AT&T at TREC-8

Amit Singhal, Steve Abney, Michiel Bacchiani,
Michael Collins, Donald Hindle, Fernando Pereira

AT&T Labs–Research
{singhal,abney,bacchiani,mcollins,hindle,pereira}@research.att.com

Abstract

In 1999, AT&T participated in the ad-hoc task and the Question Answering (QA), Spoken Document Retrieval (SDR), and Web tracks. Most of our effort for TREC-8 focused on the QA and SDR tracks. Results from SDR track show that our document expansion techniques, presented in [8, 9], are very effective for speech retrieval. The results for question answering are also encouraging. Our system designed in a relatively short period for this task can find the correct answer for about 45% of the user questions. This is specially good given the fact that our system extracts only a short phrase as an answer.

1 Introduction

Question answering and spoken document retrieval were the main areas of interest for AT&T at TREC-8. For most of our work, we used an internally modified version of the SMART retrieval system developed at Cornell University [2, 6].

Our question-answering system first retrieves the top ranked passages in response to a user question. These passages are then passed to a linguistic processing sub-system that analyzes the user question and the passages to spot entities that might answer the question. These entities are then ranked based on several heuristics that were developed on training questions. To establish a baseline, we also submitted two runs based only on passage retrieval that did not use the linguistic processing sub-system of our QA system.

For speech retrieval, we continued our experiments with document expansion using related corpora [8, 9]. Results are once again consistently good.

2 Ad-hoc Runs

Our ad-hoc runs are little changed from our 1998 ad-hoc submissions. Please refer to [8] for the details of the algorithms used.

For this task, we strongly believe that full-length TREC topics are artificially long in comparison to real user queries. Another artificiality of the TREC environment is the structure of these topics, *i.e.* the existence of separate `title`, `description` and the `narrative` sections. Such structure will not be encountered in more popular search environments. Therefore, we ignore the `narrative` section in all our runs. Additionally, we ignore the knowledge that certain words are `title` words or `description` words in a topic. We experiment with very short, `title only` queries (**t**), and longer, `title and description` (**t+d**) queries.

We submitted two runs in each category: **att99tc** and **att99te** for `title only` queries and **att99tdc** and **att99tde** for `title and description` queries. Our conservative runs **att99tc** and **att99tdc** are a repeat of our 1998 conservative collection enrichment based runs based on pseudo-feedback. Our experimental runs **att99te** and **att99tde** do some "locality-based" document selection to be used in pseudo-feedback. For these runs, we retrieve the top 50 documents using our standard vector-space ranking. We then re-ranked these fifty documents to promote documents that contain multiple query words in the same sentence or adjoining sentences (see details in Section 3 on Question Answering). Top ten documents from this reranked list are assumed relevant for pseudo-feedback.

Query Sections	Baseline *dnb.dtn*	Expansion from target collection	Expansion from TREC D12345	Conservative Collection Enrichment
t (**att99atc**)	0.2363	0.2736 (+15.8%)	0.2884 (+22.1%)	**0.2853** (+20.8%)
# Q better/worse		0/0	29/21	34/16
t (**att99ate**)	0.2363	0.2740 (+16.0%)	0.2840 (+20.2%)	**0.2835** (+20.0%)
# Q better/worse		0/0	24/26	28/22
t+d (**att99atdc**)	0.2592	0.2943 (+13.5%)	0.3170 (+22.3%)	**0.3089** (+19.2%)
# Q better/worse		0/0	30/20	34/16
t+d (**att99atde**)	0.2592	0.3024 (+16.7%)	0.3237 (+24.9%)	**0.3165** (+22.1%)
# Q better/worse		0/0	27/23	29/21

Table 1: Effect of conservative collection enrichment

Run	Average Precision	Best	>= Median	< Median
att99atc (title only)	0.2853	2	32	16
att99ate (title only)	0.2835	4	30	16
att99atdc (title+desc)	0.3089	2	41	7
att99atde (title+desc)	0.3165	4	38	8

Table 2: Results for adhoc runs

The results for our ad-hoc runs are shown in Tables 1 and 2. There are several possible variations on the database used for pseudo-feedback: expansion from the target collection (no collection enrichment), expansion from a large collection (collection enrichment), and conservative collection enrichment [8]. Even though the conservative method has somewhat somewhat poorer average precision than methods that expand from the large collection alone (collection enrichment), as the rows labeled *# Q better/worse* show, it is more stable with respect to the number of queries that improve or deteriorate in comparison to expansion from the target collection (This is the sensible baseline for this comparison since if we compare to unexpanded queries, which is the baseline used in the average precision rows, all expansion strategies will show large gains and the relative performance of different expansions will be hard to judge).

Overall, these results are quite reasonable. In our view, the minor improvements that we gain out of doing various modifications to our simple two-pass pseudo-feedback based algorithm (used in **att99atc** and **att99atdc**), are not fundamentally better and they come at an increased processing cost. Algorithms quite similar to the one used in **att99atdc** have consistently been one of the best over the last three or four TREC conferences. This leads us to wonder whether we are learning something new from the ad-hoc runs in recent years.

3 Question Answering

Our question answering system is a hybrid IR-linguistic system. The retrieval component first retrieves top ranked passages for a question, and the linguistic component then processes those passages in light of the user question to identify entities than can potentially answer the question.

3.1 Passage Retrieval

The passage retrieval system involves the following steps:

1. The top fifty documents for a question are retrieved using a straight vector match (no query expansion).
2. Each section of these top fifty documents is broken into sentences and each sentence is assigned a score based on the following algorithm:
   - the sentence score is initialized to zero, and the passage size is also initialized to zero;

- the query term weight of every question word that appears in the sentence is added to the sentence score, the passage size is set to the sentence size (in bytes);
- if a query word bigram appears in the sentence, an extra credit[1] is assigned to the sentence,
- if an adjoining sentence contains a question word not contained in this sentence, and if by adding this adjoining sentence to the passages the passage size doesn't exceed 500 bytes, half the query term weight for this word is added to the sentence score;
- if a next to adjoining sentence contains a question word not covered yet, and if by adding this adjoining sentence to the passages the passage size doesn't exceed 500 bytes, quarter of the query term weight for this word is added to the sentence score.

3. Highest scoring passage from each section is printed along with its score.
4. The highest scoring fifty passages are then selected for processing by the linguistic module.

3.2 Linguistic Processing

This section describes the natural-language processing component of the system. Input to this component is the text of the query, together with a set of (ranked) passages that are the output from the IR component. Output from this component is a list of answers ranked in order of importance. The top 5 answers were submitted to the TREC evaluation.

The basic strategy can be divided into three stages:

1. Extract a candidate set of possible answers from the passages, along with their types. The candidate set is a set of entities falling into a number of categories, including people, locations, organizations, quantities, dates, and linear measures. A full list of the types, and a description of how they are extracted, is given in section 3.2.1.
2. Produce a partial ranking of the entities according to how well their type matches the query. Usually this will involve a binary distinction of whether or not an entity is of the correct type: for example, given the query

   Who is the author of the book, The Iron Lady: A Biography of Margaret Thatcher?

   it can be assumed that entities of type `PERSON` are preferred, and a partial ranking is formed where all entities of type `PERSON` are placed ahead of other entity types.
3. Produce a final ordering of the entities by taking into account their frequency and position in the passages. The partial ordering from stage 2 contains many equally ranked entities: in the above example all `PERSON` entities would be ranked the same. The third stage produces a finer grained ranking of entities of the same type through the use of frequency and position information.

The following sections describe in detail how these three processing stages were carried out.

3.2.1 Entity Extraction

The following types of entities were extracted as potential answers to queries:

Proper Names Proper names (capitalized sequences of words) were extracted from the passages, and then classified into one of the categories `PERSON`, `LOCATION`, `ORGANIZATION` or `OTHER` using a classifier built using the method described in [4].[2]

[1] $0.25\times$(lower of the two component query term weights)

[2] The classifier makes a three way distinction between `PERSON`, `LOCATION` and `ORGANIZATION`; names where the classifier makes no decision were classified as `OTHER`.

Dates All years (4 digit numbers starting with `1...` or `20...`) were extracted from the passages. The CASS parser [1] was used to extract full dates (such as *January 1st 2000*).

Quantities Bare numbers were extracted using the CASS parser. Noun phrases involving modification by a number were also extracted by CASS: for example *The Three stooges*, *4 airports*, *270 people*. In this latter case the headword of the noun phrase (*stooges*, *airports* or *people*) was extracted; these entities could then be later identified as good answers to *How many ...* questions such as *How many stooges were there?*

Durations CASS was used to extract time durations such as *three years*, *four hours* and so on.

Linear Measures CASS was used to extract measure amounts such as *170 miles* or *180 feet*.

We should note that this list of types is almost certainly not complete. Monetary amounts (e.g., *$25 million*) were added to the system shortly after the TREC run, but other gaps in coverage remain.

3.2.2 Ranking of Entities by Type

This stage involved processing the query to identify the type that is required by the user. The following rules were used to do this:

- All queries starting with *Who, Whom* were taken to be of type `PERSON`.
- All queries starting with *Where, Whence, Whither* were taken to be of type `LOCATION`.
- All queries starting with *When* were taken to be of type `DATE`.
- All queries starting with *How few, How great, How little, How many, How much* were taken to be of type `QUANTITY`.
- All queries starting with *How long* were taken to be ambiguous between `DURATION` and `LINEAR_MEASURE`. All queries starting with *How tall, How wide, How high, How big, How far* were taken to be `LINEAR_MEASURE`.
- Queries containing the wh-words *Which* or *What* typically also involve a head noun that describes the type of entity involved. These questions fall into two formats: *What X* where *X* is the noun involved, or *What is the ... X*. Here are a couple of examples:

  What company is the largest Japanese ship builder?

  What is the largest city in Germany?

  For these queries the head noun (e.g., *company* or *city*) was extracted, and a lexicon mapping nouns to types was used to identify the type of the query. The lexicon was partly hand-built (including some common cases such as *number* → `QUANTITY` or *year* → `DATE`). A large list of nouns indicating `PERSON`, `LOCATION` or `ORGANIZATION` categories was automatically taken from the contextual (appositive) cues learned in the named entity classifier described in [4].
- In queries containing no wh-word (e.g., *Name the largest city in Germany*), the first noun phrase that is an immediate constituent of the matrix sentence is extracted, and its head is used to determine query type, as for *What X* questions.

If these rules fail to identify the type of the query, then all entities get equal ranking.

In most cases, the query classification stage implies a binary distinction (ranking) of the entities, depending on whether they are or are not of the correct type. However, there are a couple of special cases where finer distinctions are made. If a question is of the `DATE` type, and the query contains one of the words *day*, *month* or *year*, then "full" dates are ranked above years. Conversely, if the query contains the word *year*, then years are ranked above full dates. In *How many X* questions (where *X* is a noun), quantified phrases whose head noun is also *X* are ranked above bare numbers of other quantified phrases: for example, in the query *How many lives were lost in the Lockerbie air crash*, entities such as *270 lives* or *almost 300 lives* would be ranked above entities such as *270 people* or *150*.

Run	Mean Answer Length	Answer in Top-5	Mean Score
attqa50e	10.5 bytes	89/198	0.356
attqa50p	50 bytes	77/198	0.261

Table 3: Results for the 50-byte answer category

3.2.3 Ranking of Entities by Position

Finally, the frequency and position of entities in the retrieved passages was taken into account. First, some normalization of entities is done: dates are mapped to the format year-month-day, proper names are normalized by last-name, ... Then a score is calculated for each entity. Each time an entity occurs in the top-ranked passage (or passages) from the IR system, its score is incremented by 10. Each time it occurs in a lower ranked passage, its score is incremented by 1. This score is used as a secondary ranking method superimposed on the partial ranking given by the query classifier.

3.3 Results

We submitted two linguistics based runs, one in the 50-byte answer category and another in the 250-byte category. We also submitted two comparable passage retrieval based runs.

3.3.1 50-bytes

The most realistic run is our 50-byte entity based run **attqa50e**; this run extracts only the entity that the system thinks is *the answer*. The details of this run are described above in the section on linguistic processing. For comparison we also did a 50-byte passage based run (which involves no entity recognition/extraction). The passage only run trims the top ranked sentences to reduce them down to fifty bytes. It drops some function words from a sentence, and it drops the question words assuming they won't be in the answer. If the resulting trimmed sentence is still over fifty bytes, it outputs the first 50 characters.

The results from our 50 bytes runs: **attqa50e**, the entity-based run, and **attqa50p**, the passage-based run, are shown in Table 3. We expected our entity based run to be better than the passage based run, and it is. This reinforces the belief that IR system need context to do their job well. When an IR system is constrained to extract a very tiny piece of text as *the answer*, it doesn't do very well. Even with extracting about five times as much text as compared to our entity-based system (50 bytes/answer instead of just 10.5 for the entity-based system), the passage-based system gets fewer answers right and the answers are not ranked well. These results indicate that to do question answering such that the answer is just a phrase (or a very short snippet of text) we will need to enhance a purely keyword-based system with some linguistic processing.

3.3.2 250-bytes

Our 250-bytes passage based run **attqa250p** involves the following steps:

1. The passages retrieved as per the algorithm described in Section 3.1 are first refined so that no single document contributes more than one passage to the passage pool. (As described in Section 3.1, different sections of a document can each contribute a passage to the passage pool.) When one document supplies more than one passage, the highest scoring passage is selected. Ties are broken in favor of longer passages as they have a higher chance of containing the answer.

2. Near-duplicate passages are removed from the pool. If a low-scoring passage has a cosine-similarity of over 0.50 with a highly ranked passage, the low-scoring passage is removed from the pool. The main motivation behind doing this is to improve our chances of hitting the answer in one of the top five passages, instead of repeating same information in multiple passages.

Run	Mean Answer Length	Answer in Top-5	Mean Score
attqa250p	249 bytes	135/198	0.545
attqa250e	247 bytes	120/198	0.483

Table 4: Results for the 50-byte answer category

3. The top five passages from the remaining pool are printed in order of their scores. If a passage is longer than 250 bytes, the *key-sentence* of that passage (remember we build passages around a key sentence by adding previous and next sentences, and we keep adding sentences as long as we are under 500 bytes) is printed. If we still have some bytes to spare, the later bytes of previous sentence are added, and then the earlier bytes of the later sentence.

Our 250-bytes linguistic (entity) based run **attqa250e** involves the following steps:

1. The ranked list of entities as ranked for our entity-based run **attqa50e** is used as a a starting point.
2. Passages are ranked using the passage ranking algorithm described in Section 3.1.
3. The first passage that contains the top-ranked entity in the entity list is presented to the user. The top ranked entity and other entities covered by this passage are removed from the entity list.
4. This process is repeated until we have five 250 bytes passages.

Results for our 250-byte runs are shown in Table 4. Our passage based run in the 250-bytes category was one of the best runs in this track. This result indicates that when a passage-retrieval system is allowed some more text in its output, it can do a very good job answering questions. This further reinforces the belief that IR systems need context to do their job well. We believe that these results can be improved notably with more effort, and we are working in that direction.

3.4 Error Analysis of the Entity-Based System

3.4.1 Ranking of Answers

We looked first at the performance of the entity-based system, considering the queries where the correct answer was found somewhere in the top 5 answers (46% of the 198 questions). We found that on these questions, the percentage of answers ranked 1, 2, 3, 4, and 5 was 66%, 14%, 11%, 4%, and 4% respectively. This distribution is by no means uniform; it is clear that when the answer is somewhere in the top five, it is very likely to be ranked 1st or 2nd. The system's performance is thus bimodal: it either completely fails to get the answer, or else recovers it with a high rank.

3.4.2 Accuracy on Different Categories

Table 5 shows the distribution of question types in the TREC-8 test set ("Percentage of Questions"), and the performance of the entity-based system by question type ("System Accuracy"). We categorized the questions by hand, using the eight categories described in section 3.2.1, plus two categories that essentially represent types that were not handled by the system at the time of the TREC deadline: `Monetary Amount` and `Miscellaneous`.

"System Accuracy" means the percentage of questions for which the correct answer was in the top five returned by the system. There is a sharp division in the performance on different question types. The categories `Person`, `Location`, `Date` and `Quantity` are handled fairly well, with the correct answer appearing in the top five 60% of the time. These four categories make up 67% of all questions. In contrast, the other question types, accounting for 33% of the questions, are handled with only 15% accuracy.

Unsurprisingly, the `Miscellaneous` and `Other Named Entity` categories are problematic; unfortunately, they are also rather frequent. Table 6 shows some examples of these queries. They include a large tail of

Type	Percentage of Questions	System Accuracy	Type	Percentage of Questions	System Accuracy
Person	28	62.5	Other Named Entity	14.5	31
Location	18.5	67.6	Miscellaneous	8.5	5.9
Date	11	45.5	Linear Measure	3.5	0
Quantity	9.5	52.7	Monetary Amount	3	0
			Organization	2	0
			Duration	1.5	0
TOTAL	67	60	TOTAL	33	15

Table 5: Performance of the entity-based system on different question types. "System Accuracy" means percent of questions for which the correct answer was in the top five returned by the system. "Good" types are on the left, "Bad" types are on the right.

What does the Peugeot company manufacture?
Why did David Koresh ask the FBI for a word processor?
What are the Valdez Principles?
What was the target rate for M3 growth in 1992?
What does El Nino mean in spanish?

Table 6: Examples of "Other Named Entity" and "Miscellaneous" questions.

questions seeking other entity types (mountain ranges, growth rates, films, etc.) and questions whose answer is not even an entity (e.g., "Why did David Koresh ask the FBI for a word processor?")

For reference, Table 7 gives an impression of the sorts of questions that the system does well on (correct answer in top five).

3.4.3 Errors by Component

Finally, we performed an analysis to gauge which components represent performance bottlenecks in the current system. We examined system logs for a 50-question sample, and made a judgment of what caused the error, when there was an error. Table 8 gives the breakdown. Each question was assigned to exactly one line of the table.

The largest body of errors, accounting for 18% of the questions, are those that are due to unhandled types, of which half are monetary amounts. (Questions with non-entity answers account for another 4%.) Another large block (16%) is due to the passage retrieval component: the correct answer was not present in the retrieved passages. The linguistic components together account for the remaining 14% of error, spread evenly among them.

Question	Rank	Output from System
Who is the author of the book, The Iron Lady: A Biography of Margaret Thatcher?	2	Hugo Young
What is the name of the managing director of Apricot Computer?	1	Dr Peter Horne
What country is the biggest producer of tungsten?	1	China
Who was the first Taiwanese President?	1	Taiwanese President Li Teng hui
When did Nixon visit China?	1	1972
How many calories are there in a Big Mac?	4	562 calories
What is the acronym for the rating system for air conditioner efficiency?	1	EER

Table 7: A few TREC questions answered correctly by the system.

Errors	
Passage retrieval failed	16%
Answer is not an entity	4%
Answer of unhandled type: money	10%
Answer of unhandled type: misc	8%
Entity extraction failed	2%
Entity classification failed	4%
Query classification failed	4%
Entity ranking failed	4%
Successes	
Answer at Rank 2-5	16%
Answer at Rank 1	32%

Table 8: Breakdown of questions by error type, in particular, by component responsible. Numbers are percent of questions in a 50-question sample.

The cases in which the correct answer is in the top five, but not at rank one, are almost all due to failures of entity ranking.[3] Various factors contributing to misrankings are the heavy weighting assigned to answers in the top-ranked passage, the failure to adjust frequencies by "complexity" (e.g., it is significant if *22.5 million* occurs several times, but not if *3* occurs several times), and the failure of the system to consider the linguistic context in which entities appear.

4 SDR Runs

We used our own speech recognizer to process the SDR track data. In this track, we continued our experimentation with document expansion from last year [8, 9]. This year we only submitted runs based on document expansion. Our first run **att-s1** is a reproduction of the algorithm we developed in [9]. Our second run **att-s2** is also based on document expansion, but it is aimed at containing excessive increase in weights of already important document terms (see below for details).

4.1 Speech Recognizer

The speech recognition system used for the SDR track used a multi-pass search paradigm. The resulting transcriptions were obtained by performing two recognition passes, the first using both an acoustic and low complexity language model, the second retaining the acoustic scores from the first pass and using a more complex language model. The acoustic model of this system is described in section 4.1.1, the language models and search algorithm are described in section 4.1.2.

4.1.1 Acoustic model

The acoustic model was trained on all the SDR track data available from previous evaluations. The data used for training was from 14 different news programs from the period May 10, 1996 until January 31, 1998. The total amount of transcribed recordings used in training was 143 hours.

The speech waveforms were parameterized using a mel-frequency cepstral analysis and energy measurements. The system used the first 12 mel-frequency cepstral coefficients and a normalized energy parameter as well as the first and second derivatives (39 dimensions in total), computed at a rate of 100 frames per second. To compensate for channel effects, the cepstral mean of the signal was subtracted for the cepstral vectors.

[3] The sole exception was a query misclassification caused by a parse failure—miraculously, the correct answer made it to rank five despite being of the "wrong" type.

A training dictionary for all 36475 unique words seen in the training transcriptions was generated using our text-to-speech system [3] followed by hand corrections/additions. The resulting dictionary had 38616 entries (average of 1.06 entries per unique word). The used phone set consisted of 42 phone models, 1 silence model and 5 non-speech models.

All phones were modeled using three-state left-to-right HMMs except for the silence model which was a single state HMM. All state emission distributions were modeled by Gaussian mixture densities. Mixture densities were estimated by iteratively segmenting the data using the Viterbi algorithm and estimating mixture densities for the given segmentation using the Expectation Maximization (EM) algorithm (i.e. the mixture identities were hidden but the state segmentations were not). To initialize the mixture components for the EM algorithm, the k-means clustering algorithm with a Euclidean distance metric was used on variance normalized data. The final acoustic model was trained in three stages. In the first stage, a context independent system was build. To bootstrap this first training stage, an initial state-level segmentation was obtained by a Viterbi alignment using our last evaluation system. Then 20 mixture component state emission densities were estimated in three iterations. In the first iteration, 8 mixture component densities were estimated. In the second, the number of mixture components was increased to 20 and this model was refined by another iteration. In the second training stage, the sharing among triphone state emission distributions were defined. Shared state distributions were defined by decision tree clustering using a likelihood design criterion and allowing questions about the phonetic identity of the phone context. Finally, in the third stage, mixture densities were estimated for the shared state distributions. The final densities were obtained in four iterations, the first two to estimate 4 mixture component densities, the second two to estimate 12 mixture component densities.

4.1.2 Language model and Search Algorithm

In the first recognition pass, lattices were built that were rescored in a second recognition pass. The most likely transcripts were then used together with the acoustic model from the first pass to find the boundary times of the words in the transcriptions by the Viterbi algorithm.

The first recognition pass used a pruned trigram language model, the second an un-pruned 6-gram model. Both first and second pass models were Katz [5] backoff language models. The first pass trigram model was pruned using the approach of Seymore and Rosenfeld [7] using a pruning threshold of 100. In addition to the transcriptions of previous SDR evaluations we also used the transcripts of the Hub4 evaluations and two printed media sources (the LDC North American news corpus and United Press International (ClariNet)).

Different language models were constructed for every two week period in the evaluation data. A total of 11 sets of first and second pass language models were constructed for the 5 month period that the evaluation data covers. First a model was constructed for the first two week period using all available training data prior to that period. Then, for the construction of the models for each subsequent two week period, the data from the preceding two week period was added to the data used for the first two week period model. A weighting scheme was used to emphasize the contribution of the most recent two week period data as well as to emphasize the spoken news transcripts with respect to the printed sources. To accompany the two week language models, a different dictionary was used for each two week period. The dictionaries included all unique words found in the transcriptions as well as all unique words occurring with a frequency larger than two in the printed sources. The sizes of the dictionaries for the different two week periods ranged from 210340 entries to 261215 entries.

4.2 Retrieval System

We used the NA News corpus and UPI news (also used in the language model training described above) as the related corpus for document expansion as well as the large collection for conservative collection enrichment (see [8]) for query expansion. The retrieval cutoff date for this track was July 1, 1998. For 1998, the NA News corpus only has news for January to April 1998. We use all these news articles in our runs. We also added to this UPI news available through Clarinet news for the months of April to June 1998. This gave us a related corpus of 182,755 news articles.

Algorithm/Transcript details	No query expansion	Query expansion from target collection	Query expansion from Print News	Conservative Coll. Enrich.
No document expansion	0.4574	0.5103	0.5742	0.5390
ltt (Closed Captions)	—	+11.6%	+25.5%	+17.8%
No document expansion	0.4113	0.4888	0.5498	0.5194
nist-b1 (WER:27.5%)	—	+18.8%	+33.7%	+26.3%
Loss due to ASR	(–10.1%)	(–4.2%)	(–4.2%)	(–3.6%)
No document expansion	0.4058	0.4798	0.5506	0.5164
att-s1 (WER:29.3%)	—	+18.2%	+35.7%	+27.3%
Loss due to ASR	(–11.3%)	(–6.0%)	(–4.1%)	(–4.2%)
No document expansion	0.2916	0.3740	0.4123	0.3970
cmu-s1 (WER:64.4%)	—	+28.2%	+41.4%	+36.1%
Loss due to ASR	(–36.3%)	(–26.7%)	(–28.2%)	(–26.3%)
No document expansion	0.4286	0.5055	0.5667	0.5339
cuhtk-s1 (WER:20.5%)	—	+17.9%	+32.2%	+24.6%
Loss due to ASR	(–6.3%)	(–1.0%)	(–1.3%)	(–0.9%)
No document expansion	0.4233	0.4890	0.5531	0.5212
cuhtk-s1p1 (WER:26.6%)	—	+17.9%	+32.2%	+24.6%
Loss due to ASR	(–7.5%)	(–4.2%)	(–3.7%)	(–3.3%)
No document expansion	0.4226	0.5014	0.5554	0.5224
limsi-s1 (WER:21.5%)	—	+18.7%	+31.4%	+23.6%
Loss due to ASR	(–7.6%)	(–1.7%)	(–3.3%)	(–3.1%)
No document expansion	0.4001	0.4770	0.5402	0.5065
shef-s1 (WER:32.0%)	—	+19.2%	+35.0%	+26.6%
Loss due to ASR	(–12.5%)	(–6.5%)	(–5.9%)	(–6.0%)

Table 9: SDR Runs: No Document Expansion.

Algorithm-1

Our first document expansion algorithm is taken verbatim from our previous work presented in [9]. The query expansion algorithm is the same as the one used in our ad-hoc runs, only the target collection and the large collection for conservative collection enrichment are different. The target collection is the SDR collection and the large collection is the NA News and UPI news collection described above.

The results for our SDR runs are shown in Tables 9–11. Here are the main observations from these results.

1. Speech retrieval over automatically recognized speech is very viable. For reasonable transcripts, the losses in retrieval effectiveness are minimal, 1–5% (the negative numbers shown in parentheses).

2. As expected, query expansion via pseudo-feedback is useful across the board. This can be observed in the last three columns of the Tables. In each of these columns, the second entry shows the improvements of the corresponding query expansion algorithm over no query expansion.

3. Our conservative query expansion hurt us in this environment. This is evident by the consistently better results from doing query expansion from the print news vs. doing conservative collection enrichment. For example, when doing retrieval from closed caption (second row in Table 10), doing query expansion from print news yields an average precision of 0.5742, whereas our conservative query expansion yields only 0.5390, a noticeable drop.

4. Document expansion (see Table 10) is consistently beneficial. For example, our reference run **att-r1** would have been 0.5390 instead of 0.5600 had we not used document expansion.

5. Retrieval effectiveness is not very sensitive to WER of the recognizers for reasonable recognition. This is evident by looking at all our official runs (other than the one on cmu's transcripts) which have

Algorithm/Transcript details	No query expansion	Query expansion from target collection	Query expansion from Print News	Conservative Coll. Enrich.
Document expansion (Algo-1)	0.4918	0.5371	0.5804	**0.5600 att-r1**
ltt (Closed Captions)	—	+9.2%	+18.0%	+13.8%
Document expansion (Algo-1)	0.4779	0.5427	0.5646	**0.5539 att-b1**
nist-b1 (WER:27.5%)	—	+13.6%	+18.2%	+15.9%
Loss due to ASR	(–2.8%)	(+1.0%)	(–2.7%)	(–1.0%)
Document expansion (Algo-1)	0.4639	0.5207	0.5586	**0.5431 att-s1**
att-s1 (WER:29.3%)	—	+12.2%	+20.4%	+17.1%
Loss due to ASR	(–5.7%)	(–3.1%)	(–3.8%)	(–3.0%)
Document expansion (Algo-1)	0.3752	0.4369	0.4635	**0.4526 att-cr-cmus1**
cmu-s1 (WER:64.4%)	—	+16.5%	+23.5%	+20.6%
Loss due to ASR	(–23.7%)	(–18.7%)	(–20.2%)	(–19.2%)
Document expansion (Algo-1)	0.4901	0.5421	0.5715	**0.5592 att-cr-cuhtks1**
cuhtk-s1 (WER:20.5%)	—	+10.6%	+16.6%	+14.1%
Loss due to ASR	(–0.3%)	(+0.9%)	(–1.5%)	(–0.1%)
Document expansion (Algo-1)	0.4724	0.5309	0.5647	**0.5494 att-cr-cuhtks1p1**
cuhtk-s1p1 (WER:26.6%)	—	+12.4%	+19.5%	+16.3%
Loss due to ASR	(–3.9%)	(–1.1%)	(–2.7%)	(–1.9%)
Document expansion (Algo-1)	0.4717	0.5346	0.5631	**0.5516 att-cr-limsis1**
limsi-s1 (WER:21.5%)	—	+13.3%	+19.4%	+16.9%
Loss due to ASR	(–4.1%)	(–0.5%)	(–3.0%)	(–1.4%)
Document expansion (Algo-1)	0.4710	0.5277	0.5588	**0.5455 att-cr-shefs1**
shef-s1 (WER:32.0%)	—	+12.0%	+18.6%	+15.8%
Loss due to ASR	(–4.2%)	(–1.8%)	(–3.7%)	(–2.6%)

Table 10: SDR Runs: Document Expansion, Algorithm-1

average precision values in the range for 0.5431 to 0.5600. There is an insignificant 3% gap in average precision between doing retrieval on closed caption vs. doing retrieval on ASR transcripts which have up to 32% WER.

Algorithm-2

We remind you that our term weighting scheme assigns weights to words in a document based on their occurrence frequency in the document and the length of the document. The two basic factor are the *tf*-factor, which accounts for the fact that words that are repeated within a document are more important; and the document length normalization factor which is used to assign lower weights to all words in very long documents. Within one document, the document length normalization factor is same for all terms. However, the *tf*-factor changes from word to word based on the word's frequency. During the last few years, we have realized that a word that appears three times in a document is not thrice as important than a word that appears just once so we have been using a dampened *tf*-factor (a logarithmic or a double-log factor) which rises sub-linearly with the increase in word frequency. In particular, we use the double log *tf*-factor, *i.e.* a word with frequency *tf* gets a weight of $1 + ln(1 + ln(tf))$.

During the course of our experiments with document expansion, we noticed that the expansion algorithm we use above tends to create an unwanted imbalance in weights of different document terms. For example, consider a document which contains many instances of the word `information`. When we use this document as a query to find related printed documents, many of the related documents also mention the word `information`, since `information` is an important word in the query vector, *i.e.* the vector for the recognized document. When we do document expansion using Rocchio's formula, the word `information` gets a further boost in its weight and it ends up being a very heavily weighted word for this document. This is contrary to the reason for using a dampened *tf*-factor in our term weighting scheme (as described above).

To address this problem, we changed our document expansion scheme to ensure that frequent words do

Algorithm/Transcript details	No query expansion	Query expansion from target collection	Query expansion from Print News	Conservative Coll. Enrich.
Document expansion (Algo-2)	0.4821	0.5403	**0.5863**	0.5677
ltt (Closed Captions)	—	+12.1%	+21.6%	+17.7%
Document expansion (Algo-2)	0.4610	0.5406	**0.5716**	0.5634
nist-b1 (WER:27.5%)	—	+17.3%	+24.0%	+22.2%
Loss due to ASR	(–4.4%)	(+0.1%)	(–2.5%)	(–0.8%)
Document expansion (Algo-2)	0.4536	0.5208	**0.5731**	**0.5510 att-s2**
att-s1 (WER:29.3%)	—	+14.8%	+26.3%	+21.5%
Loss due to ASR	(–5.9%)	(–3.6%)	(–2.3%)	(–2.9%)
Document expansion (Algo-2)	0.3520	0.4077	**0.4643**	0.4508
cmu-s1 (WER:64.4%)	—	+15.8%	+31.9%	+28.1%
Loss due to ASR	(–27.0%)	(–24.6%)	(–20.8%)	(–20.6%)
Document expansion (Algo-2)	0.4711	0.5496	**0.5829**	0.5705
cuhtk-s1 (WER:20.5%)	—	+16.7%	+23.7%	+21.1%
Loss due to ASR	(–2.3%)	(+1.7%)	(–0.6%)	(+0.5%)
Document expansion (Algo-2)	0.4601	0.5336	**0.5678**	0.5539
cuhtk-s1p1 (WER:26.6%)	—	+16.0%	+23.4%	+20.4%
Loss due to ASR	(–4.6%)	(–1.2%)	(–3.2%)	(–2.4%)
Document expansion (Algo-2)	0.4645	0.5349	**0.5698**	0.5551
limsi-s1 (WER:21.5%)	—	+15.2%	+22.7%	+19.5%
Loss due to ASR	(–3.7%)	(–1.0%)	(–2.8%)	(–2.2%)
Document expansion (Algo-2)	0.4508	0.5229	**0.5598**	0.5452
shef-s1 (WER:32.0%)	—	+16.0%	+24.2%	+21.0%
Loss due to ASR	(–6.5%)	(–3.2%)	(–4.5%)	(–3.9%)

Table 11: SDR Runs: Document Expansion, Algorithm-2.

not end up getting very heavy weights. Under this scheme, any word is allowed an increment of *one* in it's raw frequency due to expansion. For example, if a word occurred once in the document, its pre-expansion weight was 1.0 (ignoring document length normalization). If the post-expansion weight for this words becomes 2.0, which will correspond to a post-expansion raw frequency of 5.57 (since $1 + ln(1 + ln(5.57)) = 2.0$), then this increment is not allowed and the raw frequency increase is capped at 1, yielding the post-expansion raw frequency of 2 and a post-expansion weight of 1.53. This effect is even more visible for words with very high raw frequencies (like 10).

We submitted a second run based on this document expansion scheme and the results are shown in Table 11. Comparing this document expansion algorithm **att-s2** (Algo-2) with our previous algorithm **att-s1** (Algo-1) in Table 10, we do see that this algorithm yields consistently better results than our old algorithm. It in fact yields the best results for every transcription, which are shown in column-4 of Table 11

Run	Average Precision	Best	>= Median	< Median
att-r1	0.5600	7	27	15
att-b1	0.5539	1	35	13
att-s1	0.5431	5	29	15
att-s2	0.5510	4	31	14
att-cr-cmus1	0.4626	0	18	31
att-cr-cuhtks1	0.5592	2	35	12
att-cr-cuhtks1p1	0.5494	2	32	15
att-cr-limsis1	0.5516	2	32	15
att-cr-shefs1	0.5455	3	30	16

Table 12: Results for SDR runs

Query Sections	Baseline *dnb.dtn*	Expansion from target collection	TREC D12345	Conservative Collection Enrichment
t+d (att99wtdc)	0.2470	0.2876 (+16.5%)	0.3033 (+22.8%)	**0.3091** (+25.2%)
t+d (att99wtde)	0.2470	0.2883 (+16.8%)	0.3138 (+27.1%)	**0.3113** (+26.0%)

Table 13: Effect of conservative collection enrichment

Run	Average Precision	Best	>= Median	< Median
att99wtdc (title+desc)	0.3033	2	40	8
att99wtde (title+desc)	0.3113	2	37	11

Table 14: Results for adhoc runs

(in boldface). Table 12 presents some other statistics on our official runs.

5 Web Track Runs

We submitted two runs for the small Web task: **att99wtdc** and **att99wtde**. These runs correspond to our ad-hoc runs att99atdc and att99atde, respectively. The only difference is that for the web runs, we remove duplicates from the initial list of documents used in pseudo-feedback. These runs are *content-only* runs and do not use the linkage analysis commonly used by Web search engines. For these runs, we first retrieve the top 100 documents using our standard vector-space ranking. If two documents in this list have a cosine similarity over 0.80, we assume they are duplicates of each other and remove the lower ranked document from this list.

For run **att99wtde**, the list of documents obtained above after duplicate removal is further re-ranked using sentence based locality described in our question answering effort. Top 10 documents from this reranked list are assumed relevant and are used in pseudo-feedback based query expansion. For run **att99wtdc**, no reranking is done and the top 10 documents are used in pseudo-feedback. Both these runs use the `title` and `description` sections of the queries. TREC disks 1–5 are used as the larger collection for conservative collection enrichment.

The results from our runs on the 2G web data are shown in Tables 13 and 14. These results are reasonable, especially given the fact that we did not change our retrieval algorithm in any significant manner for Web data.

We also submitted two runs for the large Web track: **att99vlci** and **att99vlcm**. Both runs are based on merging results from twenty different collection formed by dividing the 100G Web data into twenty 5G collections. The run **att99vlcm** merges the document frequencies from various collections and updates every collection so that every collection has a uniform view of the global inverse document frequency for terms. The run **att99vlci** ignores these issues and take document scores from various collections at their face value. There is no query expansion used in these runs. These runs are a straight-forward vector-space match between the query and the documents.

The precision in top 10 documents for **att99vlci** is 0.6180 and for **att99vlcm** is 0.5980. These numbers show that if a very large collection is divided into smaller sub-collections, then one can simply ignore the global-idf issues and merge results from the individual sub-collections to get effective ranking.

6 Conclusions

Our SDR work establishes the usefulness of document expansion. We are very happy to see the incorporation of the question answering track in TREC and look forward to our continuous participation in it next year.

Acknowledgments

We are thankful to to Andrej Ljolje and Michael Riley for their help in building the recognizer for the SDR track data.

References

[1] Steven Abney. Partial parsing via finite-state cascades. *J. Natural Language Engineering*, 2(4):337–344, December 1996.

[2] Chris Buckley. Implementation of the SMART information retrieval system. Technical Report TR85-686, Department of Computer Science, Cornell University, Ithaca, NY 14853, May 1985.

[3] C. Coker. A dictionary-intensive letter-to-sound program. *Journal of Acoustical Society America, Supplement 1*, pages 78–87, 1985.

[4] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *EMNLP*, 1999.

[5] S.M. Katz. Estimation of probabilities from sparse data from the language model component of a speech recognizer. *IEEE Transactions of Acoustics, Speech and Signal Processing*, pages 400–401, 1987.

[6] Gerard Salton, editor. *The SMART Retrieval System—Experiments in Automatic Document Retrieval.* Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[7] Kristie Seymore and Ronald Rosenfeld. Scalable backoff language models. In *ICSLP'96*, volume 1, 1996.

[8] Amit Singhal, John Choi, Donald Hindle, David Lewis, and Fernando Pereira. AT&T at TREC-7. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-6)*, pages 239–252, 1999.

[9] Amit Singhal and Fernando Pereira. Document expansion for speech retrieval. In *Proceedings of the Twenty Second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 34–41. Association for Computing Machinery, New York, August 1999.