Experiments with Document Retrieval from Small Text Collections Using Latent Semantic Analysis or Term Similarity with Query Coordination and Automatic Relevance Feedback

Colin Layfield, Joel Azzopardi, and Chris Staff(✉)

University of Malta, Tal-Qroqq, Msida 2080, Malta
{colin.layfield,joel.azzopardi,chris.staff}@um.edu.mt

Abstract. Users face the Vocabulary Gap problem when attempting to retrieve relevant textual documents from small databases, especially when there are only a small number of relevant documents, as it is likely that different terms are used in queries and relevant documents to describe the same concept. To enable comparison of results of different approaches to semantic search in small textual databases, the PIKES team constructed an annotated test collection and Gold Standard comprising 35 search queries and 331 articles. We present two different possible solutions. In one, we index an unannotated version of the PIKES collection using Latent Semantic Analysis (LSA) retrieving relevant documents using a combination of query coordination and *automatic* relevance feedback. Although we outperform prior work, this approach is dependent on the underlying collection, and is not necessarily scalable. In the second approach, we use an LSA Model generated by SEMILAR from a Wikipedia dump to generate a Term Similarity Matrix (TSM). Queries are automatically expanded with related terms from the TSM and are submitted to a term-by-document matrix Vector Space Model of the PIKES collection. Coupled with a combination of query coordination and *automatic* relevance feedback we also outperform prior work with this approach. The advantage of the second approach is that it is independent of the underlying document collection.

Keywords: Term similarity matrix · SEMILAR LSA Model · PIKES test collection · Log Entropy

1 Introduction

When databases contain textual documents, retrieving relevant documents can be made more accurate by pre-processing them and making available alternative indexes for search and retrieval. Stankovic *et al.* extract named entities from documents stored in a geological database in Serbian and index both the document

A. Calì et al. (Eds.): IKC 2016, LNCS 10151, pp. 25–36, 2017.
DOI: 10.1007/978-3-319-53640-8_3

content and the named entities to demonstrate that it is superior to using SQL on its own [15]. Successes are also achieved when recognised named entities are linked to 'ground truth' [4,17] through, for instance, Name Entity Linking (NEL) to Linked Open Data (LOD). In this paper, we describe alternative approaches based on Latent Semantic Analysis [5] and Term Similarity that do not need to recognise named entities, and which achieves greater accuracy on the PIKES test collection [17] than previous work [4].

Corcoglioniti *et al.* first mark up and index the text of documents and then add a number of 'semantic layers' that support Named Entity Linking and represent relations including temporal relations. Queries are similarly processed and documents are retrieved from the collection on the basis of similarity to the query and they are ranked using the semantic information [4]. The advantage of their approach is that they are using external resources to perform Named Entity Linking, which enables them to accurately identify and reason about named entities in and across documents. However, their solution is not scalable (yet).

One of our approaches is based on applying Latent Semantic Analysis (LSA) to the document collection used for retrieval. LSA is known to be computationally expensive given large enough text collections (although approximate techniques based upon ignoring certain word lexical categories, or parts of speech, are achieving good results, e.g., the SEMILAR LSA Models [16]). Our second approach uses one of the SEMILAR LSA Models, which has been produced from an early Spring 2013 English Wikipedia dump, to identify additional terms that are related to terms that appear in queries. This second approach is independent of the PIKES collection and is scalable. Although it does not give results that are as accurate as those obtained by applying LSA directly to the PIKES collection, it still outperforms prior work (see Sect. 4).

The aim of Corcoglioniti *et al.* is to "investigate whether the semantic analysis of the query and the documents, obtained exploiting state-of-the-art Natural Language Processing techniques (e.g., Entity Linking, Frame Detection) and Semantic Web resources (e.g., YAGO, DBpedia), can improve the performances of the *traditional* [our italics] term-based similarity approach" [4]. Our work demonstrates that an approach based on Latent Semantic Analysis outperforms the semantic approach taken by Corcoglioniti *et al.* on the same test collection, as does an approach based on automatic query expansion on a traditional Vector Space Model of the PIKES collection, when, using either approach, *query coordination* is incorporated into the similarity measure and *automatic relevance feedback* is used to re-rank results.

2 Literature Review

There is a significant difference between searching for any document that contains the information you seek in a massive collection like the World Wide Web, and finding all the documents that contain relevant information in small, specialised collections. In the former, many relevant documents are likely to exist,

some of which will contain terms that the user expressed in the query. As long as one of those documents is retrieved, the user may be satisfied. In the latter, different terms are likely to be used in documents to represent the concept, so if a predominantly query term-matching retrieval approach is used, only a subset of relevant documents will be retrieved. Of course, queries may be underspecified in either situation, in which case insufficient information is available to disambiguate the query. To address this problem, search results clustering may be used to partition the results list into clusters, where each cluster represents a different query *sense* (e.g., [10,14]). Attempts to improve recall and precision in small document collections include removing, reducing, or otherwise handling ambiguity in documents, and disambiguating query terms or clustering results.

The PIKES document and query test collection was built to showcase that in small collections search is limited because of the vocabulary gap - basically, relevant documents might not contain terms that the user has included in the query, but they might contain related terms [17].

Disambiguating and exposing Named Entities occurring in documents and queries and the relations between the named entities can assist with determining the degree of relevance between a document and a query [4]. Named Entities can be explicitly linked to open data sources and external resources such as WordNet or Wikipedia can be used to determine the degree of similarity between identified relations. Azzopardi and Staff also use Named Entities and relations between them to automatically cluster news reports by news event [2], eventually building a *fused* document containing the information from different reports [1]. However, this is achieved without the utilisation of external resources.

A number of studies show how Information Retrieval (IR) can be enhanced by extracting Named Entities and other semantic information. Stankovic *et al.* extract Named Entities from text files stored in a geological database [15], and index the Named Entities as well as performing full-text indexing on the files. Waitelonis *et al.* proposed marking up documents with semantic information, to help uncover semantic similarity and relatedness, usually by marking up Named Entities with Linked Open Data sources. They produced the PIKES test collection[1]. Their approach "shows that retrieval performance on less than web-scale search systems can be improved by the exploitation of graph information from LOD resources" [17]. Corcoglioniti *et al.* perform full-text indexing on the cleaned PIKES collection, perform Named Entity Linking, and extract relations from the collection, including temporal relations, which are then indexed as separate *semantic layers* [4]. When the collection is queried, the query is similarly processed, relevant documents are retrieved using and the results are ranked using the semantic layers, which usually increases precision.

Rather than using Named Entities or semantic classes/entities, purely statistical techniques can be used to enhance retrieval. One of the original aims of Latent Semantic Analyses (LSA) was to enhance IR by extracting the 'latent semantic structure' between terms and documents [5]. SEMILAR have made

[1] The original and cleaned collections are available from http://pikes.fbk.eu/ke4ir.html.

available a number of LSA models built from a Wikipedia dump [16], and identify the Wiki 4 LSA Model to be the best to perform term similarity[2]. To produce these LSA models, they systematically removed lemmas that belong to certain lexical categories. For instance, to produce the Wiki 4 LSA Model, they removed words that do not exist in WordNet[3], adverbs, adjectives, and 'light' verbs from the document collection before indexing the documents and performing Singular Value Decomposition using Latent Semantic Analysis. This produces a term vector matrix for approximately 68000 lemmas. Two term vectors can be compared using cosine similarity, for instance, the result of which indicates the 'relatedness' (or semantic similarity) between the two terms.

3 Our Approach

We begin by giving a general overview of our approach, providing detailed information in the following subsections. We experiment on the same cleaned PIKES test collection as Corcoglioniti *et al.* As described in Subsect. 3.1, we first build a standard Vector Space term-by-document Model of the unannotated PIKES document collection (which we refer to as the PIKES VSM Model). We use the PIKES VSM Model in two distinct approaches.

In the first approach, we derive an LSA Model from the PIKES VSM Model using Latent Semantic Analysis [5]. In the second approach, we use the PIKES VSM Model directly, but we automatically expand queries with related terms before retrieving a list of results. To support query expansion, we transformed an LSA Model independently generated from a Wikipedia dump (Stefanescu *et al.*'s SEMILAR Wiki 4 [16]) into a Term Similarity Matrix (Wiki4TSM).

Each query in the test collection is transformed into a query vector using Log Entropy [6] to calculate the query term weights (Subsects. 3.2 and 3.3) and a ranked list of documents is retrieved from the collection representation using the approach described in Subsects. 3.4 and 3.5. The retrieval method includes *query coordination* and document length normalisation. Inspired by the Lucene search engine approach[4], query coordination is a process through which documents in the results list may have their query-document similarity score modified to take into account the percentage of query terms present in the document.

In our approach, documents in the initial results list are subsequently re-ranked using *automatic* relevance feedback inspired by Rocchio's approach [12]. However, rather than modifying the original query, we re-rank the results in order of weighted similarity to the average vectors derived from the top-n documents in the current results list (see Subsect. 3.5). Typically, only the top-one or top-two documents in the results list are used, because the PIKES collection contains only 331 documents with between one and twelve relevant documents per query according to the Gold Standard[5]. The top-n documents used to derive

[2] Wiki 4 and other LSA Models are available from http://www.semanticsimilarity.org.
[3] http://wordnet.princeton.edu.
[4] https://www.elastic.co/guide/en/elasticsearch/guide/current/practical-scoring-function.html.
[5] http://pikes.fbk.eu/ke4ir.html.

the average document vector keep their original ranking following automatic relevance feedback.

3.1 Generating the Document Collection Representations

To produce the PIKES VSM Model used in both our approaches, stop words[6] are removed from documents, the remaining terms are stemmed using the Porter Algorithm [11], and term weights are calculated using Log Entropy [6].

In the first approach, we generate an LSA model from the PIKES VSM Model, which we refer to as the PIKES LSA Model. We decompose the PIKES VSM Model using LSA with 200-dimensions (determined empirically). In the second approach we use the PIKES VSM Model directly, with an optional preprocessing step to find related terms for each query term using a Term Similarity Matrix derived independently of the PIKES document collection, (see Subsect. 3.2). The expanded query is submitted to the PIKES VSM Model.

The motivation for trying these two approaches is to overcome the vocabulary gap present in small document collections. Given that the collection may contain documents relevant to the query but which do not contain the query terms, we compare the approaches of (i) using 'latent' semantics within the collection to expose terms that are related to each other and which are present in the collection representation itself (i.e., using the PIKES LSA Model); and, (ii) using a term similarity matrix derived from an independent collection (i.e., our Wiki4TSM is derived from SEMILAR's Wiki 4 LSA Model) to expand the query to include terms that are related to the query terms.

3.2 Finding Terms Related to Query Terms

The SEMILAR Wiki 4 LSA Model is a matrix of lemma vectors that can be used to calculate the semantic similarity or 'relatedness' between any two lemmas. Wiki 4 contains more than 68000 unique lemmas. We are unable to fold the PIKES document collection and queries into the Wiki 4 LSA Model, because the publicly available model does not support this. Instead, we process the Wiki 4 LSA Model matrix to generate a lemma-to-lemma similarity matrix consisting of the cosine similarity scores between each lemma pair in the Wiki 4 vocabulary (inspired by the approach reported in [8]). This yields the Wiki4TSM Term Similarity Matrix where for a lemmatized term the intersection of the lemma and any other lemma is the 'relatedness' between the lemmas, represented as a similarity score. Also, the elements in a lemma's vector of similarity scores can be ordered according to lemma similarity to obtain a ranked list of similar or related lemmas. As Wiki4TSM is derived from an independent source, it is possible, for any document collection, that there are lemmas in the collection (e.g., PIKES VSM Model) that are not present in Wiki4TSM and *vice versa*. Indeed, 6396 of 16677 unique lemmas in PIKES (38.35%) are missing from Wiki4TSM. However, these

[6] The stop word list is available at http://www.lextek.com/manuals/onix/stopwords1.html.

missing terms are generally non-English words, 'light' verbs, adverbs, and adjectives excluded by the SEMILAR team in the construction of the Wiki 4 LSA Model, misspelt words, hyphenated words, numerals, proper nouns (therefore including Named Entities), words that contain numerics, acronyms, abbreviations, and other terms that are not found in WordNet (which were excluded by the SEMILAR team). In our second approach, we expand the query vector with related terms obtained from the Wiki4TSM.

3.3 Query Processing

The PIKES test collection contains 35 queries. How we process the query depends on whether we are using the PIKES LSA Model, or whether we expand the query using the Wiki4TSM and submit it to the PIKES VSM Model. In either case, stop words are removed from the query before it is submitted.

To find related terms in the Wiki4TSM, query terms are lemmatized using the Stanford Core Lemmatizer[7]. As explained in Subsect. 3.2, the Wiki 4 LSA Model constructed by the SEMILAR team has a vocabulary of lemmas. For each unique lemma in the query, we extract from Wiki4TSM a similarity vector where the elements represent the similarity scores of that lemma to other lemmas in the vocabulary. We then average the similarity vectors extracted for each lemma in the query - this results in a new vector that represents the average similarity of the query as a whole to the other lemmas that are not present in the query. From this average similarity vector, we remove those lemmas whose similarity is less than 0.7, and extract the top 5-weighted lemmas from the remaining ones (both determined empirically). For example, given the original query 'bridge construction' we identify and add to the query vector terms that are related to both 'bridge' and 'construction', if any. The related lemmas we extract from the Wiki4TSM are 'construct' 'girder' 'abutment', 'truss', and 'span'. As the identification of related lemmas is totally independent from the PIKES document collection, as mentioned in Subsect. 3.2, there is no guarantee that the related lemmas are present in the collection, or that query terms have a vector for the lemmatized term in Wiki4TSM.

A query submitted to the PIKES VSM Model may contain additional related lemmas after being expanded. The lemmas in the query are stemmed using the Porter Stemmer and the unique stems are weighted using Log Entropy [6]. The term weight for stems added to the query vector as related terms is dampened by a factor of 0.5 (determined empirically).

3.4 Query Coordination and Document Length Normalisation

Queries are becoming increasingly verbose [7]. Rather than containing specific query terms, queries are becoming more natural language-like. Users may require assistance to construct a query, or wish, based upon initial results, to modify a query by increasing or reducing the importance of one or more terms.

[7] http://stanfordnlp.github.io/CoreNLP/.

In *query coordination*, users can be assisted to construct a search query [13]. Lucene's query coordination approach is automatic. Lucene retrieves documents that contain any of the search terms, 'rewarding' each document based upon the percentage of query terms the document contains[8]. In our approach, the query vector derived in the previous step (Subsect. 3.3) is submitted to the document collection representation (the PIKES LSA Model or the PIKES VSM Model, depending on the approach used, bearing in mind that the LSA Model is the VSM Model with reduced dimensionality). The scoring function given in Eq. 1, inspired by Lucene's Practical Scoring Function[9], is used to retrieve a ranked list of relevant documents, together with their similarity score $score(q, d_i)$. The query coordination score is represented as the percentage of lemmas in the *original* query that are present in the document ($coord(q, d_i)$). The scoring function adjusts the similarity scores to take document length normalisation into account (see Eq. 1). We have also experimented with performing query coordination using the related lemmas that may have been added to the query vector in the Wiki4TSM approach (see Eq. 2).

$$score(q, d) = (\boldsymbol{q} \cdot \boldsymbol{d}) \times coord(q, d) \times docLenNorm(d) \quad (1)$$

$$rscore(q, d) = score(q, d) + (0.5 \times score(q, d) \times relQueryCoord(q, d)) \quad (2)$$

where:

$$coord(q, d) = \frac{|q \cap d|}{|q|} \qquad docLenNorm(d) = \frac{1}{\sqrt{\|d\|}}$$

$$relQueryCoord(q, d) = \frac{|rel(q) \cap d|}{|rel(q)|}$$

q = Set of terms in query

d = Set of terms in document

$\boldsymbol{q}$ = Vector space representation of q

$\boldsymbol{d}$ = Vector space representation of d

$|q|$ = Cardinality of the q (number of different terms in q).

$\|d\|$ = Length of d (total number of terms in d).

$rel(q)$ = Set of related terms to q

3.5 Pseudo-Relevance Feedback and *Automatic* Relevance Feedback

Pseudo-Relevance Feedback and Query Expansion (e.g., [3,9]) can be used to re-rank a ranked list of retrieved results. Such approaches assume that the

[8] Concisely explained at https://www.elastic.co/guide/en/elasticsearch/guide/current/practical-scoring-function.html.

[9] https://lucene.apache.org/core/4_6_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html.

top-n retrieved documents are more relevant to the query than the bottom-m documents. The initial query is automatically modified to add high weighted terms present in the top-n documents that are missing from the query (similarly, removing terms from the query that are present only in the bottom-m retrieved ranked documents). The documents ranked $n + 1$ to $m - 1$ are re-ranked using the reformulated query. Mitra's approach is to modify the query assuming that of 1,000 retrieved results, the top-20 are relevant and the bottom-500 are non-relevant [9]. Automatic Query Expansion or Reformulation has also been used on structured data. Yao *et al.*'s approach [18] requires the collection to be pre-processed to identify terms in the database that can substitute or enhance terms in the query. A "*term augmented tuple graph*" (TAT) is used to model term relationships between terms (words and phrases) automatically extracted from the structured database. Random walks through the TAT are used to determine the probability of textual similarity.

Our approach to document re-ranking is motivated by Mitra's [9]. However, for small text collections, we experimented with *automatic* relevance feedback to re-rank documents in the results list with respect to the top-n documents only. In these experiments, we generate the average document vector representing just the top-1 or top-2 documents in the results list. Then the standard Cosine Similarity Measure is applied to determine the similarity score between this average vector and the other document vectors in the results list. This score is dampened by a factor of 0.7 (determined empirically) and the result is added to the document's original query-document similarity score. This step can lead to document re-ranking, but the top-n documents used to generate the average top-ranked documents vector keep their original rank.

4 Evaluation

The PIKES collection "consists of 331 articles from the yovisto blog on history in science, tech, and art. The articles have an average length of 570 words, containing 3 to 255 annotations (average 83) and have been manually annotated with DBpedia entities" [17]. We use a version cleaned by the KE4IR team that does not include the annotations[10] to generate the PIKES VSM Model and the PIKES LSA Model.

We compare the results of nine runs. The runs with their settings are given in Table 1. Runs 1–6 are performed on the PIKES LSA Model. There is one basic baseline run (Run 1 with no query coordination and no relevance feedback); two runs for relevance feedback only (using either the top-1 or top-2 documents in Runs 2 and 3 respectively); one run with query coordination only (Run 4); and two runs with relevance feedback and query coordination combined (again, using either the top-1 or top-2 documents for relevance feedback in Runs 5 and 6 respectively). Runs 7–9 are performed on the PIKES VSM Model with automatic relevance feedback and query coordination of the terms in the initial query. Run 7 does not expand the query. Runs 8 and 9 automatically expand

[10] Available from http://pikes.fbk.eu/ke4ir.html.

Table 1. Our runs and their description

	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9
PIKES LSA Model	✓	✓	✓	✓	✓	✓			
PIKES VSM Model							✓	✓	✓
Wiki4TSM								✓	✓
Query expansion								✓	✓
Relevance feedback		top-1	top-2		top-1	top-2	top-1	top-1	top-1
Query coordination original terms				✓	✓	✓	✓	✓	✓
Query coordination related terms									✓

the queries with related terms obtained from Wiki4TSM. Run 9 also performs query coordination on the related terms added by query expansion.

The results of our experiments are given in Table 2 using the same measures used by Corcoglioniti *et al.* for KE4IR [4]. The Textual results are the best of the baselines used by the KE4IR team to evaluate their approach. The measures are Precision at Rank n (P@1, P@5, P@10), Normalised Discounted Cumulative Gain (NDCG) and NDCG@10, Mean Average Precision (MAP) and MAP@10.

Table 2. Our results, compared to Textual and KE4IR

	P@1	P@5	P@10	NDCG@10	NDCG	MAP@10	MAP
Textual	0.9430	0.6690	0.4530	0.7820	0.8320	0.6810	0.7330
KE4IR	0.9710	0.6800	0.4740	0.8060	0.8540	0.7130	0.7580
Run 1	0.9714	0.6686	0.4571	0.7911	0.8427	0.6967	0.7505
Run 2	0.9714	0.6629	0.4714	0.7955	0.8478	0.7052	0.7573
Run 3	0.9714	0.6800	0.4714	0.7982	0.8502	0.7131	0.7631
Run 4	**1.0000**	0.6743	0.4765	0.8034	0.8452	0.7090	0.7568
Run 5	**1.0000**	0.6686	**0.4886**	**0.8145**	**0.8655**	0.7217	**0.7809**
Run 6	**1.0000**	**0.6914**	**0.4886**	0.8103	0.8604	**0.7230**	0.7795
Run 7	**1.0000**	0.6686	**0.4829**	0.8100	0.8621	0.7153	0.7736
Run 8	**1.0000**	**0.6857**	**0.4829**	**0.8108**	**0.8629**	**0.7217**	**0.7800**
Run 9	**1.0000**	**0.6857**	**0.4829**	0.8071	0.8593	0.7194	0.7777

In the experiments performed on the PIKES LSA Model our results only outperform KE4IR on all measures in Run 6. This indicates that query coordination and automatic relevance feedback individually are insufficient to outperform KE4IR, though query coordination on its own (Run 4) beats our baseline (Run 1) on all measures. Run 6 combines them, uses the top-2 documents for automatic relevance feedback and outperforms KE4IR on all measures.

When we use the PIKES VSM Model together with query coordination and relevance feedback but without query expansion (Run 7), we outperform KE4IR on most but not all measures. When we add automatic query expansion using Wiki4TSM (Runs 8 and 9) we outperform KE4IR on all measures. However, Run 9 shows that performing query coordination on the related terms as well harms ranking accuracy and does not improve precision compared to excluding the related terms from the query coordination step (Run 8). Indeed, it may even counteract the benefits of performing query expansion at all, as the ranking accuracy results (NDCG and NDCG@10) are worse than those achieved by Run 7 (PIKES VSM Model without query expansion)[11].

5 Discussion

We have shown that generating an LSA Model of a document collection or using an independently generated Term Similarity Matrix to automatically expand a query with related terms for use with a Vector Space representation of the same collection, both coupled with query coordination and *automatic* relevance feedback (Subsect. 3.5), outperform current approaches that perform Named Entity Linking and relationship representation to support semantic search on a small document collection. Our best performing runs on the PIKES LSA Model (Runs 5 and 6) differ only on whether the top-1 or top-2 documents in the results list are used for automatic relevance feedback to re-rank the results. However, Run 5 fares badly on the P@5 measure, failing to outperform KE4IR and Textual, and beating only Run 2 (Run 2 is identical to Run 5 except that it does not perform query coordination).

The main difference between the PIKES LSA Model and PIKES VSM Model approaches is that in the former related terms are automatically discovered from within the collection itself through the process of decomposition of the VSM Model resulting in an implicit representation of the degree of relatedness between terms. Using the PIKES VSM Model with query coordination and automatic relevance feedback is usually, but not always, sufficient to outperform prior work (compare Run 7 to KE4IR's results in Table 2), but additionally performing query expansion to include in the query vector related terms retrieved from an independently generated Term Similarity Matrix always outperforms prior work (Runs 8 and 9 compared to KE4IR's results).

[11] https://www.elastic.co/guide/en/elasticsearch/guide/current/practical-scoring-function.html explains that query coordination may not be effective when the query contains synonyms.

6 Conclusions

Small document collections generally contain documents that may be semantically relevant to a query but that may contain terms that are different from those expressed in the user query. Although this is also likely in large document collections, it is generally less of a problem unless the users needs to obtain all, rather than just some, relevant documents. In small collections, the vocabulary gap may result in too few or no relevant documents being retrieved.

Two approaches to indexing and retrieving textual documents were presented. Both yield greater accuracy than prior work on the same test collection. Stop words were removed from the collection and the remaining unigrams were stemmed and indexed as a Vector Space term-by-document matrix (PIKES VSM Model). In the first approach, related terms in the PIKES collection are automatically discovered and represented using LSA (PIKES LSA Model). In the second approach, the query is automatically expanded with related terms from the Term Similarity Matrix (derived from an LSA model of a Wikipedia dump). An adaptation of Lucene's Practical Scoring Function is used to retrieve ranked documents from the PIKES VSM Model, using query coordination to reward documents based on the percentage of query terms they contain. *Automatic* relevance feedback is used to re-rank documents in the results list.

In the PIKES LSA Model approach, the 'latent' semantics exposed in the collection is highly dependent on the documents in the collection. In the second approach, an LSA Model generated from Wikipedia is used to build a collection-independent Term Similarity Matrix that is then used to identify and expand the query with related terms. Both approaches outperform prior work when evaluated using the same PIKES test collection. Prior work had identified mentioned named entities, linked them to their open data, and identified and explicitly represented semantic relationships between named entities mentioned in the same document. Corcoglioniti *et al.* observed that their approach is not scalable [4]. Our first approach, deriving an LSA model of the text collection directly, may also prove difficult to scale given a large enough text collection (this is a known limitation of LSA). However, in our second approach, we utilised an existing, independently derived LSA model to perform automatic query expansion, which, although not as accurate as our first approach, still outperforms prior work.

References

1. Azzopardi, J., Staff, C.: Fusion of news reports using surface-based methods. In: WAINA 2012: Proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops, pp. 809–814. IEEE Computer Society, Los Alamitos (2012)
2. Azzopardi, J., Staff, C.: Incremental clustering of news reports. Algorithms **5**(3), 364–378 (2012)
3. Carpineto, C., Romano, G.: A survey of automatic query expansion in information retrieval. ACM Comput. Surv. **44**(1), 1:1–1:50 (2012)

4. Corcoglioniti, F., Dragoni, M., Rospocher, M., Aprosio, A.P.: Knowledge extraction for information retrieval. In: Sack, H., Blomqvist, E., d'Aquin, M., Ghidini, C., Ponzetto, S.P., Lange, C. (eds.) ESWC 2016. LNCS, vol. 9678, pp. 317–333. Springer, Heidelberg (2016). doi:10.1007/978-3-319-34129-3_20
5. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. **41**(6), 391–407 (1990)
6. Dumais, S.: Improving the retrieval of information from external sources. Behav. Res. Methods Instrum. Comput. **23**(2), 229–236 (1991)
7. Huston, S., Bruce Croft, W.: Evaluating verbose query processing techniques. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, pp. 291–298. ACM, New York (2010)
8. Jorge-Botana, G., Olmos, R., Barroso, A.: The construction-integration framework: a means to diminish bias in LSA-based call routing. I. J. Speech Technol. **15**(2), 151–164 (2012)
9. 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, SIGIR 1998, pp. 206–214. ACM, New York (1998)
10. Navigli, R., Vannella, D.: SemEval-2013 task 11: word sense induction and disambiguation within an end-user application. In: Second Joint Conference on Lexical and Computational Semantics (*SEM), Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), vol. 2, pp. 193–201. Association for Computational Linguistics, Atlanta, June 2013
11. Porter, M.F.: An algorithm for suffix stripping. Program **14**(3), 130–137 (1980)
12. Rocchio, J.J.: Relevance feedback in information retrieval. In: Salton, G. (ed.) The SMART Retrieval System: Experiments in Automatic Document Processing, pp. 313–323. Prentice-Hall, Englewood Cliffs (1971)
13. Spoerri, A.: How visual query tools can support users searching the internet. In: 2014 18th International Conference on Information Visualisation, pp. 329–334 (2004)
14. Staff, C., Azzopardi, J., Layfield, C., Mercieca, D.: Search results clustering without external resources. In: Spies, M., Wagner, R.R., Min Tjoa, A. (eds.) Proceedings of the 26th International Workshop on Database and Expert Systems Applications DEXA 2015, Valencia, Spain, 1–4 September 2015, pp. 276–280 (2015)
15. Stanković, R., Krstev, C., Obradović, I., Kitanović, O.: Indexing of textual databases based on lexical resources: a case study for Serbian. In: Cardoso, J., Guerra, F., Houben, G.-J., Pinto, A.M., Velegrakis, Y. (eds.) KEYSTONE 2015. LNCS, vol. 9398, pp. 167–181. Springer, Heidelberg (2015). doi:10.1007/978-3-319-27932-9_15
16. Stefanescu, D., Banjade, R., Rus, V.: Latent semantic analysis models on wikipedia and tasa. In: Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., Piperidis, S. (eds.) Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014). European Language Resources Association (ELRA), Reykjavik, May 2014
17. Waitelonis, J., Exeler, C., Sack, H.: Linked data enabled generalized vector space model to improve document retrieval. In: Proceedings of NLP and DBpedia 2015 Workshop in Conjunction with 14th International Semantic Web Conference (ISWC 2015), CEUR Workshop Proceedings (2015)
18. Yao, J., Cui, B., Hua, L., Huang, Y.: Keyword query reformulation on structured data. In: 2012 IEEE 28th International Conference on Data Engineering (ICDE), pp. 953–964. IEEE (2012)