RC22341 (W0202-049) February 19, 2002
Computer Science

IBM Research Report

Automatic Search from Streaming Data

Anni R. Coden, Eric W. Brown
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. Ithas been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distributionoutside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g, payment of royalties). Copies may be requested from IBM T. J. Watson Research Center , P. O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home

Automatic Search from Streaming Data

Anni R. Coden and Eric W. Brown
IBM, T.J. Watson Research Center
19 Skyline Drive, Hawthorne, NY 10532
{anni, ewb}@us.ibm.com

ABSTRACT

Speech is gradually becoming more important as a source of information in knowledge management and text search systems. We present a system that analyzes a speech data stream and automatically finds documents related to the current topic of discussion in the speech 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

1. INTRODUCTION

Fast processors, huge storage capacity, and sophisticated software, all packed into an impossibly small form factor, are driving information retrieval and knowledge management in a variety of new directions. One direction in particular is the integration of speech data into knowledge management systems. 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 [11] and automatic processing and analysis of radio and television news broadcasts [1].

Moving beyond archiving speech data and off-line processing of speech broadcasts, we are interested in exploiting speech as it is generated. This implies the ability to process and analyze speech in real-time. Our basic approach is to apply real-time speech recognition to convert speech into a text transcript, and then apply on-line text analysis techniques to the transcript to accomplish certain tasks.

The first task we consider (and the one we present in this paper) is the task of *finding related information relevant to the speech currently being analyzed*. If we can successfully perform this task, a variety of interesting and useful applications becomes feasible. One such application is the automatic analysis and support of meetings. This is the goal of the MeetingMiner system [6], which captures meeting discussions via microphone, converts the speech to text with ASR, 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 [9]. 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 ASR 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 identified above 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 [1]. 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 [3] and text analysis techniques [12]. 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 [16, 17, 15]. 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. [7] 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 [13, 18]. 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 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 com-

ponent 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.

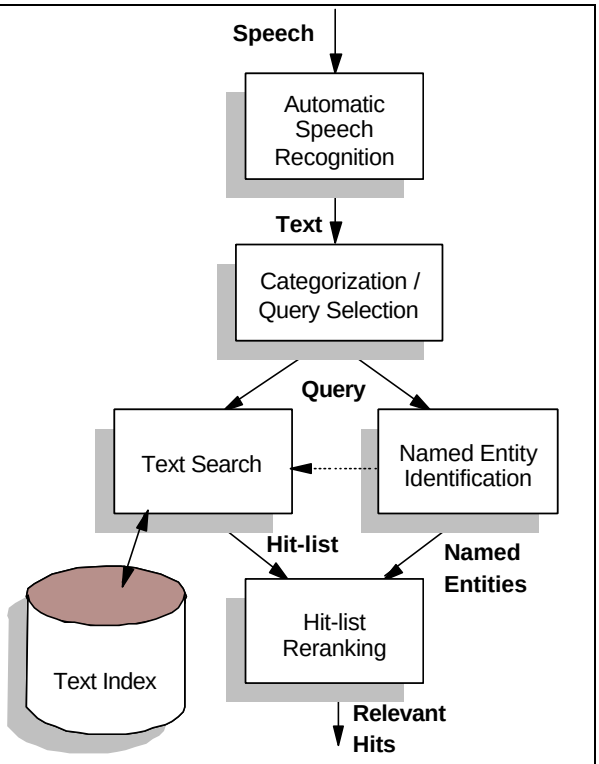

Figure 1: System architecture

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 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 [2], which must be trained with a predefined taxonomy of categories and a training set of documents that have been manually 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 TDT2 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 [5].

The system also performs named entity identification on each query using algorithms based on the Talent suite of text analysis tools [14, 10]. 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.

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 [4], 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:

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

Equation 1

where

S'	:	New score for the document
S	:	Original 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 novel 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 novel 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 judgements for the queries and the document collection. The TDT-2 corpus [8] 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) 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:

1) Q-AllDocs -- subset containing no ASR documents <2104 documents>
2) Q-ASRDocs -- subset containing only ASR transcribed documents <1170 documents>

3) Q-ASRTrans -- manually transcribed versions of the documents in Q-ASRDocs <1170 documents>

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 Q-ASRDocs 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 [4] that automatically capitalizes documents and applied it to the Q-ASRDocs 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 (labelled with the "Whole" suffix) contain whole documents as queries (modeling the case where query boundaries are explicitly given), and three of the query sets (labeled with the "Partial" suffix) contain automatically generated queries from parts of a document as described in section 3. All of the query sets are annotated in real-time. The query sets are as follows:

1) Q-AllDocsWhole – 2104 queries (375)
    a. categorized: 837 (40%)
    b. uncategorized: 1267 (60%)
2) Q-ASRDocsWhole – 1170 queries (202)
    a. categorized: 241 (20%)
    b. uncategorized: 929 (80%)
3) Q-ASRTransWhole – 1170 queries (193)
    a. categorized: 384 (33%)
    b. uncategorized: 786 (67%)
4) Q-AllDocsPartial – 8213 queries (102)
    a. categorized: 1719 (21%)
    b. uncategorized: 6494 (79%)
5) Q-ASRDocsPartial -- 3899 queries (62)
    a. categorized: 261 (7%)
    b. uncategorized: 3638 (93%)
6) Q-ASRTransPartial -- 3078 queries (78)
    a. categorized: 708 (23%)
    b. uncategorized: 2370 (77%)

A categorized query is one that could be categorized by our automatic text categorizer with sufficient confidence. The number in parenthesis after the number of queries is the average query length in words.

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 Figure 2 and Figure 3. Note that the scale of the precision axis is scaled to show the area of interest.

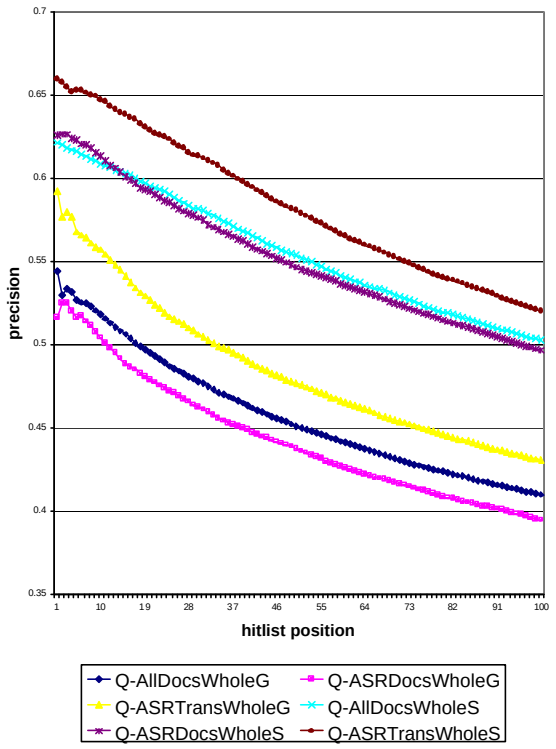

Figure 2: Baseline performance whole document queries

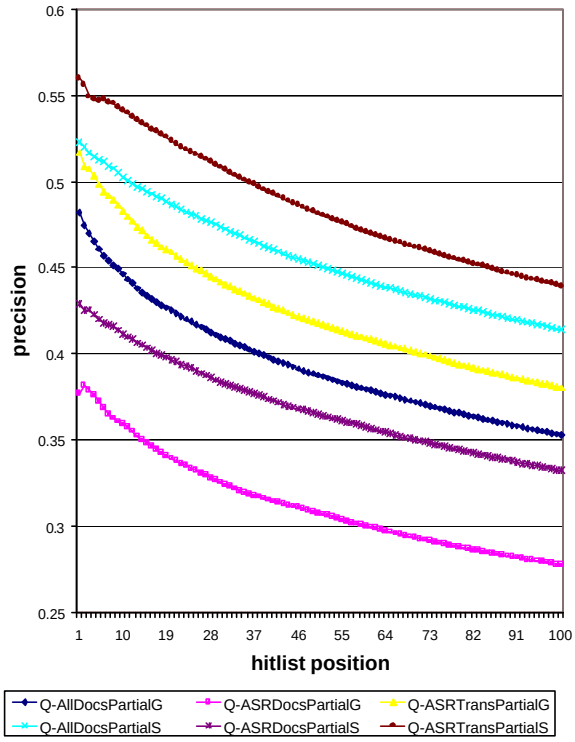

Figure 3: Baseline performance auto-generated queries

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 very similar with respect to the different query sets. The manually transcribed queries perform the best, whereas the queries based on output of an ASR system perform the worst with the spread being 5% in most cases The results for the Q-AllDocs set differ only slightly from the results for the Q-ASRDocs 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 is wider 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 and consistent with the whole document queries. The Q-ASRTrans query set achieves the highest precision, while the Q-ASRDocs query has the lowest precision. The spread in precision between the results based on the Q-ASRDocs and the Q-AllDocs query set is roughly 5% for whole document queries and closer to 10% for automatically generated queries. The same spread is observed between the Q-AllDocs and the Q-ASRTrans 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.

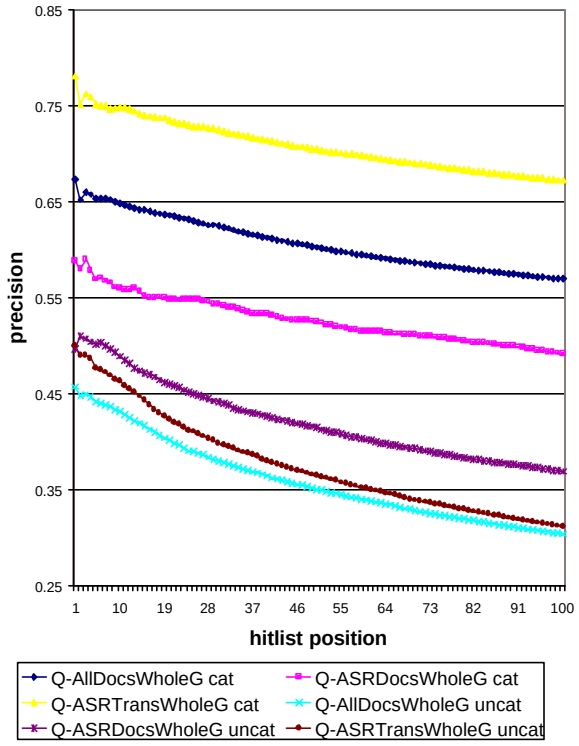

Figure 4: Categorized vs uncategorized whole document queries

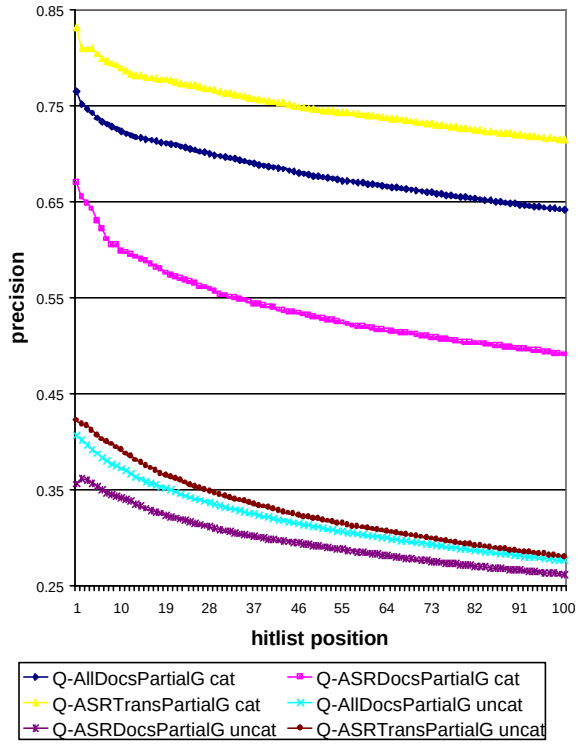

Figure 5: Categorized vs uncategorized auto-generated queries

Figure 4 shows the performance of queries that could be categorized compared with the performance of queries that could not be categorized using the NGRAM search engine. Queries that can be categorized lead to dramatically better results than queries that cannot. In particular, precision improves between 16% and 60%.

When examining auto-generated queries, very similar results are observed as shown in Figure 5.

Very similar improvements are seen when using the GURU search engine. It is noteworthy that in general the precision is best for the Q-ASRTrans query set and the worst for the Q-ASRDocs query set. Different behavior was observed for the whole document queries using the NGRAM search engine. There, the Q-ASRDocsWhole query set gave the best results, the difference in precision between the other two sets being quite small.

These results suggest strongly 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.

So far, we have shown that the baseline performance of our system is quite satisfactory. Furthermore, the performance of the base system does not change much with the varying query sets. The ability of categorizing a query improves the precision of the results dramatically, again independent 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.

The next section presents a novel re-ranking algorithm. We show that this algorithm boosts performance for all query sets.

4.4. RE-RANKING ALGORITHM

The novel 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:

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

We experimented with different values of a, b and c and the results do not differ substantially with the choice for these constants. However, empirically, it seemed 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 huge degree (even solely) on them 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 maybe 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% -12% with the smallest improvement shown for the Q-ASRTrans dataset and the biggest improvement seen for the Q-ASRDocs 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 examined the same data sets, but substituted the GURU search engine for the NGRAM search and engine and used the second set of constants for the re-ranking.

The precision at 10 hits improved between 3% and 4.6% over the original results.

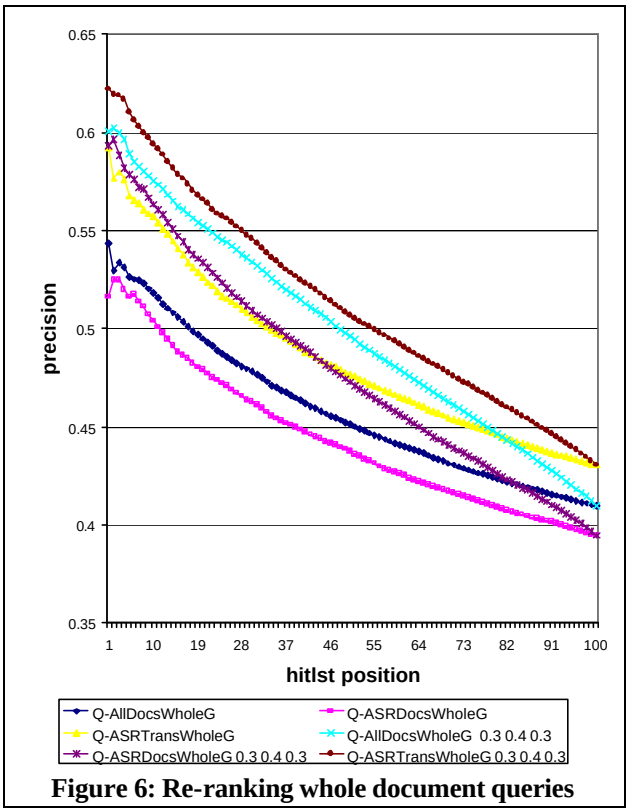

Figure 6: Re-ranking whole document queries

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.

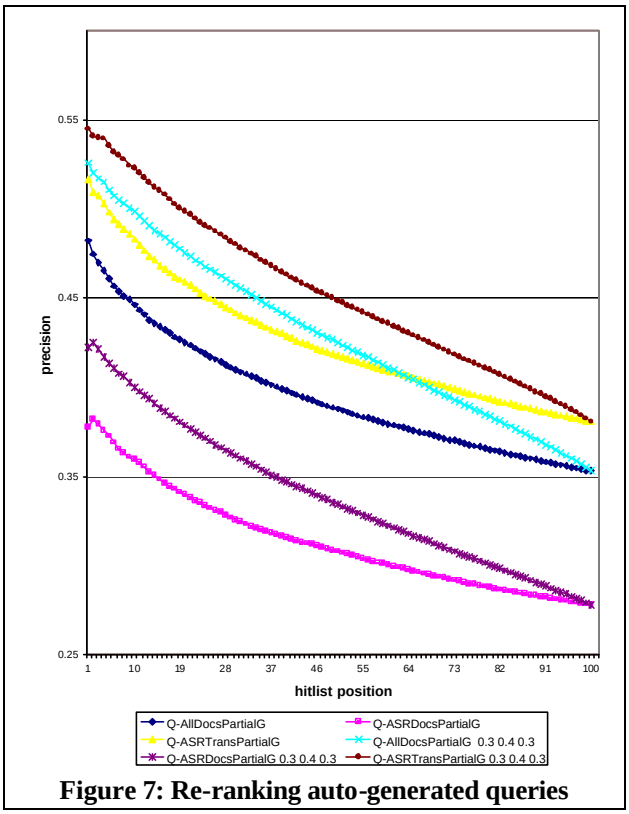

Figure 7: Re-ranking auto-generated queries

The improvements in precision at 10 hits range from 7.6% to 10% for all the query sets.

The improvements are in the range from 4% to 6% when substituting the GURU search engine for the NGRAM engine.

The next figure (Figure 8) examines whether the re-ranking algorithm improves the precision at 10 when only categorized queries are taken into account.

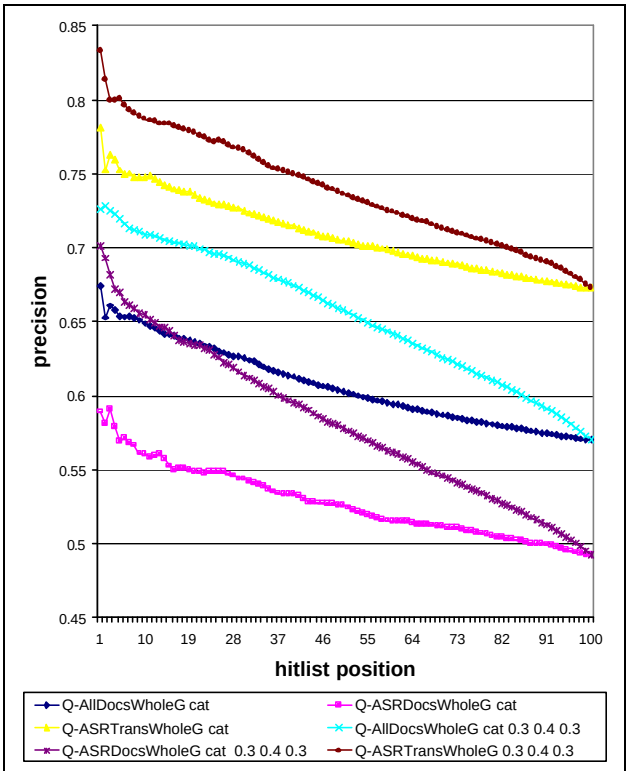

Figure 8: Re-ranking categegorized whole document queries

Here the improvements range from 5% to 14%, the biggest improvements shown for the Q-ASRDocs query set. For uncategorized queries, improvements in the range of 7% - 11% can be achieved using the NGRAM search engine using the first set of constants for the re-ranking.

All these results show that re-ranking can dramatically improve the precision of our task. There seems to be a strong indication that a common search task could also be improved using this technique. We showed, that in the absence of categorization, the re-ranking based on named entities alone still shows a big improvement in the results.

5. CONCLUSIONS

Recorded speech is gradually becoming more important as a source of information in knowledge management and text search 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.

Our experimental results on the TDT2 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.

Our experimental results also 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. For example, average precision at 10 hits for uncategorized partial queries was 38%, while average precision at 10 hits for categorized partial queries was 73%. This result suggest that an automatic text categorizer might be used to evaluate the "quality" of a query before it is even submitted to the 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.

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.

6. BIBLIOGRAPHY

[1] ***Proc. of the DARPA Broadcast News Transcription and Understanding Workshop, Lansdowne, VA, 1998.***

[2] ***C. Apte and F. Damerau, Automated Learning of Decision Rules for Text Categorization, ACM Trans. Inf. Syst., 12 (1994), pp. 233--251.***

[3] ***R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, Addison Wesley, 1999.***

[4] ***E. Brown and A. Coden, Capitalization Recovery for Text, Information Retrieval Techniques for Speech Applications, Lecture Notes in Computer Science Vol. 2273, Springer Verlag, 2002.***

[5] ***E. W. Brown and H. A. Chong, The GURU System in TREC-6, in D. K. Harman and E. M. Voorhees, eds., The Sixth Text REtrieval Conference (TREC-6), National Institute of Standards and Technology Special Publication 500-240, Gaithersburg, MD, 1998, pp. 535--540.***

[6] ***E. W. Brown, S. Srinivasan, A. Coden, D. Ponceleon, J. W. Cooper and A. Amir, Toward Speech as a Knowledge Resource, IBM Systems Journal, 40 (2001), pp. 985-1001.***

[7] ***A. Chowdhury, S. Beitzel, E. Jensen, M. Sailee, D. Grossman, O. Frieder, M. C. McCabe and D. Holmes, IIT TREC-9 - Entity Based Feedback with Fusion, Proc. of the Ninth Text REtrieval Conference (TREC 9), 2001.***

[8] ***C. Cieri, D. Graff, M. Liberman, N. Martey and S. Strassel, The TDT-2 Text and Speech Corpus, Proc. of the 1999 DARPA Broadcast News Workshop, Herndon, VA, 1999.***

[9] ***A. Coden and E. Brown, Speech Transcript Analysis for Automatic Search, Proc. of HICSS'34, 2001.***

[10] ***J. W. Cooper and R. J. Byrd, Lexical Navigation: Visually Prompted Query Expansion and Refinement, Proc. of the ACM International Conference on Digital Libraries, Philadelphia, PA, 1997, pp. 237-246.***

[11] ***J. Garofolo, E. Voorhees, V. Stanford and K. S. Jones, TREC-6 1997 Spoken Document Retrieval Track Overview and Results, The***

Sixth Text REtrieval Conference (TREC-6), NIST Special Publication 500-240, Gaithersburg, MD, 1998, pp. 83-91.

[12] C. Manning and H. Schuetze, Foundations of Statistical Natural Language Processing, MIT Press, Cambridge, MA, 1999.

[13] M. Mitra, C. Buckley, A. Singhal and C. Cardie, An Analysis of Statistical and Syntactic Phrases, Proc. of RIAO97, Computer-Assisted Information Searching on the Internet, Montreal, Canada, 1997, pp. 200-214.

[14] Y. Ravin, N. Wacholder and M. Choi, Disambiguation of Names in Text, Proc. of the ACL Conf. on Applied Natural Language Processing, Washington, D.C., 1997, pp. 202-208.

[15] T. Strzalkowski, F. Lin and J. Perez-Carballo, Natural Language Information Retrieval TREC-6 Report, Proc. of the Sixth Text REtrieval Conference (TREC-6), 1998.

[16] T. Strzalkowski, J. Perez-Carballo, J. Karlgren, A. Hulth, P. Tapanainen and T. Lahtinen, Natural Language Information Retrieval: TREC-8 Report, Proc. of the Eigth Text REtrieval Conference (TREC 8), 2000.

[17] T. Strzalkowski, G. Stein, G. B. Wise, J. Perez-Carballo, P. Tapananinen, T. Jarvinen, A. Voutilainen and J. Karlgren, Natural Language Information Retrieval: TREC-7 Report, Proc. of the Seventh Text REtreival Conference (TREC-7), 1999.

[18] A. Turpin and A. Moffat, Statistical Phrases for Vector-Space Information Retrieval, Proc. of the ACM Inter. Conf. on Research and Development in Information Retrieval, Berkeley, CA, 1999, pp. 309-310.