Text Retrieval via Semantic Forests: TREC7

Gregory D. Henderson, Patrick Schone, Thomas H. Crystal*
U.S. Department of Defense
Speech Research Branch
Ft. George G. Meade, MD 20755-6000

1. INTRODUCTION

In the second year of the use of Semantic Forests in TREC, we have raised our 30-document average precision in the automatic *Ad Hoc* task to 27% from 19% last year. We also contributed a significant number of unique relevant documents to the judgement pool [3]. Our mean average precisions on the SDR task are roughly the median performance for that task [4].

The Semantic Forests algorithm was originally developed by Schone and Nelson [1] for labeling topics in text and transcribed speech. Semantic Forests uses an electronic dictionary to make a tree for each word in a text document. The root of the tree is the word from the document, the first branches are the words in the definition of the root word, the next branches are the words in the definitions of the words in the first branches, and so on. The words in the trees are tagged by part of speech and given weights based on statistics gathered during training. Finally, the trees are merged into a scored list of words. The premise is that words in common between trees will be reinforced and represent "topics" present in the document. With minor modifications, queries are treated as documents.

Seven major changes were made in developing this year's system from last year's. (1) A number of pre-processing steps which were performed last year (such as identifying multi-word units) were incorporated into Semantic Forests. (2) A part of speech tagger was added, allowing Semantic Forests to use this additional information. (3) Semantic Forests distinguishes between queries and documents this year, since our experiments indicated they needed to be treated differently. (4) Only the first three letters of words which do not occur in the dictionary are retained, instead of the entire word. (5) A parameter directs Semantic Forests to break each document into segments containing at most a set number of words, typically 500. (6) The algorithms used by Semantic Forests to assign and combine word weights have been improved. (7) Quasi-relevance feedback was implemented and evaluated.

2. GENERAL SYSTEM OVERVIEW

Our overall system design is shown in Figure 1. First a filter is applied to the SGML documents and queries. For the documents, this removes any extraneous header and footer information and expands some common contractions (such as "u.s." into "united_states"). For the queries, we also remove some of the commonly used phrases which occur in queries, such as "relevant documents contain" or "would not be considered relevant". Although we experimented with a form of natural language processing for the queries, and feel that this should produce significant results, the filtering we used this year differs very little from last year. This year the query filter also adds a number of "class" words to the query documents. These words appear in the dictionary as a list of "synonyms" and are described in more detail in Section 3.

Next, Semantic Forests is used to produce a list of topic words for each document segment or query. The topic lists for the document segments are placed in a database, which organizes the information to allow for the efficient retrieval of all document segments having a given word in their topic lists. No significant change was made in this step from last year.

Given the database and the query topic lists, we produce a list of hopefully relevant documents by scoring the document topic lists against the query topic list. Our query system without feedback is very similar to last year's system. The system with quasi-relevance feedback adds an initial rough scoring pass with the original query topic list, after which the topic list is modified and the final scoring pass is performed.

*T.H.Crystal is an integree from the IDA Center for Communications Research, Princeton, NJ.

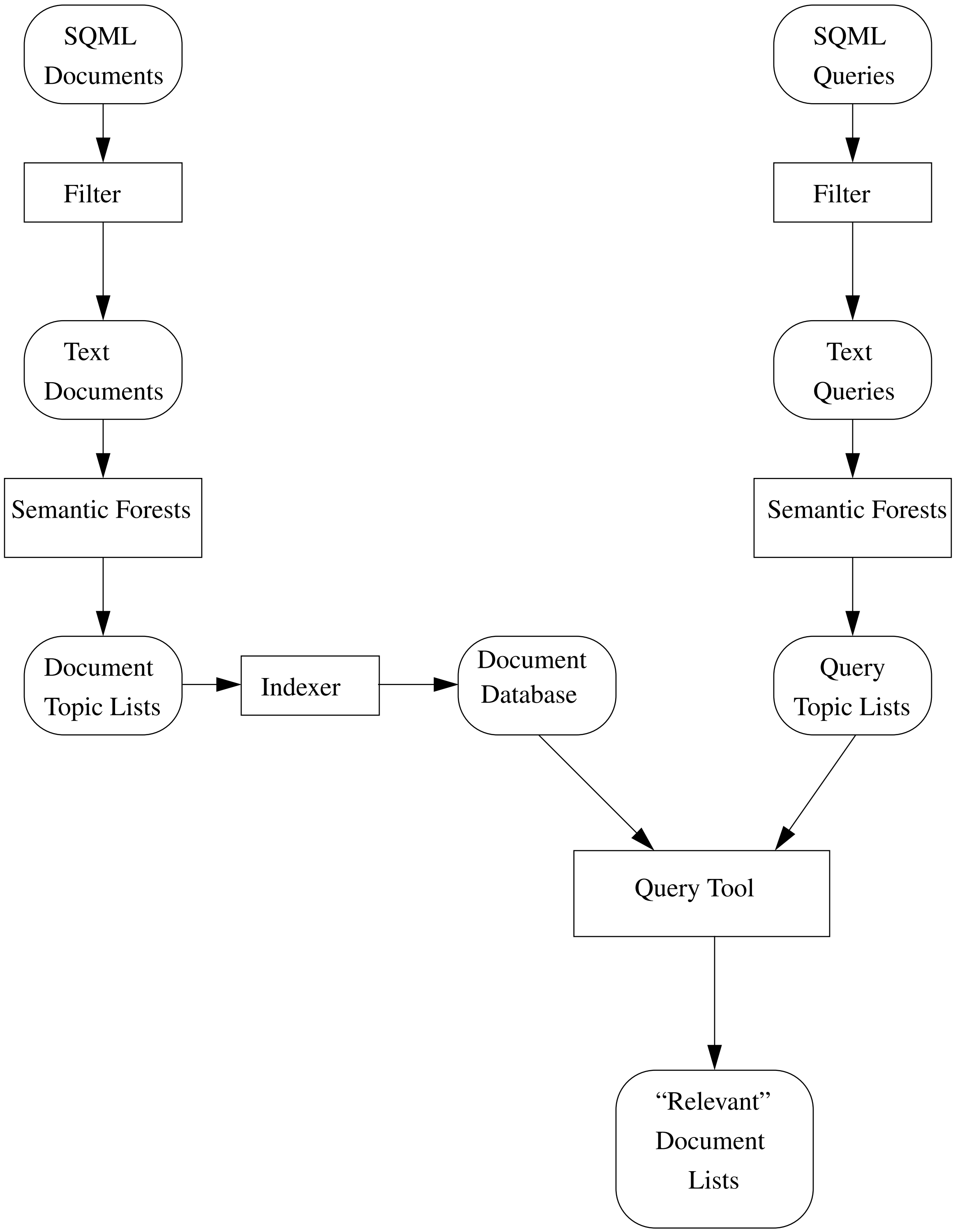

Figure 1. Schematic of the information retrieval system.

3. SYSTEM DETAILS

The Semantic Forests IR system was described in some detail in last year’s conference report [2], so this section details the changes that were made for this year.

3.1. Class Words

We added a new type of topic word (implemented as the part of speech "CLASS") this year. A class is essentially a list of topics which are, in a loose sense, synonyms of the class word. For example, the class "1970s" is the list

'70s 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979.

We reasoned that any document which contains 1972, for example, is relevant to the 1970s. This was an attempt to get "topics" instead of "words" in our topic lists, but the number of classes was somewhat limited (a total of 200). Our classes were created semi-automatically: some, like 1970s, were entered by hand based on our experience; the rest were hand selected from an "inversion" of the dictionary. For instance, the definitions of argon, hydrogen, bromine, chlorine, xenon, helium, fluorine, oxygen, neon, and nitrogen all contain the phrase "gaseous element", so a class "gaseous_element" was created containing those words. To be used effectively, however, the filter which is applied to the incoming text must insert class words when appropriate. Our experiments indicate that this idea can improve performance significantly, but we did not have the opportunity to experiment further.

3.2. Topic Labeling

Significant portions of Semantic Forests were rewritten this past year to improve efficiency and performance. The original version could handle multi-word units (such as united_states) only if they had been entered into the dictionary and joined when the documents were filtered, while the new version recognizes and joins groups of words in the text which are identified as multiword units in the dictionary. The new version also handles words not found in the dictionary differently. For instance, if "Hubble" is not in the dictionary, "hub+unk" is inserted and will match any word with first three letters "hub". This is a compromise that preserves information that would otherwise be lost without adding large numbers of new words and swamping the system. It also provides a primitive stemming for out-of-dictionary words.

Since the definition of a word depends on the way it is used, Semantic Forests should at least be able to identify parts of speech. This, however, requires error free text and natural language processing. Even though Semantic Forests was developed originally to work with imperfect speech recognizer transcripts and, given enough topic relevant text, should sort out the correct definition on its own, we felt that its performance on the *Ad Hoc* task would be improved by a part of speech tagger. To this end, a Bayesian network was added to try to identify parts of speech. While this seemed to help overall, mistakes were made. In a query about "tornados touching down", for instance, "down" was tagged as a noun and "touching" as a descriptive adjective, so "emotion" and "poignant" were given as topic words.

The algorithms for assigning weights to the root and branches of the word trees and for merging the word trees into a topic list were also changed this year.

3.2.1. The Weight of Root Words

Three statistics are used to determine the weight of a root word: *avg*, the average number of occurrences in those training documents containing the word; *fract*, the fraction of training documents which contain the word; and *beta*, the relative importance of the part of speech of the word. Given these statistics, the weight given the root word is

$$wt(word) = beta \cdot avg^2 \cdot \left[\cos\left(\frac{\pi}{2} \cdot min(1,3fract)\right)\right]^4 . \qquad (1)$$

This rewards words which occur frequently in a low fraction of documents as valuable parts of speech (such as proper nouns) and replaces Equation (2) in last year's report.

3.2.2. The Weight of Branch Words

The change here is in the "dictionary salience", *D,* between a child word and its parent. This replaces Equation (4) in last year's report. First a "Z-score" is computed:

$$Z = \left(\frac{\sum freq(parent) \cdot prob(child)}{\sum freq(parent)} - \mu(child)\right) / \sigma(child) \qquad (2)$$

The sums are over all training documents and

$freq(parent)$ is the number of times the parent word appears in the training document.

$prob(child)$ is the probability that the child word appears in the training document.

$\mu(child)$ is the expected value of the probability that the child appears, evaluated over all training documents.

$\sigma(child)$ is the standard deviation of the probability that the child appears, evaluated over all training documents.

A threshold, *Thres*, is set for each child word and

$$D(child, parent) = \begin{cases} wt(child) \cdot \frac{1}{2}(1 + erf(Y))\sqrt{Y} & Y \geq 0 \\ 0 & otherwise \end{cases}, \quad (3)$$

where $Y = Z - Thres$ and the salience of all the root word's offspring are used to distribute the parent's weight to the offspring:

$$wt(child, parent) = \frac{W \cdot D(child, parent)}{\sum D(offspring, parent)} \cdot wt(parent), \quad (4)$$

where the sum is over all offspring of the parent, and *W* is an attenuation coefficient. This algorithm is designed so that the sum of the weights of the offspring is a fixed fraction (*W*) of the parent's weight.

3.2.3. Merging Trees into a Topic List

The simplest way to combine the trees into a topic list is to sum the weights for each word that occurs in a tree. In practice, however, this does not sufficiently emphasize words in common between trees. This year we used two different methods for merging trees: the first is used for documents, and the second for the queries. The first changes the weights of the words, while the second prunes out words which don't occur close together. Our experiments indicate that this combination of algorithms performs best. The words are then sorted into a topic list and the words which are not in the original text are marked as children words. For both methods we start with a list of trees, *1,2, ...n, ... N*, where the root of the *n*th tree is the *n*th sequential word in the given text.

3.2.3.1. Documents

For documents, we compute a modified weight for each word in each tree, then sum the modified weights for each word to produce the score for that word in the document. The modified weight for a word in tree *n* is the sum of the weights for that word in trees *n* through *n+29*, unless the word is a non-root word which occurs only once in those trees. In that case the modified weight is zero.

3.2.3.2. Queries

In this method, we prune the trees of words which are not considered salient. To be salient, a word must be either the root word of the tree (the one that occurs in the query), or a child word which occurs as a child of a different parent in the tree of one of the next 29 trees. This prevents over emphasis of children words simply because their parent occurs frequently.

Next, the words in the pruned trees for the entire query are scored. It is possible for a word to occur more than once in any given tree, so we start by summing the weights for each occurrence to get a score for each word in the pruned tree. *SS(word)*, the sum of the scores over all trees, *SSSS(word)*, the square root of the sums of the squares of the scores over all trees, and *N(word)*, the number of trees in which the word occurs are all computed, and the final score for the word is given by

$$score(word) = \log \frac{SS(word)^{N(word)+1}}{SSSS(word)^{N(word)}} \quad . \qquad (5)$$

3.2.4. Training and Dictionaries

The weights and saliences that are used to make the word trees are computed from a set of training documents. We used the FT, part of FBIS, part of LA and CRE documents to train. Our experience led us to be suspicious of the Federal Register (FR) documents, and, in fact, we did not even include them in the database to be queried. We also felt that the congressional record (CRE) documents were good for training, so we left them in, although we didn't include them in the database.

The "1998 TREC-7 SPOKEN DOCUMENT RETRIEVAL (SDR) TRACK: Specification 3" document, section 10, states that "any auxiliary IR training material or auxiliary data structures such as thesauri that are used must predate the May 1, 1998 recognition/retrieval date". We had planned on modifying our dictionary for the SDR task, as we did last year, to include some common recognizer errors, but could not under our interpretation of this statement. In fact, some extra words were added to our dictionary before we received the rules and we had not kept a copy of our original dictionary. Our dictionary, therefore, complies with the spirit, but not the strict sense, of this rule.

3.3. Retrieval

The goal is to score each document topic list in the database against the query topic list, then take the high scoring messages as relevant to the query. Two scores are used: the full score and the rough score. When feedback is not being used, only the full score is used, but when feedback is being used, the rough score is used for the first pass and the full score is used for the second. Our experiments indicated that using different scores was more effective than a single score. Once we have the final scores for the document segments, they are combined into a single score for the entire document.

The following sections describe the full score, the rough score, how words in classes are handled, and how the segment scores are combined into a document score. The final section describes how the query topic list is modified between the first and second pass when feedback is being used.

3.3.1. Full Scoring of Segments

Except for the *wt* term, this is the same score we used last year:

$$score = hits^3 \cdot \sum_{words} wt \cdot mwct^2 \cdot (idf)^{1.5} \cdot 0.85^{(qrank + srank)} \quad . \qquad (6)$$

The sum is over all words which occur in both the query topic list and the segment topic list.

hits is the number of words in both the segment and query topic list. Words with rank 7 or more in the query topic list are counted as quarter hits.

wt is the weight of the word: 4 for descriptive adjectives, 1 for all others.

mwct is the number of pieces in the word. For example, "united_states" has *mwct=2.*

idf is the total number of segments divided by the number of segments which contain the word.

qrank is the rank of the word in the query topic list.

srank is the rank of the word in the segment topic list.

This score rewards segments containing words which are well ranked in both the segment and the query topic lists, occur rarely in the database, are multi-word units, or are descriptive adjectives. Segments which have many well ranked words in common with the query topic list are further rewarded.

3.3.2. Rough Scoring of Segments

We experimented with a number of scores for the first pass of feedback, with the expectation that the best score to use would give many relevant segments in the highest 10 or 15 scores. The score given here is the one which gave the best results when feedback was used.

$$score = hits^3 \cdot \sum_{words} wt \cdot 0.85^{srank} . \tag{7}$$

As before, the sum is over all words which occur in both the query topic list and the document segment topic list.

hits is the number of words in both the segment and query topic list. Words with rank 7 or more in the query topic list are counted as quarter hits.

wt is the weight of the word: 1/3 for class words, 1 for all others.

srank is the rank of the word in the segment topic list.

This score rewards documents containing words which are well ranked in the segment topic list or are not words in a class. Segments which have many well ranked words in common with the query topic list are further rewarded.

3.3.3. Class Words and Child Words

Class words and child words (those words which do not appear in the original text) are handled differently than the other words in a topic list. Our experiments indicated that using the child words was hurting performance, so we decided not to use them. They do have an implicit influence, however, since they are given a rank and so push down the ranks of subsequent words in the topic list.

For class words, only the word in the class which increases the document score the most is used. In this sense the class behaves as a single word. The term *idf* in the full score, however, uses the average number of segments in which any word in the class appears.

3.3.4. Scoring Documents

Once the segments have been scored, they are ranked, and the scores are combined into document scores:

$$score(document) = \sum_{segments} sscore \cdot e^{-0.01(srank - brank)} . \tag{8}$$

The sum is over all the segments in the document. *sscore* is the score of the segment, *srank* is the rank of the segment, and *brank* is the rank of the best scoring segment of the document.

This takes the top score among the segments in the documents and adds in the lower scores, giving less weight to the segments which score poorly. Our intent is to ensure that a document with only one well scoring segment still scores well.

3.3.5. Using Feedback to Modify the Query Topic Lists

A brief history of our feedback experiments will be useful in understanding the algorithms we used in our submitted runs. We started with an IR system whose average precision at 30 documents was 23.5% on the TREC5 queries with the CR and FT documents. After much experimentation, we achieved an average precision at 30 documents with feedback of 27.0%. At this point we tested the TREC6 queries and were disappointed to see a precision of 23.0% with no feedback drop to 21.7% with feedback. We spent the remainder of our time until the *Ad Hoc* task submission deadline trying to determine why what had helped on the TREC5 queries had hurt on the TREC6 queries. Ultimately, we proceeded on the assumption that we had over trained on the TREC5 data, and the feedback algorithm we ultimately used for the *Ad Hoc* task submission helped the TREC5 queries only moderately, but at least did not hurt the TREC6 queries. Between the *Ad Hoc* and SDR task submission deadlines, we made several refinements to strengthen the feedback algorithm.

3.3.5.1. The *Ad Hoc* (“mild”) System

The first step in modifying the query topic list by feedback is to make of list of “valuable” words from the top documents from the first pass. For the *Ad Hoc* task valuable means the word is one which occurs in 9 or more of the 12 top scoring document segments. If a word in the query topic list is valuable, its rank is improved by 1, otherwise it is made worse by 1. Valuable words which are not in the query topic list are inserted at a rank of

$$6 + \left\lfloor \frac{brank}{5} \right\rfloor ,$$

where *brank* is the best rank of the word in the top 12 segment topic lists.

3.3.5.2. The SDR ("strong") System

For the SDR task, we clustered the 5 top scoring document segments from the first pass using a similarity between segment topic lists given by

$$similarity = \sum_{words} wt \cdot mwct^2 \cdot idf^{1.5} \cdot 0.85^{(rank1 + rank2)} . \tag{9}$$

If there are three or more documents which cluster with the top scoring document, then we use the words in the topic lists of those documents to do feedback. If the cluster has only one or two documents, we do not use feedback for the query. We use a threshold of 75% of the similarity of the most similar pair to do the clustering.

Next the words are scored to determine which ones are valuable. The score we use is

$$score = \sum_{segments} 0.95^{slot} \cdot (0.95 - 0.009^{srank})^{(0.09 \cdot srank)} . \tag{10}$$

The sum is over the segments whose topic list contains the word, and

$$slot = \left\lfloor \frac{rank - 1}{20} \right\rfloor , \tag{11}$$

where *srank* is the rank (starting with *0*) of the segment among the top scoring segments and *rank* is the rank of the word in the segment topic list. A word is considered valuable if it scores better that 55% of the maximum score, which is the number of segments in the cluster.

The query topic list is now divided into a list of valuable words and a list of non-valuable words, each list ordered by rank in the original query topic list. We make a first approximation to the new list by putting the valuable list first, the non-valuable list second and reordering them top to bottom. Words which are valuable but not in the query topic list are all given the same rank as the best non-valuable word in the query topic list. Next we shift all valuable words down by $\lfloor segmentrank/4 \rfloor$, the valuable words in the original query topic list down by $\lfloor queryrank/4 \rfloor$, and the non-valuable words in the original query topic list down by $\lfloor queryrank/8 \rfloor$. If no words were valuable, we make no changes at all and so do no feedback.

4. TIMINGS

We used the same base machine as last year, namely a Sun Enterprise 5000 (250 MHz, 8 heads, and 1040 MB of RAM) with approximately 50% of the CPU time available for our work. The timings for the SDR tasks are given in Table 1. A further 6 seconds were taken in each condition to create the database.

Table 1: Time for Topic Labeling on SDR Tasks

Condition	# messages	Pre-processing &Topic Labeling
Reference	2866	1.6 min
Baseline 1	2865	2.4 min
Baseline 2	2865	2.4 min

Our *Ad Hoc* system last year required over 85 hours to train, which made experimentation difficult. This year we were able to build the whole database for the *Ad Hoc* task in just slightly under 10 hours. These 10 hours do not include the initial filtering of the data since we used the filtered output from last year. We built the database several times, and the wall-clock timings for one of the early builds are shown in Table 2. The times for the Federal Register documents are starred to indicate that we did not use the Federal Register documents in the final version of our database. It took an additional 34 minutes (2043.51 seconds of CPU time) to build the database from the topic labeled documents.

Table 2: Approximate Wall-clock Time for Topic Labeling on Ad Hoc Task

DOC TYPE	# documents	Pre-processing &Topic Labeling
FBIS (in three parts)	61,578; 26,438; 42,455	44 min; 19 min; 48 min
FR (in two parts)	26,843; 28,787	22 min *; 28 min *
FT (in two parts)	76,857; 133,301	97 min; 157 min
LA (in three parts)	43,803; 54,603; 34,210	75 min; 67 min; 41 min

Timings for the query process are given in Table 3.

Table 3: Time for Queries

TASK	Time
Ad Hoc: Without feedback	558.6 s
Ad Hoc: With feedback	1468.2 s
SDR: Reference 1 transcripts	4.2 s
SDR: Baseline 1 recognizer transcripts	4.6 s
SDR: Baseline 2 recognizer transcripts	4.5 s

5. RESULTS.

The results for the runs we submitted are given in Tables 4 and 5. We submitted two runs for the *Ad Hoc* task: our primary run used no feedback and our secondary run used feedback. Our SDR runs used feedback. The average precision at 30 documents on the *Ad Hoc* task, for instance, increased from 19% last year to nearly 27% this year. As we expected, the *Ad Hoc* feedback run shows only small changes over the non-feedback run, but in most cases it gives an improvement.

6. EXPERIMENTS

We decided to use the 50 TREC5 (1996) *Ad Hoc* queries as a development set and the 50 TREC6 (1997) *Ad Hoc* queries as a test set for our experiments. There are three document sets (CR, FR and FT) in common between these years, but our experience indicated that our system did not do well with the FR documents, so we decided to use only the CR and FT documents to query against. The CR documents are not used in TREC7, but using the FT documents alone seemed dangerous. For the SDR task, we used the 5 queries given in the TREC7 training data and queried against the IBM ltt files from set 1 and set 2.

6.1. Feedback

Table 6 shows the effect of feedback on the development and test data. “Mild feedback” refers to the method used for the *Ad Hoc* task submission (section 3.3.5.1), while “strong feedback” refers to the method used for the SDR task submission (section 3.3.5.2).

Table 4: Recall Level Precision Averages

Recall	Precision				
	Ad Hoc no feedback	Ad Hoc feedback	SDR 0% WER	SDR ~35% WER	SDR ~50% WER
0.0	0.7277	0.7184	0.7866	0.7713	0.6688
0.1	0.4203	0.4150	0.6195	0.6294	0.4909
0.2	0.2657	0.2620	0.5937	0.5414	0.4662
0.3	0.1775	0.1891	0.5387	0.4839	0.3946
0.4	0.1275	0.1441	0.4544	0.4133	0.3271
0.5	0.0824	0.1005	0.4228	0.3880	0.2830
0.6	0.0528	0.0665	0.3399	0.3080	0.2249
0.7	0.0290	0.0383	0.2476	0.2202	0.1701
0.8	0.0144	0.0238	0.2199	0.2038	0.1558
0.9	0.0032	0.0070	0.1687	0.1281	0.0946
1.0	0.0002	0.0001	0.1105	0.1002	0.0748
non-interpolated	0.1449	0.1537	0.3907	0.3640	0.2868

Table 5: Document Level Precision Averages

Number of Documents	Precision				
	Ad Hoc no feedback	Ad Hoc feedback	SDR 0% WER	SDR ~35% WER	SDR ~50% WER
5	0.4400	0.4680	0.4870	0.4870	0.3913
10	0.3700	0.3860	0.3478	0.3783	0.2957
15	0.3387	0.3360	0.2957	0.3188	0.2667
20	0.3060	0.3120	0.2696	0.2717	0.2370
30	0.2693	0.2693	0.2275	0.2362	0.2000
100	0.1612	0.1672	0.1161	0.1048	0.0983
200	0.1110	0.1171	0.0622	0.0583	0.0570
500	0.0644	0.0674	0.0277	0.0269	0.0270
1000	0.0400	0.0413	0.0146	0.0143	0.0144
Exact	0.1965	0.2048	0.3703	0.3442	0.2475

Table 6:

Task	Feedback Condition	avg. precision @30 documents	avg. precision @ R documents
Ad Hoc Development Set	none	21.481%	26.752%
	mild	22.444%	28.153%
	strong	23.259%	32.152%
Ad Hoc Test Set	none	20.069%	17.612%
	mild	20.069%	17.702%
	strong	20.069%	17.702%
TREC7 SDR training	none	28.000% (@5 docs)	23.684%
	strong	32.000% (@5 docs)	28.947%

6.2. Topics versus Words

For the SDR training data, we looked at the topic lists of the relevant documents to see how the document list for our queries could be modified to be more successful. For example, the query topic list for the third query ("What have the effects of the U.N. sanctions against Iraq been on the Iraqi people, the Iraqi economy, or world oil prices?") was:

1 - iraq<PLACE>	6 - iraqi<ADJ_NOUN>
2 - oil<NOUN>	7 - sanction<NOUN>
3 - economy<NOUN>	8 - price<NOUN>
4 - effect<NOUN>	9 - sanction<ACTION_VERB>
5 - world<NOUN>	10 - people<NOUN>

The topic lists for the documents judged relevant for this query contained a number of different words to denote the topics "iraq" (iraq<PLACE>, iraqi<ADJ_NOUN>, baghdad<PLACE>), "oil" (oil<NOUN>, gas<NOUN>, petroleum<NOUN>, crude_oil<NOUN>) and "effect" (effect<NOUN>, impact<ACTION_VERB>, impact<NOUN>). When we supplemented the topic lists for all the queries (by hand) to contain additional words from the relevant documents, our average precision at the number of relevant documents went from 28% to 50%.

We concluded that a topic labeling system which returned "topics" - collections of words, any one of which could represent the topic - had the potential of performing much better than one which only returned single words. It seems difficult to get this information, since it requires finding words which have similar meanings, which requires passing from a word to its definitions, then to words which have similar definitions. We experimented with this through the class words described above, and although our list of classes was very small it did improve performance: the average precision at 30 documents for the development set with no feedback was 20.4% with no class words and 21.3% with class words. The test set had many fewer class words (probably a deficiency in our filter), so the effect was less, but in the same direction.

6.3. Children words

The Semantic Forests algorithm is designed to produce topic words which are not necessarily in the original document ("children words"). We had hoped that these words would help performance, and were disappointed to see that they did not. For instance, on the development set, without feedback, the average precision at 30 documents was 21.3% without the children words and dropped to 20.5% with the children words. We decided not to use the children words for our submitted runs, but feel that this indicates that Semantic Forests (or the dictionary we are using) can be improved further.

6.4. Natural Language Processing of Queries

We attempted to do some very primitive natural language processing on the queries to help eliminate phrases which contained little information about the topic, such as “relevant documents will identify”. Phrases which can be anticipated can be handled by a filter, but that is not a very robust solution. We tried to identify adjective-noun groups on the premise that they were the most topical, but our initial experiments indicated this was hurting the performance and we abandoned this approach for lack of time. This still seems like an important avenue to pursue, however.

6.5. Separate *vs.* Combined Databases

We saw a modest gain in performance (23.9% average precision at 30 documents increased to 24.2%) when we ran against two separate databases (CR and FT in one and FR, FBIS and LA in another) compared to one single database. Separate databases affect the scoring only in the *idf* term of equation (6), since the counts of the number of times a word appears in the documents changes. We saw a drop in performance if the databases were too small, so there may be an optimum size for the databases, or the increase we saw may be pure chance. In any case, we had difficulty implementing feedback with multiple databases, and so chose to run with a single database.

6.6. Rough Scores

Not unexpectedly, the rough score (7) performed worse than the full score (8): on the development set, the average precision at 12 documents was 26.3% for the rough score and 28.0% for the full score. We were surprised, therefore, to find that using the rough score for the first pass (and the full score for the second pass) of feedback did better than using the full score for the both passes: 23.4% for the rough score and 20.6% for the full score. The difference was not as great on the test set, but the trend was the same. We were unable to find a good explanation for this, but chose to run with two separate scores for feedback.

6.7. Federal Register (FR) Documents

When the FR documents were added to the database, the average precision at 30 documents dropped from 21.3% to 18.4% on the development set and from 20.1% to 17.6% on the test set. We are not sure why our system has such trouble with the FR documents, but the effect was dramatic and we decided not to use them.

6.8. Document Segments

Division of the documents into contiguous segments containing no more than 500 words gave an increase in the average precision at 30 documents from 18.0% to 21.3% on the development set and from 15.6% to 20.1% on the test set, a sizeable gain. The 500 word limit was chosen to give relatively small segments and a manageable database.

6.9. Different Tree-Merging Methods

The decision to use two separate tree-merging methods for documents and queries was motivated by the experiments shown in Table 7. The two methods are described in Sect 3.2.3 as the “document method” and the “query method”, reflecting their ultimate use. The last row shows the best performance and the best improvement from feedback.

Table 7: Effect of Tree-Merging Methods on Ave. Precision at 30 Documents.

Method used for Documents	Method used for Queries	Dev. set no feedback	Dev. set feedback	Test set no feedback	Test set feedback
Query	Document	16.8%	18.7%	11.9%	12.1%
Query	Query	18.1%	17.6%	13.3%	13.3%
Document	Document	19.9%	21.5%	19.9%	17.2%
Document	Query	20.9%	23.2%	19.9%	19.8%

7. Summary

We continue to view Semantic Forests, especially with its current improvements, as an interesting and effective approach to IR. Implicit in this approach is the use of a dictionary, not only for building trees, but also in controlling the processing of words and in constructing classes of words. It should be noted that there are no specific stops words but rather the word statistics in the dictionary control what document words become topic words. Some of our new classes were developed by inverting the dictionary.

We continue to be disappointed that the explicit use of child words as topic words is not useful. (We seem to be reaccumulating the wisdom of others who have had problems with query and document expansion.) They do, however, have great influence, through the tree building and merging process, on the selection, ranking and weighting of topic words.

Our experience with Semantic Forests has helped us identify several areas for achieving marked improvement. The first of these is parsing of the queries. Ideally, we would like to automatically produce a set of filtered text to be used to generate desired topics and a set of filtered text for topics to avoid. A first pass could be made to get all documents which match the desired topics and a second pass made to eliminate the ones which discuss unwanted aspects of the topic.

Also, we need a better way of generating lists of words which constitute topics. The few experiments we were able to perform indicate that this could improve performance significantly, but these lists seem very language dependent and we are not sure how to extract the desired information from a dictionary or thesaurus. An extreme case is one of the training queries for the SDR task, which asks for documents relating to scandals in the Clinton cabinet. To return the relevant documents, our system would have to know that "Clinton cabinet" matches "Hazel O'Leary", and this kind of information is not in an ordinary dictionary.

Finally, we would like to improve our part of speech tagger and find a way to assign probabilities to multiple definitions of a word. Our primary interest, however, is in transcribed speech, where the text is imperfect and these methods would be less effective or ineffective.

8. REFERENCES

1. Schone, P., Nelson, D., "A Dictionary-Based Method for Determining Topics in Text and Transcribed speech," 1996 IEEE International Conference on Acoustics, Speech, & Signals Processing, Atlanta, Georgia, May, 1996; Vol. 1, pp. 295-298.

2. Schone, P., Townsend, J., Crystal, T., Olano, C., "Text Retrieval via Semantic Forests," The Sixth Text Retrieval Conference (TREC-6). NIST Special Publication 500-240, 1998, pp. 761-773.

3. Voorhees, E., Harmon, D. "Overview of the Seventh Text REtrieval Conference (TREC-7)", presented at TREC-7.

4. Garafolo, J., "Overview of the Spoken Document Retrieval Track", presented at TREC-7.