released. For each topic, the string enclosed within the *username* tag denotes the registered user name of the developer for this topic and the *categoryname* tag contains the name of the category selected while developing it. An example topic is shown in below.

Search Logs. A single line in the log file represents a search event by a user, where a search event can be either a click on the URL or on the bookmark button corresponding to a document, returned as part of a ranked list in response to a query execution. The logs are formatted as comma separated values and have the following structure: *user name, category, query name, document name, rank of this document, action performed on this document*, and *time stamp*. The first field i.e. the name of the user serves as an identifier

<top>
<num>1</num>
<username>*******</username>
<categoryname>Adventure sports</categoryname>
<title>rock climbing india</title>
<desc>The intent is to find general information about the rock climbing sport in India.</desc>
<narr> Relevant documents are documents that give information or news about the rock climbing sport in India. Relevant documents will also be ones that discuss achievements of rock climbers who climbed mountains in India.
</narr>
<top>

to trace the originator of the event. The second field i.e. the name of the category is used to identify the top level search category from which the event was generated. This is particularly useful in restricting investigation of browsing history within a single search session for a single user or for a group of users who chose the same search category. The next field i.e. the *query name* is the seach string for which this particular event (click or bookmark) occurred. The fourth and fifth fields are the name and the retrieval rank of the document which was clicked or bookmarked. The sixth field distinguishes between the two types of action possible which is one of *resultclick*, denoting that the user clicked on the URL of this document, and *bookmark*, which indicates that the user bookmarked this document for refering to it later. The last field records the time stamp of the event.

4 Experiments and Results

Twenty-six participants registered for the PIR track. However there were no official runs being submitted. In the absence of runs from participants, we adopted simple retrieval approaches utilizing the additional meta-information from logs. This section thus reports the experiments which we did with the collected log data to demonstrate the potential usefulness of additional meta-data information for retrieval.

4.1 Experimental Setup

The retrieval model used for all the baseline runs is Language Modeling (LM) implemented within SMART[8]. We used the standard SMART stopword list and the default stemmer of SMART, which is a variant of the Lovin's stemmer. The LM implementation of SMART employs a Jelinek Mercer smoothing [13]. The smoothing parameter λ was empirically set to 0.3.

The first retrieval run named *NoLog* is a simple ad-hoc IR run using the titles of queries and without involving any pseudo-relevance feedback (PRF). PRF involving query expansion using $R = 5$, i.e. assuming that top 5 documents are pseudo-relevant, and using $T = 5$, i.e. adding 5 selected terms from these documents to the query,

[8] `ftp://ftp.cs.cornell.edu/pub/smart/`

Table 1. Summary of the runs

Run name	PRF	Intermediate query titles	Clicked documents	Bookmarked documents
NoLog	No	No	No	No
NoLogFdbk	Yes	No	No	No
LogQry	No	Yes	No	No
LogQryFdbk	Yes	Yes	No	No
LogClicks	No	No	Yes	No
LogClicksFdbk	Yes	No	Yes	No
LogBookmarks	No	No	No	Yes
LogBookmarksFdbk	Yes	No	No	Yes

constitutes the run *NoLogFdbk*. The term selection in *NoLogFdbk* is based on LM term scores as defined by Ponte in [12]. These runs do not use any information from the logs and hence are standard IR runs to be used as baselines. To generate retrieval results using additional information from logs, we used:

i) Intermediate queries that a user entered in the search system before formulating the final test query. This has also been studied in [15].
ii) Documents clicked by a user as pseudo-relevant documents. This approach has been studied in most personalized search approaches such as [15,11,14].
iii) Documents bookmarked by a user as pseudo-relevant documents.

Table 1 summarizes each run name along with the associated methodology.

4.2 Log Processing

In order to extract out the information about the intermediate query titles and the viewed documents efficiently at the test query execution phase, we preprocessed and organized the log data into a two level hash indexed data structure. This is because given a test query string and the associated identity of the user and category of this query, respectively refered to as *current user* and *current category*, we would quickly want to narrow down on the subset of logging events which is useful for this test query. The top level of the log data structure is thus indexed by the category name, which quickly narrows down to the search to the current category. The next level of indexing is on the user name which extracts out logs only for the current user. The log records at the leaf level of this bi-level index is organized as a list, where each list element stores an intermediate query name entered into the search system and a pointer to the list of retrieved document names in response to that query. Figure 3 shows a schematic organization of the data structure. We build up this in-memory data structure only once in the preprocessing stage. We iterate through each log record from the CSV and insert it in its proper place in the bi-level hash table. Refering to the Figure 3, if we want to get log information for a test query whose category name is C_5 and user name is U_3 we first query the left-most hash table with C_5, follow the pointer and reach another hash table where we query with U_3. Following the pointer we reach the activity records for that user who

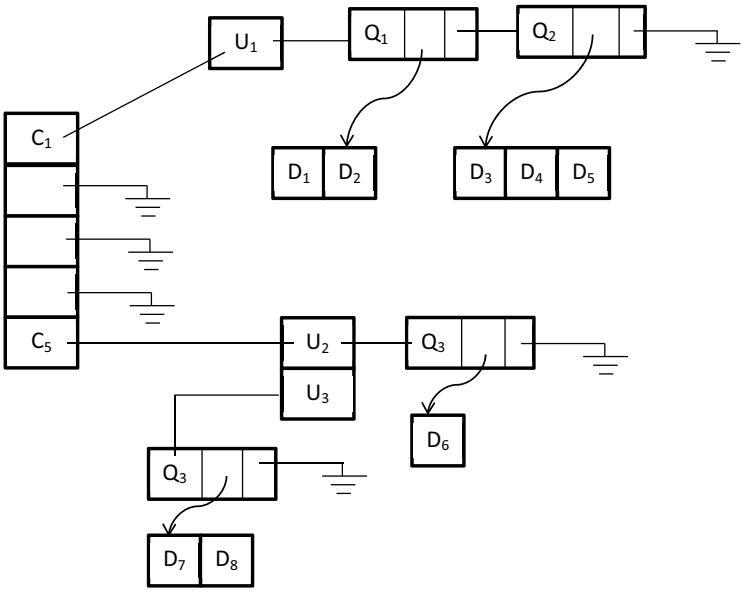


Fig. 3. Data structure for efficient log processing

in this particular example has entered only one query Q_3 and clicked on documents D_7 and D_8.

4.3 Retrieval Methodology

With this description of the organization of the log data in memory, we are now ready to outline the methodology for generating the retrieval runs. The *LogQry* runs use only the intermediate query strings i.e. the query strings marked as Q_is in Figure 3. For a test query we retrieve the intermediate query strings for the current user and the current category. This precisely constitutes the list of queries entered into the search system by the current user during the search session which led to the development of the final test query. We add these query strings into the test query title and report the retrieval run as *LogQry*. We adopted a very simple method of appending the prior query context to the current test query in contast to the reported work in [15], which involved building up language models for each prior query and then combining these models with that of the current query. A pseudo-relevant feedback run using *LogQry* as the initial retrieval run, gives *LogQryFdbk*. The rationale behind using the intermediate queries for query expansion is that these intermediate query strings serve to act as the actual intent of the user during the search session, and thus additional words can help model the user interests better and can help retrieve more relevant documents.

The *LogClick* run uses the clicked documents as the potential set of pseudo relevant documents. The rationale is that viewed documents are likely to be relevant for the intermediate query and in turn for the final test query assuming that the final test query reformulation has been affected by the contents of the viewed documents. *LogClickFdbk*

Table 2. Retrieval results

Run Name	PRF Parameters		Evaluation Metrics			
	# docs. (R)	# terms (T)	P@5	P@10	MAP	GMAP
NoLog	-	-	0.3167	0.2750	0.2604	0.1254
LogQry	-	-	**0.3833**	0.3167	0.3014	0.1558
LogClicks	# clicked docs.	5	0.3789	**0.3316**	**0.3065**	**0.2167**
LogBookmarks	# bookmarked docs.	5	0.3417	0.2792	0.2784	0.1143
NoLogFdbk	5	5	0.4167	0.3167	0.3535	0.1243
LogQryFdbk	5	5	**0.5083**	**0.3917**	**0.4056**	**0.2064**
LogClicksFdbk	# clicked docs. + 5	5	0.4750	0.3583	0.3813	0.1922
LogBookmarksFdbk	# bookmarked docs. + 5	5	0.4917	0.3792	0.4006	0.1978

is the PRF run over *LogClick* which involves standard PRF with query expansion. The *LogBookmarks(Fdbk)* runs are very similar in nature except that these use bookmarked documents instead of clicked ones.

4.4 Results

Evaluation was done by manual assessments for the pool of top 30 documents obtained from the three retrieval runs *NoLog*, *LogQry* and *LogClicks*. Table 2 shows the standard evaluation metrics averaged over the set of 25 test topics for all the retrieval runs. Results are grouped together into two categories: i) the ones which are initial retrieval results without PRF; and ii) the ones which use standard PRF over its corresponding retrieval result of the first category. Among the non-PRF runs, the highest precision at top 5 is obtained by *LogQry* which uses intermediate query strings as expansion terms over the test query. Also, the result of *LogClicks* shows that clicked documents used as relevant documents can improve retrieval effectiveness. *LogClicks* in fact achieves the highest MAP and GMAP values among the non-feedback runs. This result in fact conforms to the observations reported in [15] that clickthrough provides the maximum gains in search effectiveness.

LogQry yields the best feedback run which indicates that intermediate query strings can be effective cues in predicting the search intent, more so in the presence of PRF. An interesting observation is that *LogBookmarksFdbk* yields better results than *LogClicksFdbk* which is suggestive of the fact that the set of bookmarked documents is less noisy as compared to the set of clicked documents. This is expected because a user bookmarks a document only if he feels that he needs to refer to this document for future usage, which is in turn is highly indicative of its relevance.

Another important observation is that both *LogQry* and *LogClicks* perform worse than *NoLogFdbk*, which is the standard PRF run without using logs. This suggests that although some clicked documents and prior queries are useful for relevance feedback, a selective method of choosing a subset of these would be helpful to get improvement over standard PRF.

5 Conclusions and Outlook

The proposed methodology can act as a first stepping stone towards evaluation of different retrieval systems under the same test bed of user generated logs. The log generation process has been designed to address aspects of personalization by capturing individual information needs for a broad search category. The history of the documents viewed prior to developing the final topic makes the topic development process transparent to a retrieval system. We have shown that navigational search sessions can be generated through task based browsing activities in a constrained domain or category and have also demonstrated that such activity logs can potentially be leveraged upon to improve retrieval precision at top ranks.

In the absence of any submitted retrieval run, we undertook three approaches of harnessing the meta-information from the logs i.e. using intermediate queries for expansion of the final test query, and using viewed and bookmarked documents for relevance feedback. Results show that simple retrieval strategies which use the meta-information from logs can improve search performance.

A major problem for the track this year was the lack of participation, as a result of which the quantity of logs is very small compared to the ones which has been used for personalization research [15,14]. We therefore do not have sufficient log data to obtain conclusive results. However, our preliminary evaluation effort is a positive indication towards the effective exploitation of search logs.

A plausible reason for lack of participation can be due to the fact that the domain of news articles is not very suitable for navigational searches. A collection of informative articles such as the Wikipedia may be more suitable for the task based exploratory search. In the next year of this track, we intend to use the INEX ad-hoc collection, which comprises of the full Wikipedia collection [1] for building up log data to make the search task more interesting for the topic developers. The results should motivate participants to submit their own runs next year and thus contribute towards the development of the track. Participants from related tracks such as the Sessions Track at TREC, LogCLEF at CLEF, and the intent finding track at NTCIR may also be interested to participate in this track. The task this year focused on English, but as the methodology is language-independent, it can be applied for Indian languages also used at FIRE (e.g. Bengali or Hindi) if enough interest can be raised from FIRE participants.

Acknowledgments. This research is supported by the Science Foundation of Ireland (grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (`http://www.cngl.ie`). We would particularly like to thank Keith Curtis for developing the web interface to collect logs. We would like to express sincere gratitude to Rami Ghorab, Trinity College Dublin, for making the IR framework available to us. The authors are also grateful to Maria Eskevich, Mohd. Javed, Yalemisew Abgaz and Sudip Naskar from Dublin City University and to Gaurav Arora from DAIIT, India for contributing to test topic development.

References

1. Arvola, P., Geva, S., Kamps, J., Schenkel, R., Trotman, A., Vainio, J.: Overview of the INEX 2010 ad hoc track. In: Geva, S., Kamps, J., Schenkel, R., Trotman, A. (eds.) INEX 2010. LNCS, vol. 6932, pp. 1–32. Springer, Heidelberg (2011)
2. Bates, M.J.: The Design of Browsing and Berrypicking Techniques for the Online Search Interface. Online Review 13(5), 407–424 (1989)
3. Di Nunzio, G.M., Leveling, J., Mandl, T.: Multilingual log analysis: LogCLEF. In: Clough, P., Foley, C., Gurrin, C., Jones, G.J.F., Kraaij, W., Lee, H., Mudoch, V. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 675–678. Springer, Heidelberg (2011)
4. Harman, D.: Overview of the third text retrieval conference (TREC-3). In: TREC (1994)
5. Kanoulas, E., Carterette, B., Hall, M., Clough, P., Sanderson, M.: Overview of the TREC 2011 session track. In: TREC (2011)
6. Kanoulas, E., Clough, P., Carterette, B., Sanderson, M.: Session track at TREC 2010. In: SIMINT Workshop SIGIR 2010. ACM (2010)
7. Kellar, M., Watters, C.R., Shepherd, M.A.: A field study characterizing web-based information-seeking tasks. JASIST 58(7), 999–1018 (2007)
8. Kelly, D., Belkin, N.J.: Display time as implicit feedback: understanding task effects. In: Proceedings of SIGIR 2004, pp. 377–384. ACM (2004)
9. Li, W., Ganguly, D., Jones, G.J.F.: Enhanced information retrieval using domain-specific recommender models. In: Amati, G., Crestani, F. (eds.) ICTIR 2011. LNCS, vol. 6931, pp. 201–212. Springer, Heidelberg (2011)
10. Liu, J., Belkin, N.J.: Personalizing information retrieval for multi-session tasks: the roles of task stage and task type. In: Proceedings of SIGIR 2010, pp. 26–33. ACM (2010)
11. Matthijs, N., Radlinski, F.: Personalizing web search using long term browsing history. In: Proceedings of WSDM 2011, pp. 25–34. ACM (2011)
12. Ponte, J.M.: A language modeling approach to information retrieval. PhD thesis, University of Massachusetts (1998)
13. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: Proceedings of SIGIR 1998, pp. 275–281. ACM (1998)
14. Sontag, D., Collins-Thompson, K., Bennett, P.N., White, R.W., Dumais, S., Billerbeck, B.: Probabilistic models for personalizing web search. In: Proceedings of WSDM 2012, pp. 433–442. ACM (2012)
15. Tan, B., Shen, X., Zhai, C.: Mining long-term search history to improve search accuracy. In: Proceedings of KDD 2006, pp. 718–723. ACM (2006)
16. Teevan, J., Dumais, S.T., Liebling, D.J.: To personalize or not to personalize: modeling queries with variation in user intent. In: Proceedings of SIGIR 2008, pp. 163–170. ACM (2008)
17. White, R.W., Kelly, D.: A study on the effects of personalization and task information on implicit feedback performance. In: Proceedings of CIKM 2006, pp. 297–306. ACM (2006)

Simple Transliteration for CLIR

Sauparna Palchowdhury[1] and Prasenjit Majumder[2]

[1] CVPR Unit, Indian Statistical Institute,
203 B T Road, Kolkata 700108, India
sauparna.palchowdhury@gmail.com
[2] Computer Science & Engineering,
Dhirubhai Ambani Institute of Information and Communication Technology,
Gandhinagar 382007, India
p_majumder@daiict.ac.in

Abstract. This is an experiment in cross-lingual information retrieval for Indian languages, in a resource-poor situation. We use a simple grapheme-to-grapheme transliteration technique to transliterate parallel query-text between three morphologically similar Indian languages and compare the cross-lingual and mono-lingual performance. Where a state of the art system like the Google Translation tool performs roughly in the range of 60-90%, our transliteration technique achieves 20-60% of the mono-lingual performance. Though the figures are not impressive, we argue that in situations where linguistic resources are scarce, to the point of being non-existent, this can be a starting point of engineering retrieval effectiveness.

1 Introduction

This is an experiment in cross-lingual information retrieval for Indian languages, in a resource-poor situation. We use a simple grapheme-to-grapheme transliteration technique to transliterate parallel query-text between three morphologically similar Indian languages and compare the cross-lingual and mono-lingual performance. Where a state of the art system like the Google Translation tool[1] performs roughly in the range of 60-90%, our transliteration technique achieves 20-60% of the mono-lingual performance. Though the figures are not impressive, we argue that in situations where linguistic resources are scarce, to the point of being non-existent, this can be a starting point of engineering retrieval effectiveness.

Bengali, Gujarati and Hindi, the three languages we work with in this experiment, share some of the typical characteristics of Indian languages [1]. They are inflectional[2] and agglutinative[3]. Their writing systems use a phonetic alphabet,

[1] http://translate.google.com/

[2] Inflection - In grammar, inflection is the modification of a word for expressing tense, plurality and so on.

[3] Agglutinative - Having words derived from combining parts, each with a distinct meaning.

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 241–251, 2013.

where phonemes[4] map to graphemes[5]. There is an easily identifiable mapping between graphemes across these languages. Exploiting these similarities, we use a *grapheme-to-grapheme, rule-based transliteration* [2] technique. The rules mapping graphemes in the two alphabets are constructed manually.

The manual construction is fairly easy for these three languages because the graphemes in the Unicode chart are arranged in such a way that the similar-sounding entities are at the same offset from the table origin. For example the sound 'k' is the $22^{\text{nd.}}$ ($6^{\text{th.}}$ row, $2^{\text{nd.}}$ column) grapheme in all the three languages, and one distinct grapheme represents 'k' in each language.

Two issues in CLIR is tackling synonymy[6] and polysemy[7]. Translation addresses these issues, but it needs language resources like dictionaries, thesauri, parallel corpora and comparable corpora. On the other hand transliteration is able to move the important determinants in a query like out-of-vocabulary (OOV) words and named-entities (NE), across languages, fairly smoothly.

We retrieve from our collections using the original query and its translated (using the web-based Google Translation tool) and transliterated versions, and compare the performance in the rest of the paper. The Section 2 places our work in context, describing the related work in Indian Language IR (ILIR). Section 3 briefly mentions our benchmark collections. A detailed description of the experiments is in Section 4. Our transliteration technique is explained there. The results are discussed in Section 5. We close our exposition with conclusions, limitations and suggestions for future work in Section 6.

2 Related Work

Transliteration of query-text to a target language is an important method for cross-language retrieval in Indian languages because language resources are scarce, and transliteration can move NEs and OOV words fairly smoothly from one language to another. NEs and OOV words being important determinants of information-need in many queries, protecting them from distortion helps improve retrieval effectiveness. A common next-step to transliteration is fixing the defective NEs and OOV words. ILIR has recently been evaluated by the *Forum for Information Retrieval Evaluation*[8], where several transliteration techniques were tried ([3], [4]). Kumaran et al. [5] tries combining several machine transliteration modules. They use English, Hindi, Marathi and Kannada, and leverage a state-of-the art machine transliteration framework in English. Chinnakotla et al. [2] applies a rule-based transliteration technique using Character Sequence

[4] Phoneme - A phoneme is the indivisible unit of sound in a given language. It is an abstraction of the physical speech sounds and may encompass several different phones.

[5] Grapheme - The smallest semantically distinguishing unit in a written language. Alphabetic letters, numeric digits, punctuations are examples of graphemes.

[6] Synonymy - Being synonymous; having same meaning.

[7] Polysemy - A word having multiple meanings.

[8] `www.isical.ac.in/~fire`

Modelling (CSM) to English-Hindi, Hindi-English and Persian-English pairs. Our work is an empirical approach, focusing on a few Indian languages that share similar syntax, morphology and writing systems.

3 Benchmark Collection

The test collection[9] we used is the latest offering of the 3rd. FIRE workshop held in 2011. We used the Bengali, Gujarati and Hindi collections, and all the 50 queries in each of these languages. The queries were formulated from an information-need expressed in English and translated to six Indian languages by human translators.

4 Retrieval Runs

At the outset we describe the entire procedure in brief. We worked with Bengali (*bn*), Gujarati (*gu*) and Hindi (*hi*). We set up retrieval runs over several variations of the indexed test collections and the queries, using Terrier-3.5 [6]. The resources at hand were the test collections, queries, qrels, stop-word lists and stemmed word-lists for the three languages. We used the statistical stemmer; YASS [7].

Referring to the graphical representation of the experiment in Figure 1, Table 1, 2 and 3 may help the reader follow the description in this paragraph. Starting with a query in one language (the source language), its text was translated and transliterated to another language (the target language). The transliteration was redone by stopping and stemming the source. Thus each source language text yielded three versions of that text in the target language.

The transliteration technique simply added an offset to the hexadecimal Unicode value of each character in the alphabet. There being no strict one-to-one mapping between graphemes between the source and the target languages, manually defined mappings were used where necessary (explained in Section 4.1 on transliteration).

So, as an example, for Bengali as the target language, we ended up with 3 types of text in Bengali (*bn.gu.g*, *bn.gu* and *bn.gu.p*), sourced from Gujarati (*gu*) and 3 more (*bn.hi.g*, *bn.hi* and *bn.hi.p*) sourced from Hindi (*hi*). The prefix *bn.gu*, is of the form *target.source*, and is suffixed by letters denoting the variations. The absence of the suffix denotes the text obtained by our transliteration technique. The *.g* suffix marks the text as obtained by translation using Google Translation tool, and the *.p* suffix marks the text as obtained by our transliteration technique after pre-processing by stopping and stemming the source. Including the original Bengali query (*bn*), we had 7 (3 + 3 + 1) versions of Bengali query text. Putting all the string in a set we get R = {*bn*, *bn.gu.g*, *bn.gu*, *bn.gu.p*, *bn.hi.g*, *bn.hi*, *bn.hi.p*} for one source-target language pair.

[9] `http://www.isical.ac.in/~fire/data.html`

For each of the 7 versions in set R, we set up 3 retrieval runs by varying the query processing steps; no-processing or the empty step (e), stopping-and-stemming (sS), and query expansion (x) denoted by the set R1 = {e, sS, x}. Another 2 variations were done for each of these three; one using the topic *title* and another using the *title*-and-*description* fields of the queries, denoted by the set R2 = {T, TD}. Summing it up, we had R X R1 X R2 runs, or, 7 * 3 * 2 = 42 runs for each language. Working with 3 languages, we submitted 42 * 3 = 126 runs at the 3rd. FIRE workshop.

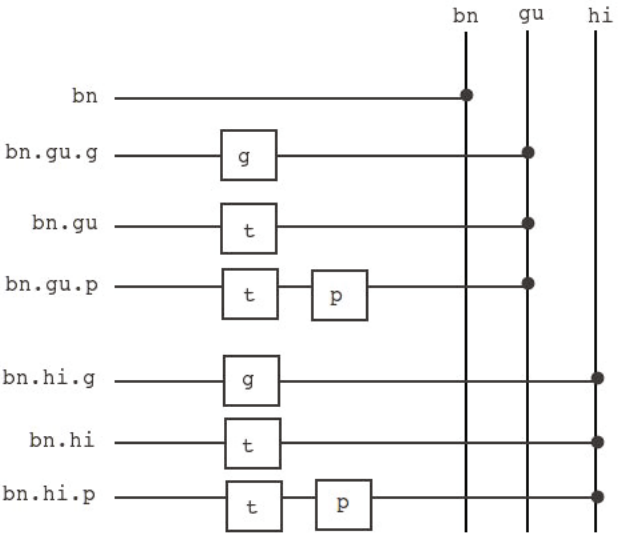


Fig. 1. The way the seven types of Bengali query text were generated. The diagram flows from right to left. The three source languages are at the top right, and lines tapped from them lead to the target language versions on the left. The *bn.gu* prefix denotes a *target.source* language pair. g is the Google Translation tool, t is our transliteration technique and p chips in as a stopping-and-stemming step of pre-processing before going through t.

4.1 Transliterating Graphemes

In *Unicode Standard 6.0*, 128 code-points are allocated to each Indian language script. The Devanagari script, used for Hindi (henceforth, we use the phrases 'Hindi script' and 'Devanagari script' interchangeably), assigns a grapheme to all code-points except one, whereas the Bengali script has 36, and Gujarati 45, missing points. The relative positions of the phonetically similar letters being identical in the Unicode chart for the three languages, adding and subtracting hex offsets worked for most cases but for the missing code-points. We had to take care of many-to-one mappings (which occurred frequently when mapping

Table 1. Set R. The seven types of Bengali query text. The strings are best read off from right to left.

1.	*bn* - The original Bengali query text
2.	*bn.gu.g* - Translating Gujarati to Bengali using the Google Translation tool.
3.	*bn.gu* - Transliterating Gujarati to Bengali using our technique.
4.	*bn.gu.p* - Transliterating, after pre-processing by stopping and stemming the query text.
5.	*bn.hi.g* - Type 2 using Hindi as the source language.
6.	*bn.hi* - Type 3 using Hindi.
7.	*bn.hi.p* - Type 4 using Hindi.

Table 2. Set R1. Three ways of retrieval. *e* is the no-processing or the empty step. The *sS* step needs the collection to be indexed with stopping and stemming enabled. For *x* we use the stopped and stemmed index.

1.	*e* - No processing whatsoever, query and document text remains as it is.
2.	*sS* - Stop-words removed and remaining words were stemmed using YASS.
3.	*x* - Query expansion (Terrier-3.5's default; Bo1).

Table 3. Set R2. Two more ways of retrieval, using the *title* and *description* fields of the queries.

1.	*T* - Retrieval using only the *title* of a query.
2.	*TD* - Retrieval using the *title* and *description* fields of a query.

Hindi to the other scripts) and mapping letters to NULL (which was equivalent to ignoring them), when a suitable counterpart was not found. Here is how we handled such situations, described for the reader who has some familirity with Indian scripts.

(a) When a grapheme had no counterpart in the target language: Devanagari vowel sign OE (0x093A) was mapped to NULL (0x0). Bengali AU (0x09D7) length mark was mapped to NULL.

(b) When a grapheme had a phonetically similar counterpart: Devanagari short A (0x0904) was mapped to A in Bengali (0x0985) and Gujarati (0x0A85). Gujarati LLA (0x0AB3) was mapped to Bengali LA (0x09B2). Hindi has a LLA (0x0933) too.

(c) When a grapheme's usage changed in the target language: ANUSVARA (for a nasal intonation) is used independently in Bengali (0x0982), but in Hindi (0x0902) it almost always resides as a dot on top of a consonant and results in pronouncing N, so it was mapped to Bengali NA (0x09A8).
(d) VA and YA was correctly assigned. VA is pronounced YA in Bengali. Bengali does not have a VA. Whereas YA in Bengali is YA in the other two languages, and not YYA, which also exists.

All in all we had to manually map 18 Bengali, 8 Gujarati and 50 Hindi graphemes, as shifting by hex offsets would not work for them. The transliteration program may be download from a public repository[10].

5 Results and Analysis

The results show all the 126 runs in Figure 2 and Table 4. The bar charts give us a quick visual comparison of the runs. Our baseline is the mono-lingual run using the original query (the leftmost bars in each of the seven stacks in each chart). It is the best possible performance in the current set-up. The output of the Google Translation tool is our cross-lingual baseline. It is a state of the art tool which is expected to have made use of language resources, helping us compare to it our resource-poor methods.

The retrieval runs show improved performance in the increasing order $e < sS < x$, and $T < TD$. Oddly, for *gu.hi* $T > TD$.

Therefore x-TD retrieval runs (retrieval with query expansion and the *title-and-description* fields) are the best results amongst all the runs. Query text translated using the Google Translation tool performs over a wide range, from 59-87% of the mono-lingual performance. In comparison our transliteration technique's performance ranges from 19-60%.

The pre-processing step of stopping and stemming the query text before transliterating them does not seem to provide any benefit. It was surmised that stopping and stemming the source text would leave behind cleaner text, as input to the translieration step, by removing the large number of inflections, but this is not corroborated by the results. YASS being a statistical stemmer, tuning it to vary its output, could well be a way to experiment further with the pre-processing.

Gujarati and Hindi seem to be morphologically closer in that the performance of queries across these two languages are better than the cases where Bengali is involved.

A per-query view of our results, in Figures 3 to 8, show how conversion between Bengali and the other two languages have not produced good results. The Bengali charts are significantly sparse, as many queries simply failed to retrieve enough relevant documents. Gujarati-Hindi conversion have been relatively better.

[10] `https://bitbucket.org/sauparna/irtools/src/26e3bed0e338/mapchar.c`

Table 4. The comparison of runs in terms of percentage of the mono-lingual performance. The first row of each block is the MAP value for the monolingual run. For a description of the run types refer to Table 1. The rest of the values are % of the mono-lingual MAP. For example, at row *hi.gu.g*, which denotes retrieval using the query translated from *gu* to *hi* using the Google Translation tool, and column *TD* and *x*, the performance is 87% of the mono-lingual Hindi run. The transliteration technique denoted by *hi.gu* in the same column but in the next row, makes it to 53%. Note that our pre-processing step does not turn out to be useful.

Bengali	T			TD		
Query type	e	sS	x	e	sS	x
bn	0.2218	0.2538	0.2910	0.2744	0.3242	0.3704
bn.gu.g	59	60	61	60	60	62
bn.gu	23	21	23	23	26	29
bn.gu.p	22	20	21	21	22	22
bn.hi.g	66	63	60	62	59	60
bn.hi	22	25	27	22	28	31
bn.hi.p	10	23	25	11	20	25
Gujarati	T			TD		
Query type	e	sS	x	e	sS	x
gu	0.2236	0.2578	0.2797	0.2611	0.2860	0.3095
gu.bn.g	64	64	64	67	69	68
gu.bn	19	20	27	20	25	30
gu.bn.p	15	18	22	17	23	29
gu.hi.g	65	67	67	72	74	77
gu.hi	54	53	60	28	30	28
gu.hi.p	25	32	37	12	17	17
Hindi	T			TD		
Query type	e	sS	x	e	sS	x
hi	0.1442	0.1532	0.1706	0.1631	0.1750	0.1877
hi.bn.g	59	59	59	69	70	76
hi.bn	19	21	19	24	23	27
hi.bn.p	18	29	31	20	32	37
hi.gu.g	65	71	71	82	83	87
hi.gu	44	42	48	49	49	53
hi.gu.p	39	39	43	42	42	49

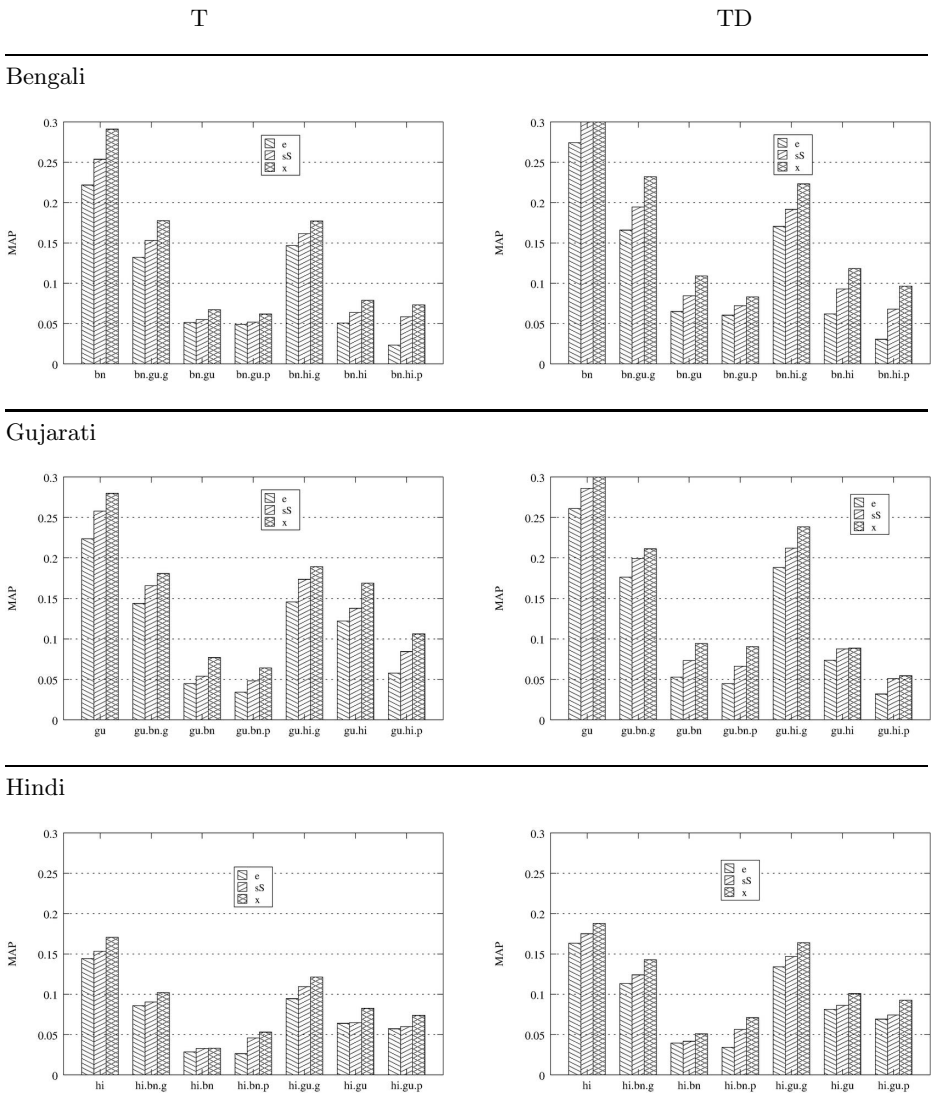


Fig. 2. The six charts show the MAP values obtained for the seven kinds of retrieval runs. Column 1 and 2 is for the T and TD runs, and a row each for the languages Bengali, Gujarati and Hindi. And in each stack of three bars in each chart, the left-to-right ordering of the patterned bars corresponds to the elements of the set R1 = $\{e$, sS, $x\}$ in order.

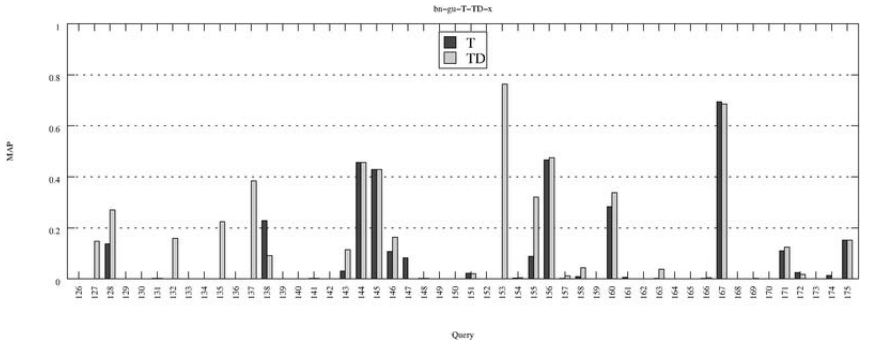


Fig. 3. bn.gu

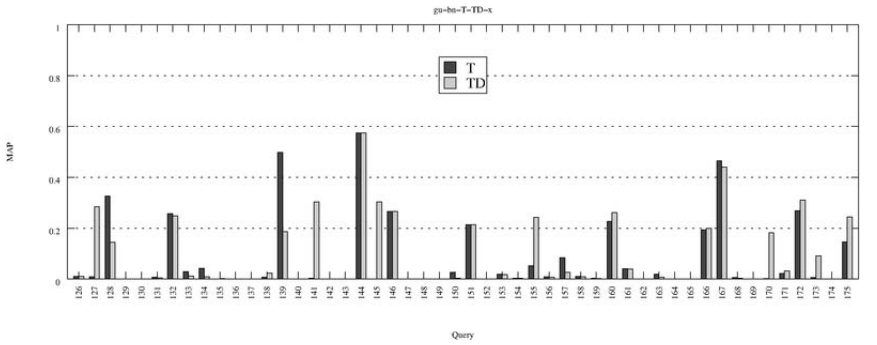


Fig. 4. gu.bn

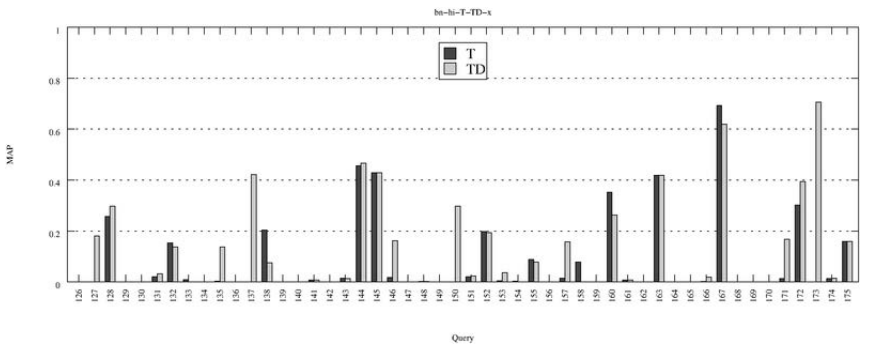


Fig. 5. bn.hi

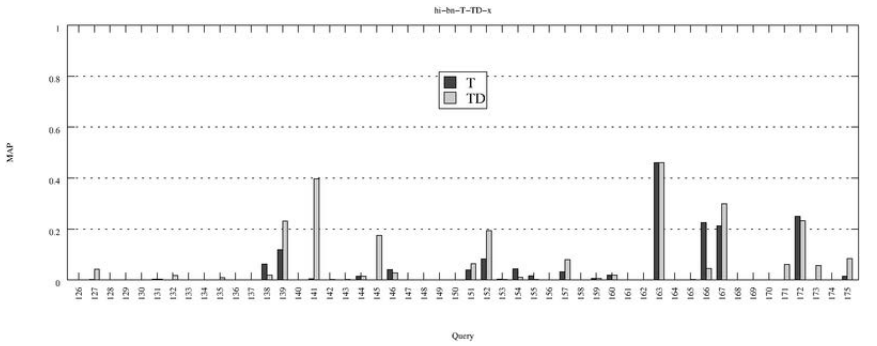


Fig. 6. hi.bn

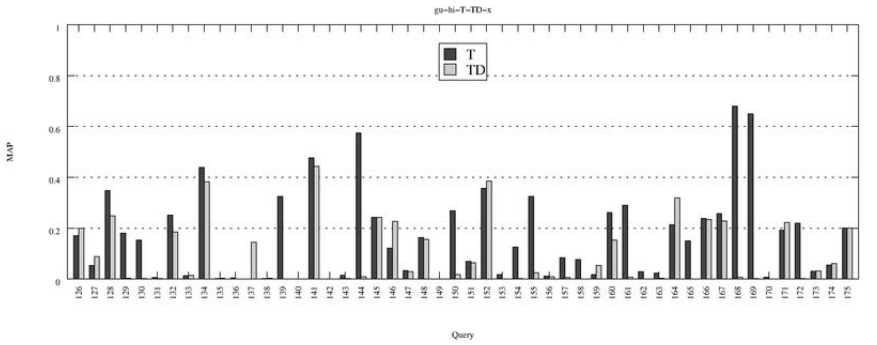


Fig. 7. gu.hi

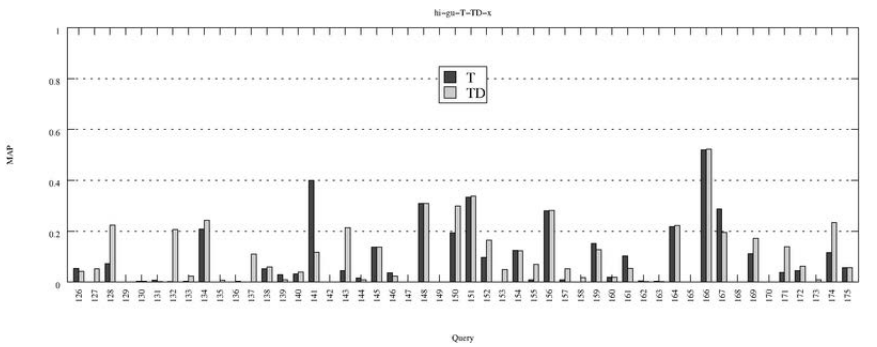


Fig. 8. hi.gu

6 Conclusion and Future Work

We have made an attempt to make use of the similarity in the scripts of a group of Indian languages to see how retrieval performance is affected. It could well have worked for any pair of language, sharing these traits, whose graphemes could be assigned a mapping manually. There are deficiencies in our methods, for example, we have not taken care of spelling variations. A spelling using I (simple 'i') in one Indian language may use II (stressed 'i') in stead. The words of same meaning, which are completely differently spelt in two languages are sure to affect the performance. For example 'vaccine' is 'tika' (English transliteration) in Bengali and Hindi, but 'rasi' (English transliteration) in Gujarati. Only a dictionary could resolve such differences. One other resource that we have not exploited in this experiment is the test collection itself. The noisy converted texts may be augmented in some way by picking evidence from the vocabulary of the test collections. An approximate string matching between noisy query words and the words in the vocabulary could be helpful in identifying the unaltered counterpart with some degree of accuracy and add or substitute them in the query text to improve it.

References

1. Majumder, P., Mitra, M., Pal, D., Bandyopadhyay, A., Maiti, S., Pal, S., Modak, D., Sanyal, S.: The fire 2008 evaluation exercise. In: Proceedings of the First Workshop of the Forum for Information Retrieval Evaluation, vol. 9(3), pp. 1–24 (2010)
2. Chinnakotla, M.K., Damani, O.P., Satoskar, A.: Transliteration for resource-scarce languages. ACM Trans. Asian Lang. Inf. Process. 9(4), 14 (2010)
3. ACM Transactions on Asian Language Information Processing (TALIP) 9(3) (2010)
4. ACM Transactions on Asian Language Information Processing (TALIP) 9(4) (2010)
5. Kumaran, A., Khapra, M.M., Bhattacharyya, P.: Compositional machine transliteration. ACM Trans. Asian Lang. Inf. Process. 9(4), 13 (2010)
6. Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., Lioma, C.: Terrier: A High Performance and Scalable Information Retrieval Platform. In: Proceedings of ACM SIGIR 2006 Workshop on Open Source Information Retrieval, OSIR 2006 (2006)
7. Majumder, P., Mitra, M., Parui, S.K., Kole, G., Mitra, P., Datta, K.: YASS: Yet another suffix stripper. ACM Trans. Inf. Syst. 25(4) (2007)

Overview of FIRE 2010

Prasenjit Majumder[1], Dipasree Pal[2],
Ayan Bandyopadhyay[2], and Mandar Mitra[2]

[1] DA-IICT, Gujarat, India
[2] Indian Statistical Institute, Kolkata, India
{prasenjit.majumder,bandyopadhyay.ayan,mandar.mitra}@gmail.com,
dipasree_t@isical.ac.in

Abstract. This is an overview of FIRE 2010, the second evaluation exercise conducted by the Forum for Information Retrieval Evaluation (FIRE). Our main focus is on the Adhoc task. The Adhoc test collections are described, and a summary of the results obtained by various participants is provided.

1 Introduction

The second installment of the FIRE evaluation exercise ran from the second half of 2009 through early 2010, and culminated in a workshop held at DAIICT, Gandhinagar, from 12–14 February, 2010. The following tracks were offered.

- Ad-hoc monolingual and cross-lingual retrieval:
  - documents in Bengali, Hindi, Marathi, and English;
  - queries in Bengali, Gujarati, Hindi, Marathi, Tamil, Telugu and English.

  Roman transliterations of Bengali and Hindi topics were also provided.
- Retrieval and classification from mailing lists and forums (MLAF). This was a pilot task offered by IBM Research India.
- Ad-hoc Wikipedia-entity retrieval from news documents. This was a pilot task offered by Yahoo! Labs, Bangalore.

The timeline for FIRE 2010 is given below.

Training data release	Aug 15 2009 (FIRE 2008 data)
Test data release	Nov 01 2009 (Topics)
Adhoc run submission	Dec 11 2009
Results release	Feb 01 2010

In the next section (Section 2), we provide some statistics about the Adhoc corpora, the FIRE queries and the relevance judgements. Section 3 concludes this paper with a very brief discussion of the submissions and results.

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 252–257, 2013.

2 Adhoc Task Data

For the second FIRE campaign, three Indian language collections (Bengali, Hindi, Marathi) were offered. The text collections were almost the same as in the first FIRE campaign [1]. A new Hindi corpus was created by crawling the archives of a national Hindi daily. This corpus covers the period from September 2004 to September 2007. All content was converted to UTF-8. The sources and sizes of various corpora are given in Table 1.

Table 1. Sources and sizes of FIRE 2010 collections

Language	Source	# docs.
Bengali	Anandabazar Patrika	123,047
Hindi	Dainik Jagran	95,215
	Amar Ujala	54,266
Marathi	Maharashtra Times, Sakal	99,275
English	Telegraph	125,586

2.1 Query Formulation

A pool of 100 topics related to various social, political, cultural and other events was generated. Manual interactive search was performed using those topics on the Bangla and Hindi corpora. Each collection was indexed using the desktop version of TERRIER [2] and manual judgements were done for all of the 100 topics to get a fair estimate of the recall base. Finally, 50 topics were selected based on their complexity in terms of information need and recall base. Topics were then translated manually into English, Gujarati, Marathi, Tamil and Telugu.

2.2 Relevance Judgements

A preliminary pool was constructed by using various retrieval models in turn (e.g. BM25, variants of DFR, and LM as implemented in TERRIER) to retrieve some number of documents per query. The top 60 documents returned per query by each model were added to the pool. For these retrieval runs, stopwords were removed and YASS [3] was used for stemming. Later, interactive search was done with the aim of adding as many relevant documents as possible to the qrel file. The interactive searchers (who also doubled as assessors) used Boolean filters, relevance feedback and supervised query expansion to achieve their aim. The searchers were instructed to check about 100 documents per query during this process. Table 2 shows the variation in pool sizes across queries for various languages. For Bengali and Hindi, each document was judged by two assessors. The final qrels were created after the judges discussed and resolved all conflicts.

Table 2. Pool size across queries

	Bengali	Hindi	Marathi	English
Minimum	96	280	233	87
Maximum	300	704	616	553
Total	8,655	22,572	20,761 +	15,135

Table 3 shows the variation in the number of relevant documents across queries and languages. On the whole, the FIRE 2010 queries have significantly fewer relevant documents than the FIRE 2008 queries, with Hindi having the maximum number of relevant documents per query. During the topic formulation stage, the assessors were asked to ensure that there are at least 5 relevant documents per topic. Unfortunately, the final topics do not all satisfy this criterion. For Marathi, only 26 topics fulfil this criterion; indeed, there are as many as 11 Marathi queries with no relevant documents at all. Table 4 shows, for each language, the number of topics with at least 5 relevant documents.

Table 3. Number of relevant documents

	Bengali	Hindi	Marathi	English
Minimum	2	2	0 (11)	1
Maximum	29	74	72	47
Mean	10	18	12	13
Median	8	14	6	11
Total	510	915	621	653
Total (FIRE 2008)	1863	3436	1095	3779

Table 4. Queries with 5 or more relevant documents

	Bengali	Hindi	Marathi	English
# queries	40	45	26	42

3 Results

3.1 Submissions

A total of 129 runs were submitted. This number was significantly higher than the number of submissions at FIRE 2008. Table 5 shows an institute wise breakup

of the submitted runs. The maximum number of submissions came from Microsoft Research India. Table 6 gives a breakup of mono- and cross-lingual runs submitted. For cross-lingual runs, the source language was English or some other Indian language, and the target language was usually Hindi or English.

Table 5. FIRE 2010 participants

Institute	Country	# runs submitted
AU-KBC	India	2
Dublin City U.	Ireland	17
IBM	India	2
IIT Bombay (1)	India	30
IIT Bombay (2)	India	3
Jadavpur U.	India	2
MANIT	India	9
Microsoft Research	India	32
U. Neuchatel	Switzerland	18
U. North Texas	USA	8
U. Tampere	Finland	6
11 (9 @ FIRE 2008)	TOTAL	129 (up from 64 @ FIRE 2008)

Table 6. FIRE 2010 runs by task

Query language	Docs retrieved	# runs
Bengali	Bengali	16
Hindi	Hindi	19
Marathi	Marathi	20
English	English	15
English	Bengali	2
English	Hindi	18
Hindi	English	21
Marathi	English	4
Tamil	English	14

3.2 Monolingual and Cross-lingual Results

Tables 7, 8 and 9 show the Mean Average Precision (MAP) values for the top five monolingual runs for Bengali, Hindi and Marathi.

Table 7. Monolingual retrieval: Bengali

RunID	Group	MAP
BBUniNE4	UniNE	0.4862
BBUniNE2	UniNE	0.4731
BBUniNE3	UniNE	0.4684
BBUniNE1	UniNE	0.4646
FBP5TD	DCU	0.4526

Table 8. Monolingual retrieval: Hindi

RunID	Group	MAP
HHUniNE1	UniNE	0.4459
HHUniNE4	UniNE	0.4373
HHUniNE2	UniNE	0.4334
FH_S2TD_QE20	DCU	0.4305
HHUniNE3	UniNE	0.4284

Table 9. Monolingual retrieval: Marathi

RunID	Group	MAP
MMUniNE1	UniNE	0.5009
MMUniNE2	UniNE	0.4897
MMUniNE4	UniNE	0.4885
MMUniNE3	UniNE	0.4817
FMP5QE	DCU	0.4373

Among the cross-lingual runs, 18 runs used English queries to retrieve Hindi documents. The results for the best English to Hindi runs are shown in Table 10, while Table 11 shows the results for the best cross-lingual runs that use Indian language queries with English as the target language. For comparison, the MAP values for the best Hindi and English monolingual runs were 0.4459 and 0.4846 respectively.

Table 10. Cross-lingual retrieval: English → Hindi

RunID	Group	Fields	MAP
FHan_P5TD_QE20	DCU	TD	0.3771
UNTclenhi	UNT	TD	0.3757
FHan_S2TD_QE20	DCU	TD	0.3747
FHgt_S2TD_QE20	DCU	TD	0.3684
FHgt_P5TD_QE20	DCU	TD	0.3647

Table 11. Cross-lingual retrieval: X → English

X	RunID	Group	MAP
HI	2010FIREHE110	MSRI	0.4376
HI	2010FIREHE112	MSRI	0.4375
HI	2010FIREHE102	MSRI	0.4369
HI	2010FIREHE101	MSRI	0.4336
HI	2010FIREHE100	MSRI	0.4042
TA	2	AUKBC	0.3980
MR	MRENFEEDBACKT50	IITBCFILT	0.2771

References

1. Majumder, P., Mitra, M., Pal, D., Bandyopadhyay, A., Maiti, S., Pal, S., Modak, D., Sanyal, S.: The FIRE 2008 evaluation exercise. ACM TALIP 9(3), 10:1–10:24 (2010)
2. Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., Lioma, C.: Terrier: A High Performance and Scalable Information Retrieval Platform. In: Proceedings of ACM SIGIR 2006 Workshop on Open Source Information Retrieval, OSIR 2006 (2006)
3. Majumder, P., Mitra, M., Parui, S.K., Kole, G., Mitra, P., Datta, K.: YASS: Yet another suffix stripper. ACM TOIS 25(4) (2007)

UTA Stemming and Lemmatization Experiments in the FIRE Bengali Ad Hoc Task

Aki Loponen, Jiaul H. Paik, and Kalervo Järvelin

School of Information Sciences, University of Tampere, Finland
Indian Statistical Institute, Kolkata, India
{aki.loponen,kalervo.jarvelin}@uta.fi, jia.paik@gmail.com

Abstract. UTA participated in the monolingual Bengali ad hoc Track at FIRE 2010. As Bengali is highly inflectional, we experimented with three language normalizers: one stemmer, YASS, and two lemmatizers, GRALE and StaLe. YASS is a corpus-based unsupervised statistical stemmer capable of handling several languages through suffix removal. GRALE is a novel graph-based lemmatizer for Bengali, but extendable for other agglutinative languages. StaLe is a statistical rule-based lemmatizer that has been implemented for several languages. We analyze 9 runs, using the three systems for the title (T) and title-and-description (TD) and title-description-and-narrative (TDN). The T runs were the least effective with MAP about 0.34 (P@10 about 0.30). All the TD runs delivered a MAP close to 0.45 (P@10 about 0.37), while the TDN runs gave a MAP of 0.50 to 0.52 (P@10 about 0.41). The performances of the three normalizers are close to each other, but they have different strengths in other aspects. The performances compare well with the ones other groups obtained in the monolingual Bengali ad hoc Track at FIRE 2010.

1 Introduction

Word inflection is a significant problem in information retrieval (IR). In monolingual IR, query and text word inflection causes mismatch problems with the database index. In addition, in cross-lingual IR (CLIR) inflected query words cannot easily be located as translation dictionary headwords. These problems plague morphologically complex languages but can affect retrieval also in simpler ones.

Word inflection has been addressed in IR using both generative and reductive methods [3]. The former are methods for generating inflectional variants into queries while stemming (e.g. [5], [10]) and lemmatization (e.g. [3], [4]) belong to the latter. Lemmatization is potentially more effective than stemming, especially in morphologically complex languages, due to reduced ambiguity of full lemmas (i.e. full dictionary headword forms instead of stems). This holds for all text-based IR and, in CLIR as well, because lemmas support accurate translation by directly matching dictionary headwords. Lemmatizers traditionally use morphological rules and dictionaries [4]. Dictionary-based methods however lose power with out-of-vocabulary (OOV) words. OOV words are significant in all IR, because they often are specific in representing the information needs especially in short queries [3, p. 31]. Contemporary approaches

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 258–268, 2013.

to dictionary independent lemmatization are based on supervised (e.g. [7], [12]) and unsupervised (e.g. [13]) machine learning techniques. Supervised techniques need a training corpus involving a large set of morphologically analyzed tokens. The disadvantage of this method is that the preparation of the training corpus is time-consuming, in particular when the tokens are tagged manually. The main limitation of unsupervised learning techniques is their corpus dependency.

In this paper, we study three morphological normalizers in the FIRE 2010 Bengali ad hoc task. One of them is the stemmer YASS, and the two others are the lemmatizers GRALE and StaLe. YASS [10] is a corpus-based unsupervised statistical stemmer capable of handling several languages through suffix removal. It produces word stems. GRALE, developed by the second author, is a novel graph-based lemmatizer for Bengali, but extendable for other agglutinative languages. StaLe [8] is a statistical rule-based lemmatizer that has been implemented for several languages. While not being completely unsupervised, both lemmatizers need only minimal supervision for treating a new language. All three normalizers are interesting for the Bengali ad hoc task because they are dictionary-independent, require no or little manual supervision, and are therefore promising for a language lacking dictionaries and other linguistic resources. All three employ a different strategy in word form normalization – making comparison interesting.

The goals of the present paper are (1) investigate the application of the three normalizers to a new language (Bengali), (2) test their effectiveness under varying query lengths (title only (T), title-and-description (TD), and title-description-and-narrative (TDN)) and metrics (P@10 and MAP), (3) examine their overall and query-by-query effectiveness. We submitted six runs to the monolingual Bengali ad hoc track at FIRE 2010. In the present paper we extend the evaluation to nine runs – all three query lengths for all three normalizers. For comparison, we employ as baselines runs with no morphological processing and the overall best runs submitted to the Bengali track.

We shall show that the performances of the three normalizers are within a narrow band and thus anyone of them could be used. However, they have varying strengths in other aspects due to their different normalization strategies. The performances compare well with the ones other groups have obtained in the monolingual Bengali ad hoc Track at FIRE 2010.

Section 2 below briefly presents some features of the Bengali language. Section 3 presents the three normalizers and Section 4 defines the UTA runs. Findings (Section 5), Discussion and Conclusion (Sections 6 and 7) follow.

2 Some Remarks on the Bengali Language

Bengali is a highly inflectional language where one root can produce 20 or more morphological variants. Unlike English, proper nouns also can have a number of variations (for example, *samir-ke, samir-i, samir-o, samir-er*are forms of *samir*(also meaning wind)). In most cases variants are generated by adding suffixes to the end of the root. Also two or more atomic suffixes combine to form a single suffix and inflect the root (for example, *samir-der-ke-o,* where *samir* is the root, and *-der, -ke, -o* are

atomic suffixes). Nouns and pronouns inflect in case, including nominative, objective, genitive and locative. The case-marking pattern for each noun being inflected depends on the noun's degree of animacy. When a definite article such as *-ta* (in singular) or *-gula* (in plural) is added, nouns also inflect in number. Some of the genitive suffixes are *-r, -er, -diger,* and *-der.* Verbs inflect more heavily than nouns. Non-finite verbs have no inflection for tense or person, while finite verbs are fully inflected for person, tense, aspect (simple, perfect, progressive) and honor (intimate, familiar and formal).

Bengali is productive in compound words, which can be formed from combinations of nouns, pronouns, adjectives and verbs. There also exist a large number of compound words having more than one root and they have a number of morphological variants. For example the word *dhan* means wealth and *haran* means robbing. These two words combine to form *dhanharan*, meaning robbing of wealth. Now *dhanharan* can produce morphological variants like, *dhanharankari* and *dhanharankarider* where *-kari* and *-der* are suffixes. Normally when two words are joined to form a compound, the corresponding inflectional suffix of each non-last component word is omitted in the final compound. However, there are some instances where a compound word's non-last component word may retain its inflectional suffixes.

New words are also formed by derivation. Derivatives and their roots may belong to different parts of speech. Derivatives may change their form significantly from the root word and they are also formed through simple suffixing. For example, the derivation of *madhurjya* (sweetness) from *modhur* (sweet) is a typical instance of the former kind and *bhadrota* (goodness) from *bhadro* (good) is of the latter kind. Derived words and their roots very often use the same suffixes to generate inflectional forms. Antonyms are also formed through derivation. Like in English, they are often formed by adding prefixes.

Due to these rich morphological features, word form normalization in Bengali through stemming or lemmatization is likely to increase the term weights of the normalized forms and therefore to benefit query-document matching.

One typical feature of Indian Languages is that proper nouns are often either abstract nouns or adjectives. This combined with the fact that Bengali letters have only one case makes it difficult to detect proper nouns. For example *mamata* is a person name and *mamata* means affection. In monolingual information retrieval this may hurt precision for short queries containing a person name as a keyword. The problem is more severe in CLIR from Bengali to other languages. Finally, there are only few resources for NLP in Bengali.

3 The UTA Experimental Systems

3.1 YASS

YASS [10] is a corpus based purely unsupervised statistical stemmer capable of handling a class of languages primarily based on suffix removal. YASS uses a string distance measure to cluster the lexicon such that each cluster is expected to contain all the morphological variations of a root word appearing in the corpus.

Given two strings YASS computes the distance between them by a formula, which rewards a long match and punishes for an early mismatch. Therefore, two words sharing a long common prefix get a high proximity. This proximity eventually serves as the foundation for determining the morphological variations. Upon computing the pairwise distances, YASS uses a clustering algorithm to group the words into equivalence classes. Since the number of clusters is unknown beforehand, hierarchical clustering is chosen over partitioning clustering algorithm to generate the equivalence classes. In particular, the complete link clustering algorithm is used since it produces compact clusters as opposed to single link clustering which produces elongated clusters. The clusters are finally created by deleting the edges above a predefined threshold. The common prefix of each cluster is then considered as the cluster representative for the cluster, i.e. as the stem.

YASS delivers stems and these stems are based on the corpus used. A new corpus of text in the same language requires new training.

3.2 GRALE

GRALE is a graph-based lemmatizer initially developed for Bengali but adaptable to other agglutinative languages as well. GRALE was developed by the second author, Jiaul Paik, of the present paper. GRALE is based on a two-step algorithm, which in its first step extracts a set of frequent candidate suffixes by measuring their n-gram frequency in a given corpus. GRALE thereafter employs case suffixes manually identified within this set by a native speaker. In the second step, words are considered as nodes of a graph and a directed edge from node u to v exists if v can be generated from u by addition of a suffix taken from the selected suffix set. The graph built over the lexicon has the following properties:

- The graph is directed and acyclic.
- A node may have a zero or larger in-degree and/or out-degree.
- Between any two nodes there may exist more than one path. (An instance where suffix sequences are employed to generate inflectional variants.)

The directed graph constructed as above is then used to find the set of equivalence classes with a specially designated node called lemma. A node with in-degree zero presumably is a lemma. We group words around such nodes. We choose a node v as an inflected form of the lemma node if an incoming edge to v exists from the lemma node and most (for our experiments we choose at least 80%) of the children of v are also children of the lemma node. The algorithm processes the nodes in topological order to respect the dependency, that is, before processing a node all its ancestors are processed. The key assumption that we make is that the lemma exists in the lexicon as a word. This assumption is not unrealistic given the properties of Bengali.

In summary, after the learning phase where suffixes are identified, GRALE is not limited to any given collection of the same language. That is, the identification of inflectional suffixes is a one time job and GRALE can be applied to a new collection without further manual intervention. GRALE yields lemmas.

3.3 StaLe

StaLe is a statistical, rule-based lemmatizer developed by Loponen [8], that can operate with out-of-vocabulary words as well as with common vocabulary and is easy to adapt into new languages; even languages with scarce linguistic resources. StaLe is based on the TRT transformation rule system by Pirkola and colleagues [11], originally designed for cross-language name and terminology matching.

StaLe has two phases [8]: a one-time creation of the transformation rules for a given language, and lemma generation from input words. StaLe creates lemmatization rules from a given corpus of token-lemma pairs where the tokens are inflected. The tokens are extracted from real texts to guarantee that the lemmatization rules would represent the language properly. One lemmatization rule is formed from each token-lemma pair by selecting the affixes with a context character from the token and the lemma. Learning is based on rule frequencies and rule confidence factors (see [8]).

StaLe does not split compound words because corpus and dictionary independent compound splitting is highly ineffective. Airio [1] also showed that decompounding in monolingual IR is not vital.

The training data set for Bengali was obtained by randomly selecting 11000 unique inflected tokens from the FIRE'08 test corpus. The training data set consisted entirely of nouns. Nouns carry most of the semantics in queries and represent the actual "things" that are retrieved [2] [9]. Loponen and Jarvelin [8] showed with StaLe that IR effectiveness is not compromised when only nouns are lemmatized. Using only nouns facilitates rule learning and makes their application more efficient. For each token, a corresponding lemma form was formed by a native Bengali speaker. From the 11 000 token-lemma pairs, a set of 1163 lemmatization rules was generated.

StaLe was shown to be competitive, reaching 88-108 % of gold standard performance of a commercial lemmatizer in IR experiments in four languages: Finnish, German, Swedish, and English, which represent morphological variation from highly complex to very simple [8]. Despite competitive performance, it is compact, efficient and fast to apply to new languages.

4 The UTA Bengali Experiments

4.1 The Test Collection and Search Engine

We used FIRE 2010 Bengali test collection with 50 topics in our experiments. The collection contains 123047 documents with an average of 362 words per document. Average query length for title (T) queries was 6, for TD queries 17, and for TDN queries 44 words. The recall base has 510 relevant documents for 50 topics. Table 1 summarizes the test collection statistics. As the search engine we used the Lemur toolkit version 4.7, the Indri search engine in particular. The Lemur Toolkit is an open-source toolkit facilitating research in language modelling and information retrieval. The Indri search engine is based on a combination for the language modelling and inference network retrieval frameworks. (See the Lemur toolkit [6].)

4.2 The UTA Runs

We conducted nine experimental runs in the monolingual Bengali ad hoc track: T, TD and TDN runs with StaLe, GRALE and YASS. The runs were conducted by treating the documents and the queries with language normalizers and matching the treated queries and documents. We also formed a baseline run for each topic length (T, TD, TDN) by matching untreated queries and documents.

Table 1. Test collection statistics for the Bengali monolingual task

Property	Value
No. of documents	123047
No. of queries	50
Recall base (total)	510
Average query length	
- T queries (words)	6
- TD queries (words)	17
- TDN queries (words)	44

5 Findings

5.1 Overall Performance

Table 2 shows the MAP and P@10 scores for the runs. In all query lengths T, TD, and TDN the MAP and P@10 improve with language normalization methods over the baseline. On average, normalization improves MAP from the baseline in T queries by 7 percent units and in TD and TDN queries by 6 percent units. In P@10, the normalization methods outperform baseline, on average, by 5 percent units in T and TD queries, and by 3 percent units in TDN queries.

Query length correlated positively with performance. The T runs were the least effective with MAP 0.337 - 0.346 (P@10 from 0.302 to 0.308). All the TD runs delivered MAPs between 0.445 and 0.451 (P@10 from 0.370 to 0.380), while the TDN runs gave a MAP between 0.500 and 0.519 (P@10 from 0.412 to 0.416).

Regarding average precision of TD queries, the statistical significance test (Friedman's test) between the baseline, the three experimental methods and the best submission confirms significance ($p=0.0056$). The three experimental methods and the best submission are all significantly different from the baseline (no morphological processing; $p<0.01$) but between them there are no significant differences. The same holds for average precision in T queries. However, in TDN queries, the only significant difference is between the best submission and all other runs. At P@10 the experimental methods are not significantly different from each other or the best run for any query length. For the shorter T and TD queries the baseline is significantly weaker than any other method.

Table 2. Run results for StaLe, GRALE, YASS and baseline, and for all query lengths. Difference to baseline significant at p<0.05 (*) or at p<0.01 (**)

System	Metric	T queries	TD queries	TDN queries
Stale	MAP	0.3374 **	0.4488 **	0.5058
	P@10	0.3080 **	0.3700 **	0.4140
GRALE	MAP	0.3458 **	0.4451 **	0.5001
	P@10	0.3020 **	0.3740 **	0.4120
YASS	MAP	0.3453 **	0.4511 **	0.5190
	P@10	0.3012 **	0.3800 **	0.4160
Best Method	MAP	0.3458 **	0.4862 **	0.5438 **
	P@10	0.3020 **	0.3680 **	0.4220 *
Baseline	MAP	0.2737	0.3892	0.4455
(no NLP)	P@10	0.2520	0.3280	0.3800

We exemplify the recall-precision behavior of our experimental methods in Figure 1 – TD queries. The overall performance of the three experimental methods is very similar across the full recall range, clearly above the baseline and a bit below the best run. The other query lengths provide roughly similar results regarding differences between the methods.

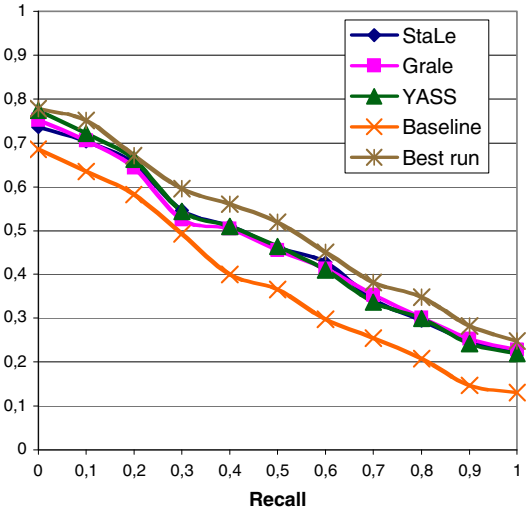


Fig. 1. Recall-precision graphs of the baseline, the three experimental methods and the best submission in the FIRE campaign (TD queries)

5.2 Query-by-Query Performance

The query-by-query performance between the experimental methods and the best method among the ones submitted to the FIRE monolingual Bengali experiment is

illustrated in Figure 2 (a)-(f). Figures (a)-(c) compare the experimental methods to each other and figures (d)-(f) compare each of the experimental methods to the best submitted run.

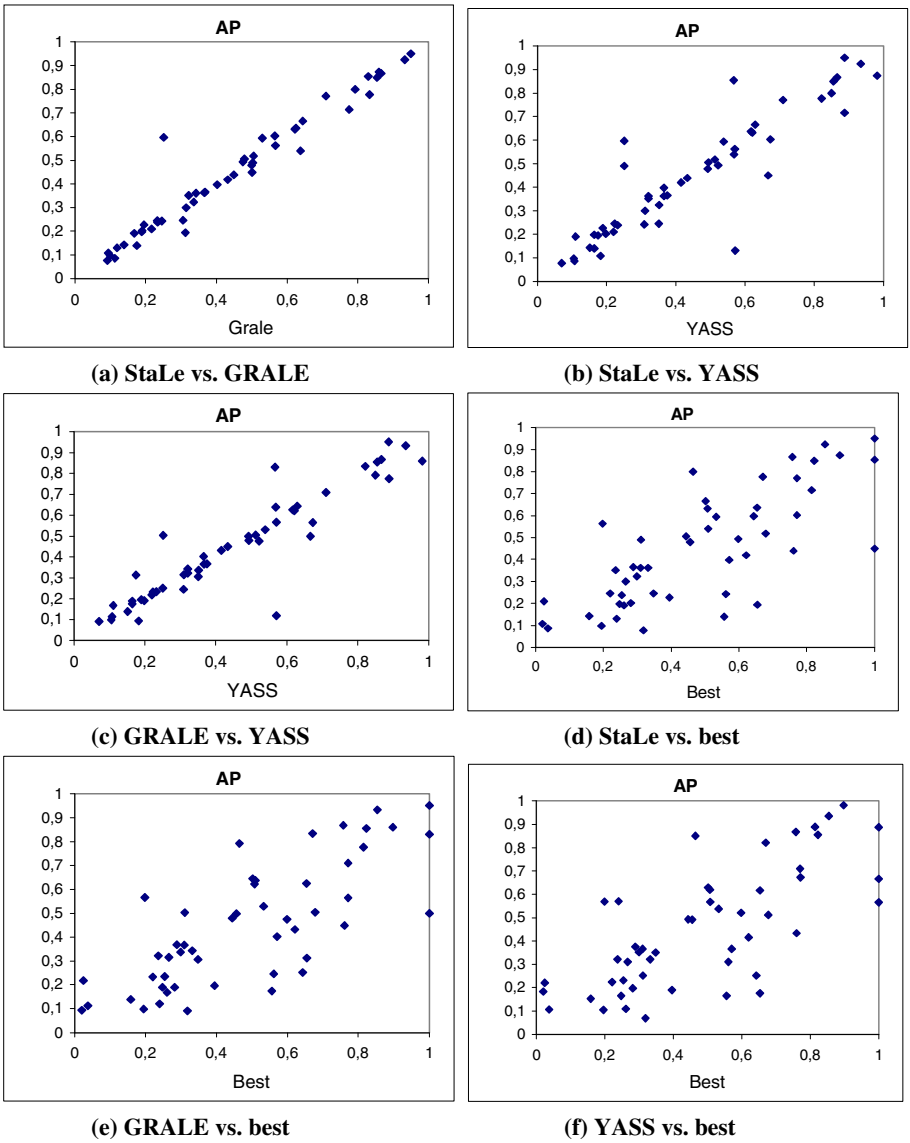


Fig. 2. Query-by-query comparison of the three experimental methods (a)-(c) and comparison to the best method in the FIRE campaign (d)-(f); average precision (AP) for TD queries

Figure 2 shows interesting trends. All our experimental methods are very close to each other for nearly all queries: most data points are close to the diagonal and only one to five queries behave clearly differently. This confirms that the overall finding (Table 2) that the experimental methods perform equally well, is not only a result of averaging but rather based on equal performance at the individual query level.

In contrast, the Figures (d)-(f) show a very different behavior. While the difference between the MAP of the experimental methods for TD queries (Table 2: ~0.45) and the MAP of the best performing run (~0.49) is negligible, the individual query level differences vary widely. The many data points far from the diagonal show this. There is a slight emphasis on the data points below the diagonal echoing the effectiveness benefit of the best method. Clearly the experimental methods and the best method are treating queries differently.

6 Discussion

The present paper had three goals. Firstly we investigated the application of the two lemmatizers StaLe and GRALE and the stemmer YASS to a new language (Bengali). Being unsupervised, YASS was directly applicable to the task, but a new learning phase would be required for a new collection. GRALE was also fast to apply. The manual phase of identifying correct suffixes from among the candidates required an effort of 3 to 4 hours from a native speaker. Regarding StaLe, we found that sampling 11,000 unique random nouns from the Bengali corpus, and providing their lemmas, was sufficient for learning a rule set for a competitive lemmatizer. Providing the lemmas was a few days' simple task for a native speaker.

Secondly, we tested the effectiveness of the three normalizers under varying query lengths (T, TD and TDN) and metrics (P@10 and MAP). The T runs were the least effective with MAP 0.337 - 0.346 (P@10 from 0.302 to 0.308). All the TD runs delivered MAPs between 0.445 and 0.451 (P@10 from 0.370 to 0.380), while the TDN runs gave a MAP between 0.500 and 0.519 (P@10 from 0.412 to 0.416). The performances compared well with the other submissions in the FIRE 2010 Bengali monolingual track. While the effectiveness of our experimental systems was lagging slightly below the effectiveness of the best submission, the differences were statistically mostly insignificant (Friedman's test) except for the longest TDN queries. The three experimental systems were significantly better than the baseline, except for the longest TDN queries. For all systems and both metrics, baseline included, increasing query length improved performance.

Thirdly, we examined their overall and query-by-query effectiveness. Interestingly, while the three experimental normalizers are based on different principles, both their overall and query-by-query performances were very similar (Figs. 1 and 2 a-c). They therefore cover similar morphological phenomena equally well. Compared to the best submission, there was in each case much more variation in query-by-query performance (Fig. 2 d-e). They therefore cover similar morphological phenomena to different degrees, or partially different morphological phenomena.

Overall, our results show that information retrieval in monolingual Bengali significantly benefits from language normalization. These methods are applicable to languages lacking robust language technology resources. The stemmer had a slight (while statistically insignificant) edge over the lemmatizers in the TD runs, while the two lemmatizers perform practically equally well. However, the stemmer's shortcoming is that it can be trained only to stem the training set, while both lemmatizers are extrapolative methods and can be applied to words from outside the training corpus.

GRALE was able to handle all word classes while StaLe handled all words as nouns. The results show that despite the exclusion of verbs from the StaLe's training set, StaLe performed at the same level as GRALE. This confirms the findings in [8].

7 Conclusion

We experimented with three language normalizers, YASS, GRALE, and StaLe in the monolingual Bengali ad hoc track at FIRE 2010. We evaluated the three normalizers for title runs, title-and-description runs, and title-description-and-narrative runs. The performances of the three normalizers are close to each other both overall and at the individual query levels. Their differences are not significant in average precision or P@10 between themselves nor with the best submitted run in the track except for the longest queries. However, they have different strengths in other aspects. YASS handles any collection in a given language, but each application is collection-specific. YASS delivers stems and this is sufficient for monolingual applications but not the best option for CLIR applications. GRALE and StaLe both deliver lemmas, which are convenient for dictionary matching. Moreover, after the learning phase, both are applicable to previously unseen collections of a language. None of the three normalizers offers compound splitting, which is a desirable property but not a must in monolingual IR [1]. The performances compare well with their rivals in the monolingual Bengali ad hoc track at FIRE 2010. Their strength is that they do not require expensive NLP resources for competitive performance.

References

1. Airio, E.: Word normalization and decompounding in mono- and bilingual IR. Information Retrieval 9, 249–271 (2006)
2. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval, 2nd edn. Addison-Wesley (2011)
3. Kettunen, K.: Reductive and Generative Approaches to Morphological Variation of Keywords in Monolingual Information Retrieval. Acta Universitatis Tamperensis 1261. University of Tampere, Tampere (2007)
4. Koskenniemi, K.: Two-level Morphology: A General Computational Model for Word-Form Recognition and Production. Ph.D. Thesis, University of Helsinki, Department of General Linguistics, Helsinki (1983)
5. Krovetz, R.: Viewing morphology as an inference process. In: Proceedings of the 16th ACM/SIGIR Conference on Research and Development in Information Retrieval, Pittsburgh, Pennsylvania, USA, pp. 191–202 (1993)

6. Lemur: The Lemur Tool-kit for Language Modelling and Information Retrieval, http://www.lemurproject.org/ (visited March 30, 2010)
7. Lindén, K.: A Probabilistic Model for Guessing Base Forms of New Words by Analogy. In: Gelbukh, A. (ed.) CICLing 2008. LNCS, vol. 4919, pp. 106–116. Springer, Heidelberg (2008)
8. Loponen, A., Järvelin, K.: A Dictionary- and Corpus-Independent Statistical Lemmatizer for Information Retrieval in Low Resource Languages. In: Agosti, M., Ferro, N., Peters, C., de Rijke, M., Smeaton, A. (eds.) CLEF 2010. LNCS, vol. 6360, pp. 3–14. Springer, Heidelberg (2010)
9. Losee, R.M.: Is 1 Noun Worth 2 Adjectives? Measuring Relative Feature Utility. Information Processing and Management 42(5), 1248–1259 (2006)
10. Majumder, P., Mitra, M., Parui, S.K., Kole, G., Mitra, P., Datta, K.: YASS: Yet Another Suffix Stripper. ACM Transactions on Information Systems (TOIS) 25(4) (2007)
11. Pirkola, A., Toivonen, J., Keskustalo, H., Visala, K., Järvelin, K.: Fuzzy translation of cross-lingual spelling variants. In: SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Toronto, Canada, pp. 345–353 (2003)
12. Plisson, J., Lavrac, N., Mladenic, D.: A rule based approach to word lemmatization. In: Proceedings of the 7th International Multi-Conference Information Society IS 2004, pp. 83–86 (2004)
13. Wicentowski, R.: Modelling and Learning Multilingual Inflectional Morphology in a Minimally Supervised Framework. Ph.D. Thesis, Baltimore, Maryland, USA (2002)

Tamil English Cross Lingual Information Retrieval

T. Pattabhi R.K. Rao and Sobha Lalitha Devi

AU-KBC Research Centre, MIT Campus of Anna University
Chrompet, Chennai, India
{sobha@au-kbc.org}

Abstract. This paper describes our work on participation in the FIRE 2010 evaluation campaign in the cross lingual information retrieval track. We describe how cross lingual information retrieval can be effectively performed between a highly agglutinative language, Tamil and English, an isolating language. Agglutination is a morphological process of adding affixes to word base. These affixations can be between noun- noun, adjective-noun, noun-case, etc. This phenomenon of the language has brought serious problems in translation, transliteration and expansion of the query into another language. To overcome these we have used a morphological analyzer which gives the root word or a word base. The word base is used in turn for translation, transliteration and query expansion. The translation of the query is done using bilingual dictionary and transliteration uses statistical method. And query expansion is performed using ontology and WordNet.

Keywords: Cross lingual Information Retrieval (CLIR), Indian languages, Tamil, English.

1 Introduction

The World Wide Web (WWW) or internet today has enormous data in various languages. This is being considered as a huge repository of information, by people all over the world. The cross lingual information retrieval in Indian languages has attracted interest of researchers and industry, only in recent times. One of the first known initiative involving Indian languages was during the TIDES surprise language exercise [9]. In this exercise Hindi, was the surprise language given to the participants. The international evaluation forum CLEF[1] had introduced a special sub-task specific for Indian languages in the year 2007. Here the Indian languages Hindi, Bengali, Marathi and Telugu were considered. In this several approaches such as language modeling coupled with probabilistic transliteration [11], iterative disambiguation [3], using zonal indexing approach [2], using word alignment learned from SMT [5], were used by different participants. The Forum for Information Retrieval Evaluation (FIRE) is an initiative in this direction, for Indian languages.

This paper, describes our participation in the FIRE 2010. Here we participated in the Ad-hoc cross-lingual document retrieval task. The task is to retrieve relevant

[1] http://www.clef-campaign.org/2007.html

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 269–279, 2013.

documents in English for a given Indian language query. We have worked on Tamil – English cross lingual information retrieval system. Here the query language is Tamil and the language of the documents to be retrieved is English. In our work, we have focused on query processing, query translation and query expansion. For the query processing we use morphological analyzer. Query translation is a dictionary based approach. During the query translation phase we have focused on proper translation of named entities. For the query expansion, we have used WordNet and the description field of the queries.

The paper is further organized as follows. In section 2 several problems encountered while developing a cross lingual information retrieval system (CLIR) is described. Section 3 describes our approach in solving these problems and how to build the Tamil – English Cross Lingual Information Retrieval. Section 4 discusses the results and finally section 5 gives the conclusion.

2 Issues in CLIR

In a cross lingual information retrieval system, the user gives queries in his/her own language, and the documents to be retrieved are in different language(s). The user query in language L1 is the query language and L2 is the document language. Here we have taken L1 as Tamil and L2 as English. Depending on the nature of the language we have to use the pre-processors. Here Tamil is an agglutinative and inflectional language. The queries in Tamil require to be processed using a morphological analyzer or a stemmer to obtain the base forms of the query terms.

The main issue in any CLIR system is the translation of the query in L1 to L2 and the performance of the system heavily depends on the accuracy of the translation. The translation of queries is not similar to that of document translation though at the outset it seems similar to and simpler than document translation. A query is a short phrase, and not a full sentence, hence language preprocessing such as part-of-speech tagging, chunking are not possible. Considering each word as an independent token, and translating each token into the target language would not be correct in all the cases. A query in most cases is a named entity, or multi-word expression embedded with named entities. In a dictionary based query translation approach, one of the major problems is the coverage of dictionaries. The coverage problem was handled using special dictionaries in the work of Pirkola [12]. Demner-Fushman and Oard [4], in their work have observed that named entities are 50% of the out-of-vocabulary(OOV) words in the query topics. They have also observed that the performance of the retrieval system reduces up to 60% if OOV terms are common in queries and if they are not handled properly. Hence in our present work we have focussed on translations of Named entities. Here we have classified Named entities into three types. Type one which can be transliterated and doesn't require translation. Type two, which requires translation and type three which needs both transliteration and translation. An example for the third type is as follows: in English the named entity “Andhra Pradesh State Road Transport Corporation”, can not be completely transliterated into Tamil. The Tamil equivalent for this is “anthira pradesa manila

pookku varathu kazhagam". This example clearly states the complexity in query translation and shows that in a query there are portions which require translation and transliteration. Another issue for a CLIR system is the ranking of retrieved documents. The objective of ranking is to display the retrieved documents in the order of relevance to the given query.

3 Our Approach

The documents used for Cross lingual retrieval are in English and it consists of 125638 documents (provided by FIRE). We have used Lucene indexer (Lucene[2] is an open source library), which consists of modules for indexing. It is a full-featured text search engine.

The main components in our cross lingual information retrieval system are
i) Language Analyzer
ii) Query Translation engine
iii) Query Expansion
iv) Ranking

3.1 Language Analyzer

The query has to be processed and translated before it is given to the search subsystem. The query language, Tamil, belongs to Dravidian family of languages and it is morphologically rich. It is a verb final language and has a relatively free word order. Its a highly agglutinative language. The words in Tamil are formed by adding suffixes successively to the root word or the base form. Morphophonemic changes occur when the suffixes are added to the root form. The main lexical categories, Nouns and Verbs take inflections. Nouns take number suffixes, case suffixes and postpositions. Nouns have 8 cases viz., Nominative, Accusative, Dative, Locative, Genetive, Instrumental, Sociative and Ablative. Verbs take tense suffix, PNG suffix (Person, Number and Gender agreement) and clitics. In Tamil we can observe a lot of compound nouns. For example the word "ativayirru" which means "abdomen" is combination of two words "ati" (in English this means "below") + "vayirru" (in English this means "stomach"). In the compound words, inflection happens to the last word. In the example stated above the inflection would happen to the last word "vayirru", such as "ativayirril" which means "in the abdomen". Here the locative case suffix "il" is added to the last word "vayirru" [16]. A more detailed description of Tamil morphology and grammar can be found in Lehman [7].

Hence the query requires to be analyzed morphologically to obtain the base form of the word. We have used a Tamil morphological analyzer [17], which is developed using paradigm based approach and uses a finite state Engine (FSA). The system was tested on the corpus obtained from Central Institute of Indian Languages (CIIL),

[2] http://lucene.apache.org/

Mysore, India. This has approximately 3 million words, consisting of several genres such as stories, politics, literature, recipes. The system performs with an accuracy of 97%. For example for the word

(1) " puththakaththil" – puththakam + thth + il
"in the book" – book + oblique stem+ Locative Case

In this example (1), the root word is "puththakam". This has past tense marker "thtth" and locative case marker "il";

The target language English is not morphologically rich and also not an agglutinative language. In English, and many related languages, morphological variation takes place at the right-hand end of a word-form [14]. Hence for this, a simple stemmer can be used. Here we use a stemmer which is an implementation of Porter stemmer algorithm [13]. This algorithm follows suffix stripping based methodology. The algorithm is very simple in concept, with 60 suffixes, two recoding rules and a single type of context-sensitive rule to determine whether a suffix should be removed. Rather than rules based on the number of characters remaining after removal, Porter uses a minimal length based on the number of consonant-vowel-consonant strings (the *measure*) remaining after removal of a suffix.

3.2 Query Translation/ Transliteration

Our approach for query translation is dictionary-based approach. We have made use of a Tamil – English bilingual dictionary, which is of 150K words. As explained in section 2, query translation is one of the most important component of a CLIR system and in our approach we have focused on the proper translation of the Named entities. We have classified Named Entities (NEs) into three types for this purpose. The first type (Type X) is the one which requires only transliteration and no translation. Transliteration is the process of mapping one language word to other language based on the pronunciation. For example the NE, "John", a name of a person written in English, whether in English or Tamil or any other language would be the same and pronounced the same. The same when written in Tamil would be "jaan". The process of transliteration is suitable for Person names, Location names. The second type (Type Y) of NE need to be translated because they have equivalents in the other language. For example the NE "Electricity Board", which is in English, has a Tamil equivalent, "minsaara vaariyam". Such NEs requiring full translation are translated using a bilingual dictionary. The NEs that require translation, instead if they are transliterated then that would lead to most cases no results not being retrieved or in some cases irrelevant results. The third type (Type Z) are the ones which require both transliteration and translation. For example consider the NE in query topic number103, "Bhaglihar hydro-electric power project". In this the word "Bhaglihar" is a place name and requires only transliteration and "hydro-electric power project" requires to be translated. Actually this NE is a case of embedded NE. The NE

identification in the query topics can be done using a automatic NE recognition (NER) engine. But most of the search topics are short and not full sentences, use of a automatic NER engine is practically difficult. The alternative to this is automatic generation of NE lexicon from a huge corpus using a NER engine and using the generated NE lexicon as a look up list during query processing time of CLIR system. We have a NE lexicon for Tamil, which was developed automatically using a huge corpus of data collected from web. A small subset of the data collected from web is manually NE tagged and a NER engine is trained. NER engine uses conditional random fields model of the machine learning techniques. The query translation algorithm is as follows:

i) If the query is a named entity of type X, then transliterate the query using transliteration engine.

ii) Else, if the query is of Type Y then match the whole query with bilingual dictionary entry,

iii) Else, if the query is of type Z then, split the query into two with n-1 terms as one and nth term as one. Now match the n-1 terms and nth term separately with the dictionary entries, if matches substitute, else the same step till all terms are substituted.

iv) Else, if no match found in the dictionary, transliterate those terms using the transliteration engine.

The transliteration engine is a statistical system, which uses a n-grams based approach [1]. This system has been trained using web corpus and tested on the web corpus. This system performs with an accuracy of 81 %. This system produces all possible correct outputs up to a maximum of ten. All these possible transliteration outputs are retained. The name for example written in Tamil as "syamallaa", in English can be represented as, "Shyamala", "Syamala", "Shyamla". This algorithm uses n-gram frequencies of the transliteration units, to find the probabilities. Each transliteration unit is pattern of consonant-vowel (C*V*) in the word.

3.3 Query Expansion

Query expansion, is the process of adding more terms or phrases to the given query. This is done to help the system in retrieving more number of relevant documents. One of the features of natural language is that we have many ways and words to express a same concept or a single object. For example in Tamil "kovil", "koyil", "aalayam" are used synonymously to mean "temple" in English. The other feature is that same word in a language can mean different, in different context. For example "bark" in English can occur as a noun or verb. Query expansion helps in adding more information which would be helpful in obtaining good search results. Query expansion is done i) using synonyms and ii) using the description field of the query document. The synonyms are obtained using WordNet.

English WordNet contains synonyms of words and is a lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory [8]. WordNet is created based on the assumption that there is a mental dictionary in which the words are organised under conceptual fields or semantic domains. In a WordNet, lexical information is organised in terms of word meanings or concepts rather than word forms. We have used two WordNets English and Tamil. The English WordNet is a large lexical database of English, developed under the direction of George A. Miller. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. (http://wordnet.princeton.edu/). We use the English WordNet 3.0 version available from the Princeton web site.

Tamil WordNet [15] is built in the similar lines of English WordNet. This shows network of semantic relations between Lexical items based on the lexical relations such as synonymy, compatibility, incompatibility (antonymy, etc.), hyponymy, hypernymy, meronymy, holonymy, troponymy, and entailment. This contains major category of words – nouns, verbs, adjectives and adverbs. This consists of total 50497 words and 41013 unique senses. Here we have also made use of the description field of the query document.

3.4 Ranking

Here, we have used the standard Okapi BM25 Model [6]. Given a keyword query Q = {q1, q2,,,qn} and document D, the BM25 score of the document D is as follows:

$$\text{score}(Q,D) = \sum \text{IDF}(qi).\ \frac{(f(qi,D).\ (k1+1))}{f(qi,D)+k1.(1-b+b.(|D|/avgdl))} \quad (1)$$

$$\text{IDF}(qi) = \log .\ \frac{N - n(qi) + 0.5}{n(qi) + 0.5} \quad (2)$$

where f(qi,D) is the term frequency of qi in D, |D| is length of document D, k1 & b are free parameters to be set, avgdl is the average length of document in corpus, N is the total no. of documents in collection, n(qi) is the number of documents containing qi. In our current experiments, we have taken k1 = 1.2 and b = 0.75.

The basic ranking algorithm is customized to suit our needs. Here we introduce a parameter called boost factor in the equation (1), given above. The boost factor is multiplied for each term in the equation (1), before the summation is done, while computing the BM25 score. The original query terms are given a boost factor of 1.5. No boost factor is given to the other new expanded terms in the query. The boost factor of 0.5 times for original query terms is to retain the importance for the user given query terms, than for the query expanded terms.

4 Experiment and Results

The FIRE document collection for Ad-hoc cross-lingual document retrieval task consists of news articles taken from "The Telegraph", which is one of the popular English news magazines available in India. The data from this news magazine is of the time period July 2004 to August 2007. The total number of documents in this collection is 125638 documents. This consists of news articles across the domains such as sports, politics, business, arts, and science. The English documents are indexed using the Lucene indexer. The documents are indexed after stemming and stop word removal. The porter stemmer algorithm is used for stemming the English documents. For ranking we use the slightly modified okapi BM25 algorithm, which includes the boost factors, to obtain better results. In Lucene, the implementation of okapi BM25 ranking function is not provided by default. There are several plugin extensions available which can be used for this purpose. Here we have used the extension plugin provided by Perez-Iglesias [10]. Here we have submitted two runs. The first run is a basic run, where the query expansion module is not implemented. The second run we have the query expansion module implemented. Here we have implemented the Tamil to English cross lingual information retrieval system. The FIRE 2010 topic set consists of 50 topics in Tamil. The queries are generated using the "Title" of the topic set.

We have used standard evaluation measures, which is used in all retrieval tasks. The following evaluation measures are used i) Mean Average Precision (MAP), ii) Precison at 5 (P@5), iii) Precision at 10 (P@10) iv) Precision at 20 (P@20) and v) Recall.

The below table, Table 1, gives the overall results of our submissions in the FIRE.

Table 1. Overall Results of the Tamil – English cross lingual information retrieval

Run ID	MAP	R-Prec	P5	P10	Recall	MAP score as percentage of English Monolingual result in FIRE 2010
2	0.3980	0.3742	0.4640	0.3900	0.9785	77.53%
1	0.2954	0.2931	0.360	0.2960	0.9372	57.54%

On analyzing the results obtained we observe that for queries such as query no. 124, 117, 93, 90 the system did not perform well. The query 124 in Tamil "inthiya maanilangal palavarril cattathirku purrampaka pothaiporull virrpaNai", means "Sale of illegal drugs in various Indian states". This query retrieved all documents consisting the term "drugs", "narcotics" even though those documents do not say about sale of illegal drugs and resulted in retrieval of irrelevant documents. For query 117, the topic is very specific, but the result for this query yields all documents related

to land controversy not just at Kalinganagar. This shows that certain terms in the query should be given negative weightage, so that those terms do not bring in irrelevant documents. Here in this query 117, the terms “land controversy” should be give negative weightage. Similarly for query 93, the system brings in documents describing bribes taken by officials, not just by parliamentarians. This shows the difficulty in tackling specific queries. Handling of specific queries is difficult compared to general queries. In the Table 2, we show all the query topics in English.

Table 2. Query Topic Titles in English

No	Query Title	No	Query Title
76	Clashes between the Gurjars and Meenas	101	Drug party at Pramod Mahajan's bungalow
77	Attacks by Hezbollah guerrillas	102	Pakistani cricketers involved in a doping scandal
78	Conflict between Advani and Singhal over the Ram Mandir issue	103	Bilateral problems surrounding the Baglihar hydro-electric power project
79	Building roads between China and Mount Everest	104	Jaya Bachchan sacked from Rajya Sabha membership
80	Babri Masjid demolition case started against Advani	105	Taj heritage corridor scandal
81	Problems related to the immunization programme against Japanese Encephalitis in India	106	Ban on Taslima Nasreen's novel "Shame"
82	Proposed bus service between Srinagar and Muzaffarabad	107	Furore over the release of a CD containing anti-Muslim sentiments in Uttar Pradesh
83	Election campaign of Laloo Prasad Yadav and Ram Vilas Paswan	108	Greater Nagaland
84	Brinda Karat's allegations against Swami Ramdev	109	New political party formed by Raj Thackeray
85	Abu Salem, accused in the Mumbai Bomb Blast case, in jail custody	110	Sino-Indian relations and border trade
86	Privatization of the Mumbai and Delhi airports	111	Dance bars banned in Mumbai

Table 2. *(Continued)*

87	Discussions between Manmohan Singh and Pervez Musharraf regarding the position of troops around Siachen	112	Links between Gutkha manufacturers and the underworld
88	Popular protests against the arrest of the accused in the Shankar Raman murder case	113	Political clashes in Bangladesh
89	Involvement of Congress ministers in the oil-for-food scam	114	Investigation of the arms scandal in the Defense Ministry
90	Indian representatives visit Bangladesh	115	Serial blasts in Varanasi
91	Allegations of financial corruption against Pratibha Patil	116	Encounter specialist Daya Nayak
92	Activities of the Tamil Tigers of Sri Lanka	117	Controversy over land at Kalinganagar
93	Taking bribes for raising questions in parliament	118	Terrorist strike at Ayodhya
94	Indian Navy accused of leaking classified information	119	Taj Mahal controversy
95	Racism row on the Big Brother show	120	Sex CD scandal involving Anara Gupta
96	Pramod Mahajan's killer	121	Blasts on Samjhauta Express
97	Quarrel between the Ambani brothers regarding ownership of the Reliance Group	122	Sanjay Dutt's surrender
98	India dismisses China's claims on Arunachal Pradesh	123	Death of Yasser Arafat
99	Laloo Prasad Yadav and the fodder scam	124	Sale of illegal drugs in various Indian states
100	Monica Bedi and the passport forgery case	125	Attack on the Lal Masjid

We see that for queries such as 76, 95, 97, 100 etc our system has performed well with MAP scores of 0.800. We observe that the MAP score results for 17 query topics is greater than 0.54 which is comparable with monolingual search result. This was possible because of proper handling of NE terms in the queries. In the query

titles we find that on an average there is at least one NE. Most of the NEs are of type X. Hence the role of transliteration engine is very significant. As explained in section 3.2, when transliterating names from Tamil to English, the correct English spelling form should be produced, else that would lead to irrelevant retrieval results which has happened in the query 77. The term "hespulla" in Tamil was not transliterated properly in English. In the query 78, we had observed that even though the query was translated/transliterated properly the results retrieved was low. The query in Tamil was "athvaani, cinkaal idaiye raamar koyil parriya karuththu veerrupaadu" and in English this was translated as "advani, singhal between ram temple issue conflict". In most of the English documents we found that instead of "temple" they had used "mandir", which is taken from Hindi. This is an interesting characteristic we find in English news articles in India. It would be interesting to study in the corpus, the percentage of such words are in use.

The overall results are encouraging; we obtain a MAP score of 0.3980 when query expansion using synonyms and description field of query is used. This is comparable with English monolingual search. In the FIRE 2010 results we observe that the maximum MAP score obtained for English monolingual search result is 0.5133. Our MAP score is 77.53 % of the monolingual result. From Table 1, we observe that query expansion helps in improving the results significantly. The second implements the query expansion.

5 Conclusion

Here we have presented Tamil to English cross lingual information retrieval system used in the FIRE Ad-hoc evaluation task. Our approach is based on bilingual dictionaries and query expansion. The use of description field of query document gives a significant increase in the recall without disturbing the precision. Here we have found that the system performs well for queries for which the query terms given are unambiguous and world knowledge has been imparted. The overall MAP score of the system is 0.3980 and R-prec is 0.3742. The results are encouraging and comparable to English monolingual system.

References

1. Afraz, M., Sobha, L.: English to Dravidian Language Machine Transliteration: A Statistical Approach Based on N-grams. In: International Seminar on Malayalam and Globalization (2008)
2. Bandyopadhyay, S., Mondal, T., Naskar, S.K., Ekbal, A., Haque, R., Godhavarthy, S.R.: Bengali, Hindi and Telugu to English Ad-hoc Bilingual Task at CLEF 2007. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, pp. 88–94. Springer, Heidelberg (2008)
3. Chinnakotla, M.K., Ranadive, S., Damani, O.P., Bhattacharyya, P.: Hindi to English and Marathi to English Cross Language Information Retrieval Evaluation. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, pp. 111–118. Springer, Heidelberg (2008)

4. Demner-Fushman, D., Oard, D.W.: The Effect of Bilingual Term List Size on Dictionary-Based Cross-Language Information Retrieval. In: 36th Annual Hawaii International Conference on System Sciences (HICSS 2003) – Track 4 (2003)
5. Jagarlamudi, J., Kumaran, A.: Cross-Lingual Information Retrieval System for Indian Languages. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, pp. 80–87. Springer, Heidelberg (2008)
6. Jones, K.S., Walker, S., Robertson, S.E.: A probabilistic model of information retrieval: development and comparative experiments (Part 1 & 2). Information Processing and Management 36(6), 779–840 (2000)
7. Lehmann, T.: A Grammar of Modern Tamil. Pondicherry Institute of Linguistics and Culture, Pondicherry (1989)
8. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.: Introduction to WordNet: An on-line lexical Database (1993)
9. Oard, D.W.: The surprise language exercises. ACM Transactions on Asian Language Information Processing (TALIP) 2(2), 79–84 (2003)
10. Perez-Iglesias, J., Perez-Aguera, J.R., Fresno, V., Feinstein, Y.Z.: Integrating the Probabilistic Models BM25/BM25F into Lucene. CoRR vol. Abs/0911.5046 (2009)
11. Pingali, P., Varma, V.: IIIT Hyderabad at CLEF 2007 Adhoc Indian Language CLIR task. In: Nardi, A., Peters, C. (eds.) Working Notes for CLEF 2007 Workshop (2007)
12. Pirkola, A.: The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. In: Proceedings of the 21st Annual International ACM SIGIR Conference, pp. 55–63 (1998)
13. Porter, M.F.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
14. Sproat, R.: Morphology and Computation. MIT Press, Cambridge (1992)
15. Thiyagarajan, S., Arulmozi, S., Rajendran, S.: Tamil WordNet. In: First Global WordNet Conference, CIIL, Mysore (2002)
16. Vijay Sundar Ram, R., Menaka, S., Sobha, L.D.: Tamil Morphological Analyser. In: Parakh, M. (ed.) Morphological Analysers and Generators, LDC-IL, Mysore, pp. 1–18 (2010)
17. Viswanathan, S., Ramesh Kumar, S., Kumara Shanmugam, B., Arulmozi, S., Vijay Shanker, K.: A Tamil Morphological Analyser. In: Proceedings of International Conference on Natural Language Processing (ICON), Mysore (2003)

Test Collections and Evaluation Metrics Based on Graded Relevance

Kalervo Järvelin

School of Information Sciences, University of Tampere, Finland
kalervo.jarvelin@uta.fi

Abstract. In modern large information retrieval (IR) environments, the number of documents relevant to a request may easily exceed the number of documents a user is willing to examine. Therefore it is desirable to rank highly relevant documents first in search results. To develop retrieval methods for this purpose requires evaluating retrieval methods accordingly. However, the most IR method evaluations are based on rather liberal and binary relevance assessments. Therefore differences between sloppy and excellent IR methods may not be observed in evaluation. An alternative is to employ graded relevance assessments in evaluation. The present paper discusses graded relevance, test collections providing graded assessments, evaluation metrics based on graded relevance assessments. We shall also examine the effects of using graded relevance assessments in retrieval evaluation, and some evaluation results based on graded relevance. We find that graded relevance provides new insight into IR phenomena and affects the relative merits of IR methods.

1 Introduction

Information retrieval (IR) research is heavily based on experimental evaluation of retrieval methods. This is seen as the corner-stone for progress in the field. IR evaluation traditionally follows the Cranfield paradigm, thus using test collections containing a document corpus, a set of requests (or topics) and relevance assessments identifying which documents are relevant to each topic. In most IR experiments, including the TREC-experiments, documents are judged binarily relevant or irrelevant with regard to the request. However, binary relevance cannot reflect the possibility that documents may be relevant to a different degree; some documents contribute more information to the request, some less without being totally irrelevant. In some studies relevance judgments are allowed to fall into more than two categories, but only a few tests actually take advantage of different relevance levels (e.g. [4]). More often relevance is conflated into two categories at the analysis phase because of the calculation of precision and recall (e.g., [1] [19]).

In current large search environments, the number of relevant documents to a request may easily exceed by orders of magnitude the number of documents a user is willing to examine. It is therefore desirable to rank highly relevant documents first in search results. To learn to do that one needs to evaluate IR methods accordingly, by

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 280–294, 2013.

highly relevant documents or by graded relevance. However, the prevalent test collections offer rather liberal binary assessment of topical relevance. In the test TREC collections, a document needs to have at least one sentence pertaining to the request to count as relevant [2]. At such a low level we would often consider the document at most as marginal unless the information need is simply factual. Experiments based on such assessments may not tell sloppy retrieval methods from excellent ones.

The present paper discusses graded relevance, test collections providing graded assessments, evaluation metrics based on graded relevance assessments. We shall also examine the effects of using graded relevance assessments in retrieval evaluation, and some evaluation results based on graded relevance. We find that graded relevance provides new insight into IR phenomena and affects the relative merits of IR methods.

The paper is organized as follows: Section 2 discusses types and degrees of relevance. Section 3 presents test collections providing graded assessments and Section 4 evaluation metrics based on graded relevance. We discuss some results of using graded relevance assessments in retrieval evaluation in Section 5. Discussion and conclusion in Section 6 follow.

2 Graded Relevance

The concept of graded relevance. Relevance has been a difficult problem in Information Science throughout the years. The following characterization is based on Kekäläinen & Järvelin [11] and Ingwersen & Järvelin [5] who extensively cite relevant original research and review papers. First, relevance is a *multidimensional* (e.g., objective, subjective) and *dynamic* (i.e. assessments may change over time) phenomenon. There are several *factors* and *criteria* that affect relevance judgments. Some researchers have argued that there are various *kinds* of relevance (e.g., algorithmic, topical, cognitive, situational, motivational relevance). Test collections employ topical relevance.

Relevance also is a *multilevel* phenomenon. Therefore some documents are more relevant than others to an information need of a user. Multiple *degrees* of relevance and their expression have been studied in laboratory settings already in the 1960's as well as in field studies of information seeking and retrieval. Tang and colleagues [22] found that a seven-point scale for relevance assessments is optimal in terms of the assessors' confidence in their assessments.

Until 2000 in the practice of IR evaluation, the binary scale has been the norm. Even in cases where multiple degree scales are used in assessments – they have been collapsed into binary scales for IR method evaluation. This is unfortunate since it does not allow testing whether some IR methods are better than others at a particular degree of relevance. This neither allows rewarding IR methods the more, the better (the more relevant) documents they are able to rank high in retrieval results.

Scales of measurement of relevance. The binary measurement of relevance implies a category scale while the multiple degree measurement an ordinal scale (in some papers still referred to as a category scale). An ordinal scale does not allow to inferences like "a document of relevance degree 3 is three times as relevant as a document

of relevance degree 1". This approach leads to the evaluation of IR systems at each level of relevance separately. Kekäläinen and Järvelin [11] proposed such evaluation (see below).

However, a single combined measure is more desirable. It should take the relevance levels into account and credit IR systems for retrieving documents of all levels of relevance, but more for the highly relevant ones. This however requires accepting that "a document of relevance degree 3 is three times as relevant as a document of relevance degree 1". Continuous relevance scales have in fact been studied empirically and analytically, and used for the measurement of relevance. Therefore Kekäläinen and Järvelin [11] propose free weighting of relevance levels so that the evaluator can compare IR methods through several weighting schemes from neutral (collapsing relevance levels to binary) to sharp (heavier weighting of more relevant documents). Therefore evaluation measures that allow any number of relevance degrees and consistent weighting are needed. Kekäläinen and Järvelin [11] propose such an evaluation measure, called generalized recall and precision, which allows computing generalized MAP over graded assessments (see below).

3 Test Collections with Graded Relevance

3.1 Construction – The TREC-UTA Experience

The University of Tampere has created two IR test collections based on graded relevance assessments. These are TUTK [20] and TREC-UTA [21]. The former, created originally by Sormunen in 1992, contains some 55K Finnish newspaper articles and has assessments for 35 topics on a four-point scale. The latter is a 500K sub-collection of TREC-7 and TREC-8 with 41 topics assessed on a four-point scale. We discuss the construction of the latter below.

The goal in the creation of the TREC-UTA collection was to obtain a sub-collection of TREC 7 and 8 where the capability of IR systems to retrieve highly relevant documents could be studied. To achieve this, documents in the recall base for initially 38 topics (three were added later) were *reassessed* with a 4-point scale. A subset of TREC topics 351-450 was chosen observing the following principles:

- The number of TREC relevant documents required to exceed 30 per topic.
- A cost consideration: the number of relevant documents augmented with a 5% sample of the known non-relevant docs not to exceed 200 in a topic.
- A cultural requirements: selection of general topics rather than topics requiring knowledge of the American society or culture because the re-assessors were Finnish.

The four grades of relevance were:

- (0) The document does not contain information on topic.
- (1) The document only points to the topic - no more information than in the topic text (marginal relevance). Typical extent of relevant material: one sentence or fact.

- (2) The document is topical but not exhaustive (fair relevance). Typical extent of relevant material: one paragraph, 2-3 sentences or facts.
- (3) The document is exhaustive on the topic (high relevance). Typical extent of relevant material: several paragraphs, 4+ sentences or facts.

The relevance assessment process was as follows. Six students of information science were hired to do the assessments. They were given an introduction to the task and assessment guidelines. They were trained through two topics. For the assessments, all except very long documents were printed. The latter were examined on screen. For each topic, the process involved (a) an initial scanning of documents to establish stable relevance criteria, (b) note-taking on topic interpretations and resolution of conflicts between assessors, (c) partially parallel assessments by two assessors for quality control, (d) rough marking of relevant passages on documents, and (e) the assessment. Later, the assessments were compared to original TREC assessments for divergence analysis. [21]

Table 1. Relevance assessments in the TREC-UTA collection [21]

Level of relevance	TREC relevant		TREC nonrelev		UTA rel
	#	%	#	%	%
Rel=3	353	13	11	0	16
Rel=2	724	26	40	1	34
Rel=1	1004	36	134	5	50
Rel=0	691	25	2780	94	
Total	2772	100	2965	100	100

Table 1 indicates that nearly 6000 documents were reassessed. Among the originally TREC-relevant documents, a quarter was assessed non-relevant, over a third marginal, and 26 % and 13 % as fair or highly relevant, respectively. Among the originally TREC-non-relevant documents, 94 % were assessed non-relevant. In the final TREC-UTA collection, one half of the relevant documents were marginal, one third fair, and one sixth highly relevant. This indicates that findings in experiments using the TREC7-8 collections are heavily based on the IR methods' ability to retrieve marginal documents. This makes test collections based on graded (or just more stringent) relevance assessments important.

However, the downside of test collections providing graded assessments lies in their economics. Judging relevance liberally is fast. In graded assessment, extra work is required to specify the degree of relevance of each document. The total time to assess slightly over 7000 documents was about 20 person months (training etc. included). Note that about one half of the documents were relevant in this secondary analysis. This slows down the process – normally the density of relevant documents is much less. Experience improved judgment speed.

3.2 Sample Collections Providing Graded Relevance

We list in Table 2 some IR test collections that provide graded relevance assessments. The first two are from UTA and were discussed above. TREC WT10g [3] is a 10-gigabyte web collection with three-point relevance assessments. The NTCIR-4 Web is a 100 gigabyte web collection with four-point relevance assessments [17]. The table is not exhaustive.

Table 2. Some test collection providing graded relevance assessments

Collection	Collection size	No of relevance grades	No of topics	Average no of relevant/topic
TUTK	55K docs	4	35	36
Trec-UTA	500K docs	4	41	60
TREC WT10g	10 GB	3	50	52
NTCIR-4 Web	100 GB	4	80	4

4 Evaluation Metrics for Graded Relevance

Evaluation metrics for graded relevance are required in order to benefit from the assessments. Several metrics have been proposed. Below we shall introduce popular approaches and demonstrate that the metrics chosen may greatly affect what one observes. Firstly, the simplest approach is to collapse the scale of graded relevance into a binary one, which implies that a threshold for relevance and irrelevance is introduced within the relevance grades. Secondly, generalized recall and precision measures can be defined to directly handle varying degrees of relevance among the retrieved documents. Consequently, average precision and MAP are also generalized for graded relevance. These measures facilitate evaluation where IR methods are credited more for retrieving highly relevant documents. Thirdly, metrics based on cumulated gain, such as NDCG [6], may be used. Each approach facilitates also visual evaluation based on graphs. The following subsections provide a more detailed discussion.

4.1 Binarization of Graded Relevance

Even if the available relevance assessments may have had multiple degrees, these may be, and often are, collapsed into two for evaluation. For example, assume relevance assessments on a four-point scale (0 to 3 points, 3 denoting highly relevant, see Section 3). A liberal approach based on traditional metrics is to liberally consider documents of all degrees of relevance 1-3 as relevant and only documents of the degree 0 as non-relevant. A stringent approach only accepts highly relevant documents as relevant and takes the rest (0-2) as non-relevant. Figure 1 provides a comparison for visual analysis. The liberal case is on left and the stringent on the right side. The curves are for unexpanded and expanded versions of bag-of-words (SUM) queries,

Boolean structured (BOOL) queries, and bags-of-synonym-sets (SSYN) structured queries. On both graphs, the underlying queries and results are the same, only the binary relevance is different. One may observe that the best method (expanded SSYN) stands out on both graphs, whereas the differences between the other methods shrink significantly as one moves from liberal to stringent evaluation.

Because this approach involves binary relevance in evaluation, all traditional IR metrics, e.g. as provided by the *trec_eval* package[1], can be applied directly.

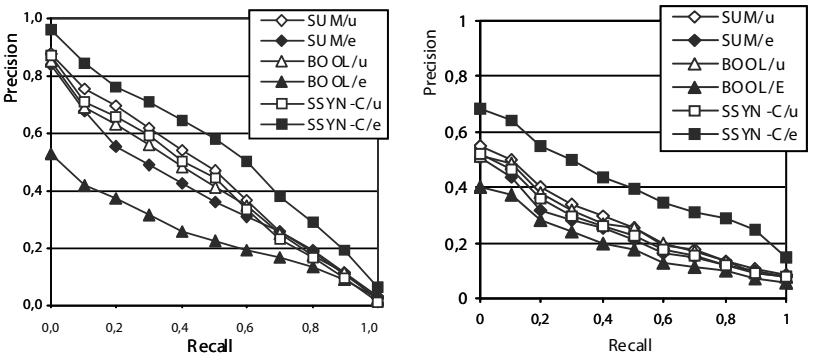


Fig. 1. Recall-precision curves of unexpanded (u) and expanded (e) SUM, BOOL, and SSYN-C queries (a) at relevance threshold 0/1-3, and (b) at relevance threshold 0-2/3. TUTK collection, 30 topics. [11]

4.2 Mean Average Precision Based on Graded Relevance

Another approach to employ graded relevance level is to generalize the binary measures of recall and precision to measures that credit the documents according to their degree of relevance. One must also allow inferences like "a document of relevance degree 3 is three times as relevant as a document of relevance degree 1". However, one may wish to assign varying weights to relevance degrees and one therefore needs to allow their weighting. One may thus gain insight into the behavior of IR methods under different evaluation scenarios. [11]

The *generalized, non-binary recall and precision* are defined as follows [11]. Let R be the set of documents retrieved from a database $D = \{d_1, d_2, \ldots, d_N\}$ in response to a query on some topic, $R \subseteq D$. Let the documents d_i in the database have relevance scores $r(d_i)$, being real numbers ranging from 0.0 to 1.0 with as many intermediate points as used in the study, with respect to the request behind the query. Generalized recall gR and generalized precision gP may now be computed by:

$$gP = \sum_{d \in R} r(d) / n \qquad gR = \sum_{d \in R} r(d) / \sum_{d \in D} r(d)$$

[1] http://trec.nist.gov/trec_eval/index.html

These can be computed and used like the traditional binary recall and precision. The generalized measures allow for any number of ranks on an ordinal scale, or a continuous scale of relevance assessments. They also allow reweighing of ordinal measurements to produce non-linear relationship of document value to its assessment rank. [11]

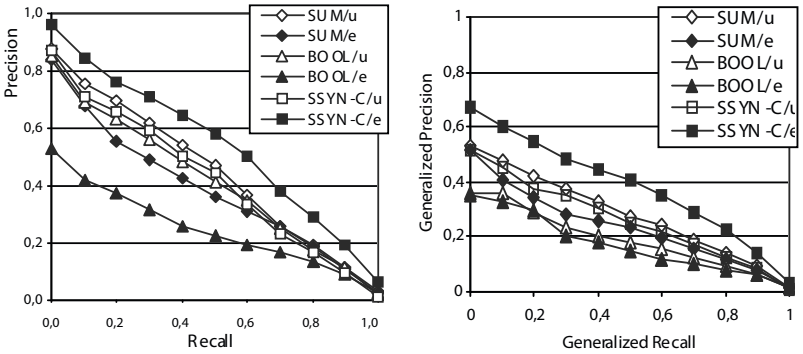


Fig. 2. Recall-precision curves of unexpanded (u) and expanded (e) SUM, BOOL, and SSYN-C queries (a) at relevance threshold 0/1-3 and (b) with generalized relevance and weights (0-1-5-10). TUTK collection, 30 topics. [11]

Figure 2 provides a comparison for visual analysis. The underlying query data are as in Figure 1, and Figure 2a is the same as Figure 1a. However, Figure 2b is based on generalized recall and precision so that the four relevance degrees 1-3 are assigned weights 0, 1, 5, and 10, respectively. Therefore fair documents are assessed as five times, and highly relevant documents as ten times, more valuable than marginal documents. One may observe that the best method (expanded SSYN) stands out as in Figure 1b, but the differences between the other methods in Figure 2b remain greater than in Figure 1b. By adjusting the weights, the graphs turn different. The weights should reflect the purpose of evaluation.

Because this approach allows calculating precision as a function of recall in evaluation, all traditional IR metrics, can be applied directly.

4.3 Cumulated Gain Based Metrics

Discounted Cumulated Gain (DCG) is an IR evaluation metric based on non-binary relevance assessments of documents in ranked retrieval. DCG assumes that highly relevant documents are more valuable than marginally ones for a searcher. It further assumes, that the greater the ranked position of a relevant document, the less valuable it is for the searcher, because the less likely it is that (s)he will ever examine the document – at least more effort is required to find it. DCG formalizes these assumptions by crediting a retrieval system (or a query) for retrieving relevant documents by their weighted degree of relevance, which is discounted by a factor dependent on the logarithm of the

document's ranked position. The steepness of the discount is controlled by the base of the logarithm and models the searcher's patience in examining the retrieval result. A small base (say, 2) models an impatient searcher while a large base (say, 10) a patient searcher. The Normalized Discounted Cumulated Gain (nDCG), the actual DCG performance for a query is divided by the ideal DCG performance for the same topic, based on the known relevant documents for the topic in a test collection. [6] [7]

Evaluation by (n)(D)CG assumes one query per topic/session. In real life however, interactive searchers often use multiple queries through reformulation and/or relevance feedback until they are satisfied or give up. Evaluation metrics assuming one query per topic are insufficient in multiple query session evaluation, where the searcher's reformulation and feedback effort matters. To overcome the single-query limitation, Jarvelin and colleagues [8] proposed a session-based metric sDCG for multiple interactive queries. This metric allows (normalized) DCG-style of cumulation and discounting over a sequence of query results. Because each query reformulation (or relevance feedback) involves searcher's effort, which is variable but always limited, sDCG supports further, and progressively stronger, discounting of any relevant documents found only after one or more reformulations. The rationale here is that an IR system (or searcher-system combination) should be rewarded less for relevant results found by later queries. Also the sDCG metric may by normalized.

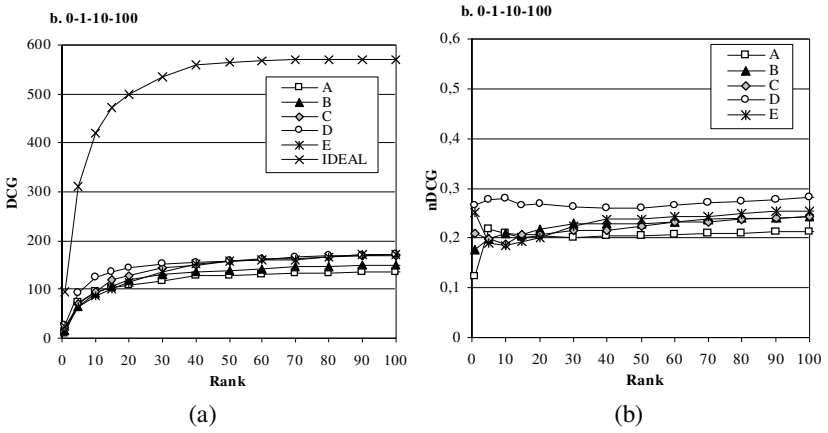


Fig. 3. DCG (a) and nDCG (b) curves based on weighting 0-1-10-100 and discounting log=2, top-100 results. The TREC-UTA collection, 20 topics. [6]

Figure 3 shows two graphs, (a) DCG curves by five participants A-E from TREC-7 ad hoc manual track and the ideal performance curve, and (b) the nDCG curves by the same five participants A-E. The DCG curves imply that the actual runs A-E all lag far below the ideal. Yet they have interesting differences: A never (within top-100) achieves the gain that D achieves at rank 15. After normalization, the superiority of D becomes more prominent. The normalized curves can be tested for significance of statistical differences. This may be based on the gain at some given rank, e.g. 10, or

on the average gain across a rank range, say 1 to 50. Modifications of the parameters for discounting and weighting reflect immediately as changes of the graphs. Note that the binary relevance case may be studies by assigning the same weight to all positive degrees of relevance. [6]

5 Doest it Matter? Evaluation Results Based on Graded Relevance

In the present section discuss some findings that allow the assessment of the significance of applying graded relevance in IR evaluation. We shall look at ranking IR systems by their performance using binary and graded relevance assessments. Next we shall look at graded relevance in relevance feedback. This is followed by graded relevance based evaluation of cross-language IR. Finally, we take up the issue of negatively weighting non-relevant documents. All these examples are from studies based on test collections. We shall however begin with user study, focusing on the searchers' ability to recognize documents of varying degrees of relevance as relevant.

5.1 Searchers' Ability to Recognize Degrees of Relevance

Vakkari & Sormunen [23] studied the effect of graded relevance on the results of interactive information retrieval (IR) experiments based on assigned search tasks in a test collection – the TREC-UTA collection. A group of 26 test persons searched for four topics using automatic and interactive query expansion based on relevance feedback. As a part of the study, the searcher-suggested pools of relevant documents were compared to the four-point relevance assessments in the test collection. The results indicated that the searchers were able to identify nearly all highly relevant documents and about half of the marginal ones. The searchers also selected a fair number of irrelevant documents for query expansion. The findings in [23] suggest that the effectiveness of query expansion is closely related to the searchers' success in retrieving and identifying highly relevant documents for feedback. This suggests that graded relevance is a meaningful category for human searchers and affects retrieval performance. At least this is so, if test collections provide liberal relevance judgments. One may question the worth of marginal documents for relevance feedback. However, this issue is tricky in the light of later findings (see below Section 5.3).

5.2 Ranking IR Systems by Graded Relevance

Voorhees [24] was the first to test extensively the effects of graded relevance and metrics based on it in IR experiments. For her experiment, the assessors for the TREC web track used a three-point relevance scale of non-relevant, relevant, and highly relevant. First, the relevant document sets were formed from all relevant documents and from only highly relevant documents. The relative effectiveness of IR techniques evaluated by these relevant document sets differed. Therefore, different retrieval techniques work better for retrieving only highly relevant documents. Voorhees found

this type of evaluation however, unstable since there are relatively few highly relevant documents per test topic. The discounted cumulative gain measure was found to increase evaluation stability by incorporating all relevance judgments while still giving precedence to highly relevant documents. The weighting of highly relevant documents was shown to affect the relative preferences order of IR techniques.

Kekäläinen [10] compared the rankings of IR systems based on binary and graded relevance in TREC 7 and 8 data, i.e., the above-mentioned TREC-UTA collection offering the four-point relevance scale. Twenty-one topics and 90 IR techniques (runs) from TREC 7 and 20 topics and 121 techniques from TREC 8 form the data. Binary precision, and cumulated gain, discounted cumulated gain and normalized discounted cumulated gain were the metrics compared. Several weighting schemes for relevance levels were tested. Kendall's rank correlations were computed to determine to what extent the rankings produced by different metrics are similar. Weighting schemes from flat (i.e. all relevant documents are equally relevant) to emphasizing highly relevant documents formed a continuum, where the metrics correlated strongly in the flat end, and less in the heavily weighted end.

The findings by Voorhees [24] and Kekäläinen [10] suggest that, if relevance grades are seen important in evaluation, the cumulated gain based metrics can make a difference in IR technique preference order.

5.3 Graded Assessments in Relevance Feedback

Keskustalo and colleagues [12] studied the effectiveness of relevance feedback (RF) based on simulation of interactive users in a test collection, the TREC-UTA collection. They first defined a user model, which helps to quantify essential interaction decisions involved in simulated RF. The model has three components. First, the relevance criterion defines the relevance level (0-3 in the four-point scale) the user requires of documents to be relevant to his/her needs and RF. Second, the browsing effort refers (as the number of documents) to the patience of the user to browse through the initial retrieved documents for giving feedback. Third, the feedback effort refers to the user's willingness (as the number of documents) to point out documents as suitable for RF. This model allowed the construction of several simulated RF scenarios in a laboratory setting. In addition to using graded relevance as a component in the user model, it was also employed in the evaluation of search results in the way suggested in Section 4.1.

Keskustalo and colleagues studied the effects of the quality and quantity of the feedback documents on the effectiveness of the RF and compared this to the pseudo-relevance feedback (PRF). After RF, the revised query was constructed by extracting new search keys (to be appended to the original query) from the full texts of the RF documents. Their initial results suggested that small amounts of high quality RF could compensate large amounts of low quality RF. When evaluation was by highly relevant documents, high-quality feedback paid off, but the trend was reversed when evaluation was by liberal relevance. PRF was only effective when evaluated by liberal relevance. The seen initial result documents were not frozen to their ranks but were re-ranked by the revised query.

In a later study Keskustalo and colleagues [13], using the same basic approach but different evaluation metrics (cumulated gain) and freezing of the seen documents, however found out that giving RF liberally (i.e. of mixed quality) and early (i.e. only a few documents) is most effective in various RF scenarios – also when the simulated users prefer finding highly relevant documents. Figure 4 exemplified this with two RF runs contrasted with the initial query baseline. The topmost run is based on examining at most 10 documents and liberally giving up to 10 relevant (levels 1-3) documents as RF (model <1,10,10>). The middle run differs in that RF is stringent, only highly relevant documents (level 3) are valid as RF (model <3,10,10>). The bottommost run is the initial query baseline, without any feedback. The relevance levels 0/1/2/3 were weighted in evaluation as follows: 0/1/10/100, respectively. Liberal feedback gives the best results because there is much more of it than in the stringent case.

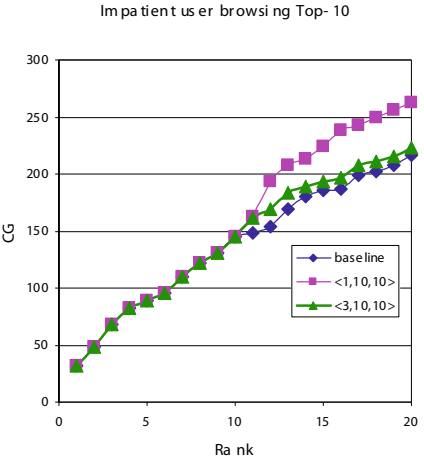


Fig. 4. The effect of RF shown in cumulated gain (CG) over result ranks 1-20 for three runs in TREC-UTA, top-to-bottom: Liberal feedback, stringent feedback and baseline (initial result without feedback). [13]

Järvelin [9] confirmed these findings, using a different approach. Here the query reformulation approach was based on query-biased summarization of RF documents identified in the initial query results. The finding was that a small amount of mixed-quality feedback from a short browsing window improves the final ranking the most. Longer browsing allows more feedback and better queries but also consumes the available relevant documents thereby reducing the chances for RF to improve results.

5.4 Graded Assessments in Cross-Language IR Evaluation

Lehtokangas and colleagues [15] investigated the effectiveness of dictionary-based in cross-language IR (CLIR) using graded relevance assessments in a best match retrieval environment. The test database was TUTK (see Section 3.2) containing 55K newspaper articles and a related set of 35 search topics were used in the tests. First,

monoligual baseline queries were formed automatically for the topics. Second, source language topics (in English, German, and Swedish) were automatically translated into the target language (Finnish) using both structured and unstructured queries. The unstructured queries were simple bag-of-words queries. In the structured ones, a synonym operator was used to combine the translation equivalents of each single source language word, as given by the dictionary. The queries were structured as bags of synonym sets [18]. Effectiveness of the variously translated queries was compared to that of the monolingual Finnish queries. CLIR performance (using MAP) was evaluated on three relevance thresholds: stringent (i.e. only highly relevant documents accepted), regular, and liberal (i.e. even marginal documents accepted). When liberal criteria were used, a reasonable performance – 80% - 90% of the monolingual – was achieved. However, under stringent criteria, considerable loss in performance compared to monolingual performance was observed – only 65% - 79% of the monolingual performance was achieved.

Lehtokangas and colleagues [16] continued to investigate the effectiveness of *transitive* dictionary-based CLIR using graded relevance assessments in a best match retrieval environment. The test database was TUTK (see above). Source language topics (in English, German, and Swedish) were automatically translated into the target language (Finnish) via an intermediate (or pivot) language. Effectiveness of the transitively translated queries along several translation routes was compared to that of the directly translated and monolingual Finnish queries. Pseudo-relevance feedback (PRF) was also used to expand the original transitive target queries. CLIR performance was again evaluated on three relevance thresholds: stringent, regular, and liberal. The transitive translations performed well achieving, on the average, 85-93% of the direct translation performance, and 66-72% of monolingual performance. The performance was the best under liberal evaluation and worst under strict evaluation. However, PRF was successful in raising the performance of transitive translation routes in absolute terms as well as in relation to monolingual and direct translation performance applying PRF.

In summary, graded relevance assessments seem to give interesting insight in the CLIR process suggesting that retrieving highly relevant documents may be more of a challenge than retrieving marginal documents for a topic.

5.5 Negative Weighting of Non-relevant Documents

Keskustalo and colleagues [14] proposed a new approach to IR evaluation, acknowledging that real users experience both gains and costs in retrieval. Traditional IR evaluation relying on binary topical insufficiently represents the user's viewpoint and thus insufficiently allows analyzing, e.g., why the user continues searching, or why and when the user stops. Therefore, traditional IR evaluation would only partially support understanding users' searching behavior. The authors argued that honest evaluation of retrieval results from the user's viewpoint requires taking into account both the user's gains and costs/frustrations. They therefore defined an extension to the (normalized) DCG measure involving explicit negative gain values for non-relevant documents. The extended metric was utilized it to represent the frustration of

searchers encountering non-relevant documents. They demonstrated the effects of negative gains in IR evaluation through traditional binary-scale relevance data (50 topics of the TREC8 test collection). They used also graded relevance and the 41 topics of the TREC-UTA collection. Figure 5 illustrates the latter case with patient and impatient users and in both cases, with positive-only and mixed gain weighting.

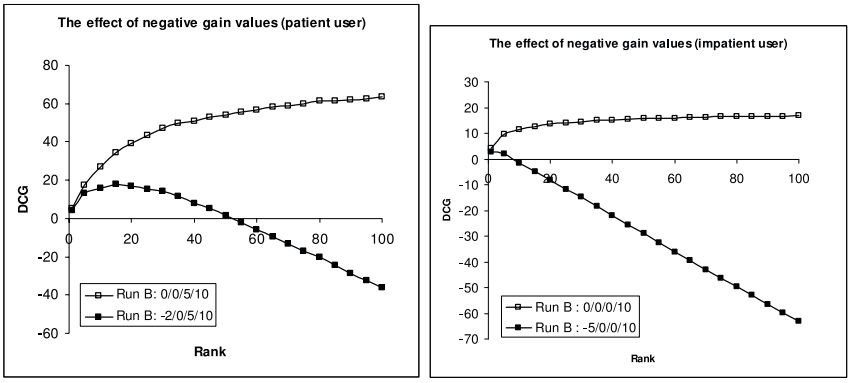


Fig. 5. Average DCG results for 41 topics (TREC-UTA collection) using graded relevance assessments. The left graph patient user (logarithm base 10): the same retrieval results illustrated based on positive-only weighting 0/0/5/10 (the upper curve), and with mixed weight values -2/0/5/10 (the lower curve). The right graph, impatient user (logarithm base 2): the same retrieval results illustrated based on positive-only weighting 0/0/5/10 (the upper curve), and with mixed weight values -2/0/5/10 (the lower curve). [14]

All four curves in Figure 5 illustrate the same average search result for 41 topics. The only differences are is user patience – high on the left, low on the right – and in relevance gain weighting on non-relevant documents – the lower curves in each graph employing a negative gain of -2 for non-relevant documents. While the positive-only curves implicitly suggest that further gain may be earned by digging deeper into the search results, the curves representing negative weights for non-relevant documents make it clear that one would sooner or later abandon browsing as counter-productive. User's stopping thus becomes understandable.

6 Discussion and Conclusion

We have discussed graded relevance and building test collections that provide graded relevance assessments. We have also presented evaluation approaches and metrics based on graded relevance assessments. These varied from variable binarization of graded relevance scales for traditional evaluation to using evaluation metrics directly relying on graded assessments. We have also examined the effects of using graded relevance assessments in retrieval evaluation and discussed some evaluation results based on graded relevance. These covered ranking IR systems by graded relevance, graded assessments in relevance feedback, graded assessments in cross-language IR

evaluation, and negative weighting of non-relevant documents. In each case we found that evaluation based on graded relevance assessments tell a different story about what is happening in IR (experiments). Also differences between sloppy and effective IR methods may be observed in evaluation.

With the metrics directly relying on graded assessments (like generalized MAP), investigators may use as many degrees of relevance as needed. The evaluation scenario of the study is then used to determine, which weighting schemes for document relevance are appropriate. By trying out several schemes, the evaluator gains better insight into IR method behaviour with more or less emphasis on highly relevant documents.

In conclusion, we find that graded relevance provides new insight into IR phenomena and affects the relative merits of IR methods. Thus graded assessments facilitate the formulation of novel research questions. However, if one's interest in solely in finding which search engine (or retrieval technique) is best for a given test collection, graded relevance assessments may not cause a radical change in the findings – compared to traditional evaluation.

References

1. Blair, D.C., Maron, M.E.: An evaluation of retrieval effectiveness for a full-text document-retrieval system. Communications of the ACM 28(3), 289–299 (1985)
2. Harman, D.: Private communication on TREC relevance judgments (February 1, 2001)
3. Hawking, D.: Overview of the TREC-9 Web Track. In: Voorhees, E., Harman, D. (eds.) The Ninth Text REtrieval Conference, TREC 9 (2011), `http://trec.nist.gov/pubs/trec9/t9_proceedings.html` (visited April 10, 2011)
4. Hersh, W.R., Hickam, D.H.: An evaluation of interactive Boolean and natural language searching with an online medical textbook. Journal of the American Society for Information Science 46(7), 478–489 (1995)
5. Ingwersen, P., Järvelin, K.: The Turn: Integration of Information Seeking and Retrieval in Context. Springer (2005)
6. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems 20(4), 422–446 (2002)
7. Järvelin, K., Kekäläinen, J.: Discounted Cumulated Gain. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems. Springer (2009)
8. Järvelin, K., Price, S.L., Delcambre, L.M.L., Nielsen, M.L.: Discounted Cumulated Gain Based Evaluation of Multiple-Query IR Sessions. In: Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., White, R.W. (eds.) ECIR 2008. LNCS, vol. 4956, pp. 4–15. Springer, Heidelberg (2008)
9. Järvelin, K.: Interactive Relevance Feedback with Graded Relevance and Sentence Extraction: Simulated User Experiments. In: Cheung, D., et al. (eds.) Proceedings of the 18th ACM Conference on Information and Knowledge Management, pp. 2053–2056. ACM, New York (2009)
10. Kekäläinen, J.: Binary and graded relevance in IR evaluations - Comparison of the effects on ranking of IR systems. Information Processing & Management 41(5), 1019–1033 (2005)
11. Kekäläinen, J., Järvelin, K.: Using graded relevance assessments in IR evaluation. Journal of the American Society for Information Science and Technology 53(13), 1120–1129 (2002)

12. Keskustalo, H., Järvelin, K., Pirkola, A.: The Effects of Relevance Feedback Quality and Quantity in Interactive Relevance Feedback: A Simulation Based on User Modeling. In: Lalmas, M., MacFarlane, A., Rüger, S.M., Tombros, A., Tsikrika, T., Yavlinsky, A. (eds.) ECIR 2006. LNCS, vol. 3936, pp. 191–204. Springer, Heidelberg (2006)
13. Keskustalo, H., Järvelin, K., Pirkola, A.: Evaluating the Effectiveness of Relevance Feedback Based on a User Simulation Model: Effects of a User Scenario on Cumulated Gain Value. Information Retrieval 11(5), 209–228 (2008)
14. Keskustalo, H., Järvelin, K., Pirkola, A., Kekäläinen, J.: Intuition-Supporting Visualization of User's Performance Based on Explicit Negative Higher-Order Relevance. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 675–682. ACM, New York (2008)
15. Lehtokangas, R., Keskustalo, H., Järvelin, K.: Dictionary-based CLIR loses highly relevant documents. In: Losada, D.E., Fernández-Luna, J.M. (eds.) ECIR 2005. LNCS, vol. 3408, pp. 421–432. Springer, Heidelberg (2005)
16. Lehtokangas, R., Keskustalo, H., Järvelin, K.: Experiments with Transitive Dictionary Translation and Pseudo-Relevance Feedback Using Graded Relevance Assessments. Journal of the American Society for Information Science and Technology 59(3), 476–488 (2008)
17. NTCIR-4 WEB test collection, `http://research.nii.ac.jp/ntcir/permission/ntcir-4/perm-en-WEB.html` (visited April 10, 2011)
18. Pirkola, A.: The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. In: Proceedings of the 21st Annual International ACM Sigir Conference on Research and Development in Information Retrieval, pp. 55–63. ACM, New York (1998)
19. Saracevic, T., Kantor, P., Chamis, A., Trivison, D.: A study of information seeking and retrieving. I. Background and methodology. Journal of the American Society for Information Science 39(3), 161–176 (1988)
20. Sormunen, E.: A Method for measuring Wide Range Performance of Boolean Queries in Full-Text Databases. University of Tampere, Acta Electronica Universitatis Tamperensis (2000), `http://acta.uta.fi/pdf/951-44-4732-8.pdf` (visited April 10, 2011)
21. Sormunen, E.: Liberal relevance criteria of TREC—Counting on negligible documents? In: Beaulieu, M., Baeza-Yates, R., Myaeng, S.H., Järvelin, K. (eds.) Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 324–330. ACM, Tampere (2002)
22. Tang, R., Shaw, W.M., Vevea, J.L.: Towards the identification of the optimal number of relevance categories. Journal of the American Society for Information Science 50(3), 254–264 (1999)
23. Vakkari, P., Sormunen, E.: The Influence of Relevance Levels on the Effectiveness of Interactive Information Retrieval. Journal of the American Society for Information Science 55(11), 963–969 (2004)
24. Voorhees, E.: Evaluation by Highly Relevant Documents. In: Proceedings of the 24th ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 74–82. ACM, New Orleans (2001)

Term Conflation and Blind Relevance Feedback for Information Retrieval on Indian Languages

Johannes Leveling, Debasis Ganguly, and Gareth J.F. Jones

CNGL, School of Computing, Dublin City University, Dublin 9, Ireland
{jleveling,dganguly,gjones}@computing.dcu.ie

Abstract. For the first participation of Dublin City University (DCU) in the FIRE 2010 evaluation campaign, Information Retrieval (IR) experiments on English, Bengali, Hindi, and Marathi documents were performed to investigate term conflation, Blind Relevance Feedback (BRF), and manual and automatic query translation. The experiments are based on BM25 and on language modeling (LM) for IR. Results show that term conflation always improves Mean Average Precision (MAP) compared to indexing unprocessed word forms, but different approaches seem to work best for different languages. For example, in monolingual Marathi experiments indexing 5-prefixes outperforms our corpus-based stemmer; in Hindi, corpus-based stemming approach achieves a higher MAP. For Bengali, the LM retrieval model with the rule based stemmer achieves a higher (but not significantly higher) MAP than BM25 with a corpus based stemmer (0.4583 vs. 0.4526). In all experiments, BRF yields considerably higher MAP in comparison to experiments without it. Bilingual IR experiments (English to Bengali and English to Hindi) are based on query translations obtained from native speakers and the Google translate web service. For the automatically translated queries, MAP is slightly (but not significantly) lower compared to experiments with manual query translations. The bilingual English to Bengali (English to Hindi) experiments achieve 81.7%-83.3% (78.0%-80.6%) of the best corresponding monolingual experiments.

1 Introduction

The Forum for Information Retrieval Evaluation[1] (FIRE) provides document collections, topics, and relevance assessments for information retrieval (IR) experiments on Indian languages. Similar to other IR evaluation initiatives such as TREC[2], NTCIR[3], or CLEF[4], FIRE aims at comparing the retrieval performance of different systems and approaches and at investigating evaluation methods for IR [1]. FIRE started in 2008 with document collections for English, Bengali, Hindi, and Marathi.

[1] http://www.isical.ac.in/~fire/
[2] http://trec.nist.gov/
[3] http://research.nii.ac.jp/ntcir/
[4] http://www.clef-campaign.org/

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 295–309, 2013.

This paper describes the participation of Dublin City University (DCU) in the FIRE 2010 evaluation. Monolingual and bilingual IR experiments for English and for the Indian languages Bengali, Hindi, and Marathi have been performed to investigate aspects of term conflation (stemming and prefix indexing), blind relevance feedback (BRF), and manual and automatic query translation.

The rest of this paper is organized as follows: Section 2 introduces related work, Section 3 discusses term conflation approaches in detail. Section 4 describes the BRF algorithm, Section 5 describes preparations for the retrieval experiments and the experimental setup. The experiments and results for monoliongual and bilingual IR experiments are presented in Section 6. The paper concludes with an outlook on future work in Section 7.

2 Related Work

There has been little research on IR for Indian languages before the start of FIRE. Larkey, Connell et al. normalized Hindi multi-byte characters using manually crafted rules for the TIDES (Translingual Information Detection, Extraction, and Summarization) surprise language exercise for Hindi IR experiments [2]. More recently, research on information retrieval on Indian languages has been encouraged by the FIRE 2008 evaluation campaign.

Dolamic and Savoy [3] used language modeling (LM) and divergence from randomness (DFR) for Indian language IR in the FIRE 2008 evaluation. Their approach employs light stemming [4], stopword removal based on small stopword lists, and Rocchio-style blind relevance feedback with $\alpha = \beta = 0.75$.

Xu and Oard [5] applied a Perl Search engine on the FIRE data for English-Hindi CLIR. They employed a stopword list with 275 words for Hindi IR.

McNamee employed n-grams and skipgrams (n-grams with wildcards) as indexing units for IR on English, Bengali, Hindi, and Marathi documents using language modeling (LM) as a retrieval model [6]. He experimented with different but fixed numbers of expansion terms for different indexing methods: 50 feedback terms for words, 150 for 4-grams and 5-grams, and 400 for skip-grams. Additional experiments on the FIRE 2008 data used n-grams on running text, and word truncation (prefixes) [7]. Significant improvements for indexing n-grams compared to the baseline of indexing words were observed. The best effectiveness for Hindi and Bengali was achieved when using 4-grams, highest MAP for Marathi by word-internal 4-grams.

Stemming approaches can be classified into different categories, e.g. by the results produced by the stemmer (light stemming [8] vs. aggressive stemming [9]) or by the resources used (corpus-based [10] vs. dictionary-based [11]).

The most widely used stemming approach for English is the rule-based Porter stemmer [12], which successively applies rules to transform a word form into its base form. The successive removal of affixes means that words with a recursive morphological structure are reduced to their base form, e.g. words such as *'hopelessness'* may be reduced to *'hope'* by removing the suffixes *'ness'* and *'less'*.

Light stemming focuses on removing only a few but the most frequent suffixes from word forms. Recently, light stemming has been researched as a less aggressive means to reduce words to their root form. For English, the *s*-stemmer which removes only the *'-s'*, *'-es'*, and *'-ies'* suffixes from words and other light stemming approaches have been proposed (see, for example, [13] and [4]).

YASS is a clustering-based suffix stripper which has been applied to documents in English, French, and Bengali [14]. YASS identifies clusters of equivalence classes for words by calculating distance measures between strings. This stemmer relies on multiple word lists which have to be extracted from documents, i.e. all words starting with the same character have to be collected in the same word list in a scan over all documents.

Xu and Croft [10] use a combination of aggressive suffix removal with co-occurrence information from small text windows to identify stemming classes. This technique is corpus-based and requires little knowledge about the document language. The original stemmer was developed for a Spanish document collection [10] and shows an increase in recall for Spanish.

Goldsmith [15] identified suffixes employing a minimum description length (MDL) approach. MDL reflects the heuristic that words should be split into a relatively common root part and a common suffix part. Every instance of a word (token) must be split at the same breakpoint, and the breakpoints are selected so that the number of bits for encoding documents is minimal.

Oard, Levow et al. [16] apply the Linguistica tool by Goldsmith [15] to create a statistical stemmer. Suffix frequencies are computed for a subset of 500,000 words in a document collection. The frequencies of suffixes up to a length of 4 were adjusted by subtracting the frequency of subsumed suffixes. Single-character suffixes were sorted by the ratio between their final position likelihood and their unconditional likelihood. Suffixes were sorted in decreasing order of frequency, choosing a cutoff value where the second derivative of the frequency vs. rank was maximized.

3 Term Conflation

Three approaches to term conflation were investigated for the experiments described in this paper as follows: i) *n*-prefixes, ii) corpus-based stemming, and iii) rule-based stemming. The following subsections provide details of these approaches.

3.1 N-prefixes

The goal of term conflation is to reduce different derivational or inflectional variants of the same word to a single indexing form to increase effectiveness (and efficiency). Full word forms can be conflated by truncating words after n characters. This approach is inexpensive and language-independent, because it uses word and character n-gram strings and does not rely on additional external language resources. For experiments on the FIRE 2008 document collections and

topics, we found that $n = 5$ or $n = 6$ produced the highest MAP for all languages tested (English, Bengali, Hindi and Marathi).

3.2 Corpus-Based Stemming

A corpus-based, language-independent stemming approach was implemented following the morpheme induction approach described by Dasgupta and Ng [17], which has been evaluated for English and Bengali. On a manually annotated set of Bengali words this approach achieved a substantially higher F-score than Linguistica [15].

For the retrieval experiments described in this paper, the first steps of the morpheme induction approach were implemented to obtain a stemmer. The additional morpheme induction steps described in [17] mainly test the validity of composite suffix candidates and suffix attachments. The morpheme induction approach produces a list of candidate suffixes based on a frequency analysis of potential word roots and suffixes. For example, the English word *'hopeful'* is split into the root-suffix pairs *'hop'*+*'eful'*, *'hope'*+*'ful'*, and *'hopef'*+*'ul'* (+ denotes the concatenation of strings). The middle variant is chosen, because its root and suffix frequency are highest.

In a second step, suffix combinations (forming composite suffixes) are determined via the frequency of potential root forms, which allows for a recursive morphological word structure. A word is stemmed by removing the longest suffix found in the generated suffix lists or by not removing a suffix, otherwise.

The list of candidate suffixes is produced following a method suggested by [18]. In the first step, all words w are analyzed by successively selecting all possible segmentation points, splitting into a potential root form r and a suffix s. Thus, w is the concatenation of r and s. If the potential root form r can also be found in the set of raw word forms (e.g. it is part of the collection vocabulary and the root frequency is higher than 0), s is added to the list of suffix candidates and r is added to the list of root candidates. Candidate suffixes are filtered as follows:

1. As a minor variation of the approach proposed by Dasgupta and Ng [17], suffixes with a frequency less than a given threshold t_f are removed (in this case, $t_f < 5$).
2. A score is assigned to each suffix by multiplying the suffix frequency and the suffix length in characters. Using suffix length as a scoring factor is motivated by the observation that short, low-frequency suffixes are likely to be erroneous [15].

The suffix candidates are then ranked by their score to obtain the top K suffixes. For the experiments described here, a fixed value of $K = 50$ was used for all languages tested. Dasgupta and Ng used the same number of suffixes for morpheme induction for English [17]. Considering that about 60 affix removal rules are defined by the Porter stemmer this seems a plausible setting for mildly aggressive stemming.

In a second step, composite suffixes are detected by combining all suffixes in the induced candidate list, e.g. *'less'*+*'ness'* in *'fearlessness'*. The detection of

composite suffixes s_1+s_2, builds on the assumption that a root form r will also combine with part of the suffix (s_1). This property typically does not hold for non-composite suffixes. The morpheme induction method presumes that s_1+s_2 is a composite suffix if s_1+s_2 and s_1 are similar in terms of the words they can combine with. Specifically, s_1+s_2 and s_1 are considered to be similar if their similarity value – which is calculated as shown in Equation 1 – is greater than a threshold t_s (specifically, $t_s > 0.6$ was used).

$$similarity(s_i + s_j, s_i) = P(s_i|s_i + s_j) = \frac{|W^{iji}|}{|W^{ij}|} \quad (1)$$

where $|W^{iji}|$ is the number of distinct words that combine with both s_i+s_j and s_i, and $|W^{ij}|$ is the number of distinct words that combine with s_i+s_j.

The corpus-based stemmer reads the lists of suffixes and processes words which are longer than a given threshold t_l ($t_l = 3$). All other words remain unstemmed. The stemmer determines the one longest suffix in the suffix lists (if any) and removes it from the word to produce a root form. Some example suffixes which are removed by the corpus-based stemmer for Bengali words are shown in Table 1. Throughout this paper we use the ITRANS[5] encoded phonetic transliterations of Indian language texts. Stemming rules are specified as A[*suffix*] → A where 'A' denotes a string in Bengali.

Table 1. Rules mined for Bengali using corpus-based stemming

Stemming rule	Example
A[খানি] (khAni) → A	ছবিখানি (Chabi[khAni]) → ছবি (Chabi)
A[খানা] (khAnA)→ A	টুকরোখানা (Tukro[khAnA]) → টুকরো (Tukro)
A[ভাবে] (bhAbe)→ A	বিস্তারিতভাবে (bistArita[bhAbe]) → বিস্তারিত (bistArita)
A[দের] (der) → A	ছেলেদের (Chele[der]) → ছেলে (Chele)
A[গণের] (gaNer) → A	জনগণের (jana[gaNer]) → জন (jana)
A[গামী] (gAmI) → A	কলকাতাগামী (kolkAtA[gAmI]) → কলকাতা (kolkAtA)

3.3 Rule-Based Stemming for Bengali

The advantages of a rule-based stemmer over a corpus based approach are: i) it is much faster because it does not require any pre-processing step on the indexed documents; ii) the corpus based methods are error prone due to under-training in the presence of a corpus, which is not large enough for statistical training; and iii) highly frequent proper nouns might lead to stemming errors. A disadvantage is that the stemming rules may have to be created manually and for each language. For the rule-based stemmer employed for Bengali IR, the stemming rules were compiled manually by one of the authors who is a native Bengali speaker.

[5] `http://www.aczoom.com/itrans/`

Bengali is an Indo-Aryan language spoken by more than 200 million people in Bangladesh and the Indian state of West Bengal. Bengali is a highly inflectional language with frequent compound suffixes which makes it necessary to apply rules in steps. Morphogical affixing in Bengali can be categorized into: a) Inflectional, where the part-of-speech of the inflected word remains unchanged; and b) Derivational, where the part-of-speech of the inflected word changes.

Since nouns, typically due to their higher Inverse Document Frequency (*idf*) values, are more important in IR than other parts-of-speech [19], for inflectional morphology we restrict our investigation to nouns only. Bhattacharya et al. [20] show that noun inflections can grouped into:

i) *Title markers:* These are the titles such as দেবী (Mrs.), বাবু (Sir) etc. which are added as suffixes to proper nouns.
ii) *Classifier:* Used to denote plurality and specificity of a noun e.g. a root word ছবি (picture) may be inflected as ছবিগুলো (pictures) or ছবিটা (the picture). A classifier can also indicate the gender of a noun e.g.ছাত্র (student) may be inflected to ছাত্রী to particularly denote a female student.
iii) *Case marker:* Used to denote possessive or accusative relations with other words. The possessive case marker for English is the apostrophe character. English does not use accusative markers. An example of an possessive marker is পরিবারের where the suffix ের is added to the root form পরিবার (family) to mean "family's".
iv) *Emphasizer:* These markers are used to emphasize the current word e.g. ছবি may be inflected to ছবিই to denote an equivalent of "only a picture" in English.

All of the above suffix types can appear in a word but only in the specified order e.g. ছবিগুলোকেও, where গুলো (a plurality classifier), কে (an accusative marker) and ও (an emphasizer) have been used to derive "'also those pictures" from the root word ছবি With reference to the above example, we see that for English language IR "also" and "those" can be easily removed since these are stopwords and the trailing "s" which is the plural classifier, can be removed by a simple rule. But in Bengali it is difficult since an English phrase can map to a single word and not normalizing this word to the base form can result in a poor retrieval performance. Title markers if present come before the case markers.

Algorithm 1 shows the algorithm to remove the suffixes for Bengali and Table 2 illustrates a particular case in the control flow of the former. To handle compound suffixes rules are applied in a series of steps.

4 Blind Relevance Feedback

Blind relevance feedback (or Pseudo-Relevance Feedback) builds on the assumption that the top-ranked documents provide useful information for query expansion (QE) or for query rewriting. Typically, additional terms are extracted from the top ranked documents and all query terms are reweighted.

Algorithm 1. BengaliSuffixStripper(w)

```
len ← len[w]
{Drop the emphasizers}
if w[len-1] = ও or w[len-1] = ই then
    len ← len-1;
end if
{Drop the classifiers and case markers}
x = {তা, টা, টি, টুকু, কে, র, ের, দের ভাবে}
while ∃ x: w=w'x do
    w ← w'; len ← len[w'];
end while
{Drop title markers}
x = {কারী, শীল, দেবী, বাবু, ভাই}
while ∃ x: w=w'x do
    w ← w'; len ← len[w'];
end while
{Drop the plural suffixes}
x = {রা, গুলো, গুলি, গুলোতে, গুলিতে}
if ∃ x: w=w'x then
    w ← w'; len ← len[w'];
end if
{Drop the derivational suffixes}
V = {Bengali Vowels} ∪ {Bengali Matras}∪ {য়}
while w[len-1] ∈ V do
    len ← len-1
end while
if len>2 then
    w ← w[0...len-1]
end if
return w
```

For the experiments with the BM25 retrieval model, 20 feedback terms were extracted from the top ranked 10 documents. The blind relevance feedback approach employed follows the one described in [21] and [22].

For our LM runs we employed the LM retrieval module of SMART. It uses log likelihood for document generation probabilities involving Jelineck-Mercer smoothing as outlined in Equation 2 where λ_i is the smoothing parameter for the i^{th} query term.

$$P(d|q) = \sum_{t_i \in q} \log(1 + \frac{\lambda_i P(t_i, d)}{(1 - \lambda_i) P(t_i)}) \tag{2}$$

Feedback for the LM retrieval is carried out by the steps enumerated in Algorithm 2. Since the λ_i values are indicative of the importance of the ith query term and since the expansion terms are not chosen directly by the user, an intuitive approach is to set these λ_i to a somewhat lower value and simultaneously giving the λ_i of the original query terms a boost. Thus, we propose to use a higher value for β and a lower value for α as compared to the λ of the initial retrieval.

Table 2. Rules for simple suffixes with Bengali examples

Lines	Suffix type	Bengali notation	ITRANS notation
4	Emphasizer	আধিক্যই → আধিক্য	Adhikya[i] → Adhikya
4, 19	Emphasizer and plural classifier	মন্ত্রীরাও → মন্ত্রী	mantrI[rAo] → mantrI
9	Specific classifier	মুখোশটা → মুখোশ	mukhosh[TA] → mukhosh
9	Possessive case marker	ভারতের → ভারত	bhArat[er] → bhArat
9	Plural accusative case marker	শিল্পীদের → শিল্পী	shilpI[der] → shilpI
9, 9	Specific classifier and Possessive case marker	দুনিয়াটার → দুনিয়া	duniyA[TAr] → duniyA
14	Derivational	স্থিতীশীল → স্থিতী	sthitI[shIl] → sthitI
14	Title marker	করুনাদেবী → করুনা	karunA[debI] → karunA
9, 21	Plural accusative case marker and derivational	ভারতীয়দের → ভারতীয়	bhAratiya[der] → bhAratiya

Algorithm 2. $\mathtt{LM}_{\mathtt{BRF}}(R, T, \alpha, \beta)$

1: Score every term t in the top R documents by $L(t) = \sum_{d \in R} \log \frac{P(t|d)}{P(t)}$ [23].
2: Select T feedback terms having the top $L(t)$ scores.
3: Choose λ_i for the ith query term q_i as follows:

$$\lambda_i = \begin{cases} \beta & \text{if } q_i \text{ is a term in the original query} \\ \alpha & \text{otherwise} \end{cases} \quad (3)$$

5 Experimental Settings

5.1 Query Translation

Several cross-lingual IR experiments were conducted to compare CLIR performance for automtatic and manual query translation. The Bengali and Hindi queries were manually translated from English by native speakers. The Google translate web service[6] has been used for automatic query translation from English to Hindi.

5.2 Processing and Indexing

The FIRE document collection for ad hoc IR contains newspaper articles on various topics including sports, politics, business, and local news. The articles are represented as structured XML documents in TREC format, using UTF-8 encoding.

[6] `http://translate.google.com/`

FIRE topics resemble topics from other retrieval campaigns such as TREC in format and content. They comprise a brief phrase describing the information need (topic title, T), a longer description (topic description, D), and a part with information on how documents are to be assessed for relevance (topic narrative, N). Retrieval queries are always generated from the title and description fields of topics (TD) for the experiments described in this paper. For each language, fifty unique topics and the relevance assessments were provided together with the corresponding document collection. For all FIRE topics, relevant documents have been assessed by pooling submissions from systems participating in the FIRE retrieval track.

Not all documents could be indexed properly: some files include invalid XML characters or contain otherwise invalid XML; others contain no valid text at all. These documents have not been indexed at all, but they make up only a small portion of each collection.

The stopword lists used for the experiments described in this paper (stopword lists for English, Bengali, Hindi, and Marathi) originate from different sources. First, special characters (e.g. punctuation marks) were compiled in a list. For example, "।" (Danda, U+0964) can be used as a end of sentence marker in Bengali, similar to a full stop in English. A second list is created during indexing, containing the most frequent index terms. Terms occuring in more than half of all documents in the document collection are considered as stopwords.

The third source for stopwords is Jacques Savoy's web page on multilingual resources for IR at the University of Neuchâtel[7]. These stopword lists have been generated following an approach to obtain a general stopword lists for general text [24,8], in which the N most frequent words are extracted from the document collection, numbers are removed from the list, and the resulting stopword list is manually extended with additional word forms. The resulting stopword lists contain 571 words for English (the SMART stopword list), 119 for Bengali, 163 for Hindi, and 98 for Marathi. For the LM experiments, the stopword list for Bengali provided on the FIRE web site was emplyed (384 stopwords).

Unnormalized text encoded with UTF-8 may use different multi-byte character encodings for the same character. For example, the character *é* in the Spanish name *San José* may be encoded as a single byte (for *é*), as the byte sequence for e + ´ or as the byte sequence for ´ + e. For the experiments described in this paper, encoded text was normalized by following the guidelines for canonical decomposition followed by canonical composition from the International Components for Unicode (ICU) implementing the standard normalization forms described in the Unicode Standard Annex #15 - Unicode Normalization Forms[8]. These normalization steps guarantee a fixed order of characters where multiple variants are allowed.

In addition, text was processed by applying the following normalization rules.

1. Internal word space is removed (e.g. characters U+200c and U+200d)
2. 'Chandra bindu' and 'anusvara' are mapped to 'anusvara'.

[7] `http://members.unine.ch/jacques.savoy/clef/index.html`
[8] `http://www.unicode.org/unicode/reports/tr15/`

3. 'Chandra' followed by a vowel is mapped to the corresponding vowel.
4. 'Virama' is removed from the text.
5. Combinations of 'nukta' and a consonant are replaced by the corresponding consonant character.
6. Long vowels are mapped to the corresponding short form.
7. Some character sequences visually similar to a single glyph are mapped to a single character (e.g. letter A + sign O, letter A + sign AA + sign E, letter A + sign E + sign AA are mapped to the letter O).
8. Accents (which are typically part of transcribed foreign names) are removed.
9. Digit symbols in Bengali and Devanagari are mapped to Arabic numeric literals, because the FIRE data contains both forms.

These rules serve as a means to normalize orthographic variants.

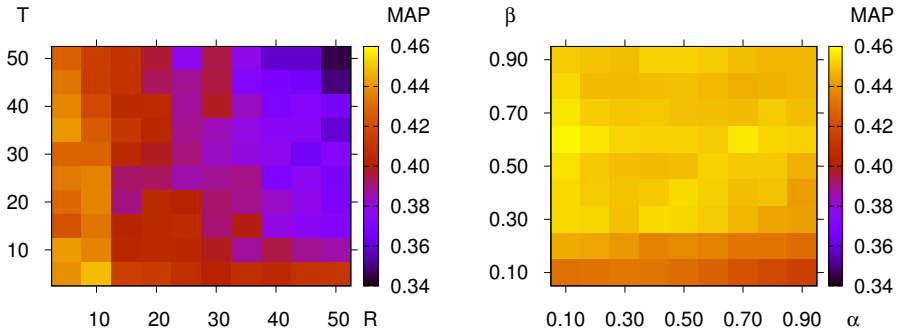
6 Retrieval Experiments and Results

For the experiments described in this paper, the Lucene IR toolkit was employed.[9] Lucene does not (yet) include state-of-the-art IR models or blind relevance feedback. Support for the BM25 model [21,22] and for the corresponding blind relevance feedback approach was implemented for Lucene by one of the authors. Runs with BM25 used 10 feedback documents and 20 feedback terms.

Additional experiments for the Bengali monolingual adhoc track were performed using LM implemented within the SMART system by one of the authors [25]. The submitted official LM runs used 35 feedback documents and terms. The LM experiments used different seetings for λ: $\lambda = 0.3$ for the baseline run and $\lambda = 0.25$ for the run including blind relevance feedback. The parameters for the official submissions were set in an adhoc way. After the release of manual assessments, we did some more experiments to determine the optimal settings of λ and the BRF parameters - R (number of pseudo-relevant documents) and T (number of expansion terms). The results show that the highest MAP is achieved by using $\lambda = 0.5$. We then vary (R, T) in the range of $[5, 50]$ in steps of 5 using $\lambda = 0.5$ for both the original and expansion terms. The MAPs are reported in Figure 1a. We then experimented with the methodology of assigning different λs to the original and new terms as outlined in the algorithm of Section 4. Figure 1b plots the MAPs for different settings of α and β. Optimal results are obtained for (α, β)=$(0.1, 0.6)$, thus verifying the hypothesis that in the feedback step one should assign lower importance to the expansion terms and a higher importance to the original ones.

Stemming was done by the rule-based stemmer. It can be seen from Table 4 that the Bengali rule based stemmer improves MAP significantly. The last row also suggests that assigning a higher λ for the original terms and lower for the expansion terms improves retrieval effectiveness further. The following parameters were varied in our FIRE 2010 experiments:

[9] `http://lucene.apache.org/`



(a) Effect of varying (R, T) **(b)** Effect of varying (α, β)

Fig. 1. LM experiments on Bengali FIRE-2010 topics

- the source and target language for topics/queries (EN: English, MR: Marathi, BN: Bengali, HI: Hindi);
- the translation method for bilingual experiments (Nat: manual translation by native speaker, GT: the Google translate web service);
- the type of word processing before indexing (PS: Porter stemming, P5: 5-prefixes, CS: corpus-based stemming, RS: rule-based stemming, NP: no processing, raw word forms);
- the retrieval model (BM25: Okapi BM25, LM: Language modeling); and
- the use of blind relevance feedback (Y: yes, N: no).

The results shown in Table 3 include the number of relevant and retrieved documents (rel_ret), mean average precision (MAP), geometric MAP (GMAP), and precision at 10 and 20 documents (P@10 and P@20, respectively).

6.1 Results for Monolingual Experiments

Results for the monolingual IR experiments on the FIRE document collections are shown in Table 3 and Table 4. Both stemming approaches, the corpus-based stemmer and the rule-based stemmer, signifcantly outperform the baseline (no stemming) for all languages (measured using Wilcoxon test with confidence=99%). The highest individual result was achieved for English (0.4846 MAP). However, for Marathi retrieval experiments, indexing 5-prefixes yields the highest MAP. Query expansion additionally increases IR performance in these cases.

Interestingly, the performance of experiments using stems or 5-prefixes, combined with query expansion, is similarly high in most cases. This could indicate that even a crude method such a truncating words after a fixed number of characters works well in general. For languages with few resources or complex morphology, using 5-prefixes can be considered as an alternative to stemming and might even make the development of a stemmer redundant.

Table 3. Results for monolingual and bilingual official runs for FIRE 2010

Run	Parameters					Results			
ID	lang.	transl.	index	retrieval	BRF	rel_ret	MAP	GMAP	P@10
S1	EN	-	PS	BM25	N	650	0.465	0.361	0.432
S1TD_QE20	EN	-	PS	BM25	Y	652	**0.485**	0.354	0.438
P5TD_QE20	EN	-	P5	BM25	Y	652	0.477	0.352	0.442
B0	MR	-	NP	BM25	N	550	0.236	0.012	0.272
P5TD_QE20	MR	-	P5	BM25	Y	614	**0.341**	0.023	0.328
S2TD_QE20	MR	-	CS	BM25	Y	601	0.291	0.013	0.302
B0	BN	-	NP	BM25	N	489	0.386	0.291	0.330
P5TD_QE20	BN	-	P5	BM25	Y	502	**0.453**	0.372	0.380
S2TD_QE20	BN	-	CS	BM25	Y	499	0.400	0.327	0.342
B0	HI	-	NP	BM25	N	846	0.390	0.227	0.452
P5TD_QE20	HI	-	P5	BM25	Y	895	0.445	0.296	0.444
S2TD_QE20	HI	-	CS	BM25	Y	902	**0.468**	0.330	0.480
P5TD_QE20	EN→BN	Nat	P5	BM25	Y	451	0.370	0.282	0.341
S2TD_QE20	EN→BN	Nat	CS	BM25	Y	447	**0.377**	0.279	0.313
P5TD_QE20	EN→HI	Nat	P5	BM25	Y	725	**0.377**	0.252	0.448
S2TD_QE20	EN→HI	Nat	CS	BM25	Y	724	0.375	0.244	0.444
P5TD_QE20	EN→HI	GT	P5	BM25	Y	724	0.365	0.240	0.436
S2TD_QE20	EN→HI	GT	CS	BM25	Y	722	0.368	0.236	0.432

Table 4. Post-submission monolingual LM runs for Bengali for FIRE 2010

Run	Parameters							Results			
ID	lang.	λ	R	T	(α, β)	index	BRF	rel_ret	MAP	GMAP	P@10
NOSTEM	BN	0.5	0	0	(0.5, 0.5)	NP	N	486	0.372	0.288	0.326
INIT	BN	0.5	0	0	(0.5, 0.5)	RS	N	500	0.433	0.360	0.356
FDBK_CONST	BN	0.5	10	5	(0.5, 0.5)	RS	Y	503	0.449	0.369	0.360
FDBK_VAR	BN	0.5	10	5	(0.1, 0.6)	RS	Y	501	**0.458**	**0.382**	**0.368**

6.2 Results for Bilingual Experiments

Two sets of bilingual IR experiments have been performed (English to Bengali and English to Hindi). Experiments using manual translation of FIRE topics for Bengali to English achieved 81.7%-94.3% of the MAP for the corresponding monolingual experiments. Manual query translation for English to Hindi shows 87.0%-92.9% of the MAP for the corresponding monolingual experiments. In comparison, query translation by the Google translate web service shows a slightly (but not significantly) lower MAP and achieves 85.6%-89.8% of the MAP for the best monolingual Hindi run. Thus, contrary to our expectation, manual

query translation does necessarily not lead to a higher MAP. Possible explanations are that the automatic machine translation has already the same quality as a manual translation or that the queries are unambiguous and not difficult to translate.

7 Conclusions and Outlook

This paper described adhoc IR experiments at FIRE 2010, evaluating stemming, blind relevance feedback and query translation.

In combination with blind relevance feedback, both stemming approaches performed significantly better than the baseline of indexing words in all tested languages (English, Bengali, Hindi, and Marathi). An obvious improvement of the corpus-based stemming approach will be to determine the cut-off point for the suffixes dynamically. In all experiments and for all languages, a fixed set of 50 suffixes was used. For morphologically rich languages such as Bengali (with many different and composite morphological suffixes), this number is probably too low. The rule-based stemmer needs some further improvement because the inflections in a language like Bengali can not only be suffixial but also prefixial. Moreover two Bengali words often get merged together to form a compound word making it a challenging task to obtain compound constituents for indexing. We would like to explore on these language-specific issues in the coming years of participation at FIRE. Additional directions for future research for ad-hoc IR on Indian languages include indexing at the phrase level for common phrases, using common phrases as stopwords, and expanding queries with the help of an external resource such as a thesaurus.

Acknowledgments. This material is based upon works supported by the Science Foundation Ireland under Grant No. 07/CE/I1142. as part of the Centre for Next Generation Localisation (CNGL) project. Special thanks to Ankit Srivastava and Sudip Naskar for translating the queries.

References

1. Majumder, P., Mitra, M., Pal, D., Bandyopadhyay, A., Maiti, S., Mitra, S., Pal, S.: Text collections for FIRE. In: SIGIR 2008, pp. 699–700 (2008)
2. Larkey, L.S., Connell, M.E., Abduljaleel, N.: Hindi CLIR in thirty days. ACM Transactions on Asian Language Information Processing 2(2), 130–142 (2003)
3. Dolamic, L., Savoy, J.: UniNE at FIRE 2008: Hindi, Bengali, and Marathi IR. Working Notes of the Forum for Information Retrieval Evaluation, Kolkata, India, December 12-14 (2008)
4. Savoy, J.: Light stemming approaches for the French, Portuguese, German and Hungarian languages. In: Haddad, H. (ed.) Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23-27, pp. 1031–1035. ACM (2006)

5. Xu, T., Oard, D.W.: FIRE-2008 at Maryland: English-Hindi CLIR. Working Notes of the Forum for Information Retrieval Evaluation, Kolkata, India, December 12-14 (2008)
6. McNamee, P.: N-gram tokenization for Indian language text retrieval. Working Notes of the Forum for Information Retrieval Evaluation, Kolkata, India, December 12-14 (2008)
7. McNamee, P., Nicholas, C., Mayfield, J.: Addressing morphological variation in alphabetic languages. In: Allan, J., Aslam, J.A., Sanderson, M., Zhai, C., Zobel, J. (eds.) Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, July 19-23, pp. 75–82. ACM, Boston (2009)
8. Savoy, J.: A stemming procedure and stopword list for general French corpora. Journal of the American Society for Information Science 50(10), 944–952 (1999)
9. Lovins, J.B.: Development of a stemming algorithm. Mechanical Translation and Computation 11(1-2), 22–31 (1968)
10. Xu, J., Croft, B.: Corpus-based stemming using co-occurence of word variants. ACM Transactions on Information Systems 16(1), 61–81 (1998)
11. Krovetz, R.: Viewing morphology as an inference process. In: Korfhage, R., Rasmussen, E., Willett, P. (eds.) Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 191–202. ACM, Pittsburg (1993)
12. Porter, M.F.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
13. Harman, D.: How effective is suffixing? Journal of the American Society for Information Science 42(1), 7–15 (1991)
14. Majumder, P., Mitra, M., Parui, S.K., Kole, G., Mitra, P., Datta, K.: YASS: Yet another suffix stripper. ACM Transactions on Information Systems (TOIS) 25(4), 18:1–18:20 (2007)
15. Goldsmith, J.: Unsupervised learning of the morphology of a natural language. Computational Linguistics 27, 153–198 (2001)
16. Oard, D.W., Levow, G.-A., Cabezas, C.I.: CLEF experiments at maryland: Statistical stemming and backoff translation. In: Peters, C. (ed.) CLEF 2000. LNCS, vol. 2069, pp. 176–187. Springer, Heidelberg (2001)
17. Dasgupta, S., Ng, V.: High-performance, language-independent morphological segmentation. In: Sidner, C.L., Schultz, T., Stone, M., Zhai, C. (eds.) Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (NAACL HLT 2007), April 22-27, pp. 155–163. ACL, Rochester (2007)
18. Keshava, S., Pitler, E.: A simpler, intuitive approach to morpheme induction. In: PASCAL Challenge Workshop on Unsupervised Segmentation of Words Into Morphemes - MorphoChallenge 2005, Venice, Italy, April 12 (2006)
19. Xu, J., Croft, W.B.: Improving the effectiveness of informational retrieval with Local Context Analysis. ACM Transactions on Information Systems 18, 79–112 (2000)
20. Bhattacharya, S., Choudhury, M., Sarkar, S., Basu, A.: Inflectional morphology synthesis for bengali noun, pronoun and verb systems. In: Proceedings of the National Conference on Computer Processing of Bangla (NCCPB), pp. 34–43 (2005)
21. Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M.M., Gatford, M.: Okapi at TREC-3. In: Harman, D.K. (ed.) Overview of the Third Text Retrieval Conference (TREC-3), pp. 109–126. National Institute of Standards and Technology (NIST), Gaithersburg (1995)

22. Robertson, S.E., Walker, S., Beaulieu, M.: Okapi at TREC-7: Automatic ad hoc, filtering, VLC and interactive track. In: Harman, D.K. (ed.) The Seventh Text REtrieval Conference (TREC-7). NIST Special Publication 500-242, pp. 253–264. National Institute of Standards and Technology (NIST), Gaithersburg (1998)
23. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: SIGIR 1998: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 275–281. ACM, New York (1998)
24. Fox, C.: Lexical analysis and stoplists, pp. 102–130. Prentice-Hall, NJ (1992)
25. Ganguly, D.: Implementing a language modeling framework for information retrieval. Master's thesis, Indian Statistical Institute, India (2008)

Improving Cross-Language Information Retrieval by Transliteration Mining and Generation

K. Saravanan, Raghavendra Udupa, and A. Kumaran

Multilingual Systems Research
Microsoft Research India
Bangalore, India
{v-sarak,raghavu,kumarana}@microsoft.com

Abstract. The retrieval performance of Cross-Language Retrieval (CLIR) systems is a function of the coverage of the translation lexicon used by them. Unfortunately, most translation lexicons do not provide a good coverage of proper nouns and common nouns which are often the most information-bearing terms in a query. As a consequence, many queries cannot be translated without a substantial loss of information and the retrieval performance of the CLIR system is less than satisfactory for those queries. However, proper nouns and common nouns very often appear in their transliterated forms in the target language document collection. In this work, we study two techniques that leverage this fact for addressing the problem, namely, Transliteration Mining and Transliteration Generation. The first technique attempts to mine the transliterations of out-of-vocabulary query terms from the document collection whereas the second generates the transliterations. We systematically study the effectiveness of both techniques in the context of the Hindi-English and Tamil-English ad hoc retrieval tasks at FIRE2010. The results of our study show that both techniques are effective in addressing the problem posed by out-of-vocabulary terms with Transliteration Mining technique giving better results than Transliteration Generation.

Keywords: Cross-Language Information Retrieval System, FIRE 2010, Transliteration Mining, Transliteration Generation.

1 Introduction

With the exponential increase in non-English user population on the Internet over the last two decades, Cross-Language Information Retrieval (CLIR) has gained importance both as a research discipline and as an end-user technology. The importance of this discipline is evidenced by increased number of research publications, workshops and shared tasks, focusing on various aspects of information retrieval tasks in multilingual or cross-language settings. While there has been substantial progress in the core cross-language retrieval algorithms, the retrieval performance of any CLIR system is a function of the coverage of the translation lexicon used by the system. As query terms (or their statistics) must be translated in CLIR before the retrieval of

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 310–333, 2013.

documents, translation lexicon plays a large role in determining the retrieval performance no matter what retrieval algorithm is ultimately employed by the CLIR system. When query terms cannot be translated to the target language, there could be a loss of information and consequently, a loss in retrieval performance. The loss of information is substantial when the query terms are proper nouns or common nouns which are often the information bearing terms in the query.

Unfortunately, most translation lexicons do not provide a good coverage of proper nouns (names) and it turns out that names appear often in queries and constitute the largest class of out-of-vocabulary terms in CLIR [43].[1] This is not surprising because names form an open set in a language and every day new names enter into a language. Hence, it is practically impossible to keep a translation lexicon up-to-date. Further, the same source language name can have multiple variants in the target language due to the difference in the sounds used in the two languages and also due to inflections and agglutination. Therefore, even when the translation lexicon has an equivalent for a source language name, it might not be the variant that appears in the documents relevant to the query.

In cultures where a foreign language like English is widely used (say as a second language), many common nouns in the foreign language are imported to the native language by a phenomenon called code-mixing – the foreign language words are transliterated to the native language and used instead of their equivalents in the native language. As with proper nouns, most translation lexicons do not provide a good coverage of such imported common nouns.

Proper nouns and common nouns are often the most information bearing terms of a query and can cause severe loss in retrieval performance when left untranslated. However, in many cases, proper nouns and common nouns (those which are imported from a foreign language) are found in their transliterated forms in the target language document collection. In this work we study two techniques that leverage this fact to address the problem posed by out-of-vocabulary terms in CLIR:

1. Transliteration Mining
2. Transliteration Generation.

Transliteration Mining is a novel technique that was proposed in [43] and shown to give significant improvements in retrieval performance over a language model based CLIR baseline. It employs a two-pass retrieval approach. The source language query is translated using the translation lexicon ignoring all out-of-vocabulary (OOV) terms and a first pass retrieval is done. Transliterations for the out-of-vocabulary terms are then mined from the top results using a statistical transliteration model. The source language query is now retranslated using the translation lexicon and the transliterations thus mined and a second pass retrieval is done.

Transliteration Generation has been employed in CLIR by many works including [1, 46]. It generates the transliterations of the out-of-vocabulary terms using Machine Transliteration and uses them along with the translation lexicon to translate the source

[1] In fact, 60% of the topics in the 2000-2007 Cross-Language Evaluation Forum (CLEF) ad hoc retrieval tasks had at least one name and 18% of them had at least three.

language query. In this work, we employ a state of the art Machine Transliteration technique [19].

We systematically study the above mentioned techniques and evaluate their effectiveness relative to a language model based CLIR baseline. We also compare the two techniques to two oracles, which can identify for every out-of-vocabulary term, its equivalent in English topic and in the target collection, respectively. Our study is a continuation of the initial study done as part of FIRE 2010 ad hoc CLIR tasks [40].

The rest of the paper is organized as follows: In Section 2 we discuss some works that are related to our work. In Section 3 we discuss the CLIR system and two techniques for handling OOV terms. In Section 4 we discuss the experimental setup and in Section 5 we report the results.

2 Related Work

Basic CLIR systems that use translation lexicons have been studied in several works in early literature, for example [4, 18], but they suffer from the problem of OOV terms in the queries, which often are names. Leaving the OOV query terms untranslated is well recognized to have a significant negative impact on the performance of CLIR systems [8, 25, 26, 47]. Broadly, there are two distinct approaches taken to address the problem of OOV terms: The first approach is to employ a Machine Transliteration system to generate the transliteration equivalents in the target language [1, 2, 15, 17, 23, 46] , and use them for retrieval. The second approach is to enhance the translation lexicon offline, by mining the transliteration equivalents from parallel or comparable corpora [3, 10, 11, 29, 37, 43].

Several methods have been reported in CLIR literature where Machine Transliteration was employed on OOV query terms. They differ in the exact technique used for transliterating the OOV query terms to the target language [1, 14, 15, 34, 46, 51]. While most of the above works report improved retrieval performance, the improvements are modest. This is because Machine Transliteration, to be effective in CLIR, has to produce the exact string used in the document collection and not just any acceptable similar sounding string. However, we note that in the recent past there has been significant progress in Machine Transliteration as evidenced by the results of the shared task on Machine Transliteration at NEWS 2009, 2010, and 2011 workshops [20, 22, 50]. Thanks to this series of workshops, it is now possible to calibrate different Machine Transliteration techniques and use the best among them in CLIR.

Mining based approaches are used in Machine Translation to augment the translation lexicon with name transliterations. In the literature, there are many interesting corpus-based techniques for mining both translation equivalents and transliteration equivalents [5, 18, 29, 37, 39, 44, 45][2]. While many such techniques have addressed the OOV problem in Machine Translation and improved the quality of translated text, none of them are effective in ad hoc retrieval tasks. This is because corpus-based mining techniques might not always be successful in finding the transliteration

[2] For a more detailed discussion on corpus-based mining techniques, please see [45].

equivalents of the specific names and common nouns used in the topics of an ad hoc retrieval task. Therefore, the best place to look for transliterations of OOV terms of a query are the top results from the target collection itself for the query as hypothesized in [43]. In this approach, the source language query is translated using the translation lexicon ignoring all out-of-vocabulary (OOV) terms and a first pass retrieval is done. Transliterations for the out-of-vocabulary terms are then mined from the top results using a statistical transliteration model. This is the approach we also take in this study.

We at Microsoft Research India (MSR India) [28] fielded a CLIR system without addressing the OOV query terms in the CLEF 2007 [41] campaign for the Hindi-English track [13]. In FIRE 2008 campaign, we fielded a CLIR system that employed Transliteration Mining in Hindi-English track [42]. In 2010, FIRE organized several ad hoc monolingual and cross-language retrieval tracks, and we fielded a CLIR system that used both Transliteration Mining and Transliteration Generation – in the cross-language Hindi-English and Tamil-English ad hoc retrieval tracks [40].

3 Retrieval System

In this section, we outline the various components of our CLIR system that participated in FIRE2010.

3.1 Monolingual Retrieval System

Our monolingual retrieval system is based on the well-known Language Modeling framework for Information Retrieval [35, 49]. In this framework, queries as well the documents are viewed as probability distributions. The similarity of a query (q) with a document (d) is measured in terms of the likelihood of the query under the document language model (or equivalently, as the Kullback-Leibler divergence of query and document unigram language models).

$$Score(q,d) = \sum_{w} p(w \mid q) \log p(w \mid d) \tag{1}$$

where, w is the term in the lexicon. For a detailed description and discussion of the Language Modeling framework, please see [35, 48, 49]. We smooth the document language model by interpolating with a corpus language model:

$$p_{sm}(w \mid d) = (1-\alpha)\, p_{mle}(w \mid d) + \alpha p(w \mid C) \tag{2}$$

where $0 \leq \alpha \leq 1$.

3.2 Cross-Language Retrieval System

We translate the query distribution in the source language (q_s) to the target language using a probabilistic translation lexicon:

$$p(w_t \mid q_s) = \sum_{w_s} p(w_s \mid q_s)\, p(w_t \mid w_s) \tag{3}$$

where w_s is a source language term and w_t is a target language term. Note that the target language translation (q_t) of the query need not have a surface realization. Nevertheless, the similarity of the translated query (q_t) with a document (d_t) is measured in terms of the Kullback-Leibler divergence of the query and the document language models:

$$\begin{aligned} Score(q_s, d_t) &= \sum_{w_t} p(w_t \mid q_t) \log p(w_t \mid d_t) \\ &= \sum_{w_t, w_s} p(w_s \mid q_s)\, p(w_t \mid w_s) \log p(w_t \mid d_t) \end{aligned} \tag{4}$$

3.3 Handling Out-of-Vocabulary terms

Like any cross-language system that makes use of a translation lexicon, we too faced the problem of OOV query terms. As we observed earlier, many of the OOV terms are names that can be transliterated to the target language and some are imported common nouns. To handle these OOV terms, we used two different techniques, (i) Transliteration Mining and (ii) Transliteration Generation. The details of the above two techniques are given in subsequent sections.

3.4 Transliteration Mining

The mining algorithm issues the translated query minus OOV terms to the target language information retrieval system and mines transliterations of the OOV terms from the top results of the first-pass retrieval. Hence, in the first pass, each query-result pair is viewed as a "comparable" document pair, assuming that the retrieval brought in a reasonably good quality results set based on the translated query without the OOV terms. The algorithm hypothesizes a match between an OOV query term and a document term in the "comparable" document pair and employs a transliteration similarity model (Section 3.4.1) to decide whether the document term is a transliteration of the query term [43, 45]. Transliterations mined in this manner are then used to retranslate the query and issued again, for the final retrieval.

For each topic, we considered top-100 documents returned by the cross-language retrieval system for the purpose of mining. We refer to [43] for details of the mining technique.

3.4.1 Transliteration Similarity Model

Our transliteration similarity model is an extension of W-HMM word alignment model presented in [12], which had been shown to perform well on Transliteration Mining tasks [43, 44, 45]. It is a character-level hidden alignment model that makes use of a

richer local context in both the transition and emission models compared to the classic HMM model [25]. The transition probability depends on both the jump width and the previous source character as in the W-HMM model. The emission probability depends on the current source character and the previous target character unlike the W-HMM model. The transition and emission models are not affected by data scarcity unlike Machine Translation as the character lexicon of a language is typically several orders smaller than its word lexicon. Instead of using any single alignment of characters in the pair (w_S, w_T), we marginalize over all possible alignments:

$$p(t_1^m \mid s_1^n) = \sum_A \prod_{j=1}^{m} p(a_j \mid a_{j-1}, s_{a_{j-1}}) p(t_j \mid s_{a_j}, t_{j-1}) \quad (5)$$

Here, t_j (respectively, s_i) denotes the j^{th} (respectively, i^{th}) character in target word w_T (respectively, source word w_S) and $A \equiv a_1^m$ is the hidden alignment between w_T and w_S where t_j is aligned to s_{a_j}, $j = 1,\ldots,m$. We estimate the parameters of the model by learning over a training set of transliteration pairs. We use the EM algorithm to iteratively estimate the model parameters. The transliteration similarity score of a pair (w_S, w_T) is $\log p(w_T \mid w_S)$ appropriately transformed.

3.5 Transliteration Generation

We experimented with two different techniques of generating transliterations in a target language – Direct and Compositional. In direct transliteration, the OOV terms are directly transliterated using a Machine Transliteration system trained on source-target language parallel names corpora, as detailed in Section 3.5.1.

In compositional transliteration, we use a two-stage system as outlined in Section 3.5.2, for generating transliterations of a given OOV term in source language into the target language, by transitioning through an intermediate language.

3.5.1 Direct Transliteration Generation

Systematic comparison of the various transliteration systems in the NEWS-2009 workshop [22] showed conclusively that orthography based discriminative models like Conditional Random Fields [21] performed best in a language-neutral manner. Hence, we decided to adopt a conditional random fields-based approach using purely orthographic features. In addition, we introduced a word origin detection module to identify specifically Indian origin names. Use of such classifiers allowed us to train a specific CRF-based transliteration engine for Indian origin names, and thus producing better quality transliterations. All other names are transliterated through an engine that is trained on non-Indian origin names. Such a system was used for generating transliterations of OOV terms between source and target languages.

For word origin detection, we manually classified 3000 words from the training set (detailed in Section 4.3.1) into words of Indic origin and Western origin. Two trigram language models were built, one for the Indic origin names and another for Western origin names, to help classify all the name pairs in the training set as Indic or Western names. Manual verification showed that this method about 97% accurate, yielding good quality data that is used for training two distinct CRF-based modules for transliterating Indic and Western names.

Conditional Random Fields are undirected graphical models used for labeling sequential data [21]. Under this model, the conditional probability distribution of the target string given the source string is given by:

$$p(Y / X; \lambda) = \frac{1}{Z(X)} \cdot e^{\sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k (Y_{t-1}, Y_t, X, t)} \tag{6}$$

where,

X = *source string*
Y = *target string*
T = *length of source string*
K = *number of features*
λ_k = *feature weights*
$Z(X)$ = *normalization constant*
f = *1 if the feature is active, 0 otherwise*

We used CRF++, an open source implementation of CRF for training and further transliterating the names (top-n most probable sequences) [7]. We used the alignment model developed by [43] to get the character level alignments for the parallel names in the training corpora. Under this alignment, each character in the source word is aligned to zero or more characters in the corresponding target word. We trained a transliteration engine, based on a rich feature set generated from this character-aligned data; the feature set includes aligned characters in each direction within a small distance (typically, 2) and source and target bigrams and trigrams.

3.5.2 Compositional Transliteration Generation

Compositional transliterations systems combine multiple direct transliterations systems serially to produce transliterations between source language to target language [16, 19]. Specifically, we assume that parallel names corpora are available between the language pair, X and Y, and the language pair, Y and Z; we train two CRF based transliteration systems (as outlined in the earlier section), between the language X and Y, and Y and Z. We provide every name in the test set (in language X) as an input to the X→Y transliteration system, take the top-10 candidate output strings (in language Y) and provide each as an input to the Y→Z system. The output of the Y→Z system for the top-10 candidate strings (in language Z) were merged and re-ranked by their

probability scores. Finally, the top-10 of the merged output was taken as the final output of the compositional transliteration system.

4 Data for Experimental Setup

In this section, we specify all the data used in our experiments.

4.1 FIRE Data

The English document collection provided by FIRE2010 was used in all our runs [9]. The English document collection consists of ~124,000 news articles from "The Telegraph India" from 2004-07. All the English documents were stemmed using the Porter stemmer [36]. We ignored the stop words in the documents as well as the queries. We did not stem the query terms, due to the non-availability of good stemmers in these languages. We plan to experiment with language-neutral stemming techniques for Indian languages in our future work [27].

Totally 50 topics were provided in each of the languages, each topic having a title (T), description (D) and narrative (N), successively expanding the scope of the query. Table 1 shows a typical topic in Hindi, and the TDN components of the topic, for which relevant English documents are to be retrieved from the aforementioned English news corpus.

Table 1. A FIRE2010 Topic in Hindi

Type	Topic
Title	गुटखा मालिकों का अन्डरवर्ल्ड के साथ उलझाव
Description	प्रसिद्ध गुटखा कम्पनी (माणिकचन्द और गोवा)के साथ दाऊद इब्राहिम के सम्बन्ध
Narration	प्रासंगिक प्रलेख में माणिकचन्द गुटखा और गोवा गुटखा मालिकों का अन्डरवर्ल्ड डॅन दाऊद इब्राहिम के साथ सम्बन्ध, से सम्बन्धित सूचनाएँ यहाँ होनी चाहिये। अन्य कम्पनियों के साथ दाऊद इब्राहिम के सम्बन्ध यहाँ अप्रासंगिक हैं।

Table 2. A FIRE2010 Topic in English

Type	Topic
Title	Links between Gutkha manufacturers and the underworld.
Description	Links between the Goa and Manikchand Gutkha manufacturing companies and Dawood Ibrahim.
Narration	A relevant document should contain information about the links between the owners of the Manikchand Gutkha and Goa Gutkha companies and Dawood Ibrahim, the gangster. Information about links between Dawood Ibrahim and other companies is not relevant.

It should be noted that FIRE has also released a set of 50 English (i.e., target language) topics, equivalent to each of the source language topics. The purpose of such topics is to have monolingual (in target language) runs that may provide an upper bound on the retrieval performance of the CLIR runs. A typical English topic (equivalent of the one in the table above) is given in Table 2.

4.2 Bilingual Dictionaries for CLIR

For both the Hindi-English and Tamil-English cross-language retrieval tasks, statistical dictionaries were used; these statistical dictionaries were generated by training statistical word alignment models on Hindi-English parallel corpora (~100 K parallel sentences) and Tamil-English parallel corpora (~50 K parallel sentences) using the GIZA++ tool [32]. We used 5 iterations of IBM Model 1 and 5 iterations of HMM [32]. In the Hindi-English language pair, the training ultimately yielded a statistical dictionary consisting of ~59 K Hindi words and ~63 K English words. In the Tamil-English language pair, the training yielded a statistical dictionary consisting of ~107 K Tamil words and ~45 K English words. We used only top 4 translations for every source word, an empirically determined limit to avoid generation of noisy terms in the query translations.

4.3 Training Data for Transliteration Generation

4.3.1 Training Direct Transliteration Systems

The direct transliteration systems were trained with about 15 K parallel names in Hindi and English and Tamil and English. As reported in [16], the quality of a Machine Transliteration system trained with 15 K corpora is similar to that of a system trained with much larger training data, and hence we used about 15 K parallel names for training a CRF-based transliteration generation system, as described in Section 3.5.1.

4.3.2 Training Compositional Transliteration Systems

The compositional transliteration systems chains two distinct transliteration systems, as described in Section 3.5.2, each trained with about 15 K of appropriate parallel names corpora [16, 19]. In our case, we used Kannada, an Indian language of Dravidian family, as the intermediate language, and trained two separate systems: one between Hindi and Kannada, and another between Kannada and English. Kannada was chosen as the intermediate language as it has a near superset of phoneme inventory of Hindi and English, and hence captures the phonetic essence of the source name to reproduce in the target language. The compositional transliteration technique was used only for Hindi-English cross-language runs. We used top 5 results from transliteration generation for query translation.

4.4 Training Data for Transliteration Mining

We trained Hindi-English and Tamil-English transliteration similarity models on 16 K parallel single word names in Hindi-English and Tamil-English language pairs respectively, and ran 15 iterations of Expectation Maximization training.

5 Results and Analysis

In this section, we present our experimental results and also an analysis of the results.

5.1 Metrics for Measuring Performance

We use Mean Average Precision (MAP) as the measure for the topic set, Average Precision (AP) for individual topics and Precision at top-10 (P@10).

5.2 An Illustrative Analysis of the Impact of different Techniques

In this section we show one example Hindi topic and discuss how various approaches affected the retrieval performance. Note that the results of our CLIR experiments, as presented in Tables 7 & 8, indicate that the approaches generally help in improving the CLIR performance.

Consider the topic number 112 in Hindi shown in table 3. The OOV terms in the Hindi topic are shown in bold, and those OOV terms that are transliteratable are underlined; hence Transliteration Mining and Transliteration Generation can potentially help the retrieval performance, by providing equivalents in the target language for these words.

Table 3. Hindi Topic No. 112

Type	Topic
Title	गुटखा मालिकों का **अन्डरवर्ल्ड** के साथ **उलझाव**
Description	प्रसिद्ध गुटखा कम्पनी (**माणिकचन्द** और गोवा)के साथ दाऊद इब्राहिम के सम्बन्ध
Narration	प्रासंगिक **प्रलेख** में **माणिकचन्द** गुटखा और गोवा गुटखा मालिकों का **अन्डरवर्ल्ड डॅन** दाऊद इब्राहिम के साथ सम्बन्ध, से सम्बन्धित सूचनाएँ यहाँ होनी चाहिये। अन्य कम्पनियों के साथ दाऊद इब्राहिम के सम्बन्ध यहाँ अप्रासंगिक हैं।

The Hindi topic has five OOV terms (namely, '**अन्डरवर्ल्ड**', '**उलझाव**', '**माणिकचन्द**', '**प्रलेख**' and '**डॅन**'), out of which two of them ('**अन्डरवर्ल्ड**', a common noun imported to Hindi from English by transliterating the word '*underworld*' and '**माणिकचन्द**', a proper noun '*manickchand*') are terms that may occur in the transliterated form in the target language document collection. Transliteration Mining was able to identify the valid English equivalents for both of these two terms from the top results of the first pass

retrieval (‘*underworld’* for ‘**अन्डरवर्ल्ड**’ and ‘*manikchand’* for ‘**माणिकचन्द**’), whereas generation produced only one English equivalent (‘*manikchand’* for ‘**माणिकचन्द**’) correctly.

Table 4 shows the generated and mined equivalents for the OOV terms of the Hindi topic 112 (shown in Table 3). The OOV terms that are underlined are those that occur in their transliterated form in the target corpus, and their valid English equivalents by Transliteration Generation or Transliteration Mining are shown in bold.

Table 4. OOV terms in Hindi topic 112, and their top-5 generated transliterations (Direct and Compositional) and mined English equivalents

OOV in Topic	Generation (Direct)	Generation (Compositional)	Mining
अन्डरवर्ल्ड	andrverld, anderverld, andrverd, anderverd, andrvorld	anderverld, onderverld, enderverld, inderverld, xanderverld	**underworlds, underworld**
उलझाव	uljhav, ulwav, ulzav, ulqav, ulav	ullav, ulwav, uljav, ullau, ulzav	-
माणिकचन्द	manikchanda, **manikchand**, maanikchanda, maanikchand, manikanda	**manikchand**, manikchandh, manikchande, manikchandr, maanikchand	**manikchand, manickchand**
प्रलेख	pralekh, pralekha, prlekh, pralaekh, pralakh	pralekh, pralekha, pralekh, pralekh, pralek	palekar
डॅन	dann, dan, den,denn, danne	don, den, dan, dn, dian	

Note each of the OOV terms were handled slightly differently by the two competing transliteration techniques: The English equivalent for the Hindi OOV term ‘**अन्डरवर्ल्ड**’ (code-mixed Hindi word for the English word, ‘*underworld’*) was not generated correctly by Transliteration Generation techniques, but its two equivalents were mined correctly by Transliteration Mining. The English equivalent for the Hindi OOV term ‘**माणिकचन्द**’ (a transliteration for the proper noun, ‘*manikchand’*) was generated correctly by both the generation techniques, but Transliteration Mining was able to mine multiple variants of the name from the target corpus.

The two Hindi OOV terms, namely ‘**उलझाव**’ and ‘**प्रलेख**’, are not transliteratable (that is, they are proper Hindi words that were not translated by our query translation engine, due to the lack of coverage, and its transliterated form is unlikely to be in the target corpus). For the term ‘**उलझाव**’, the Transliteration Generation techniques produced some English strings (which clearly will not be found in the target English corpora), and Transliteration Mining also could not mine any equivalents. However, for the term ‘**प्रलेख**’, Transliteration Mining did find a near phonetic equivalent ‘*palekar’* which occurs in the target corpus but semantically unrelated to the source word ‘**प्रलेख**’; The generated equivalents for the OOV term ‘**प्रलेख**’ may have had relatively small negative effect on retrieval performance, as they are noisy terms.

We observe that both Transliteration Generation and Transliteration Mining introduced some noise words as well along with the correct transliterations. However, it is

important to note that the positive effect of handling OOV terms correctly outweighs the negative effect of noisy terms which are in general uncorrelated with the query terms. As can be observed in Figure 1, the overall retrieval performance of topic 112 is significantly improved when either of the techniques is employed.

5.3 Performance of Various Configurations of Integrated CLIR System

As shown in Table 1, each of the 50 topics in Hindi and Tamil has a title (T), description (D) and narrative (N), successively expanding the scope of the query. We ran our experiments taking progressively each of (title), (title and description), and (title, description and narrative), calibrating the cross-language retrieval performance at each stage, to explore whether expanding the query adds useful information for retrieval or just noise. Table 5 shows the notation used in our description of various configurations to interpret the results presented in Tables 6, 7 and 8.

Table 5. Notations used

T	Title
TD	Title and Description
TDN	Title, Description and Narration
M	Transliteration Mining
G_D	Transliteration Generation – Direct
G_T	Transliteration Generation – Compositional

Tables 6, 7 and 8 show the MAP and precision@10 of our monolingual as well as cross-language official runs submitted to FIRE 2010 shared task. The format of the run ids in the results table is 'Source-Target-Query-Technique', where 'Query indicates the type of the query, and is one of {T, TD, TDN} and 'Technique' indicates the technique and from the set {M, G_D, G_T, M+G_D, M+G_T}. The '+' refers to the combination of more than one approach. The symbols double star (**) and single star (*) indicate statistically significant differences with 95% and 90% confidence respectively according to the paired t-test over the baseline. The best results achieved are highlighted in bold.

5.4 Monolingual English Retrieval

We submitted 3 official runs for the English monolingual track, as shown in the Table 6. For these runs, the English topics provided by the FIRE 2010 organizers were used.

Table 6. English Monolingual Retrieval Performance (Official submissions for the FIRE 2010 Shared Task)

Run	MAP	P@10
English-English-T	0.3653	0.344
English-English-TD	0.4571	0.406
English-English-TDN	**0.5133**	**0.462**

With the full topic (TDN), our monolingual IR system achieved a MAP score of 0.5133. Generally this performance is thought to be the upper bound for cross-language performance, presented in Tables 7 and 8.

5.5 Hindi-English Cross-Language Retrieval

We submitted totally 18 official run results on Hindi-English cross-language track, as shown in Table 7.

Table 7. Hindi-English Cross-Language Retrieval Performance (Official submissions for the FIRE 2010 Shared Task)

Run	MAP	P@10
Hindi-English-T	0.2931	0.26
Hindi-English-T[GD]	0.3168**	0.282
Hindi-English-T[GT]	0.3140**	0.276
Hindi-English-T[M]	**0.3390****	**0.304**
Hindi-English-T[M+GD]	0.3388**	0.302
Hindi-English-T[M+GT]	0.3388**	0.302
Hindi-English-TD	0.4042	0.356
Hindi-English-TD[GD]	0.4336**	0.386
Hindi-English-TD[GT]	0.4369**	0.382
Hindi-English-TD[M]	0.4376**	**0.388**
Hindi-English-TD[M+GD]	**0.4378****	0.386
Hindi-English-TD[M+GT]	0.4375**	0.386
Hindi-English-TDN	0.4748	0.424
Hindi-English-TDN[GD]	0.4942**	0.434
Hindi-English-TDN[GT]	0.4970**	0.438
Hindi-English-TDN[M]	**0.4977****	0.442
Hindi-English-TDN[M+GD]	0.4971**	**0.444**
Hindi-English-TDN[M+GT]	0.4965**	**0.444**

The first run under each of the 'T', 'TD' and 'TDN' sections in Table 7 present the results of the runs of our baseline CLIR system without handling the OOV terms, and hence provide a baseline for measuring the improvement in retrieval performance due to Transliteration Generation or Transliteration Mining, provided subsequently. From the results, we observe that the usage of all of the components of the topic, namely T, D and N, produced the best retrieval performance. The basic Hindi-English cross-language run 'Hindi-English-TDN' (without Transliteration Generation or Transliteration Mining), achieved the MAP score 0.4748, and our best cross-language run 'Hindi-English-TDN[M]' with Transliteration Mining achieved a MAP score of 0.4977. We observe similar trends in the other runs that use only the title, or title and description sections of the topics. It should be noted that our basic TDN run achieves 92% of the monolingual performance, and the cross-language TDN run enhanced with Transliteration Mining, 97% of the monolingual retrieval performance.

In practice, user queries are more likely to be the topics restricted to 'T'. We note that for the 'T' runs, Transliteration Mining gives superior retrieval performance than Transliteration Generation.

5.6 Tamil-English Cross-Language Retrieval

We submitted totally 12 official Tamil-English cross-language runs, as shown in Table 8. As with the Hindi-English runs, the first run under each of the 'T', 'TD' and 'TDN' sections in Table 8 present the results of the runs without handling the OOV terms, and is a baseline.

Table 8. Tamil-English Cross-Language Retrieval Performance (Official submissions for the FIRE 2010 Shared Task)

Run	MAP	P@10
Tamil-English-T	0.2710	0.258
Tamil-English-T[GD]	**0.2891***	**0.268**
Tamil-English-T[M]	0.2815**	0.258
Tamil-English-T[M+GD]	0.2816*	0.268
Tamil-English-TD	0.3439	0.346
Tamil-English-TD[GD]	0.3548*	0.35
Tamil-English-TD[M]	**0.3621****	0.346
Tamil-English-TD[M+GD]	0.3617**	**0.362**
Tamil-English-TDN	0.3912	0.368
Tamil-English-TDN[GD]	0.4068**	0.378
Tamil-English-TDN[M]	**0.4145****	0.368
Tamil-English-TDN[M+GD]	0.4139**	**0.394**

From the results presented in Table 8, we observe that the usage of all of the components of the topic, namely T, D and N, produced the best retrieval performance. The basic Tamil-English cross-language run 'Tamil-English-TDN' achieved the MAP score 0.3912, and our best cross-language run 'Tamil-English-TDN[M]' with Transliteration Mining achieved a MAP score of 0.4145. We observe, in general, similar trends in the other runs that use only the title, or title and description sections of the topics. While the cross-language performance of Tamil-English achieves ~81% of our monolingual English retrieval performance, we observe that this is not as high as the Hindi-English retrieval, perhaps due to the highly agglutinative nature of Tamil as we explain in section 5.7.2.

Given that Transliteration Mining performed generally better than Transliteration Generation, we take a deeper look at Transliteration Mining in the next section.

5.7 Mining OOV Terms and Its Effect on CLIR Performance

In this section, we analyze the volume of the OOV terms in FIRE topics, and to what extent they are handled by Transliteration Mining, which clearly emerged as the better

technique for addressing the OOV problem. Also, we show the effect of handling the OOVs on the cross-language retrieval performance, for both the Hindi-English and Tamil-English CLIR runs.

5.7.1 Profile of OOV Terms in Hindi-English Cross-Lingual Task

Table 9 gives the profile of OOV terms. We see that there are a large number of OOV terms in all three query configurations and a good number of queries are affected. We also see that majority of the OOV terms are transliteratable, i.e. they are either names or imported common nouns. Further, Transliteration Mining is able to mine at least one correct transliteration for most of the transliteratable OOV terms.

Table 9. Profile of OOV terms in Hindi-English CLIR

Type	All OOV terms		Transliteratable OOV terms		Transliteratable OOV terms handled correctly	
	No. of Terms	No. of Topics	No. of Terms	No. of Topics	No. of Terms	No. of Topics
T	15	13	11	11	11	11
TD	35	24	23	17	21	15
TDN	73	50	31	19	24	17

Table 10. Performance Improvements in Hindi-English CLIR

Type	MAP as % of monolingual	MAP improvement in % over the baseline	
		All topics	Only topics with OOV
T	92.8	15.7	60.7
TD	95.7	8.3	14.1
TDN	97.0	4.8	4.8

Table 10 shows the performance improvements that Transliteration Mining brings relative to the baseline. The percentage of MAP score improvement over the monolingual performance is shown in column 2. The percentage improvements over the baseline CLIR system are measured in two contexts: with all topics in the FIRE cross-slingual task, and with only those topics that have at least one OOV in them. These two are shown in columns 3 and 4 respectively.

In the TD configuration, the 50 Hindi topics contained 35 distinct OOV terms that appeared in 24 topics. Out of these 35 terms, 23 were proper or common nouns and appeared in 17 topics. Transliteration Mining produced at least one transliteration equivalent for 21 of these OOV terms (91.3%), which appeared in totally15 topics. As shown in Table 7 & 10, handling these OOV terms resulted in MAP score improvement of 8.3% when considering all 50 topics and 14.1% when considering only those 24 topics that have at least one OOV term, over the baseline CLIR system. Similarly,

in the T configuration, Transliteration Mining produced at least one transliteration equivalent for all 11 transliteratable terms (that is, 100%) that appeared in 11 topics. As a consequence, the MAP improved by 15.7% when considering all 50 topics and by 60.7% when considering only those 13 topics that had at least one OOV term. This highlights the significance of our method in practice where most queries are short.

5.7.2 Profile of OOV Terms in Tamil-English Cross-Lingual Task

Table 11 gives the profile of OOV query terms and Table 12 shows the performance improvements for our Tamil-English CLIR system.

Table 11. Profile of OOV terms in Tamil-English CLIR

Type	All OOV terms		Transliteratable OOV terms		Transliteratable OOV terms handled correctly	
	No. of Terms	No. of Topics	No. of Terms	No. of Topics	No. of Terms	No. of Topics
T	24	19	13	13	5	5
TD	58	33	29	21	15	12
TDN	129	45	47	28	24	17

As shown in Table 11, in the TDN configuration, the 50 Tamil topics contained 129 unique OOV terms that appeared in 45 topics, out of which 47 (in 28 topics) were proper or common nouns. Transliteration Mining produced at least one transliteration equivalent for 24 of these OOV terms (51.06%) that appeared in 17 topics. As shown in Table 8 and 12, handling these OOV's resulted in MAP score improvement by 6% when all 50 topics are considered and 6.5% when considering only the 45 topics that had at least one OOV term.

As we observed in Hindi-English task, short queries benefit most (13.6%) from Transliteration Mining in Tamil-English task also. However, this is relatively low when compare to Hindi-English task (60.7%). This is due to the reason that in Hindi-English task 100% of transliteratable OOV terms were handled correctly whereas it is only 38.46% in Tamil-English task. Also, note that the performance of our Tamil-English CLIR system with Transliteration Mining is ~81% of the monolingual performance in TDN configuration.

Table 12. Performance Improvements in Tamil-English CLIR

Type	MAP as % of monolingual	MAP improvement in % over CLIR baseline	
		All topics	Only topics with OOV
T	77.1	3.9	13.6
TD	79.2	5.3	8.7
TDN	80.8	6.0	6.5

Tamil poses specific challenges compared to Hindi: First, the transliteratable terms mentioned in the fourth column of the Table 11 excludes some terms whose equivalents are multiword expression in English; mining such multiword transliteration equivalents is beyond the scope of our work and hence they were not handled. Second, 26 out of the 47 terms that are transliteratable were inflected or agglutinated. While Transliteration Mining algorithm could mine some of them, many could not be at the threshold used by the transliteration similarity model. By relaxing the threshold we could mine more such terms, but that introduced many more noise terms, affecting the overall retrieval performance. We believe that the use of a good stemmer for inflectional languages like Tamil may help our Transliteration Mining algorithm and thereby the cross-language retrieval performance.

5.8 Mining OOV Terms and Its Effect on Individual Topic Performance

In this section, we discuss the effect of Transliteration Mining on the retrieval performance for individual topics of the FIRE 2010 shared task. Figure 1 shows the difference in the Average Precision – topic-wise – between the baseline CLIR system and the one that employs Transliteration Mining. We see that many topics benefitted from Transliteration Mining; for example, in the Hindi-English language pair, in T configuration, 8 topics benefitted substantially (with improvement of ≥ 0.2 in AP) whereas only 2 topics were negatively impacted (a drop of ≥ 0.2 in AP). Similar trends could be seen for all configurations, in both Hindi-English and Tamil-English language pairs.

In order to observe the impact of Transliteration Mining on individual topics, let us consider some topics in the Hindi-English and Tamil-English test collections (in TDN configuration), specifically those topics which are affected most. Such topics are shown in the Table 13. The OOV terms of these topics are shown in bold, and those OOV terms that are transliteratable are underlined. Subsequently, we show how the Transliteration Mining handled each of such OOV terms, and present the resulting change on the retrieval performance over the baseline CLIR. Table 14 shows the OOV terms of the above topics and the outputs of Transliteration Mining. The valid English equivalents mined are shown in bold.

In Hindi-English task, the AP of topic 112 was increased by +0.36 in the TDN setup. This topic has five Hindi OOV terms, out of which two of them ('**अन्डरवर्ल्ड**' and '**माणिकचन्द**') are transliteratable. Transliteration Mining was able to produce all the valid English equivalents from the top results, but it also mined a noisy term for non-transliteratable OOV term '**प्रलेख**'. Still, the overall effect on the retrieval performance was positive. On the other hand, topic 123 has two OOV terms, namely '**फलस्तीनी**' and '**प्रलेख**', out of which only one ('**फलस्तीनी**') is transliteratable; Transliteration Mining was able to produce two equivalents. However, as in topic 112, Transliteration Mining produced a noisy term for the non transliteratable OOV '**प्रलेख**'; the overall effect on the retrieval performance was negative (-0.11). When we investigated this topic further, we found that the relevant documents had the

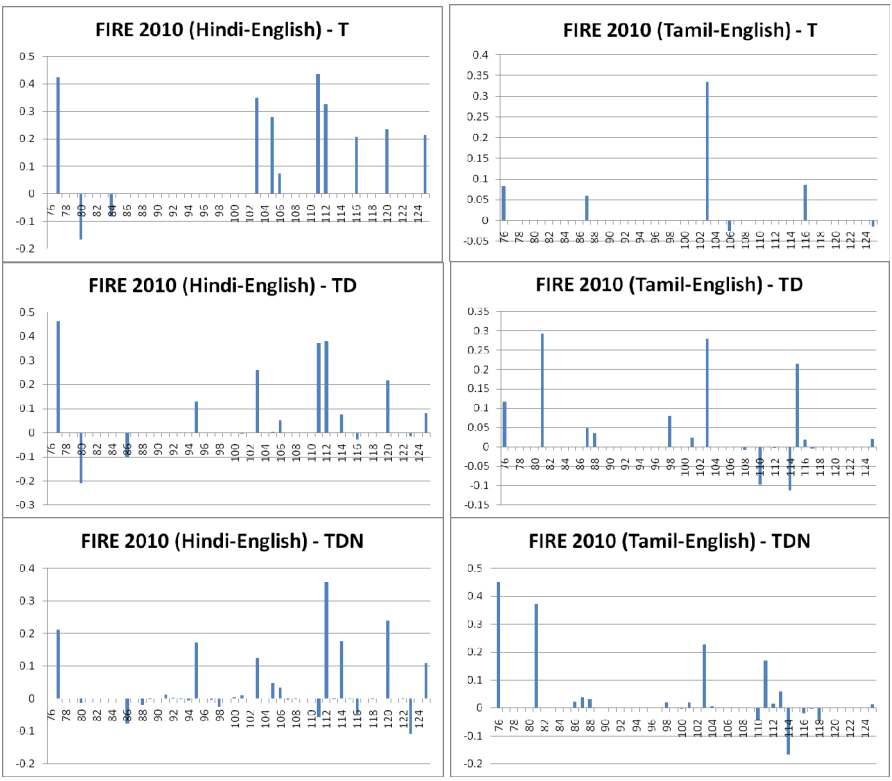


Fig. 1. Differences in Average Precision between the baseline and CLIR with Transliteration Mining

term '*palestinian'* which was being stemmed as '*palestinian'* where as the mined transliterations, '*palestines'* and '*palestine'*, were both stemmed as '*palestin'*. Had the stemmer produced the same stems for '*palestinian'*, '*palestines'* and '*palestine'*, the retrieval performance would have gone up for this topic.

In the Tamil-English task, the topic 76 had 7 OOV terms, out of which only 3 are transliteratable. Transliteration Mining produced the equivalent for one of them (which occurred 4 times in the topic) and as a consequence, the AP increased by 0.45. Topic 114 has 6 OOV terms, out of which only two are transliteratable. Transliteration Mining produced a valid transliteration '*fernandess'* for the transliteratable OOV term '**பெர்ணான்டஸ்**'; but this mined term is different from that in the equivalent English topic, which is '*fernandez'* and AP went down by-0.17).

5.9 Hybrid Approach: Mining with Transliteration Generation

In addition to generation and mining of transliteration equivalents we conducted a few experiments that employed a combination of both techniques. In this technique, we

first mine the transliteration equivalents using Transliteration Mining and employ

Table 13. Topics affected most by Transliteration Mining

Topic ID	Topic
Hindi 112	(T) गुटखा मालिकों का **अन्डरवर्ल्ड** के साथ **उलझाव** (D) प्रसिद्ध गुटखा कम्पनी (**माणिकचन्द** और गोवा)के साथ दाऊद इब्राहिम के सम्बन्ध (N) प्रासंगिक **प्रलेख** में **माणिकचन्द** गुटखा और गोवा गुटखा मालिकों का **अन्डरवर्ल्ड डॅन** दाऊद इब्राहिम के साथ सम्बन्ध, से सम्बन्धित सूचनाएँ यहाँ होनी चाहिये। अन्य कम्पनियों के साथ दाऊद इब्राहिम के सम्बन्ध यहाँ अप्रासंगिक हैं।
Hindi 123	(T) यासर अराफात की मृत्यु (D) **फलस्तीनी** नेता यासर अराफात की मृत्यु (N) प्रासंगिक **प्रलेख** में फलस्तीनी नेता यासर अराफात की मृत्यु से सम्बन्धित सूचनाएँ होनी चाहिये। फलस्तीनी नेता की मृत्यु से फैली राजनीतिक अशांति से सम्बन्धित सूचनाएँ यहाँ अप्रासंगिक हैं
Tamil 76	(T) குஜ்ஜார், **மீனாஸ்** இடையே மோதல் (D) பழங்குடியினர் பட்டியலில் **குஜ்ஜாரை இணைத்ததற்கு மீனாஸ்** தலைவர்களின் எதிர்ப்பு. (N) பழங்குடியினர் பட்டியலில் **இணைக்க** வேண்டும் என **குஜ்ஜார்களின்** கிளர்ச்சி. **மீனாஸ்** தலைவர்கள் இதற்கு கடும் எதிர்ப்பு. **மீனாஸ்** தலைவர்கள் எதிர்ப்பதற்கான பின்னணி காரணங்கள் என்ன? இவ்விரு **பிரிவினரிடையேயான** போராட்டத்திற்கு **மூலக்காரணம்** பற்றிய செய்திகள் இந்த ஆவணத்தில் இடம்பெறலாம்.
Tamil 114	(T) பாதுகாப்புத் துறையின் ஆயுத ஊழல் விசாரணை (D) ஜார்ஜ் **பெர்ணான்டர்ஸ்**, டெனில் இடையேயான ஆயுத ஒப்பந்தம், இந்த **முறைக்கேடிற்கு** பிரணாப் முகர்ஜியின் **விசாரணைத்** தேவை என்ற **கோரிக்கைப்** பற்றிய செய்திகள் இந்த ஆவணத்தில் இடம்பெறலாம். (N) முன்னாள் அமைச்சர் ஜார்ஜ் **பெர்ணான்டர்ஸ்**, தென்னாப்பிரிக்க நிறுவனமான **டெனிலுடனான** ஆயுத ஒப்பந்தம். இந்த ஒப்பந்தம் குறித்த பாதுகாப்பு அமைச்சர் பிரணாப் முகர்ஜியின் **விசாரணைப்** பற்றிய தகவல்கள் இந்த ஆவணத்தில் இடம்பெறலாம்.

Table 14. Topics with their OOV terms and the impact on retrieval performance

Topic ID	Hindi/Tamil OOV	Mined English words	Change in AP
Hindi 112	**अन्डरवर्ल्ड**	**underworlds, underworld**	+0.36
	उलझाव	**-**	
	माणिकचन्द	**manikchand, manick-chand**	
	प्रलेख	palekar	
	डॅन	**-**	
Hindi 123	**फलस्तीनी**	**palestines, palestine**	-0.11
	प्रलेख	palekar	

Table 14. *(Continued)*

Tamil 76	மீனாஸ்	menace, **meenas**	+0.45
	குஜ்ஜாரை	-	
	இணைத்ததற்கு	-	
	இணைக்க	-	
	குஜ்ஜார்களின்	-	
	பிரிவினரிடையேயான	-	
	மூலக்காரணம்	-	
Tamil 114	பெர்ணான்டர்ஸ்	**fernandess**	-0.17
	முறைக்கேடிற்கு	-	
	விசாரணைத்	-	
	கோரிக்கைப்	-	
	டெனிலுடனான	-	
	விசாரணைப்	-	

Transliteration Generation for those OOV terms for which Transliteration Mining produced no results. In Table 7 and 8, the run ids with '$M+G_D$' and '$M+G_T$', refer that they are combination of mining and generation. We observe that the hybrid approach did not produce significantly better results than Transliteration Mining in both Hindi-English and Tamil-English.

5.10 Comparison of Transliteration Mining against Oracles

Finally, in this section, we compare the performance of our best performing configuration –Transliteration Mining – against two oracular systems, which can identify the right transliterations from equivalent English topic and relevant documents from the target collection, respectively, and thus can indicate an upper bound for the cross-language retrieval performance. We devised two oracular CLIR systems, as described below. The first oracular system identified the correct transliterations for OOV terms from the equivalent English topic (which was provided as a part of the FIRE 2010 test collection). The second oracular system identified the correct transliteration equivalents for the OOV terms from the relevant documents for that topic (provided as a part of the FIRE 2010 test collection. Note that in both the above oracles, we used the same statistical dictionaries used in our previous experiments. The MAP figures for the runs are summarized in Table 15. The best performances are highlighted in bold.

The results presented in Table 15 indicate that the performance Transliteration Mining is nearly equivalent to that of oracular experiments in Hindi-English and fairly close to that for Tamil-English. One curious result needs a bit of explanation: Transliteration Mining outperforms the best oracle in 'Hindi-English-T' setup. On further examination, we found that Transliteration Mining was able to identify two valid equivalents for the transliteratable OOV term '**हिजबुल्लाह**' (a transliteration for the

Table 15. Comparison of MAP of Transliteration Mining and two oracular CLIR systems

Collection	Oracle-1	Oracle-2	Trans-litera-tion Mining	As % of Best Oracle
FIRE 2010 Hindi-English-T	0.3385	0.3374	**0.3390**	100.15
FIRE 2010 Hindi-English-TD	**0.4406**	0.4349	0.4376	99.32
FIRE 2010 Hindi-English-TDN	**0.5026**	0.4942	0.4977	99.03
FIRE 2010 Tamil-English-T	0.3136	**0.314**	0.2815	86.65
FIRE 2010 Tamil-English-TD	**0.3962**	0.3955	0.3621	91.39
FIRE 2010 Tamil-English-TDN	**0.4562**	0.4507	0.4145	90.86

proper noun, '*hezbollah'* or '*hizbolla'*) in topic number 77, specifically, '*hizbollahs'* and '*hizbollah'* from the first-pass retrieval; in comparison to the corresponding English topic for the same topic, has only the word '*hezbollah'*. However, we found that a document in the relevant document set contains '*hizbollah'* but not '*hezbollah'*. Thus, Transliteration Mining could outperform the oracle!

6 Conclusion

In this paper, we underlined the need for handling proper and common nouns for improving the retrieval performance of cross-language information retrieval systems. We proposed and outlined two techniques namely Transliteration Mining and Transliteration Generation for handling out of vocabulary (OOV) words to enhance a state of the art baseline CLIR system. Such an enhanced system was used by our team in Microsoft Research India in our participation in the FIRE 2010 shared task [9], for cross-language Hindi-English and Tamil-English retrieval tasks. We presented the performance of our system under various topic configurations, specifically for English monolingual task and two cross-language tasks, Hindi-English and Tamil-English on the standard FIRE 2010 dataset. We showed that each of the two techniques improved retrieval performance, but consistently more so by Transliteration Mining. We also showed specific sample topics to explain and highlight how each technique affects the retrieval performance – positively or negatively, but our experimental evaluation indicate that the overall effect is significantly positive. Finally, we showed that Transliteration Mining performs almost as well as two oracular systems that can identify the transliterations from the equivalent English topic or from the relevant documents from the target collection.

References

1. AbdulJaleel, N., Larkey, L.S.: Statistical transliteration for English-Arabic cross language information retrieval. In: CIKM (2003)

2. Al-Onaizan, Y., Knight, K.: Machine transliteration of names in Arabic text. In: ACL Workshop on Computational Approaches to Semitic Languages (2002)
3. Al-Onaizan, Y., Knight, K.: Translating named entities using monolingual and bilingual resources. In: 40th Annual Meeting of ACL (2002)
4. Ballesteros, L., Croft, B.: Dictionary Methods for Cross-Lingual Information Retrieval. In: Thoma, H., Wagner, R.R. (eds.) DEXA 1996. LNCS, vol. 1134, pp. 791–801. Springer, Heidelberg (1996)
5. Cao, G., Gao, J., Nie, J.Y.: A system to mine large-scale bilingual dictionaries from monolingual Web pages. In: Proceedings of the 11th MT Summit (2007)
6. Chinnakotla, M.K., Vachhani, V., Gupta, S., Raman, K., Bhattacharyya, P.: IITB CFILT @ FIRE 2010: Discriminative Approach to IR. Working Notes for the Forum for Information Retrieval Evaluation (FIRE) Workshop (2010)
7. CRF++, `http://crfpp.sourceforge.net`
8. Demner-Fushman, D., Oard, D.W.: The effect of bilingual term list size on dictionary based cross-language information retrieval. In: 36th Hawaii International Conference on System Sciences (2002)
9. Forum for Information Retrieval Evaluation, `http://www.isical.ac.in/~fire`
10. Fung, P.: A pattern matching method for finding noun and proper noun translations from noisy parallel corpora. In: ACL (1995)
11. Fung, P.: Compiling bilingual lexicon entries from a non-parallel English-Chinese corpus. In: 3rd Work-shop on Very Large Corpora (1995)
12. He, X.: Using word dependent transition models in HMM based word alignment for statistical machine translation. In: 2nd ACL Workshop on Statistical Machine Translation (2007)
13. Jagarlamudi, J., Kumaran, A.: Cross-Lingual Information Retrieval System for Indian Languages. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, pp. 80–87. Springer, Heidelberg (2008)
14. Järvelin, A., Järvelin, A.: Comparison of *s*-gram Proximity Measures in Out-of-Vocabulary Word Translation. In: Amir, A., Turpin, A., Moffat, A. (eds.) SPIRE 2008. LNCS, vol. 5280, pp. 75–86. Springer, Heidelberg (2008)
15. Joshi, T., Joy, J., Kellner, T., Khurana, U., Kumaran, A., Sengar, V.S.: Crosslingual location search. In: SIGIR, pp. 211–218 (2008)
16. Khapra, M., Kumaran, A., Bhattacharyya, P.: Everybody loves a rich cousin: An empirical study of transliteration through bridge languages. In: NAACL (2010)
17. Knight, K., Graehl, J.: Machine Transliteration. Computational Linguistics (1998)
18. Kraiij, W., Nie, J.-Y., Simard, M.: Emebdding Web-based Statistical Translation Models in Cross-Language Information Retrieval. Computational Linguistics (2003)
19. Kumaran, A., Khapra, M., Bhattacharyya, P.: Compositional Machine Transliteration. ACM Transactions on Asian Language Information Processing, TALIP (2010)
20. Kumaran, A., Khapra, M., Li, H.: Report of NEWS 2010 Transliteration Mining Shared Task. In: 2010 Named Entities Workshop, ACL (2010)
21. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: International Conference on Machine Learning (2001)
22. Li, H., Kumaran, A., Pervouchine, V., Zhang, M.: Report of NEWS 2009 Machine Transliteration Shared Task. In: 2009 Named Entities Workshop: Shared Task on Transliteration (2009)

23. Li, H., Sim, K.C., Kuo, J., Dong, M.: Semantic Transliteration of Personal Names. In: ACL (2007)
24. Majumder, P., Mitra, M., Pal, D., Bandyopadhyay, A., Maiti, S., Mitra, S., Sen, A., Pal, S.: Text collections for FIRE. In: SIGIR (2008)
25. Mandl, T., Womser-Hacker, C.: How do named entities contribute to retrieval effectiveness? In: Cross Language Evaluation Forum Campaign (2004)
26. Mandl, T., Womser-Hacker, C.: The Effect of named entities on effectiveness in crosslanguage information retrieval evaluation. In: ACM Symposium on Applied Computing (2005)
27. Mayfield, J., McNamee, P.: Single n-gram stemming. In: SIGIR (2003)
28. Microsoft Research India, http://research.microsoft.com/en-us/labs/india/
29. Munteanu, D., Marcu, D.: Extracting parallel sub-sentential fragments from non-parallel corpora. In: ACL (2006)
30. Nardi, A., Peters, C.: Working Notes for the CLEF 2007 Workshop (2007)
31. NTCIR, http://research.nii.ac.jp/ntcir
32. Och, F., Ney, H.: A systematic comparison of various statistical alignment models. Computation Linguistics (2002)
33. Peters, C.: Working Notes for the CLEF 2006 Workshop (2006)
34. Pirkola, A., Toivonen, J., Keskustalo, H., Järvelin, K.: Frequency-based identification of correct translation equivalents (FITE) obtained through transformation rules. ACM Transactions on Information Systems (TOIS) 26(1), article 2 (2007)
35. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: SIGIR (1998)
36. Porter, M.F.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
37. Quirk, C., Udupa, R., Menezes, A.: Generative models of noisy translations with applications to parallel fragments extraction. In: 11th MT Summit (2007)
38. Rao, P.R.K., Devi, S.L.: AU-KBC FIRE2010 Submission - Cross Lingual Information Retrieval Track: Tamil- English. In: Working Notes for the Forum for Information Retrieval Evaluation (FIRE) Workshop (2010)
39. Rapp, R.: Automatic identification of word translations from unrelated English and German corpora. In: ACL (1999)
40. Saravanan, K., Udupa, R., Kumaran, A.: Crosslingual Information Retrieval System Enhanced with Transliteration Generation and Mining. In: Working notes for Forum for Information Retrieval Evaluation (FIRE) Workshop (2010)
41. The Cross-Language Evaluation Forum (CLEF), http://clef-campaign.org
42. Udupa, R., Jagarlamudi, J., Saravanan, K.: Microsoft Research India at FIRE 2008: Hindi-English Cross-Language Information Retrieval. In: Working notes for Forum for Information Retrieval Evaluation (FIRE) Workshop (2008)
43. Udupa, R., Saravanan, K., Bakalov, A., Bhole, A.: They Are Out There, If You Know Where to Look: Mining Transliterations of OOV Query Terms for Cross-Language Information Retrieval. In: ECIR (2009)
44. Udupa, R., Saravanan, K., Kumaran, A.: Mining Named Entity Transliteration Equivalents from Comparable Corpora. In: CIKM (2008)
45. Udupa, R., Saravanan, K., Kumaran, A., Jagarlamudi, J.: MINT: A Method for Effective and Scalable Mining of Named Entity Transliterations from Large Comparable Corpora. In: EACL (2009)

46. Virga, P., Khudanpur, S.: Transliteration of proper names in cross-lingual information retrieval. In: ACL Workshop on Multilingual and Mixed Language Named Entity Recognition (2003)
47. Xu, J., Weischedel, R.: Empirical studies on the impact of lexical resources on CLIR performance. Information Processing and Management (2005)
48. Zhai, C., Lafferty, J.: Two Stage Language Models for Information Retrieval. In: SIGIR (2002)
49. Zhai, C., Lafferty, J.: A study of smoothing algorithms for language models applied to information retrieval. ACM Transactions on Information Systems 22(2), 179–214 (2004)
50. Zhang, M., Li, H., Kumaran, A., Liu, M.: Report of NEWS 2011 Machine Transliteration Shared Task. In: 2011 Named Entities Workshop, IJCNLP (2011)
51. Zobel, J., Dart, P.: Phonetic string matching: lessons from information retrieval. In: SIGIR (1996)

Information Retrieval with Hindi, Bengali, and Marathi Languages: Evaluation and Analysis

Jacques Savoy, Ljiljana Dolamic, and Mitra Akasereh

Computer Science Department,
University of Neuchatel,
Rue Emile Argand 11, 2000 Neuchatel, Switzerland
{Jacques.Savoy,Ljiljana.Dolamic,Mitra.Akasereh}@unine.ch

Abstract. Our first objective in participating in FIRE evaluation campaigns is to analyze the retrieval effectiveness of various indexing and search strategies when dealing with corpora written in Hindi, Bengali and Marathi languages. As a second goal, we have developed new and more aggressive stemming strategies for both Marathi and Hindi languages during this second campaign. We have compared their retrieval effectiveness with both light stemming strategy and *n*-gram language-independent approach. As another language-independent indexing strategy, we have evaluated the trunc-*n* method in which the indexing term is formed by considering only the first *n* letters of each word. To evaluate these solutions we have used various IR models including models derived from Divergence from Randomness (DFR), Language Model (LM) as well as Okapi, or the classical *tf idf* vector-processing approach.

For the three studied languages, our experiments tend to show that IR models derived from Divergence from Randomness (DFR) paradigm tend to produce the best overall results. For these languages, our various experiments demonstrate also that either an aggressive stemming procedure or the trunc-*n* indexing approach produces better retrieval effectiveness when compared to other word-based or *n*-gram language-independent approaches. Applying the Z-score as data fusion operator after a blind-query expansion tends also to improve the MAP of the merged run over the best single IR system.

Keywords: Hindi, Bengali and Marathi information retrieval, retrieval effectiveness with Indian Languages, FIRE evaluation campaign, automatic indexing.

1 Introduction

During the last ten years, the IR group at University of Neuchatel was involved in designing, implementing and evaluating various indexing and search strategies for various natural languages, including popular European [1], Far-East (e.g., Chinese, Japanese, and Korean) [2], as well as Indian languages [3]. This objective also includes bilingual IR (the topics are then written in one language, the retrieved documents in another) or multilingual IR systems (targeted information items are written

P. Majumder et al. (Eds.): FIRE 2010 and 2011, LNCS 7536, pp. 334–352, 2013.

in different languages). In our participation in the second FIRE campaign (www.isical.ac.in/~fire/), our main motivation is to promote new tools and to improve existing ones for monolingual IR when facing with Hindi, Marathi and Bengali languages.

The rest of this paper is organized as follows: Section 2 presents an overview of the corpora used in the FIRE-2010 *ad hoc* track. Section 3 outlines the main aspects of various IR models used with these test-collections together with the stopword lists and stemming strategies we developed for these languages. Section 4 presents the evaluation carried out on the various query formulations, stemming and indexing strategies using various IR models for the Hindi, Bengali and Marathi corpora. Finally, Section 5 describes our official runs and their evaluation while Section 6 gives a general conclusion.

2 Overview of the Corpora

The corpora used in our experiments are based on newspaper articles extracted from four main sources, namely *Anandabazar Patrika* (2004-2007) for the Bengali language, *Maharashtratimes* & *Esakal* (2004-2007) for the Marathi corpus, and *Dainik Jagran* & *Amar Ujala* (2004-2007) for the Hindi collection. The latest newspaper forms a new source of information in the second FIRE 2010 *ad hoc* campaign. The encoding system used for both documents and topic formulations is UTF-8.

In order to obtain an overall picture of the three corpora, we have reposted some statistics in Table 1. In this table, we can see that the Hindi collection is the largest with around 1.3 GB of data. The number of documents is however similar between this language and the Bengali corpus. When inspecting the document length, the Hindi, Bengali and Marathi corpora show similar mean lengths. These values range from 300.7 for the Hindi corpus to 264.6 for the Marathi collection. It is interesting to note that even though the Marathi collection is the smallest (487 MB), it contains a larger number of distinct indexing terms (511,550) when compared to both the Hindi and Bengali corpora. This fact is certainly related to a more complex inflectional morphology for this language.

Based on the TREC model [4], each topic formulation consists of three logical sections, namely a brief title (denoted T), a one-sentence description (D), and a narrative part (N) used mainly to specify more precisely the relevance judgment. Available topics reflect a diversity of information needs having mostly a national coverage (e.g., Topic #86 “Privatization of the Mumbai and Delhi airports”, Topic #106 “Ban on Taslima Nasreen's novel “Shame””, or Topic #119 “Taj Mahal controversy”). The real user information need behind the topic description is sometimes difficult to determine, at least based on the title of the topic formulation (e.g., Topic #89: “Involvement of Congress ministers in the oil-for-food scam”).

In the bottom part of Table 1, we have indicated the number of relevant documents (label “#Rel. doc.”) per topic, with the mean always being greater than the median (e.g., for the Marathi collection, the average number of relevant documents per query is 15.9, with the corresponding median being 10). These findings indicate that each

collection contains numerous queries, yet only a rather smaller number of relevant items are found. For each collection, 50 queries were created (numbered from #76 to #125), and then manually translated into the other languages, including also an English version. Relevant documents could not however be found for each request and each language. For the Marathi language, eleven topics (#95, #98, #103, #107, #108, #113, #117, #119, #120, #121, and #125) do not have any relevant item in the collection.

Table 1. FIRE Test-Collection Statistics

	Hindi	Bengali	Marathi
Size	1,300 MB	732 MB	487 MB
# documents	149, 481	123,047	99,357
# terms	230,578	249,215	511,550
Number of indexing terms per document			
Mean	300.7	291.88	264.6
Std dev.	337.13	180.62	188.96
Median	220	265	222
Max	6,998	2,928	5,077
Min	0	0	28
Topics			
Number	50	50	39
# Rel. doc.	913	510	621
Mean	18.26	10.2	15.9
Std dev.	15.3	6.6	18.5
Median	14	8	10
Max	74 (T #93)	29 (T #89)	72 (T #88)
Min	2 (T #78, #87)	2 (T #84)	1 (T #79, #102, #122, #124)

The largest number of relevant items is 74 for Topic #93 ("Relations between Congress and its allies") in the Hindi collection. On the other hand, and for the Marathi corpus only, Topic #79 ("Building roads between China and Mount Everest"), Topic #102 ("Pakistani cricketers involved in a doping scandal"), Topic #122 ("Sanjay Dutt's surrender") and Topic #124 ("Sale of illegal drugs in various Indian states") have only one relevant document.

3 IR Models and Stemming Strategies

3.1 IR Models

Instead of being limited to a single indexing and search strategy, our aim is to obtain a relatively large overview of the relative merits of different IR models. To achieve this, we have considered adopting different weighting schemes for the terms included in document or query representatives. These different IR schemes take account for term occurrence frequencies (denoted tf_{ij} for indexing term t_j in document D_i), as well as their inverse document frequency ($idf_j = \log(n/df_j)$ with n indicating the number of

documents in the corpus, and df_j the number of documents in which the term t_j occurs). To define the first IR model, we have normalized each indexing weight using the cosine in order to obtain the classical *tf idf* formulation.

In addition to this classical vector-space approach, we also considered probabilistic models such as Okapi (or BM25) [5] that also takes document length into account. As a second probabilistic approach we have implemented four variants of the DFR (*Divergence from Randomness*) paradigm proposed by Amati & van Rijsbergen [6]. In this framework, the indexing weight w_{ij} attached to term t_j in document D_i combines two information measures as follows:

$$w_{ij} = \text{Inf}_{ij}^1 \cdot \text{Inf}_{ij}^2 = -\log_2\left(\text{Prob}_{ij}^1(tf)\right) \cdot \left(1 - \text{Prob}_{ij}^2\right) \tag{1}$$

As a first model, we have implemented the PB2 scheme defined as follows:

$$\text{Prob}_{ij}^1 = \left[\frac{e^{-\lambda_j} . \lambda_j^{tf_{ij}}}{tf_{ij}!}\right] \quad \text{with} \quad \lambda_j = \frac{tc_j}{n} \tag{2}$$

$$\text{Prob}_{ij}^2 = 1 - \left[\frac{tc_j + 1}{df_j \cdot \left(tfn_{ij} + 1\right)}\right] \quad \text{with} \quad tfn_{ij} = tf_{ij} \cdot \log_2\left[1 + \frac{c \cdot mean\ \ dl}{l_i}\right] \tag{3}$$

where tc_j indicates the number of occurrences of term t_j in the collection, l_i the length (number of indexing terms) of document D_i, *mean dl* the average document length, n the number of documents in the corpus, and c is a constant. Table 2 depicts the exact values of these parameters used in our experiments.

Table 2. Parameter Settings for the Various Test-Collections

	Okapi			DFR	
Language	*b*	K_1	*avdl*	*c*	*mean dl*
Hindi	0.55	1.2	300	1.5	300
Bengali	0.55	1.2	292	1.5	292
Marathi	0.75	1.2	265	1.5	265

For the GL2 model, the implementation of Prob^1_{ij} is shown in Equation 4, and Prob^2_{ij} in Formula 5.

$$\text{Prob}_{ij}^1 = 1 - \left[\frac{1}{1+\lambda_j}\right] \cdot \left[\frac{\lambda_j}{1+\lambda_j}\right]^{tfn_{ij}} \tag{4}$$

$$\text{Prob}_{ij}^2 = \frac{tfn_{ij}}{tfn_{ij} + 1} \tag{5}$$

For the PL2 model, the implementation was carried out using Formula 2 for $Prob^1_{ij}$ and Equation 5 for $Prob^2_{ij}$. Finally for the fourth model denoted I(n_e)C2, the implementation is based on the following two equations:

$$Inf^1_{ij} = tfn_{ij} \cdot \log\left[\frac{n+1}{n_e+0.5}\right] \quad \text{with} \quad n_e = n \cdot \left[1-\left(\frac{n-1}{n}\right)^{tc_j}\right] \tag{6}$$

$$\text{Prob}^2_{ij} = 1 - \frac{tc_j + 1}{df_j \cdot (tfn_{ij} + 1)} \tag{7}$$

Finally we also considered an approach based on a statistical language model (LM) [7], [8], known as a non-parametric probabilistic model (the Okapi and DFR are viewed as parametric models). Probability estimates would thus not be based on any known distribution (e.g., as in Equation 2 or 4), but rather estimated directly based on the term occurrence frequencies in document D_i or the whole corpus C. Within this language model paradigm, various implementations and smoothing methods might be considered, although in this study we adopted a model proposed by Hiemstra [8], as described in Equation 8, combining an estimate based on the document (Prob[t_j|D_i]) and on the corpus (Prob[t_j|C]) combining using the Jelinek-Mercer smoothing [9] scheme.

$$\text{Prob}[D_i \mid Q] = \text{Prob}[D_i] \cdot \prod_{t_j \in Q} \left[\lambda_j \cdot \text{Prob}[t_j \mid D_i] + (1-\lambda_j) \cdot \Pr ob[t_j \mid C]\right] \tag{8}$$

$$\Pr ob[t_j \mid D_i] = tf_{ij} / l_i \quad \text{and} \quad \Pr ob[t_j \mid C] = df_j / l_c \quad \text{with} \quad lc = \sum_k df_k$$

where λ_j is a smoothing factor (constant for all indexing terms t_j, and fixed at 0.35) and lc an estimate of the size of the corpus C.

3.2 Stopword Lists and Stemmers

During this evaluation campaign, our stopword lists for the Hindi and Bengali languages were the same as those used during our FIRE 2008 participation [3]. These stopword lists contain 165 Hindi terms and 114 Bengali words. During the second FIRE evaluation campaign, we have proposed a new stopword list for the Marathi language. This list, created in the same way as the stopword lists for other two languages [10], contains 99 terms. We may mention that compared to other Indo-European languages, these lists are rather short (e.g., the SMART system used a list of 471 words for the English language). We may certainly include additional words to speed up the query processing and to enhance the quality of the final ranked list of retrieved items. Recent studies tend to show that the elaboration of such stopword list may have a clear impact on the retrieval effectiveness of a search engine [11].

The light stemming procedures we have employed this year are the same as those used for the Indian languages during the first FIRE evaluation campaign. These stemming procedures remove inflectional suffixes attached to both nouns and

adjectives, while completely ignoring the verbal morphology of the underlying language. This reflects our belief and prior experiments with other languages [12] that nouns and adjectives are the Part-Of-Speech (POS) categories covering the most important part of the semantic content of both documents and queries. Moreover, including numerous verbal suffixes in a suffix-striping approach might hurt retrieval effectiveness, especially when we know that such stemmer does not consider the underlying POS or does not involve a complex morphological analysis. Finally, for the English language at least, we do not find that a deeper morphological analysis proposes a better retrieval effectiveness than simple stemmers [13], [14] approaches [15].

Finally, our participation in the FIRE evaluation campaigns was also motivated by our wish to promote and evaluate new and more aggressive stemming procedures for the Hindi, Marathi and Bengali languages. These procedures, apart from removing inflectional suffixes from nouns and adjectives, remove also some frequently used derivational suffixes found in the grammar of the corresponding language. In the web site `members.unine.ch/jacques.savoy/clef/` we can find the proposed stopword lists and various stemmers.

4 Evaluation and Analysis

To evaluate various indexing and search strategies, Section 4.1 exposes the evaluation measures and methodology used in our experiments. Section 4.2 presents the performance achieved by seven retrieval models based on three topic formulations (T, TD and TDN). Section 4.3 describes the performance that can be achieved by using three different stemming strategies (none, light or aggressive) while Section 4.4 reports the evaluation achieved by using three document and query representations (word-based, *n*-gram, and trunc-*n*). Section 4.5 analyzes some queries in an attempt to understand the impact of various stemming and indexing strategies as well as the effect of adding search terms to the current query. The last section evaluates the impact of two automatic blind-query expansion techniques to hopefully improve the retrieval effectiveness.

4.1 Evaluation Methodology

As a measure of retrieval effectiveness, we have adopted mean average precision (MAP) (computed by the `TREC_EVAL` software based on a maximum of 1,000 retrieved records). This performance measure has been used by all evaluation campaigns for around 20 years in order to objectively compare various IR models, particularly regarding their ability to retrieve relevant items (*ad hoc* tasks) [16]. Using this evaluation tool, some differences may occur in the values computed according to the official measure. The latter always takes 50 queries while in our presentation we did not account for queries having no relevant item, as for the Marathi collection owning only 39 queries. In the following tables, best performances under given conditions (same indexing scheme and same collection) are listed in bold type.

Using the mean as a measure of the system's performance signifies that we attached an equal importance to all queries. Comparisons between two IR strategies

will therefore not be based on a single query with respect to those available in the underlying test-collection or specifically created in order to demonstrate that a given IR approach must be rejected. Thus we believe that it is important to conduct experiments involving the largest possible number of observations (between 39 and 50 queries in our evaluations, depending on the language).

To statistically determine whether or not a given search strategy would be better than another, we applied the bootstrap methodology [17] showing very similar conclusion than the *t*-test but without requiring parametric assumption [18]. In our statistical tests, the null hypothesis H_0 stated that both retrieval schemes produce similar MAP performance. Such a null hypothesis would be accepted if two retrieval schemes returned statistically similar MAP, otherwise it must be rejected. Thus, in the experiments presented in this paper, statistically significant differences were detected by a two-sided test (significance level $\alpha = 5\%$).

4.2 Evaluation of Different Query Formulations

Based on the Hindi corpus, Table 3 shows the MAP obtained by seven IR models with three different query formulations (T, TD, and TDN) using a word-based and a light stemming approach. Tables 4 and 5 depict the same information for the Bengali and Marathi languages respectively. Across the three different corpora and three query formulations, we can see that the best IR model is usually I(n_e)C2, an implementation of the DFR family. We must recognize that the performance differences are not important when comparing this model with the DFR-PB2 scheme.

Table 3. MAP of Various IR Models (Light Stemming, Hindi Corpus, 50 queries)

	Mean Average Precision		
Light Stemming Model	Hindi T	Hindi TD	Hindi TDN
Okapi	0.3011	0.3717	0.4533 †
DFR-PB2	0.2851	0.3737	0.4630
DFR-GL2	0.2730 †	0.3524 †	0.4337 †
DFR-PL2	0.2843 †	0.3621	0.4430 †
DFR-I(n_e)C2	**0.3054**	**0.3836**	**0.4732**
LM	0.2533 †	0.3310 †	0.4223 †
tf idf	0.1427 †	0.1830 †	0.2367 †
Average	0.2635	0.3368	0.4179
Change % over T	base	+27.79%	+58.56%

To verify whether these performance differences are statistically significant, we compare the various IR schemes to the best performing model depicted in bold. We then marked with a cross ("†") the performance values depicting statistically significant differences. In this case, the classical *tf idf* vector-space model and the language model (LM) offer a performance level that is significantly lower for all languages and topic formulations. For the other models, the answer depends on the language and the

IR model. We can see however that the performance differences between the DFR-PB2 and DFR-I(n_e)C2 are never significant. With the Okapi model, we found just one significant performance difference (Hindi corpus and with TDN topic formulation). Finally we can observe mixed results when analyzing the retrieval effectiveness differences with the I(n_e)C2 and the two remaining DFR implementations, namely DFR-GL2 and DFR-PL2. These differences are usually significant with the Hindi corpus, and usually not when considering the Bengali or Marathi language.

Table 4. MAP of Various IR Models (Light Stemming, Bengali Corpus, 50 queries)

	Mean Average Precision		
Light Stemming Model	Bengali T	Bengali TD	Bengali TDN
Okapi	0.3527	0.4256	0.4925
DFR-PB2	**0.3586**	**0.4405**	0.4980
DFR-GL2	0.3350	0.4081	0.4697 †
DFR-PL2	0.3434	0.4221 †	0.4872
DFR-I(n_e)C2	0.3543	0.4383	**0.5026**
LM	0.3102 †	0.3946 †	0.4720 †
tf idf	0.1750 †	0.2061 †	0.2456 †
Average	0.3185	0.3908	0.4525
Change % over T	base	+22.70%	+42.10%

Table 5. MAP of Various IR Models (Light Stemming, Marathi Corpus, 39 queries)

	Mean Average Precision		
Light Stemming Model	Marathi T	Marathi TD	Marathi TDN
Okapi	0.2986	0.3446	0.3855
DFR-PB2	0.2921	0.3247	0.3910
DFR-GL2	0.2884	0.3336	0.3745
DFR-PL2	0.2849	0.3177 †	0.3658 †
DFR-I(n_e)C2	**0.3075**	**0.3502**	**0.4137**
LM	0.2892 †	0.3185 †	0.3824 †
tf idf	0.2024 †	0.2286 †	0.2535 †
Average	0.2804	0.3168	0.3666
Change % over T	base	+12.98%	+30.73%

In the bottom part of these three tables, we can find under the label "Average" the average MAP across the seven IR models. The last line shows the relative percentage of variation obtained when compared to the short (T) query formulation.

From these tables we can see that enlarging the query formulation (from T to TD, and from TD to TDN) brings the improvement of the retrieval effectiveness for all three languages in question. This improvement is less for the Marathi language (see Table 5) than for both the Hindi (see Table 3) and Bengali corpus (see Table 4). The

longest topic formulation improve the MAP from 30.73% for the Marathi language to 58.56% for the Hindi collection, showing clearly the need of more search terms in order to perform an effective search.

When applying our statistical test with the performance achieved under the Title-only topic formulation as baseline, we always found statistically significant differences with either TD or TDN topic formulation. Thus when the user is able to enlarge the query, the retrieval effectiveness is significantly improved.

4.3 Evaluation of Various Stemming Strategies

In the previous section we compared different topic formulations using the light stemmer. In order to investigate whether others stemming strategies may improve the retrieval effectiveness, we need to consider a more aggressive stemmer on the one hand and, on the other, ignoring this word normalization process (no stemming). For the Hindi language, Table 6 depicts the MAP obtained by various IR models using these three different stemming approaches (TD query formulation). Similar conclusions can be obtained with T or TDN query formulations.

Table 6. MAP of Various Stemming Strategies, TD queries (Hindi Corpus)

	Mean Average Precision		
TD Model	Hindi no stemmer	Hindi light	Hindi aggressive
Okapi	0.3835	0.3717	0.3986
DFR-PB2	0.3908	0.3737	0.3875
DFR-GL2	0.3627	0.3524	0.3684
DFR-PL2	0.3740	0.3621	0.3873
DFR-I(n_e)C2	**0.3917**	**0.3836**	**0.4067**
LM	0.3481	0.3310 †	0.3523
tf idf	0.1975	0.1830 †	0.1833 †
Average	0.3498	0.3368	0.3549
Change %	base	-3.17%	+1.46%

Using the same condition, MAP obtained with the Bengali corpus is reported in Table 7 and Table 8 depicted the same information with the Marathi language. While for the Hindi the light stemming hurts MAP in mean (-3.17% compared to "no stemmer" scheme), for the Bengali and Marathi languages this indexing scheme tends to produce, in mean, better retrieval effectiveness compared to an indexing strategy ignoring the stemming stage. On the other hand, the aggressive stemming results in an improvement in MAP for all three languages, the performance difference comparing to no stemming being rather small for Hindi language (e.g., +1.46% in mean over 7 IR models) while being much more important for both Bengali and Marathi languages (e.g., +19.57% and +33.76% respectively).

When applying our statistical test to verify whether these performance differences are statistically significant, we used the performance achieved without any stemming normalization as baseline (values shown under the label "no stemmer"). We marked with a cross ("†") the performance values depicting statistically significant differences.

Table 7. MAP of Various Stemming Strategies, TD queries (Bengali Corpus)

	Mean Average Precision		
TD Model	Bengali no stemmer	Bengali light	Bengali aggressive
Okapi	0.3640	0.4256 †	**0.4446** †
DFR-PB2	0.3629	**0.4405** †	0.4366 †
DFR-GL2	0.3498	0.4081 †	0.4405 †
DFR-PL2	0.3550	0.4221 †	0.4311 †
DFR-I(n_e)C2	**0.3673**	0.4383 †	0.4443 †
LM	0.3402	0.3946 †	0.4078 †
tf idf	0.2179	0.2061	0.2136
Average	0.3367	0.3908	0.4026
Change %	base	16.05%	19.57%

Table 8. MAP of Various Stemming Strategies, TD queries (Marathi Corpus)

	Mean Average Precision		
TD Model	Marathi no stemmer	Marathi light	Marathi aggressive
Okapi	0.2872	0.3446 †	0.3958 †‡
DFR-PB2	0.2947	0.3247	0.3928 †‡
DFR-GL2	0.2937	0.3335 †	0.3829 †
DFR-PL2	0.2767	0.3177 †	0.3891 †‡
DFR-I(n_e)C2	**0.2976**	**0.3502** †	**0.4118** †‡
LM	0.2907	0.3185 †	0.3824 †‡
tf idf	0.2023	0.2286	0.2435
Average	0.2775	0.3168	0.3712
Change %	base	14.17%	33.76%

With the Hindi language (see Table 6), the performance differences between the three stemming approaches are usually not significant, except with the classical *tf idf* vector-space model. As depicted in Table 7, the Bengali corpus presents a different situation where all performance differences are significant, except with the *tf idf* model. With the Marathi language (see Table 8), we found a similar conclusion demonstrating that applying a stemmer tends to improve the retrieval effectiveness with language owning a more complex inflectional morphology than English or Hindi.

Finally, we want to analyze the differences in retrieval effectiveness when applying the light stemmer (baseline) with the aggressive stemming approach. Each statistically

significant difference is marked with a double cross ("‡") in Table 6 to 8. As we can see, for both the Hindi and Bengali languages, we detect no significant difference between the light and a more aggressive stemmer. For the Marathi corpus however, the aggressive stemming scheme tends to produce significantly better results as depicted in Table 8.

4.4 Evaluation of Various Indexing Strategies

Finally we will compare different document and query representation strategies. Instead of being limited to a word-based surrogate, we want to evaluate the effectiveness of two language-independent indexing strategies, namely 4-gram [19] and trunc-4 [20]. For the Hindi (see Table 9), Bengali (see Table 10), and Marathi language (see Table 11), we have computed the MAP of these indexing approaches and compared them to the word-based indexing scheme with the aggressive stemmer and using TD query formulation.

Table 9. MAP of Various Indexing Strategies, TD queries (Hindi Corpus)

	Mean Average Precision		
TD Model	Hindi aggressive	Hindi 4-gram	Hindi trunc-4
Okapi	0.3986	0.3674	0.3770
DFR-PB2	0.3875	0.3704	0.3687
DFR-GL2	0.3684	0.3479	0.3569
DFR-PL2	0.3873	0.3554	0.3719
DFR-I(n_e)C2	**0.4067**	**0.3841**	**0.3780**
LM	0.3523	0.3366	0.3378
tf idf	0.1833	0.1837	0.1844
Average	0.3549	0.3351	0.3392
Change %	base	-5.58%	-4.40%

Table 10. MAP of Various Indexing Strategies, TD queries (Bengali Corpus)

	Mean Average Precision		
TD Model	Bengali aggressive	Bengali 4-gram	Bengali trunc-4
Okapi	**0.4446**	0.3803 †	**0.4522**
DFR-PB2	0.4366	0.3875 †	0.4395
DFR-GL2	0.4405	0.3740 †	0.4260
DFR-PL2	0.4311	0.3841	0.4377
DFR-I(n_e)C2	0.4443	**0.3876** †	0.4493
LM	0.4078	0.3557	0.4063
tf idf	0.2136	0.2143	0.1921
Average	0.4026	0.3548	0.4004
Change %	base	-11.89%	-0.55%

Results depicted in these tables tend to indicate that both language-independent indexing strategies result in similar performance levels when facing with the Hindi language (e.g., in average, -5.58% for 4-gram, and -4.40% for trunc-4, see Table 9). Applying our statistical test, all performance differences compared to the word-based with an aggressive stemmer are not significant. When comparing with either the light or no stemmer, we reached the same conclusion: no significant performance differences for all IR models with the Hindi corpus.

For both Bengali and Marathi language, results depicted respectively in Table 10 and 11 tend to indicate that trunc-4 language-independent indexing strategy result in similar retrieval effectiveness, in average, when compared to a word-based indexing scheme with an aggressive stemmer (e.g., -0.55% for the Bengali, +3.18% for the Marathi corpus). We can also find that the trunc-*n* tends to produce better retrieval results than the *n*-gram scheme that is also more complex to implement and require more query and indexing processing time. The statistical tests can however detect significant performance differences only when comparing word-based (aggressive stemmer) with 4-gram with the Bengali corpus (indicated by a cross "†" in Table 10).

Table 11. MAP of Various Indexing Strategies, TD queries (Marathi Corpus)

	Mean Average Precision		
TD Model	Marathi aggressive	Marathi 4-gram	Marathi trunc-4
Okapi	0.3958	0.3525	0.4161
DFR-PB2	0.3928	0.3329	0.4191
DFR-GL2	0.3830	0.3653	0.3881
DFR-PL2	0.3891	0.3440	0.3972
DFR-I(n_e)C2	**0.4118**	**0.3744**	**0.4347**
LM	0.3824	0.3592	0.3920
tf idf	0.2435	0.2204	0.2337
Average	0.3712	0.3355	0.3830
Change %	base	-9.61%	3.18%

When using a light stemmer as baseline and with the Bengali language, the performance differences are usually significant with the 4-gram approach, and not significant with the trunc-4 scheme. For Marathi (see Table 11) and using the light stemmer as baseline, no significant difference can be found with the 4-gram indexing scheme, but when compared with the trunc-4, the retrieval effectiveness differences are always significant, except with the *tf idf* model.

4.5 Some Query-by-Query Analysis

To obtain a better understanding of effects associated with different stemming and indexing strategies, we analyzed a few Hindi queries. As a first example we can inspect Topic #108 "Greater Nagaland" (owning 13 relevant items). This query performs poorly with Title-only topic formulation, achieving an average precision (AP)

of 0.0963 with word-based indexing (light stemmer). The performance increases when considering TD topic formulation (AP = 0.7393) or with TDN topic formulation (AP = 0.8516). The internal representation of this query is limited to one term ("वृहत्तर" or "greater") when using the Title-only topic formulation, and this ineffective surrogate cannot retrieve pertinent information. When including related terms from the descriptive or narrative section (such as the name "एनएससीएन"), the overall performance increases.

To analyze the effect of an aggressive stemming procedure in representing the query, we can look at Topic #77 "Attacks by Hezbollah guerrillas" (with eight relevant items). When using a word-based indexing scheme ignoring the stemming normalization, the average precision is rather low (0.1063). Using a light stemmer, we degrade the retrieval effectiveness for this query (AP = 0.0340). After employing an aggressive stemmer, the performance increases to 0.3238. The main reason is related to the word "Hezbollah" ("हिजबुल्लाह") that cannot retrieve many relevant documents owning the related expression "हिजबु" ("Hezbo"). This second term can be found only after applying the aggressive stemmer, thus offering more matching possibilities between the query and other relevant articles.

To illustrate the difference between word-based and *n*-gram indexing strategies, we analyze Topic #107 "Furore over the release of a CD containing anti-Muslim sentiments in Uttar Pradesh" (owning ten relevant items). Using a word-based representation with a light stemmer, we can achieve an AP of 0.6491, but using a 4-gram scheme the performance is clearly lower (AP = 0.0692). In the relevant articles, we can usually find the state name "Uttar Pradesh" and "Muslim". With the 4-gram indexing scheme, many non-relevant items appear in the top of the ranked list due to the fact that they have the term "प्रदेशा" ("Pradesh", meaning *state*) instead of "प्रदेश" (the same form but without an additional letter at the end). With word-based, the search system can discriminate between the correct answers and the irrelevant ones. This capability is less accurate with the 4-gram representation, and thus the performance decreases.

4.6 Pseudo-relevance Feedback

Previous experiments with different languages and corpora tend to indicate that pseudo-relevance feedback (PRF or blind-query expansion) seemed to be a useful technique for enhancing retrieval effectiveness. In this study, we have adopted Rocchio's approach [21]) with $\alpha = 0.75$, $\beta = 0.75$, whereby the system was allowed to add *m* terms extracted from the *k* best ranked documents from the original query (see Table 12). From our previous experiments we learned that this type of blind query expansion strategy does not always work well. More particularly, we believe that including terms occurring frequently in the corpus (because they also appear in the top-ranked documents) may introduce more noise, and thus be an ineffective means of discriminating between relevant and non-relevant items [22]. Consequently we also chose to apply our *idf*-based query expansion model [23].

Using the Rocchio's method, Table 12 shows that the retrieval performance after applying a pseudo-relevance feedback approach can be improved for both the Hindi and Marathi corpus, not with the Bengali. The best result for the Hindi language indicates an enhancement of +16.8% (from 0.4067 to 0.4750), while for the Marathi we

were able to increase the MAP of +5.8% (from 0.4118 to 0.4359). To verify whether these performance differences are statistically significant, we applied our statistical test with the performance before blind-query expansion as baseline (values shown in the third row). We marked with a cross ("†") the MAP values depicting a statistically significant difference. As depicted in Table 12, only a few parameter settings were able to achieve a significant performance difference over the baseline.

Table 12. MAP of Different Blind-Query Expansions, Rocchio's method, TD queries

	Mean Average Precision		
TD Model	Hindi aggressive	Bengali aggressive	Marathi aggressive
DFR-I(n_e)C2	0.4067	**0.4443**	0.4118
10 docs / 10 terms	0.4465	0.3729 †	0.4224
10 docs / 30 terms	0.4688	0.4085	0.4303
10 docs / 50 terms	0.4714 †	0.4060	**0.4359**
10 docs / 70 terms	0.4729 †	0.4137	0.4347
10 docs / 100 terms	**0.4750** †	0.4131	0.4453
15 docs / 10 terms	0.4327	0.3966 †	0.4041
15 docs / 30 terms	0.4691 †	0.4007	0.4162
15 docs / 50 terms	0.4647	0.3965	0.4195
15 docs / 70 terms	0.4642	0.4016	0.4192
15 docs / 100 terms	0.4608	0.4014	0.4193

Table 13. MAP of Different Blind-Query Expansions, *idf*-based method, TD queries

	Mean Average Precision		
TD Model	Hindi aggresive	Bengali aggressive	Marathi aggressive
DFR-I(n_e)C2	0.4067	0.4443	0.4118
10 docs / 10 terms	0.4333	0.4270	0.4328
10 docs / 30 terms	0.4721 †	0.4416	0.4490
10 docs / 50 terms	0.4766 †	0.4544	0.4551
10 docs / 70 terms	**0.4829** †	0.4430	0.4495
10 docs / 100 terms	0.4760 †	0.4422	0.4550
15 docs / 10 terms	0.4123	0.4133	0.4281
15 docs / 30 terms	0.4695	**0.4578**	0.4577
15 docs / 50 terms	0.4801 †	0.4539	**0.4605**
15 docs / 70 terms	0.4703 †	0.4550	0.4473
15 docs / 100 terms	0.4633 †	0.4510	0.4406

Based on our *idf*-based blind-query expansion [23], Table 13 reports the results using different parameter settings across the three languages. The best improvement was obtained with the Hindi corpus (+18.7%, from 0.4067 to 0.4829) followed by the Marathi language (+11.8%, from 0.4118 to 0.4605). For the Bengali, the enhancement was smaller (+3%, from 0.4443 to 0.4578). Compared to the results obtained

with the Rocchio's method (see Table 12), the *idf*-based approach performs better. When applying our statistical test to verify whether these performance differences are statistically significant, we select the performance before blind-query expansion as baseline (MAP values shown in the third row). We marked with a cross ("†") the retrieval effectiveness values depicting a statistically significant difference. As depicted in Table 13, such a significant differences occurs usually only for the Hindi corpus.

5 Official Results

Table 14 shows the exact specifications of our 6 official monolingual runs for the Hindi *ad hoc* monolingual evaluation task. These runs are based on three probabilistic models (Okapi, DFR and statistical language model (LM)). In each case, we then applied a pseudo-relevance feedback stage, and finally we merged the three individual ranked lists into a fused common list based on the Z-score merging strategy [1]. Table 15 lists the same information for Bengali, showing our 6 official submissions while Table 16 reports our official experiments for the Marathi language.

To propose effective search strategies, we selected three IR probabilistic models and enlarged the query by adding 20 to 150 terms retrieved from the 3 to 10 best-ranked articles contained in the Hindi (see Table 14), Bengali (see Table 15) or Marathi (see Table 16) collection. In the last column of Table 16 we have given in brackets the official results taking into account all 50 available topics.

Table 14. Description and Mean Average Precision (MAP) for our Official Hindi Monolingual Runs

	Index	Model	PRF	MAP	MAP
1	trunc-4	PL2	3 / 20	0.4440	Z-score
TD	4-gram	LM	10 / 50	0.3741	**0.4904**
	aggressive	I(n_e)C2	10 / 70	0.4904	
2	trunc-4	Okapi	10 / 100	0.4201	Z-score
TD	4-gram	PL2	5 / 50	0.4338	0.4836
	aggressive	LM	3 / 20	0.3936	
3	light	PB2	10 / 20	0.4504	Z-score
TD	aggressive	PL2	10 / 50	0.4431	0.4686
	trunc-4	PB2	3 / 20	0.4027	
4	light	PL2	10 / 20	0.4546	Z-score
TD	trunc-4	PL2	3 / 20	0.4440	0.4879
	4-gram	PL2	5 / 100	0.4477	
5	trunc-4	PB2	10 / 50	0.5193	Z-score
TDN	aggressive	PL2	5 / 50	0.5142	0.5339
	light	Okapi	10 / 100	0.4747	
6	4-gram	LM	5 / 50	0.4468	Z-score
TDN	trunc-4	Okapi	3 / 100	0.4942	**0.5467**
	aggressive	I(n_e)C2	3 / 50	0.4981	

Table 15. Description and Mean Average Precision (MAP) for our Official Bengali Monolingual Runs

	Index	Model	PRF	MAP	MAP
1	trunc-4	PL2	10 / 70	0.4679	Z-score
TD	4-gram	LM	10 / 70	0.4100	0.4646
	aggressive	PB2	5 / 50	0.4352	
2	light	I(n_e)C2	10 / 70	0.4327	Z-score
TD	aggressive	PL2	5 / 20	0.4590	0.4731
	trunc-4	LM	10 / 50	0.4234	
3	4-gram	Okapi	5 / 150	0.3899	Z-score
TD	trunc-4	PL2	10 / 70	0.4679	0.4684
	aggressive	PB2	5 / 50	0.4352	
4	trunc-4	Okapi	5 / 100	0.4660	Z-score
TD	aggressive	PB2	10 / 50	0.4576	**0.4862**
5	trunc-4	PB2	5 / 20	0.4866	Z-score
TDN	4-gram	Okapi	5 / 150	0.4594	0.5329
	aggressive	PL2	10 / 70	0.5021	
6	light	PL2	10 / 70	0.5165	Z-score
TDN	trunc-4	PB2	5 / 20	0.4866	**0.5438**
	4-gram	GL2	5 / 50	0.4659	

Table 16. Description and Mean Average Precision (MAP) for our Official Marathi Monolingual Runs

	Index	Model	PRF	MAP	MAP
1	trunc-4	GL2	5 / 20	0.4437	Z-score
TD	4-gram	I(n_e)C2	3 / 50	0.4273	**0.5009**
	aggressive	PB2	10 / 50	0.4541	(0.3907)
2	trunc-4	LM	10 / 20	0.4718	Z-score
TD	4-gram	Okapi	5 / 50	0.4197	0.4897
	aggressive	PL2	10 / 50	0.4610	(0.3820)
3	trunc-4	LM	10 / 20	0.4718	Z-score
TD	aggressive	PB2	10 / 50	0.4541	0.4817
	light	Okapi	10 / 20	0.4079	(0.3757)
4	4-gram	I(n_e)C2	3 / 50	0.4273	Z-score
TD	light	GL2	5 / 70	0.4113	0.4885
	trunc-4	PL2	10 / 70	0.4790	(0.3810)
5	trunc-4	Okapi	3 / 70	0.4474	Z-score
TDN	aggressive	LM	10 / 70	0.5412	**0.5355**
	light	LM	10 / 100	0.4729	(0.4177)
6	trunc-4	LM	10 / 20	0.4910	Z-score
TDN	4-gram	Okapi	5 / 150	0.4182	0.5126
	aggressive	PL2	5 / 50	0.4878	(0.3998)

For the Hindi, Marathi and Bengali corpora, we have submitted four runs with TD formulation and two additional runs with the longest TDN query formulation in order to enhance the quality of the final pool. All runs were fully automated using our

stopword lists and different word-based and language-independent indexing strategies. Furthermore, in order to improve the overall retrieval effectiveness, we may consider different merging strategies [24], [25]. In all cases the same Z-score data fusion approach (see details in [1]) was applied.

6 Conclusion

The results achieved in FIRE 2010 evaluation campaign confirm the retrieval effectiveness of models derived from *Divergence from Randomness* (DFR) paradigm. Implementations of the DFR-I(n_e)C2 or DFR-PB2 tend to produce high MAP when facing different test-collections, in this case Hindi, Marathi, and Bengali collections. Moreover, the effectiveness of these models proves to be independent of underlying indexing strategy or query formulation. After applying a statistical test, we can conclude that the retrieval effectiveness differences were always significant when comparing the best result (DFR-I(n_e)C2 or DFR-PB2) with either the *tf idf* or LM approach. Usually, other implementations of the DFR family (DFR-GL2, DFR-PL2) tend to achieve lower performance levels for which the differences are however usually not statistically significant.

For all three languages studied we have found that enlarging the topic formulation from T to TD, or from TD to TDN (and of course from T to TDN) will improve retrieval effectiveness (up to 58% in mean, over 7 models when comparing T to TDN query formulation for Hindi collection). When enlarging the query from the Title-only topic description, performance differences were always statistically significant.

For each language and based on our experiments we have reached following conclusions regarding usage of various indexing strategies. For the Hindi language, all stemming strategies produce similar levels of performance and the differences were usually not significant. The light stemming or language-independent indexing strategies resulted in lower MAP when compared to no stemming approach. However, incorporating the aggressive stemming brings slight but not significant improvement in MAP.

For the Marathi language our experiments tend to show that an aggressive stemming approach performs significantly better than a light stemmer. This last approach is better than no stemming. While light stemming and 4-gram approaches result in comparable performances, aggressive stemming or trunc-4 brings a clear improvement in MAP for this language presenting more complex morphology.

For the Bengali language, usage of a light or aggressive stemming generates significantly better results than an indexing scheme ignoring the stemming normalization. The performance differences between the light and the aggressive stemmer are not significant. For this language, a trunc-4 language-independent approach or a word-based with an aggressive stemmer result in usually significant better performance levels than a 4-gram indexing scheme.

Of course we do not have the needed resources to investigate all possible and pertinent research questions dealing with the Hindi, Bengali, and Marathi languages. We must also mention that the elaboration of test-collections with other languages

families (e.g., Dravidian languages such as Telugu or Tamil) is, from our point of view, an important task in order to have a better understanding of the underlying problems dealing with the automatic processing of various Indian languages.

Acknowledgement. The authors would like to thank the FIRE-2010 task organizers for their efforts in developing various Indian language test-collections. This research was supported in part by the Swiss National Science Foundation under Grant #200020-129535/1.

References

1. Savoy, J.: Combining Multiple Strategies for Effective Monolingual and Cross-Lingual Retrieval. IR Journal 7, 121–148 (2004)
2. Savoy, J.: Comparative Study of Monolingual and Multilingual Search Models for Use with Asian Languages. ACM - Transactions on Asian Languages Information Processing 4, 163–189 (2005)
3. Dolamic, L., Savoy, J.: UniNE at FIRE 2008: Hindi, Marathi and Bengali IR. FIRE 2008 Working Notes (2008)
4. Voorhees, E.M., Harman, D.K. (eds.): TREC. Experiment and Evaluation in Information Retrieval. The MIT Press, Cambridge (2005)
5. Robertson, S.E., Walker, S., Beaulieu, M.: Experimentation as a Way of Life: Okapi at TREC. Information Processing & Management 36, 95–108 (2002)
6. Amati, G., van Rijsbergen, C.J.: Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness. ACM Transactions on Information Systems 20, 357–389 (2002)
7. Hiemstra, D.: Using Language Models for Information Retrieval. Ph.D. Thesis (2000)
8. Hiemstra, D.: Term-Specific Smoothing for the Language Modeling Approach to Information Retrieval. In: Proceedings of ACM-SIGIR, pp. 35–41. The ACM Press (2002)
9. Zhai, C., Lafferty, J.: A Study of Smoothing Methods for Language Models Applied to Information Retrieval. ACM Transactions on Information Systems 22, 179–214 (2004)
10. Fox, C.: A Stop List for General Text. ACM-SIGIR Forum 24, 19–35 (1990)
11. Dolamic, L., Savoy, J.: When Stopword Lists Make the Difference. Journal of the American Society for Information Sciences and Technology 61, 200–203 (2010)
12. Savoy, J.: Light Stemming Approaches for the French, Portuguese, German and Hungarian Languages. In: Proceedings of ACM-SAC, pp. 1031–1035. The ACM Press (2006)
13. Harman, D.K.: How Effective is Suffxing? Journal of the American Society for Information Science 42, 7–15 (1991)
14. Porter, M.F.: An Algorithm for Suffix Stripping. Program 14, 130–137 (1980)
15. Fautsch, C., Savoy, J.: Algorithmic Stemmers or Morphological Analysis: An Evaluation. Journal of the American Society for Information Sciences and Technology 60, 1616–1624 (2009)
16. Buckley, C., Voorhees, E.M.: Retrieval System Evaluation. In: Voorhees, E.M., Harman, D.K. (eds.) TREC. Experiment and Evaluation in Information Retrieval, pp. 53–75. The MIT Press, Cambridge (2005)
17. Savoy, J.: Statistical Inference in Retrieval Effectiveness Evaluation. Information Processing & Management 33(4), 495-512

18. Abdou, S., Savoy, J.: Statistical and Comparative Evaluation of Various Indexing and Search Models. In: Ng, H.T., Leong, M.-K., Kan, M.-Y., Ji, D. (eds.) AIRS 2006. LNCS, vol. 4182, pp. 362–373. Springer, Heidelberg (2006)
19. McNamee, P., Mayfield, J.: Character N-gram Tokenization for European Language Text Retrieval. IR Journal 7, 73–97 (2004)
20. McNamee, P., Nicholas, C., Mayfield, J.: Addressing Morphological Variation in Alphabetic Languages. In: Proceedings of ACM-SIGIR 2009, pp. 75–82. The ACM Press (2009)
21. Buckley, C., Singhal, A., Mitra, M., Salton, G.: New Retrieval Approaches Using SMART. In: Proceedings of TREC-4, pp. 25–48. NIST Publication #500-236, Gaithersburg (1996)
22. Peat, H.J., Willett, P.: The Limitations of Term Co-Occurrence Data for Query Expansion in Document Retrieval Systems. Journal of the American Society for Information Science 42, 378–383 (1991)
23. Abdou, S., Savoy, J.: Searching in Medline: Stemming, Query Expansion, and Manual Indexing Evaluation. Information Processing & Management 44, 781–789 (2008)
24. Vogt, C.C., Cottrell, G.W.: Fusion via a Linear Combination of Scores. IR Journal 1, 151–173 (1999)
25. Fox, E.A., Shaw, J.A.: Combination of Multiple Searches. In: Proceedings of TREC-2, pp. 243–249. NIST Publication #500-215, Gaithersburg (1994)

Author Index