Inf Retrieval (2006) 9: 95–109
DOI 10.1007/s10791-005-5723-3

Automatic search from streaming data

Anni R. Coden · Eric W. Brown

Received: January 9, 2003 / Revised: December 23, 2004 / Accepted: January 12, 2005

Abstract Streaming data poses a variety of new and interesting challenges for information retrieval and text analysis. Unlike static document collections, which are typically analyzed and indexed off-line to support ad-hoc queries, streaming data often must be analyzed on the fly and acted on as the data passes through the analysis system. Speech is one example of streaming data that is a challenge to exploit, yet has significant potential to provide value in a knowledge management system. We are specifically interested in techniques that analyze streaming data and automatically find *collateral information*, or information that clarifies, expands, and generally enhances the value of the streaming data. We present a system that analyzes a data stream and automatically finds documents related to the current topic of discussion in the data stream. Experimental results show that the system generates result lists with an average precision at 10 hits of better than 60%. We also present a hit-list re-ranking technique based on named entity analysis and automatic text categorization that can improve the search results by 6%–12%.

Keywords Speech retrieval · Text mining · Information retrieval

1. Introduction

Fast processors, huge storage capacity, and sophisticated software, all packed into a small form factor, are driving information retrieval, text analysis, and knowledge management in a variety of new directions. One direction of interest is the application of information retrieval and text analysis techniques to on-line streaming data. In this scenario, the data has a "timeliness" aspect to it where the value of the data to the end user diminishes rapidly over time. To provide maximum value from the data, the analysis system must analyze the streaming data as it arrives and generate results in real time.

A.R. Coden · E.W. Brown
IBM, T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532
e-mail: {anni, ewb}@us.ibm.com

One example of this kind of streaming data is speech. Automatic speech recognition (ASR) software has matured to the point where it can transcribe speech into text with sufficient accuracy to allow indexing, search, and retrieval of spoken documents (Garofolo et al. 1998), and automatic processing and analysis of radio and television news broadcasts (DARPA 1998). While there is certainly value in treating speech like a traditional document collection, we are interested in analyzing speech on the fly to enhance the value of that speech as it is generated. Specifically, we are interested in techniques that find *collateral information*, or information that is related to the current data stream. Collateral information enhances the value of the original data by providing additional context, answering questions raised by the original data, or providing more details.

The ability to automatically analyze speech and accurately identify collateral information enables a number of interesting applications. One such application is the automatic analysis and support of meetings. This is the goal of the MeetingMiner system (Brown et al. 2001), which captures meeting discussions via microphone, converts the speech to text with automatic speech recognition, then applies a series of analyzers to the text to track the topic, automatically answer questions, and automatically find information related to the current discussion.

Another application made possible by automatic analysis of speech is Data Broadcasting (Coden and Brown 2001). Data Broadcasting is the process of using the extra bandwidth in a television broadcast to send arbitrary data along with the audio and video program. A key challenge in Data Broadcasting is deciding what data to send. Although the data can be assembled and scheduled manually, a more useful approach is to analyze the television program and automatically identify relevant data to send with the program. Using the same underlying approach as in the MeetingMiner system, we use automatic speech recognition to convert the audio program to text, analyze the text to identify the current topic of discussion in the program, and automatically find related information to send down the data channel.

To support the task of finding collateral informaiton (and the two applications just mentioned) we have developed a core system that takes speech audio as input and produces a list of relevant information sources as output. The system uses continuous automatic speech recognition to transcribe the speech, analyzes the resulting text transcript to decide when to search for related information, automatically generates a query when a search is triggered, and assembles the search results for output to the controlling application.

Although similar to the traditional information retrieval search task, our problem differs in a number of key ways. First, rather than process queries explicitly posed by a user, we must automatically generate queries by analyzing a data input stream, e.g., the transcript of the discussion. Second, since the transcript is generated by ASR it may contain errors. Our solution, therefore, must be robust in the face of word recognition errors. Finally, the goal of this system is to provide information relevant to the current discussion such that the information can be quickly integrated into the end user task (whether it be a meeting, viewing a television program, or some other yet to be conceived application). In this scenario, a small number of highly relevant results is much more important than finding all possible results. As such, our system (and our evaluation) focuses on precision and ignores recall.

Our contributions are as follows. First, we have introduced a new application area of information retrieval and knowledge management where speech data is captured and exploited as it is generated. Second, we present a system that accomplishes the capture and analysis task using new techniques to automatically generate queries and process search results. Moreover, our search result processing includes techniques for re-ranking results based on automatic categorization and named entity extraction that are applicable to traditional text

search systems as well. Finally, we present an evaluation of our techniques using a standard evaluation collection.

The rest of this paper is organized as follows. In the next section, we discuss related work. In Section 3, we present our system in more detail. In Section 4, we present our experimental evaluation, and in Section 5, we offer concluding remarks.

2. Related work

The problem that we are trying to solve here, namely how to analyze a data stream in real time and augment the data stream with relevant, collateral information, is somewhat similar to the problems explored in the various Topic Detection and Tracking workshops (DARPA 1998). In TDT, the goal is to analyze news broadcasts (text articles or text transcripts generated automatically from audio and video) and identify previously unseen news events, or topics. Topics are then tracked by linking subsequent news stories covering the same event. This is accomplished using a variety of off-line text processing, language modeling, and machine learning algorithms.

TDT differs from our work in two ways. First, our goal is not to detect and track new events, but rather to identify information objects (e.g., documents) in a knowledge repository that are relevant to the current data stream being analyzed. Rather than track a series of related events, we want to generate a short list of highly relevant objects for a given segment of the data stream.

The second way in which our work differs from TDT is that our system must operate on-line in order to augment the data stream with collateral information as the data stream is generated. The typical TDT system operates off-line.

The techniques we use to accomplish this task are drawn largely from traditional information retrieval (Baeza-Yates and Ribeiro-Neto 1999) and text analysis techniques (Manning and Schuetze 1999). To satisfy the response time requirements of the task, however, we are limited to techniques that can be performed on-line at the same rate that the data stream is fed into the system.

A significant contribution of our work is our hit-list re-ranking procedure based on named entity extraction and automatic text categorization. This procedure is required for two reasons. First, the ASR generated text transcript may contain word recognition errors. Second, the queries are automatically generated from a full text transcript and are generally less precise than queries created by real users. Both of these factors can cause problems for traditional text search engines, which are tuned for short, user-generated queries without word recognition errors.

Our re-ranking procedure has similarities to previous work on incorporating natural language processing and statistical and syntactic phrases into the document retrieval process. Strzalkowski et al. have demonstrated improvements in retrieval effectiveness using natural language processing techniques (Strzalkowski et al. 1998, 1999, 2000). Their work explores the use of full linguistic analysis to identify higher level conceptual indexing terms combined with query expansion techniques to incorporate the concepts into the query. Their results have been mixed, but they generally found that their techniques are more effective with longer queries. Our natural language processing is confined to named entity identification where the named entities are used to refine the results generated from a bag of words query. Our technique appears to be more robust and consistently provides an improvement. Our original queries are rather long, however, increasing the likelihood that our technique will be effective.

Chowdhury et al. (2001) use entities to improve precision as part of their automatic relevance feedback technique. They extract entities from the top documents returned by an initial query, add them to the query, and run the query again. In our system, we use only the named entities that appear in the original query stream.

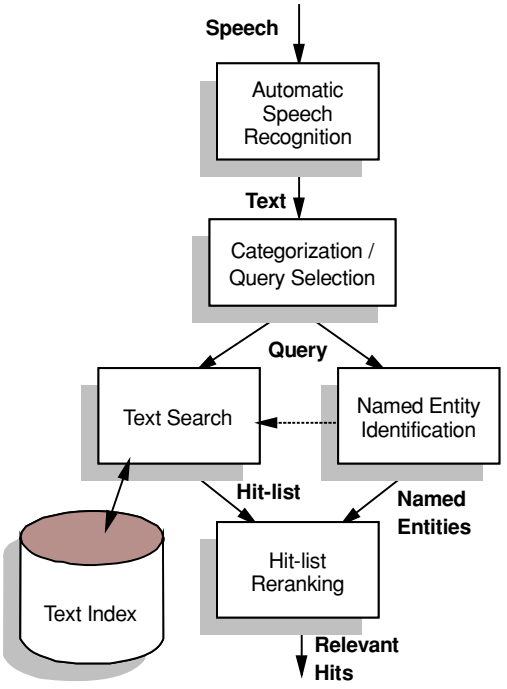
A number of authors have found only marginal improvements in retrieval effectiveness using statistical and syntactic phrases (Mitra et al. 1997, Turpin and Moffat 1999). Rather than try to automatically identify phrases and incorporate them as terms in the document retrieval model, we identify named entities and use them to adjust the results returned by the search engine. Our more accurate phrase (entity) identification combined with a more conservative application of the information yields a more robust, consistently beneficial technique.

3. System description

The overall system architecture is depicted in Figure 1. The system receives a speech audio signal as input and sends it to an automatic speech recognition system, which in turn generates a text transcript of the speech. In an actual application of this system, the ASR component must be able to process an audio signal in real-time, apply a continuous dictation speech recognition model, and generate the transcript. For the work presented in this paper, we assume that such an ASR system exists and do not consider it further.

The system feeds the text transcript one sentence at a time into the query selector, which generates queries using an automatic text categorization system. The query selector appends the current sentence to a context buffer then runs the categorizer on the context buffer. If the categorizer can categorize the context buffer with sufficiently high confidence, the query selector outputs the buffer as a query and clears the buffer for the next sentence. The query

Fig. 1 System architecture



selector will append at most seven sentences to the context buffer, at which time the selector generates a query regardless of the categorization result. The categorizer used in the query selector is a rule-induced categorizer (Apte and Damerau 1994, Johnson et al. 2002), which must be trained with a predefined taxonomy of categories and a training set of documents that have been previously categorized into the taxonomy.

The taxonomy used for the evaluation below is a set of 38 broadcast news related categories (e.g., "Domestic Politics", "Business", "Crime", etc.) trained with approximately 1200 manually categorized documents from the January TDT-2 documents (see Section 4 below).

The system sends each generated query to the text search engine, which performs a free text search on the text index and returns a hit-list of relevant documents. Any free text search engine may be used for this component. For our prototype system and for the experiments presented below we use two different search engines. The first search engine is an n-gram based, tf*idf style ranking search engine called NGRAM. The second search engine is a probabilistic ranking search engine called GURU (Brown and Chong 1998).

The system simultaneously performs named entity identification on each query using algorithms based on the Talent suite of text analysis tools (Cooper and Byrd 1997, Ravin et al. 1997). The analysis identifies proper names, places, and certain technical terms in our on-line system. The named entities could be used directly by the search engine and factored into its internal ranking algorithm. In our current implementation, they are used by the hit-list re-ranking process to refine the hit-list returned by the search engine. This post-processing approach both facilitates experimentation and enables integration with existing search engines. Ideally, the re-ranking process would be more integrated into the core search engine ranking algorithm to provide better execution performance.

The Talent algorithms rely heavily on clues provided by capitalization. If the speech recognition engine does not produce properly capitalized text, we apply an automatic capitalization recovery process to the text (Brown and Coden 2002), which uses a number of syntactic rules, punctuation, and statistical analysis to restore proper capitalization to the text.

For many search engines, the rank associated with a document in the hit-list is a function of the frequencies of the words of the query, the document and the collection as a whole. Otherwise, all words in the query (with the exception of stop words) are treated equally. However, when we examined the application scenarios discussed in this paper, it became apparent that other factors should also play a role in ranking the resulting documents. We postulated:

(1) If a result document has the same category associated with it as the query, the ranking of such a document should be increased.
(2) If a result document contains all of the same named entities as the query, it should be ranked higher than a document that has only some or none of the named entities.

For each document on the hit-list, the re-ranking process works by considering the relevance score returned by the search engine, the number of matching named entities between the query and the document, and whether or not the query and the document share the same category. A new relevance score for the document is calculated using the following formula:

$$S' = aS + b\frac{E_d}{E_q} + cC \tag{1}$$

Table 1 Constants for Eq. (1)

S'	New score for the document
S	Original non-zero score of the document
E_d	Number of distinct named entities in the document that match those in the query
E_q	Number of distinct named entities in the query
C	1 if document and query have same category, 0 otherwise
a, b, c	Constants in the range 0 to 1 inclusive such that $a + b + c = 1$

Note that our hit-list re-ranking procedure assumes that the documents in the text search index have been processed by the same automatic text categorizer and named entity identifier that we apply to the queries.

After a new score for each document is calculated, the hit-list re-ranking component sorts the hit-list according to the new scores and returns the hit-list to the calling application. The constants a, b, and c determine the influence of each component of the re-ranking formula on the final score. We will consider appropriate values for these constants in Section 4.

4. Experimental results

4.1. Goals

The system evaluated here is one that automatically retrieves collateral information based on automatic query generation from a text stream. The different components of the system and their interactions are evaluated as well as the variability of the results based on the quality of the text stream and the underlying search engine. We show that the re-ranking algorithm always gives an improvement in the precision of the results independent of the search engine and the quality of the text stream. Hence, it could also be applied to a general search task.

In particular, the following questions are studied and the evaluation results are presented in subsequent sections.

(1) What is the effect of the query generation process on the precision of the results?
(2) What is the effect of the underlying search engine?
(3) What is the effect of the re-ranking algorithm?
(4) What is the effect of the quality of the text-stream?
(5) What is the effect of categorization?

We will show that the precision can be improved using categorization, named entity identification and the re-ranking results.

4.2. Corpus

To evaluate our system we need a document collection, a set of queries (preferably drawn from a speech data stream), and relevance judgments for the queries and the document collection. The TDT-2 corpus (Cieri et al., 1999) meets all of these requirements. The TDT-2 corpus is a collection of text and speech from several sources: newswire, radio and television news broadcast programs. It contains approximately 60,000 stories. These stories were transcribed in three different ways: (1) Manual transcription (2) ASR (Automatic Speech Recognition)

Table 2 Types of query sources

All	subset containing no ASR documents (2104 documents)
ASR	subset containing only ASR transcribed documents (1170 documents)
ASRTr	manually transcribed versions of documents in ASR (1170 documents)

Table 3 Query sets

Query type	Number (av. length)	Categorized	Uncategorized
AllW	2104 (375)	837 (40%)	1267 (60%)
ASRW	1170 (202)	241 (20%)	929 (80%)
ASRTrW	1170 (193)	384 (33%)	786 (67%)
AllP	8213 (102)	1719 (21%)	6494 (79%)
ASRP	3899 (62)	261 (7%)	3638 (93%)
ASRTrP	3078 (78)	708 (23%)	2370 (77%)

transcription (3) Closed Caption transcription. The corpus also includes 96 topics for which every document in the collection has been manually judged as relevant or not relevant.

We created our text collection (referred to below as the "training data") using the "sgm" versions from all English data sources dated January to April inclusive (i.e., newswire text and manual transcriptions of broadcast sources). The queries are drawn from English documents dated May and June that were assigned to one or more qualifying TDT-2 topics, where a qualifying topic is one where at least 10 documents in the training data were assigned to the topic. Although the TDT-2 corpus includes 96 fully evaluated topics, only 46 topics have more than 10 training documents in our split of the corpus. Three separate query sources were created for our experiments (see Table 2).

These three different query sources enabled us to study the effect of ASR on the search task as outlined at the beginning of this section.

The ASR collection contains only documents that are mono-case. However, for one part of our system (named entity identification) capitalization is required. We used a system (Brown and Coden 2002) that automatically capitalizes documents and applied it to the ASR query set. This part of the system could be easily replaced with another subsystem not requiring capitalization. The results of the evaluation would not change with such a substitution.

We define a document to be relevant collateral information for a particular query if and only if the query and the document have the same TDT topic. The topic of a query is the topic of the document from which the query is drawn.

Based on the three different query sources we generated six different query sets. Three of the query sets (labeled with the "W" suffix) contain whole documents as queries (modeling the case where query boundaries are explicitly given), and three of the query sets (labeled with the "P" suffix) contain automatically generated queries from parts of a document as described in Section 3. The generation of the "partial" queries is done in real-time as is the annotation of queries. The annotations are the category of a query and the named entities that occur in it. The query sets shown in Table 3.

A categorized query is one that could be categorized with sufficient confidence by our automatic text categorizer. The number in parenthesis after the number of queries is the average query length in words. The percentages in the second and third columns indicate the percentage of categorized and uncategorized queries.

Each story in the corpus itself (i.e., the data which is searched) is also annotated by the categorizer and the named entity identifier. The corpus has 40,932 documents, 14,537 (35%)

which have a category attributed to them, 26,395 (65%) of which were not categorized. We will explore the effects of categorization in more depth in the next section.

4.3. Evaluation

The first set of results presented here establishes the baseline for our system. All results presented below show precision at *n* hits. Since our target application is to augment a data stream with collateral information, we are emphasizing precision. The goal is to find relevant collateral information for a query. We will show that our system can produce results that have precision of 40%–65% in the top 10 hits. These results can be achieved with different search engines and different query sets as shown in Figures 2 and 3. Note that the precision axis is scaled to show the area of interest.

The baseline performance was established using two different search engines. The datasets G (labels ending with "G") used the IBM NGRAM Search Engine, whereas the datasets S (labels ending with "S") used the GURU search engine. In general, the GURU search engine performs better, however both search engines perform similarly with respect to the different query sets. The manually transcribed queries perform the best, while the queries based on output of an ASR system perform the worst, with the reduction in precision, or spread, being 5% in most cases. The results for the *All* set differ only slightly from the results for the ASR set, which seems to imply that ASR systems are quite adequate for the task under evaluation. Results in Figure 2 are based on whole document queries while the results shown in Figure 3 are based on automatically generated queries.

The spread in the precision for auto-generated queries is wider as shown in Figure 3. The GURU search engine again performs better than the NGRAM search engine. However, for both search engines the relative performance of the different query sets is the same

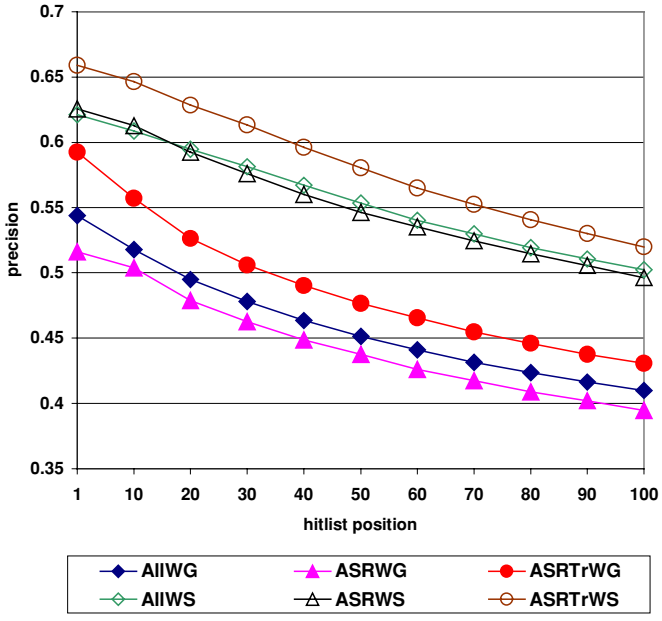


Fig. 2 Baseline performance whole document queries

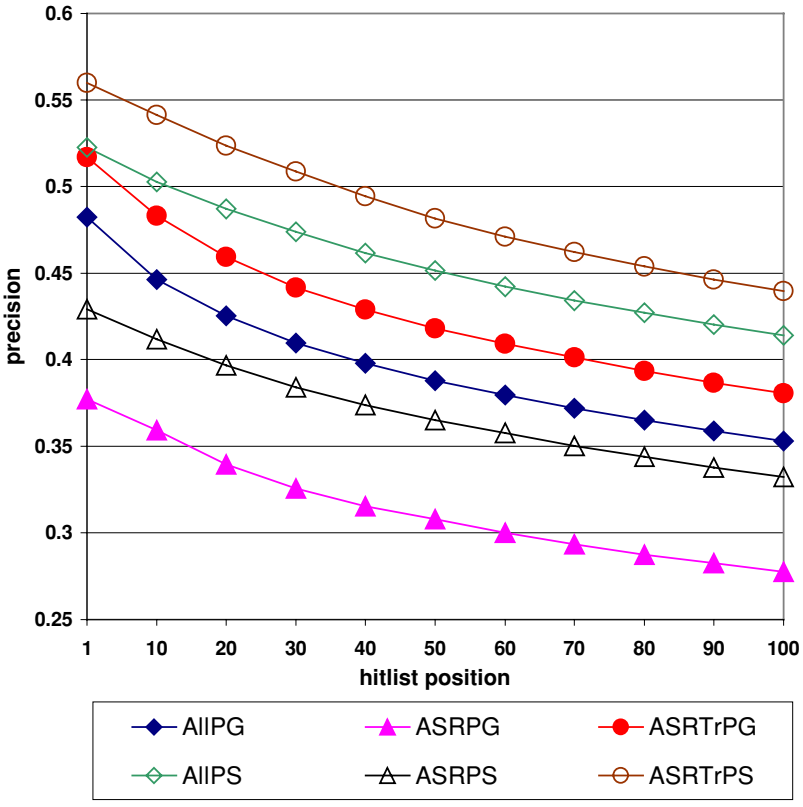


Fig. 3 Baseline performance auto-generated queries

and consistent with the whole document queries. The ASRTr query set achieves the highest precision, while the ASR query set has the lowest precision. The spread in precision between the results based on the ASR and the *All* query set is roughly 5% for whole document queries and closer to 10% for automatically generated queries. The same spread is observed between the *All* and the ASRTr datasets.

The second set of results examines the effects of categorization. Again, we look at how the results vary as a function of the various query sets and search engines. Figure 4 shows the performance of queries that could be categorized compared with the performance of queries that could not be categorized (search is performed by the NGRAM search engine). Queries that can be categorized lead to substantially better results than queries that cannot. In particular, precision improves between 16% and 60%. When examining auto-generated queries, similar results are observed as shown in Figure 5.

Similar improvements (not shown here) are seen when using the GURU search engine instead of the NGRAM engine and performing the same experiments. It is noteworthy that in general the precision is best for the ASRTr query set and the worst for the ASR query set.

These results suggest that a carefully crafted and trained taxonomy can lead to improved results by supporting a categorization-screening step to indicate the likelihood of success for the query. Although one cannot guarantee that a query is categorized, one can attempt to augment a query in those instances. The augmentation can be done either automatically or through user interaction.

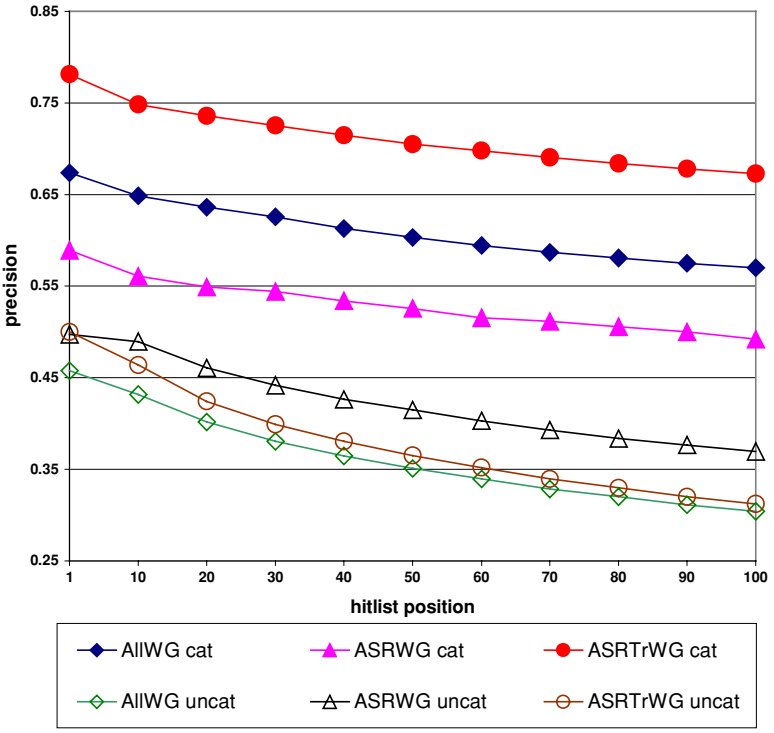


Fig. 4 Categorized vs. uncategorized whole document queries

So far, we have shown that our baseline system produces an average precision at 10 hits of around 52%. The performance of the base system does not change much with the varying query sets, the spread being between 5% and 10% on average. Furthermore, when the system is able to categorize the query, precision at 10 improves by 10% on average, regardless of the underlying query set and search engine. The length of the query (whole document vs. auto-generated query) does not have a major influence on the results.

4.4. Re-ranking algorithm

Our re-ranking algorithm was introduced in Section 3, where the formula for computing the rank of a document is presented. In this section, the performance results are presented. In Section 3, we suggested that a new score for a document should be computed based on the original score, the overlap of named entities between the query and the document, and the categorization of the query and the document.

The following study shows that indeed an improvement of up to 14% in precision at 10 hits can be achieved considering all three factors. Furthermore, we show that an improvement is achieved for all query sets, independent of search engine and query length. Recall that the new rank of a hit-list document is computed using the following formula:

$$S' = aS + b\frac{E_d}{E_q} + cC$$

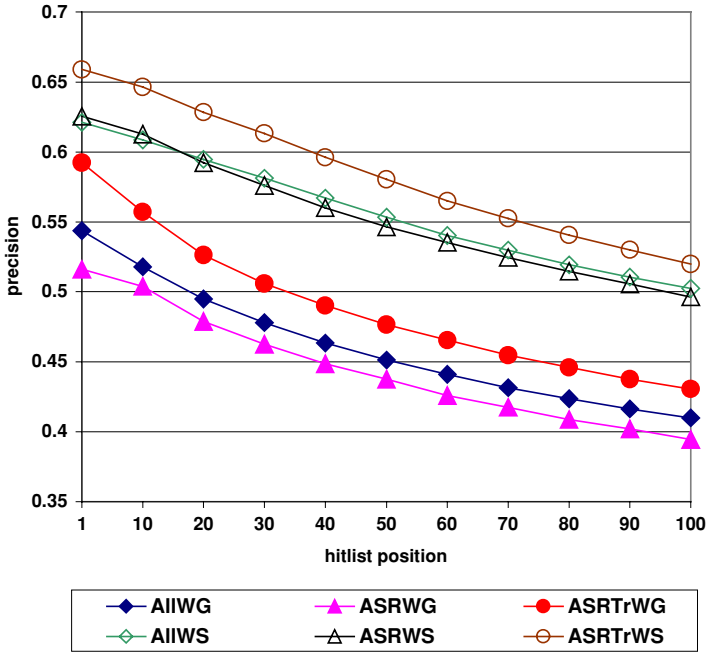


Fig. 5 Categorized vs. uncategorized auto-generated queries

We experimented with different values of a, b and c and the results do not differ substantially with the choice for these constants. However, we determined empirically, that the following sets of constants give the best results:

(1) $(a, b, c) = (0.3, 0.4, 0.3)$
(2) $(a, b, c) = (0.1, 0.9, 0.0)$

The constants indicate that queries with many named entities can rely to a substantial degree (even solely) on the named entities in the re-ranking process. However, in the presence of a good taxonomy and a small number of named entities, the first set of constants may be more advisable. The experiments presented here consider the first 100 returned hits for the re-ranking process. Our goal is to improve the precision in the top 10 results.

The experimental results shown in Figure 6 examine the improvements in precision for all the query sets. The improvement (at 10 hits) ranges from 5% to 12%, with the smallest improvement shown for the ASRTr dataset and the largest improvement seen for the ASR query set. Here the NGRAM engine performs the basic search using whole document queries. The first set of constants is used for the re-ranking.

We also examined the same data sets using the GURU search engine instead of the NGRAM search and engine and used the second set of constants for the re-ranking (detailed results not shown here). The precision at 10 hits improved between 3% and 4.6% over the original results.

Next, we examine how much the re-ranking algorithm improves the precision if auto-generated queries form the input. The results shown in Figure 7 use the NGRAM search engine and the first set of constants for the re-ranking process. The improvements in precision at 10 hits range from 7.6% to 10% for all the query sets. The improvements are in the range

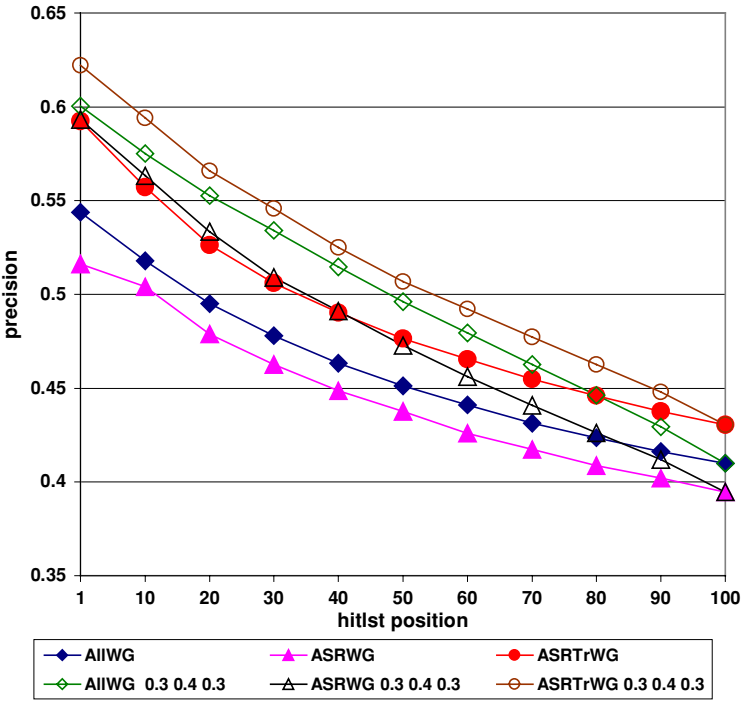


Fig. 6 Re-ranking whole document queries

from 4% to 6% when substituting the GURU search engine for the NGRAM engine (not shown here).

The next figure (Figure 8) examines whether the re-ranking algorithm improves the precision at 10 when only categorized queries are considered. Here the improvements range from 5% to 14%, with the largest improvements obtained using the ASR query set. For uncategorized queries, improvements in the range of 7%–11% can be achieved using the NGRAM search engine and the first set of constants for the re-ranking.

In Table 4 we summarize some of the results of our experiments. They reveal a strong correlation between the ability of the automatic text categorizer to categorize a query (i.e., assign the query to a category with sufficiently high confidence) and the quality of the result returned by the text search engine. The following table shows the precision at 10 hits, where a query is a whole document.

The results are similar for automatically generated queries as shown in the next table.

For example, average precision at 10 hits for uncategorized partial queries is 37%, while average precision at 10 hits for categorized partial queries is 72%. This result suggests that

Table 4 Precision at 10 for whole document queries

	Baseline (%)	Categorized (%)	Uncategorized (%)
AllW	52	65	43
ASRW	50	56	49
ASRTrW	55	75	46

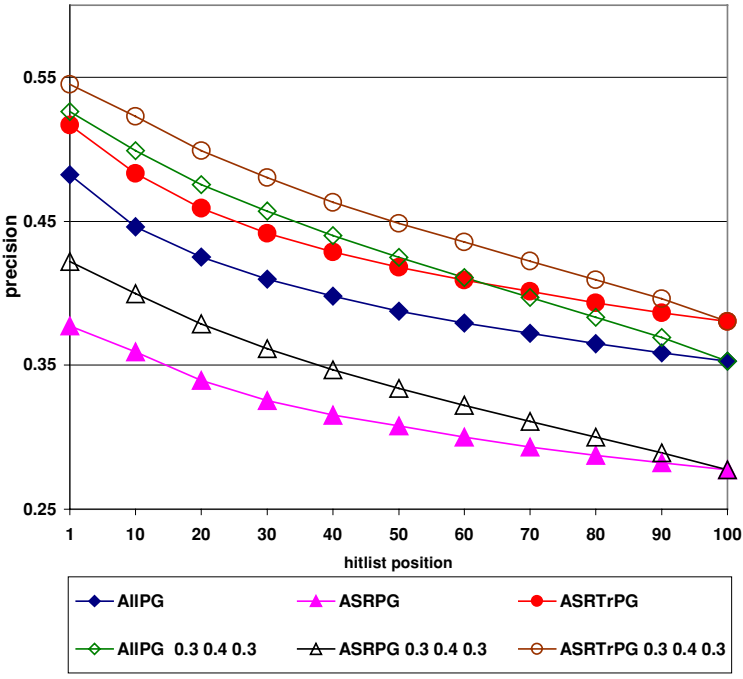


Fig. 7 Re-ranking auto-generated queries

an automatic text categorizer might be used to evaluate the "quality" of a query before it is even submitted to a search engine. Queries that cannot be categorized might go through additional processing, e.g., iteration with the user, automatic query expansion, automatic relevance feedback, etc.

The re-ranking algorithm improves the precision at 10 hits even further. Table 6 shows the precision at 10 hits where the queries are whole documents. Re-ranking improved the results between 5% and 10%.

Table 5 Precision at 10 categorized vs. uncategorized queries

	Baseline (%)	Categorized (%)	Uncategorized (%)
AllP	45	72	37
ASRP	36	60	34
ASRTrP	48	79	39

Table 6 Re-ranking results for precision at 10

	Categ. (%)	Re-rank categ. (%)	Uncateg. (%)	Re-rank uncateg. (%)
AllW	65	72	43	49
ASRW	56	66	49	54
ASRTrW	75	80	46	50

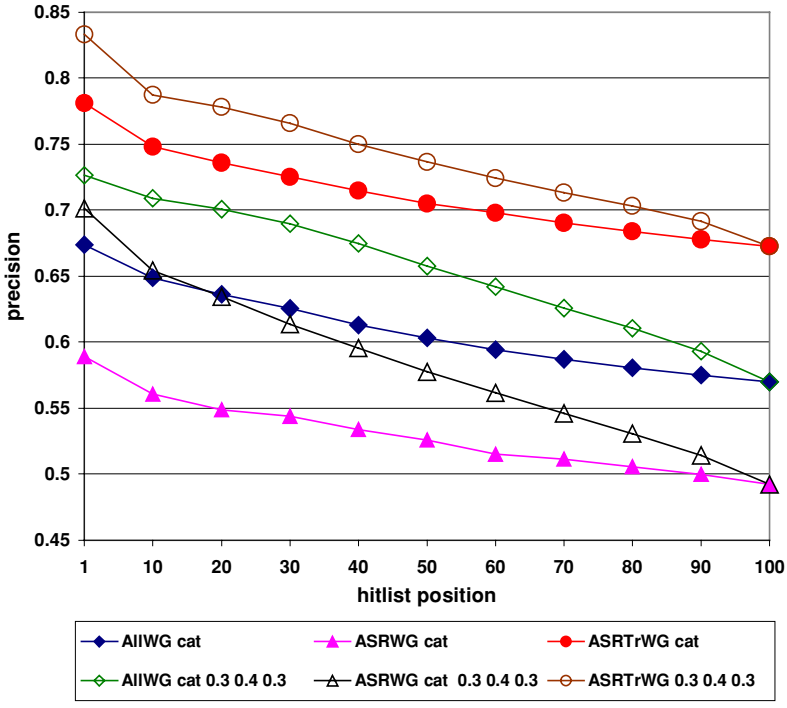


Fig. 8 Re-ranking categorized whole document queries

5. Conclusions

Streaming data is gradually becoming more important as a source of information in knowledge management and text search systems. Speech is a prime example of streaming data that has tremendous potential for enhancing knowledge management systems. Previously most of the information retrieval work related to speech involved searching spoken documents or analyzing transcribed news broadcasts for topic detection and tracking. In the work presented here, we are exploring new approaches to exploiting speech as a knowledge resource. In particular, we are interested in analyzing speech as it is generated and augmenting the speech stream with additional collateral information from related knowledge sources.

We have presented a system that captures speech, transcribes the speech to text using continuous automatic speech recognition, analyzes the text transcript, and finds documents that are relevant to the current topic of discussion in the speech stream. All of this processing occurs on-line (i.e., fast enough to keep up with the speech generation rate), and the processing that occurs on the text transcript can be applied to any data stream with a textual representation.

The focus of the work presented here is the accurate identification of collateral information given a speech data stream. Our experimental results on the TDT-2 collection show that the baseline system can automatically generate queries and find relevant documents with an average precision in the top ten results of 50% to 60% depending on the underlying text search engine. We also presented an approach for re-ranking the search engine results using named entity analysis and automatic text categorization that produces an improvement of 4%-10% in precision at 10 hits, for final results of up to 65% precision at 10 hits. We

showed that the re-ranking technique provides improvement for two different text search engines, and it is robust on ASR output that contains word recognition errors. Moreover, this technique is applicable to traditional text search systems and might provide similar precision improvements. We plan to run more experiments to confirm this.

We showed that the precision at 10 hits of categorized queries is between 7% and 21% higher than of uncategorized queries. In the absence of categorization, the re-ranking based on named entities alone still shows an improvement in precision at 10 hits between 4% and 7%.

Our main goal in this work was to validate our approach to augmenting speech data with relevant information. Based on the experimental results, we believe we have achieved this goal. Much work remains in this area. In particular, we are interested in exploring the effect of storing spoken documents in the text collections being searched. For all of our experiments here, the system searched a text collection comprising written documents or manually transcribed speech. In the future, we plan to explore the effect of searching automatically transcribed documents along with written documents and manually transcribed speech.

References

Apte C and Damerau F (1994) Automated learning of decision rules for text categorization: ACM Trans. Inf. Syst., 12:233–251.

Baeza-Yates R and Ribeiro-Neto B (1999) Modern information retrieval. Addison Wesley, New York.

Brown EW and Chong HA (1998) The guru system in TREC-6. In: Proceedings of the Sixth Text REtrieval Conference (TREC-6), pp. 535–540.

Brown EW and Coden AR (2002) Capitalization recovery for text. In: Coden AR, Brown EW and Srinivasan S (eds.) Information Retrieval Techniques for Speech Applications. LNCS 2273. Springer, Berlin, pp. 11–22.

Brown EW, Srinivasan S, et al. (2001) Toward speech as a knowledge resource. IBM Systems Journal, 40:985–1001.

Chowdhury A, Beitzel S, et al. (2001) IIT TREC-9 - entity based feedback with fusion. In: Proceedings of the Ninth Text REtrieval Conference (TREC 9).

Cieri C, Graff D, et al. (1999) The TDT-2 text and speech corpus. In: Proceedings of the 1999 DARPA Broadcast News Workshop.

Coden A and Brown E (2001) Speech transcript analysis for automatic search. In: Proceedings of HICSS'34.

Cooper JW and Byrd RJ (1997) Lexical navigation: Visually prompted query expansion and refinement. In: Proceedings of the ACM International Conference on Digital Libraries, pp. 237–246.

DARPA (1998) Proceedings of the DARPA broadcast news transcription and understanding workshop. In: Proceedings.

Garofolo J, Voorhees E, et al. (1998) TREC-6 1997 spoken document retrieval track overview and results. In: Proceedings of The Sixth Text REtrieval Conference (TREC-6), pp. 83–91.

Johnson DE, Oles FJ, et al. (2002) A decision-tree-based symbolic rule induction system for text categorization. IBM Systems Journal, 41:428–437.

Manning C and Schuetze H (1999) Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA.

Mitra M, Buckley C, et al. (1997) An analysis of statistical and syntactic phrases. In : Proceedings of RIAO97, Computer-Assisted Information Searching on the Internet, pp. 200–214.

Ravin Y, Wacholder N, et al. (1997) Disambiguation of names in text. In: Proceedings of the ACL Conf. on Applied Natural Language Processing, pp. 202–208.

Strzalkowski T, Lin F, et al. (1998) Natural language information retrieval TREC-6 report. In: Proceedings of the Sixth Text REtrieval Conference (TREC-6).

Strzalkowski T, Perez-Carballo J, et al. (2000) Natural language information retrieval: TREC-8 report. In: Proceedings of the Eigth Text REtrieval Conference (TREC 8).

Strzalkowski T, Stein G, et al. (1999) Natural language information retrieval: TREC-7 report. In: Proceedings of the Seventh Text REtreival Conference (TREC-7).

Turpin A and Moffat A (1999) Statistical phrases for vector-space information retrieval. In: Proceedings of the ACM Inter. Conf. on Research and Development in Information Retrieval, pp. 309–310.