FDU at TREC-9: CLIR, Filtering and QA tasks

Lide Wu, Xuan-jing Huang, Yikun Guo, Bingwei Liu, Yuejie Zhang
Fudan University, Shanghai, China

This year Fudan University takes part in the TREC-9 conference for the first time. We have participated in three tracks of CLIR, Filtering and QA.

We have submitted four runs for CLIR track. Bilingual knowledge source and statistical-based search engine are integrated in our CLIR system. We varied our strategy somewhat between runs: long query (both title and description field of the queries involved) with pseudo relevance feedback (FDUT9XL1), long query with no feedback (FDUT9XL2), median query (just description field of queries involved) with feedback (FDUT9XL3) and, the last, mono long query with feedback (FDUT9XL4).

For filtering, we participate in the sub-task of adaptive filtering and batch filtering. Vector representation and computation are heavily applied in filtering procedure. 11 runs of various combination of topic and evaluation measure have been submitted: 4 OHSU runs, 1 MeSH run and 2 MSH-SAMPLE runs for adaptive filtering, and 2 OHSU runs, 1 MeSH run and 1 MSH-SAMPLE run for batch filtering.

Our QA system consists of three components: Question Analyzer, Candidate Window Searcher and Answer Extractor. We submitted two runs in the 50-byte category and two runs in the 250-byte category. The runs of "FDUT9QL1" and "FDUT9QS1" are extracted from the top 100 candidate windows. The other two runs of "FDUT9QL2" and "FDUT9QS1" are extracted from the top 24 candidate windows.

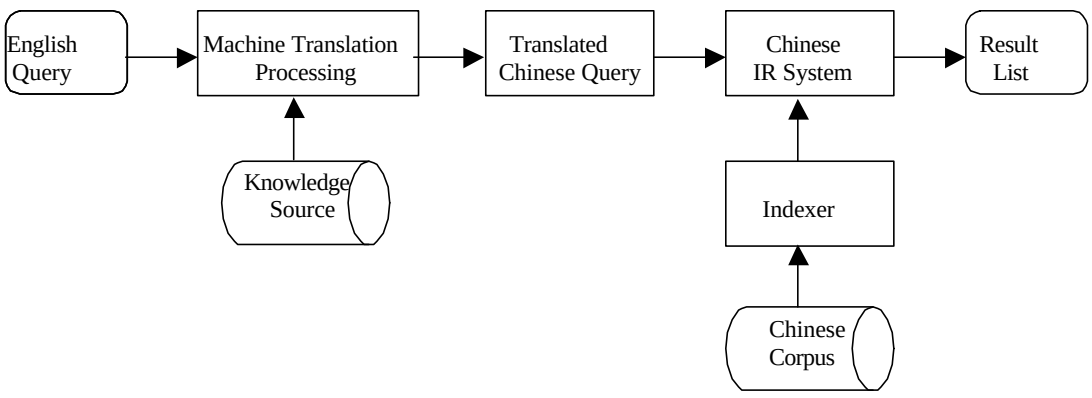
1. Cross-Language IR

We focused our attention on Chinese document indexing and query translation. All query processing was fully automatic and both long and short query translation are covered.

Our overall strategy to CLIR task is to translate English query into Chinese word list, since we feel it is not feasible to build a document translation system in such a short period, while it is much more reasonable to disambiguate word sense in context of long query by statistical approach, such as POS and knowledge. Once queries have been translated, we use IR techniques, which is a variant of MIT's approach[1] and probabilistic methods to obtain relevant document list. The whole corpus has been indexed with Chinese NLP techniques developed by our group in recent years [2]. Finally, we also explore the weight of words in both of the title and description fields.

The system infrastructure is illustrated in figure 1.1. Description of each part is followed.

Figure 1.1 System architecture for CLIR



1.1 Indexer

We explored two different indexing methods for our CLIR task, one is word-based Chinese indexing module, and the other is n-gram based indexing module. Because Chinese is different from English in that there are no extra spaces between Chinese words, we must first segment Chinese character sequence into words or n-grams in order to index documents.

1.1.1 Chinese word segmentation module

Figure 1.2 illustrates the architecture of our word segmentation sub-system. Given a document, it is first divided into a sequence of sentences (sub-sentences) by punctuation such as full stop, comma, etc. Each sentence then passes through the sentence segmentor and is segmented into a sequence of words. Finally, a text-level post-processor will act on the word sequence and generate the final segmentation result.

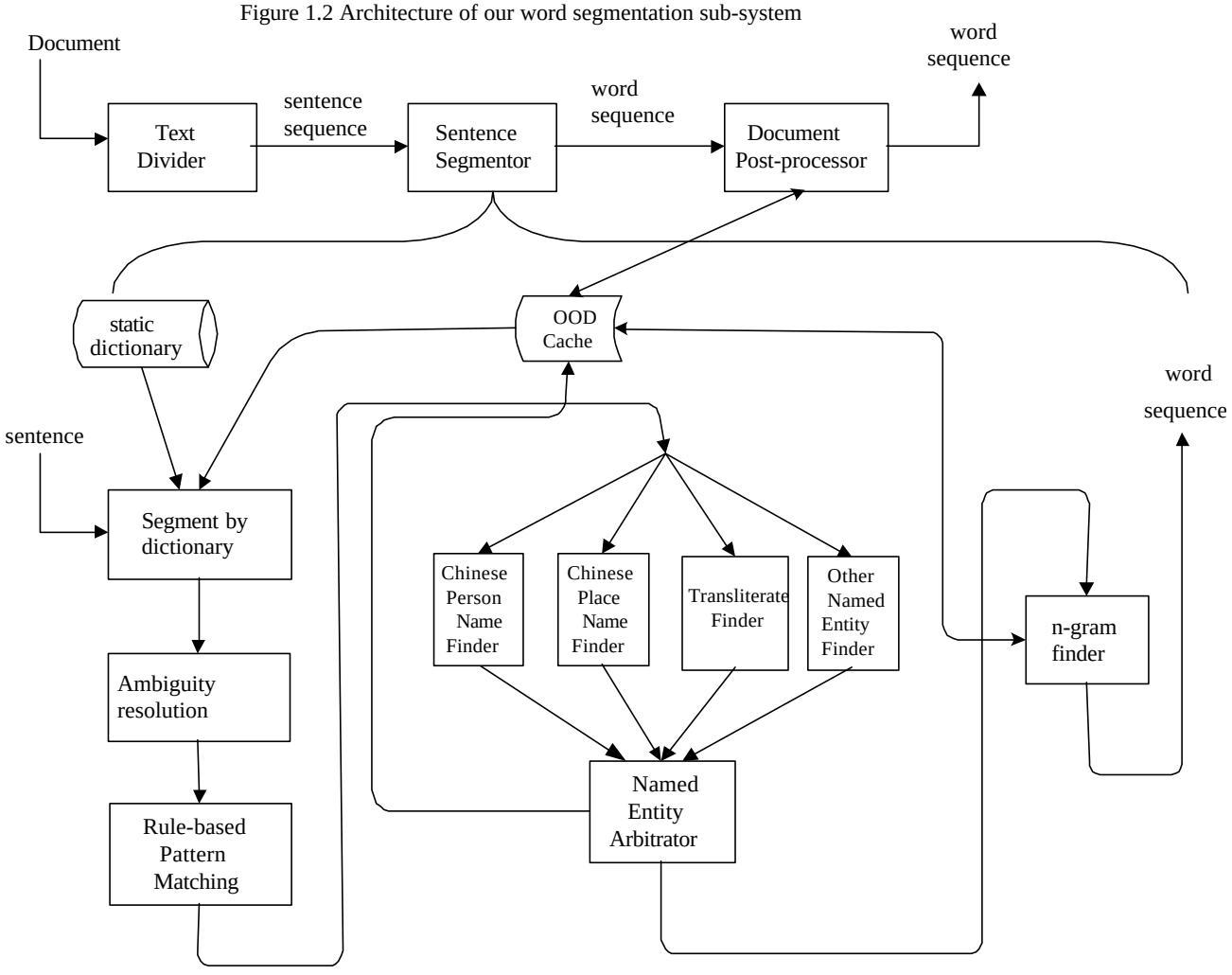
- Dictionary

Two kinds of dictionaries are used during the segmentation process. One is the static dictionary, which records Chinese lexical words and is unchangeable. The other is the OOD (out-of-dictionary words) cache, which records the newly found OOD and changes dynamically.

- Sentence segmentor

The input sentence is first segmented by both static and dynamic dictionary. Ambiguous strings are handled at the same time. We use a pattern-matching module to recognize those OODs with fixed structure pattern, such as money, date, time, percentage and digit.

Figure 1.2 Architecture of our word segmentation sub-system



The recognition module of person's name, place, organization and transliteration is more complex. Contextual and structural information both play important roles in identifying these kinds of OODs. The former can provide external evidence for deciding word boundary and predicting the category of OOD. The latter can provide internal evidence for suggesting and validating the appearance of certain OOD. For example, in Chinese, family surnames are stereotypical. We've made lots of statistical analysis on various categories of OOD and built correspondent identification modules for each. Each module works independently. A named entity arbitrator will take effect when two or more kinds of name entities conflict with each other and select the most probable one.

Beside these OOD types mentioned above, there are still many other kinds of OODs. We can also recognize some OODs according to its string frequency and internal characters' mutual information.

- Document post-processor

During the sentence-by-sentence segmentation, some OODs will not be recognized until they occur several times. Therefore, the OOD cache changes continuously until the whole document passes through the sentence segmentor. After that, a document post-processor based on the final content of the cache is necessary to detect missed or mistakenly segmented words before.

1.1.2 n-Gram based Tokenization

We also implement n-Gram based tokenization process, which does not need sophisticated segmentation method. The document is simply cut into sequence of bigrams. We want to know whether the effectiveness of IR based on n-gram is comparable, inferior or superior to that based on word segmentation.

1.1.3 Word Indexing

Every document in the corpus is cut into no more than 64K segment to make indexing procedure more robust and normalize the document length. After being segmented, text id, term frequency, document frequency and term position are stored for the task. No stop word is removed from the invert file, since the corpus is rather small.

In order to optimize the disk space and I/O in retrieval time, we have also implemented invert file compression. The file was then decrease to about one half of its original size.

1.2 Query Translation

The essence of cross-language information retrieval is to use queries in one language to retrieval documents from a pool of documents written in other languages. This may be achieved by using query translation, document translation, or by using both query and document translation.

Here, we adopt query translation as the dominant strategy, use English query to be translated object, and utilize English-Chinese bilingual dictionary as the important knowledge resource to acquire correct translations. So by using our Chinese Information Retrieval system, the complete English-Chinese CLIR process can be implemented successfully.

1.2.1 Knowledge Source Construction

The knowledge source used in English-Chinese-oriented CLIR system mainly includes dictionary knowledge and Chinese Synonym Dictionary. In addition, stopword list and word morphological resumption list are also utilized in our system. In fact, dictionary is a carrier of knowledge expression and storage, which involves almost all information about vocabulary, namely static information.

(1) English-Chinese Bilingual Dictionary

This dictionary is mainly used in translation processing in word level and phrase level. And it consists of three kinds of dictionary component as follows:

- Basic Dictionary--A basic knowledge source independent of particular field, which records basic linguistic vocabulary;
- Technical Terminology Dictionary –Recording terminology knowledge in a particular technical field, which is mainly referred to Hong Kong commercial terminology knowledge and incorporated in the basic dictionary;

- Idiom Dictionary--Recording familiar fixed matching phenomena, such as idiom and phrase.
- The whole bilingual dictionary involves almost 50,000 lexical entries. And each entry is established as the following data structure:

English Lexical Information	Part-of-Speech Information	Subcategory Information	Concept Number	Matching Information	Semantic Class Code	Chinese Lexical Information

Two examples of particular entry representation form in dictionary are listed as the following:

**happiness || n || ng || 0 || M ;[U]; || bbaaa ||* 幸福 *(felicity) ||||*

**handle || v || vt || 5 || Wv3;T1]; || CBBC ||*处理 *(processing) ||||*

(2) Chinese Synonym Dictionary (同义词词林)

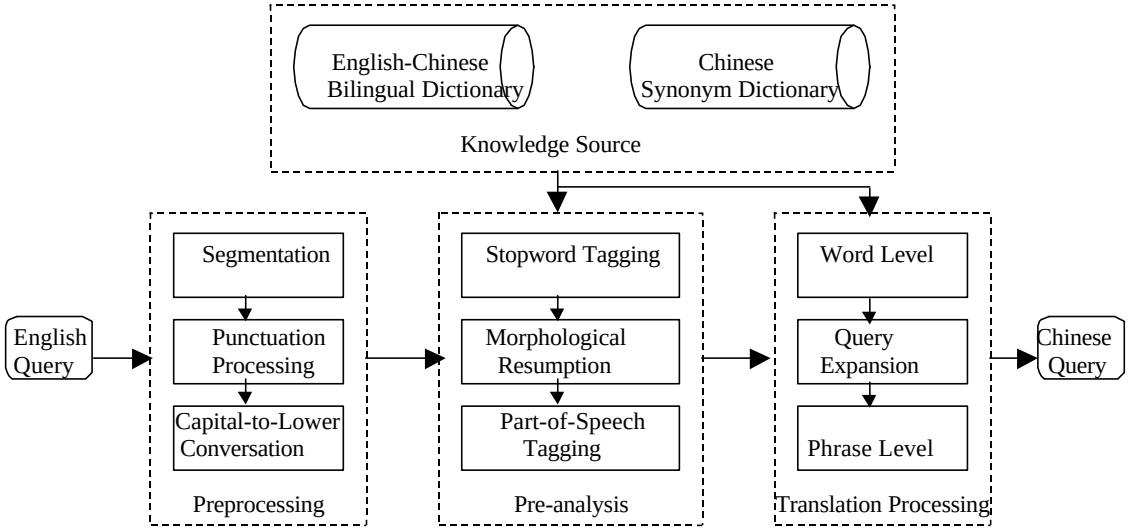
Actually, this dictionary is a thesaurus, which involves nearly 70,000 entries. All entries are arranged according to specified semantic relations. It is mainly used in expanding translation that has passed through translation processing, namely query expansion.

While the stopword list is used in tagging the stopwords in English query, and the English morphological resumption list which describes all irregular varieties about vocabulary is used in morphological resumption of words with irregular variety forms.

1.2.2 Translation algorithm

The basic framework of English-Chinese-oriented translation algorithm is mainly divided into three parts, as shown in Figure 1.3.

Figure 1.3 Basic framework of English-Chinese-oriented query translation algorithm



- Preprocessing -- including sentence segmentation, punctuation tagging and capital-to-lower letter conversation for English query;
- Pre-analysis -- including stop words tagging, word morphological resumption and POS tagging processes;

Considering that translation processing is related with some stopwords, the stopwords must be tagged by stopword list. Because there are some words with variety forms in English query, translation knowledge cannot be induced correctly. So by using English-Chinese bilingual dictionary, morphological resumption lists for irregular variety and heuristics for regular variety, we get words' original form from the process called "morphological resumption". To analyze word part-of-speech, we develop a HMM-based (Hidden Markov Mode) Part-of-Speech Tagger.

- Translation processing -- including translation processes in two levels: word level and phrase level.

Word level translation: By using the basic vocabulary part of English-Chinese bilingual dictionary, this process mostly implements translation word by word. For word disambiguation, a word may correspond with several kinds of different sense. Word sense is related with particular word, and

cannot be given without particular linguistics environment. The condition of linguistics environment may be syntactic and semantic parameters. When selecting a particular word, the difference mark of word should be chosen. This difference mark represents a certain syntactic and semantic feature, and identifies the sense of word uniquely, namely Concept Code. The concept code together with lexical entry can decide a certain word sense to accomplish word sense disambiguation. For machine translation, word disambiguation should be a very important problem. But in our CLIR system, in some degree, word disambiguation has not taken some obvious affect to retrieval efficiency. At the same time, in order to provide more query information to retrieval system, by using "Chinese Synonym Dictionary", expansion operation is done for translation knowledge through translation processing. According various synonymous relations described in the dictionary above, all synonyms corresponding with translation knowledge is listed, namely completing query expansion process. Thus, more affluent query information can be provided to retrieval system. So the retrieval efficiency is increased greatly, and the retrieval performance is improved.

Phrase level translation: This process is implemented based on the idiom dictionary part of English-Chinese bilingual dictionary. The recognition of near distance phrase and far distance phrase is an important problem. Here, by adopting Greedy Algorithm, the recognition and translation processing of near distance phrase is mainly completed, shown as the following:

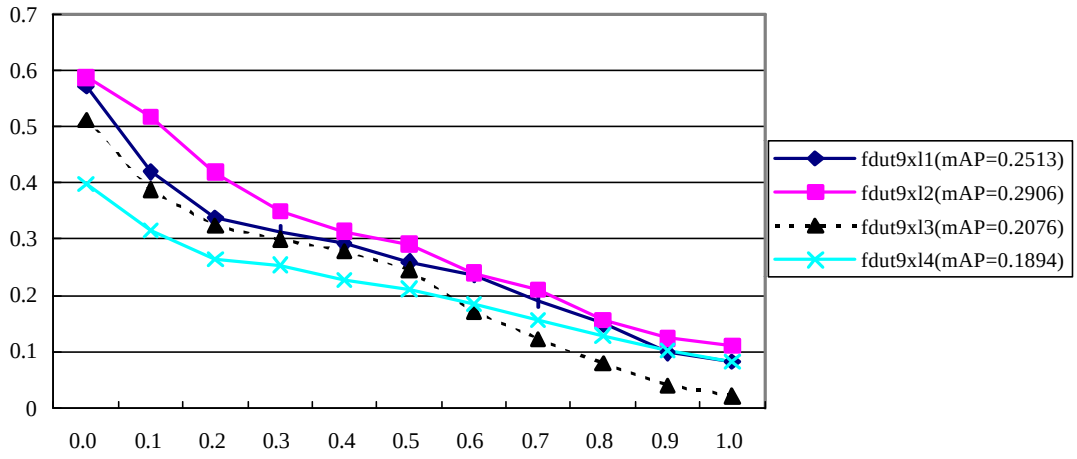
- Acquiring phrase set which have some query word as the head word of the idiom from English-Chinese bilingual dictionary;
- Identifying the phrases which have the same word as head word and the same number of word as the phrase in the above set;
- Comparing each one of the identified phrases and every member in the correspondent phrase set and finding out the matched phrase with the maximum length.

1.3 Experiment

Our search engine scores document by maximum likelihood ratio, put forward by Spoken Language Systems Group in MIT [1]. In our retrieval experiment, we use the TREC-5 Chinese task as the "training" data set for tuning and optimizing our retrieval model. Finally, our best run has achieved the mAP (mean average precision) of 0.3869, which is about the same as the best result at that time.

After that, we submit four runs for CLIR official evaluation this year. Figure 1.4 is the official precision and recall curve and the mAP score of our 4 CLIR runs. The first three of them are automatic query translation run, using our word segmentation approach for indexing, while the monolingual run we submit uses n-gram based segmentation. Although the results are not as good that of training results, the run of "fdut9xl2" still can achieve the mAP of near 0.30.

Figure 1.4 official Prec and Recall Curve and mAP score on TREC-9 CLIR Task



We have outstanding performance for automatic query translation run: most of the queries

outperform the average in the run "fdut9xl1". However, the monolingual run is not as good as we expected. We speculate that it may be due to our sophisticated segmentation method, which could correctly segment the names of people, place and organization etc. In other word, indexer based on word segmentation performs much better than indexer based on n-gram.

2. Filtering

For filtering, we participate in the sub-task of adaptive filtering and batch filtering. Our research focuses on how to create the initial filtering profile and threshold and then modify them adaptively.

Our batch runs share the same adaptation module with adaptive runs. Therefore, our batch runs are actually batch adaptive runs.

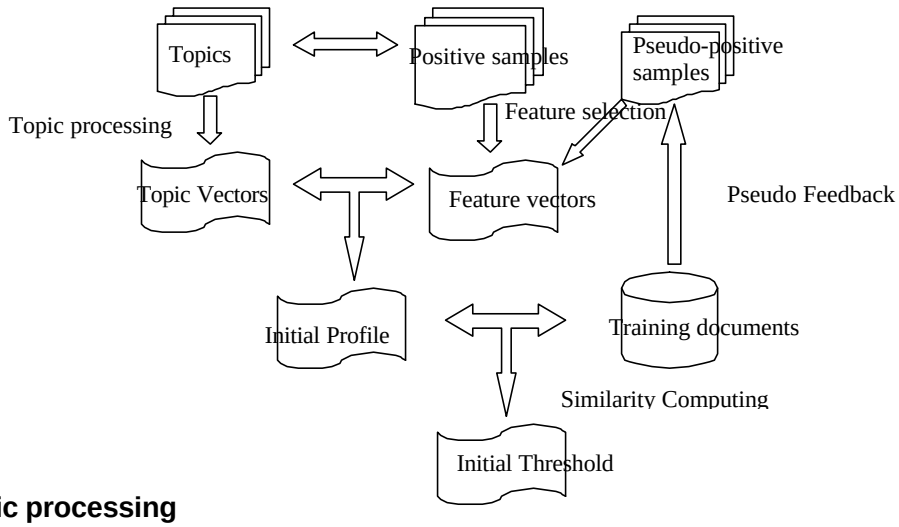
There is only a slight difference in the initialization (in other word, training) module of our batch and adaptive runs. Full relevance judgement is provided in batch filtering, while only a small proportion of relevance judgement is provided in adaptive filtering. As a result, for batch run, we can obtain a set of "Negative" documents, which are those irrelevant documents with high similarity to the filtering profile, and then make use of such documents. For adaptive run, we try to discover more pseudo-relevant documents based on the topic and limited relevance judgement in order to optimize the initial profile.

Following is the detailed introduction to our training and adaptation module of our adaptive and batch task.

2.1 Training of adaptive filtering

Figure 2.1 shows the architecture of the training in adaptive filtering. At first, topics are changed into topic vectors, while feature vectors are extracted from positive and pseudo-positive document samples. The initial profile is the weighted sum of topic and feature vectors. Then we compute the similarity between the initial profile and all training documents to find the optimal initial threshold for every topic.

Figure 2.1 Architecture of the training in adaptive filtering



2.1.1 Topic processing

Topic is being processed as such: Firstly, every word in the topic is labeled with one of four attributes: title words; description words; negative words (words which behinds the word "without"); domain dependent stopwords such as "*document*" and "*describe*". Each kind of attribute is assigned with a coefficient. If one word occurs several times in the topic and different attributes are labeled, the maximum coefficient is chosen for it. Different coefficients are chosen for OHSU and MeSH topics.

As we know, for OHSU topic, title is the description of patient, and description is the information request. Both are important. However, for MeSH topic, title is MeSH concept name and description is the definition of the concept. We have found the description part is not as important as the title. For example, the description of the concept of "*abdomen*" is that "*the portion of the body that lies between*

the thorax and the pelvis". If we expand the initial query with such word as "*thorax*" or "*pelvis*". the performance will even be hurt.

In our experiment, the coefficients of OHSU topics are set to 1, 1, -1 and 0 respectively, while those of MeSH topics are 1, 0, 0 and 0 respectively.

The weight of each topic word is set to be the product of its coefficient multiplied with Smart's *ltc* weight: *ltc*=log(*N/n*)[3], where *N* is the total number of documents and *n* is the number of documents in which the word occurs. We adopt *ltc* weight because the simplicity in computing. We have also tried other weighting formulas with relevant information and do find they can lead to better performance when only topic information is utilized in profile creation. However, once topic vector is combined with feature vectors from the training documents to form the initial profile, more complicated weight algorithm no longer ensures better performance.

2.1.2 Feature selection

Since the total number of all words is very large and then it cost more time in similarity computation, we decide to select some important words from them. First, we use Porter's stemmer to get the root form of every word. Then we remove the stopwords and low frequent words (occur no more than 6 times in the training document sets). Then we compute the *logarithm Mutual Information* between remaining words and topics:

$$\log MI(w_i, T_j) = \log\left(P(w_i \mid T_j) \Big/ P(w_i) \right) \qquad (2.1)$$

Where, w_i is the ith word and T_j is the jth topic. Higher logarithm Mutual Information means w_i and T_j are more relevant. P(w_i) is estimated by *maximal likelihood method*. Since the total number of relevant documents is very small, $P(w_i \mid T_j)$ is estimated by *Turing-Good method.*

For each topic, we select those words with logarithm Mutual Information higher than 3.0 and occurs more than once in the relevant documents. Thus the average feature number of each topic is around 100. Logarithm Mutual Information is not only used as the selection criterion, but also as the weight of feature words.

2.1.3 Creating initial profile

Each topic profile is represented by a vector which is the weighted sum of topic vector, feature vector from positive (relevant) documents and feature vector from pseudo relevant documents with the coefficient of A, B and C.

We add a procedure of **pseudo feedback** to acquire more relevant documents for training. Those documents that have highest similarity and don't occur in the positive documents are regard to be relevant.

The initial profile is created by two-phase method. First we get the pseudo relevant documents; then we get the initial profile and re-compute the optimal initial threshold.

During the first phase, A, B and C are set to be 0.4:1.0:0. We set C to 0 because we have no similarity score at this time. Then some documents are selected as pseudo positive documents. As for how many documents should be appended, we adopt two methods. The first method choose the N highest similar documents which don't occur in the positive documents; while the second method choose those documents whose similarity is higher than a fixed scale (α) of the highest similar positive document. These two method lead to similar results. Here we set α = 0.45, or N=10.

During the second phase, A, B and C are set to be 0.25:1.0:0.25. The parameter of A becomes smaller because now we have so much positive document that topic vector becomes relatively less important. Therefore, we have got the initial profile.

2.1.4 Similarity Computation

The similarity between the profile and training documents is computed by the cosine formula:

$$Sim(d_i, p_j) = Cosq = \sum_k d_{ik} * p_{jk} \Big/ \sqrt{(\sum_k d_{ik}^2)(\sum_k p_{jk}^2)} . \quad (2.2)$$

Where, p_j is the profile vector of the jth topic and d_i is the vector representation of the ith document. d_{ik}, the weight of the kth word in d_i, is computed as such: $d_{ik} = 1 + \log tf_{ik}$, where tf_{ik} is the term frequency of the kth word in the ith document.

Documents are first removed of the redundant tag information and then stemmed. Only the document identifier, tile and abstract are reserved, while MeSH headings and other fields are all removed.

2.1.5 Setting initial threshold

Once the threshold is set, those documents with similarity greater than the threshold are regarded to be relevant and those documents with similarity smaller than the threshold are regarded to be irrelevant. Then we can compute the evaluation criteria such as T9U and T9P under different threshold. Thus the initial threshold are set to be the threshold which can result in the largest T9U or T9P.

2.2 Training of batch filtering

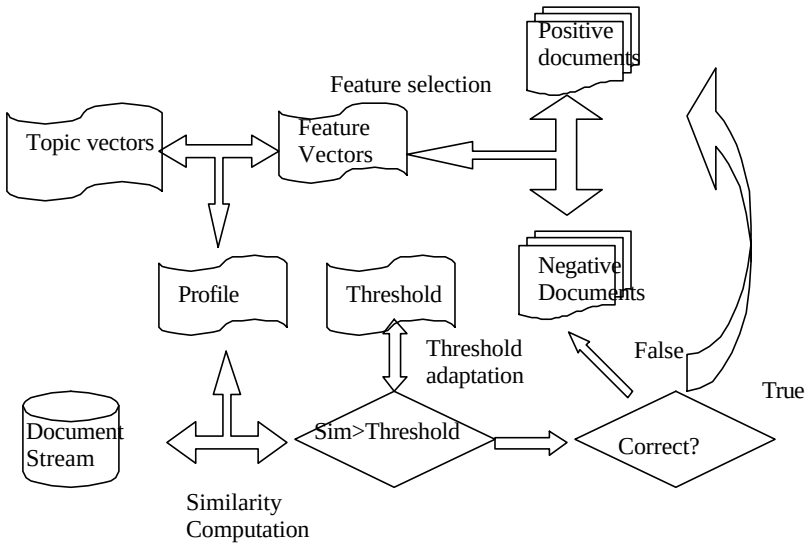
Training of batch filtering is quite similar to adaptive filtering. The only difference is that feature vectors are now extracted from positive and negative (irrelevant but with high similarity) document samples. Each topic profile is represented by a vector which is the weighted sum of topic vector, feature vector from positive documents and feature vector from negative documents with the coefficient of A, B and C.

Initial threshold is also set in a two-phase method. At the first phase, A, B and C are set to be 0.25:1.0:0. For each vector profile, we calculate its similarity with every training document and then set the temporary similarity threshold. After that, negative documents are selected to be those irrelevant documents with similarity higher than the temporary threshold and could lead to wrong judgement. During the second phase, A, B and C are set to be 0.25:1.0:-0.25.

2.3 Adaptation

For adaptive and batch filtering we adopt the same adaptation procedure. Figure 2.2 shows the architecture for the adaptation. For each document in the stream, its similarity with the specific topic profile is computed. If the similarity is greater than the threshold, it is assumed to be relevant. Then we search the "qrel" file to see whether it is really relevant and do some adaptation accordingly.

Figure 2.2 Architecture for the adaptation



2.3.1 Adaptation of threshold-T9P

Thresholds are adjusted after b documents have been processed (for this experiment, b =8000). For different evaluation measure of T9P and T9U, the adaptation is also different.

In order for the optimization of T9P, the purpose of threshold adaptation is to make sure that about 50 documents are retrieved during 4 years. Therefore M documents should be retrieved in the b-document interval. For each topic, we define:

Cor: # of documents correctly retrieved in the interval
Rtv: # of documents retrieved in the interval
Cor1: # of documents correctly retrieved heretofore
Rtv1: # of documents retrieved heretofore
M1: # of documents should be retrieved heretofore
T: Similarity threshold
Algorithm:
If Cor<Rtv*0.20 && Rtv>max(M,4) , then T*=1.2
(If the precision is too slow, the threshold should be increased quickly)
If Rtv>M && Rtv1>M1, then T*=1.1
(If documents are retrieved more than required, the threshold should be increased)
If Rtv<M && Rtv1<M1, then T*=0.9
(If documents are retrieved less than required, the threshold should be lowered)

We have supposed that we can retrieve fewer documents at first and then retrieved more documents after profiles are updated. However, such experiment cannot lead to better results.

2.3.2 Adaptation of threshold-T9U

In order for the optimization of T9U, the purpose of threshold adaptation is to make sure that documents should be retrieved with high accuracy. But if the precision is too high, thresholds should also be decreased to retrieval more documents and then get larger T9U.

Algorithm:
If Cor<Rtv*0.10 && Rtv>max(M,4) , then T*=1.2
(If the precision is too slow, the threshold should be increased quickly)
If Rtv-1>M && Cor+1>Rtv*0.33, then T*=1.1
(If enough documents have been retrieved and precision is too low, the threshold should be increased)
If Rtv<M && Cor-1>Rtv*0.25 or Cor-1>Rtv*0.33 or Cor=0, then T*=0.9
(If documents are retrieved less than required with moderate precision, or precision is too high, or no document is retrieved, the threshold should be lowered)
In addition, if two irrelevant documents are retrieved continuously, T*=1.1.

Here, M represents the number of documents should be retrieved in the b-document interval. However, adaptive filtering systems cannot take into account the percentage that are relevant over the entire test set for a particular query in building their retrieval rules. Under such condition, M is estimated from the training corpus while relevant and pseudo relevant documents are taken into account. Although M is actually variable among different topics, we just use the average value for the convenience for computation.

2.3.3 Adaptation of topic profile

Once a retrieved document has been judged to be relevant, it is added to the positive document set for further adaptation, otherwise it is added to the negative document set. During profile adaptation, feature vectors are extract from positive documents and negative documents. The new topic profile is the weighted sum of topic vector, feature vector from positive documents and feature vector from negative documents. Thus not only the weight of features but also the feature words can be adjusted. The coefficient of A, B and C are still 0.25, 1.0, and -0.25.

Since relevant document is too scarce, we adjust the topic profile only after b *4 documents have

been processed. In fact, after processing b document, adaptation is triggered. Among 4 successive adaptation, the first 3 are threshold adaptation and the last one is profile adaptation. We don't adjust threshold and profile simultaneously because the threshold is optimized for the original profile.

2.4 Evaluation results

This year Fudan University has submitted 11 runs for adaptive filtering and batch filtering. We submit no routing runs. Table 2.1~2.3 summarize our adaptive and batch filtering runs.

Table 2.1 shows the results of OHSU topics. The "*score*" column is the score of each run under different evaluation measure. Micro recall and precision are calculated globally for all the topics, while macro recall and precision are averaged across all the topics[4]. The last columns give the number of topics in which our runs perform better, equal and worse than median ones. And the numbers inside the parentheses shows the number of topics in which our runs perform best.

Task	*Measure*	*Run*	*Score*	*Recall*		*Precision*		*Comparison with median*		
				Micro	*Macro*	*Micro*	*Macro*	*>(Best)*	=	<
Adaptive	T9U	FDUT9AF2	9.6	0.212	0.181	0.473	0.319	51(7)	6	6
	T9P	FDUT9AF1	0.264	0.277	0.300	0.283	0.271	37(6)	9	17
		FDUT9AF3	0.265	0.276	0.301	0.286	0.273	39(5)	8	16
		FDUT9AF4	0.249	0.263	0.285	0.278	0.259	34(9)	2	27
Batch	T9U	FDUT9BF1	13.6	0.276	0.245	0.492	0.390	37(20)	11	15
	T9P	FDUT9BF2	0.317	0.331	0.379	0.326	0.322	45(21)	7	11

Table 2.1 Adaptive and batch filtering for OHSU topics

Table 2.2 and 2.3 show the results of MeSH and MeSH sample topics. Our MeSH and sample runs don't perform as good as OHSU runs.

Task	*Measure*	*Run*	*Score*	*Comparison with median*		
				>(Best)	=	<
Adaptive	T9P	FDUT9AF6	0.356	134(134)	148	218
	T9U	FDUT9AF7	29.3	120(85)	72	308
Batch	T9P	FDUT9BF3	0.430	169(61)	151	180
	T9P	FDUT9BF4	0.440	215(101)	138	147

Table 2.2 Adaptive and batch filtering for MeSH sample topics

Task	*Run*	*T9P*	*Comparison with median*		
			>(Best)	=	<
Adaptive	FDUT9AF5	0.351	1297(1297)	1072	2535
Batch	FDUT9BF3	0.418	2297(2297)	0	2607

Table 2.3 Adaptive and batch filtering for MeSH topics

3. Question Answering

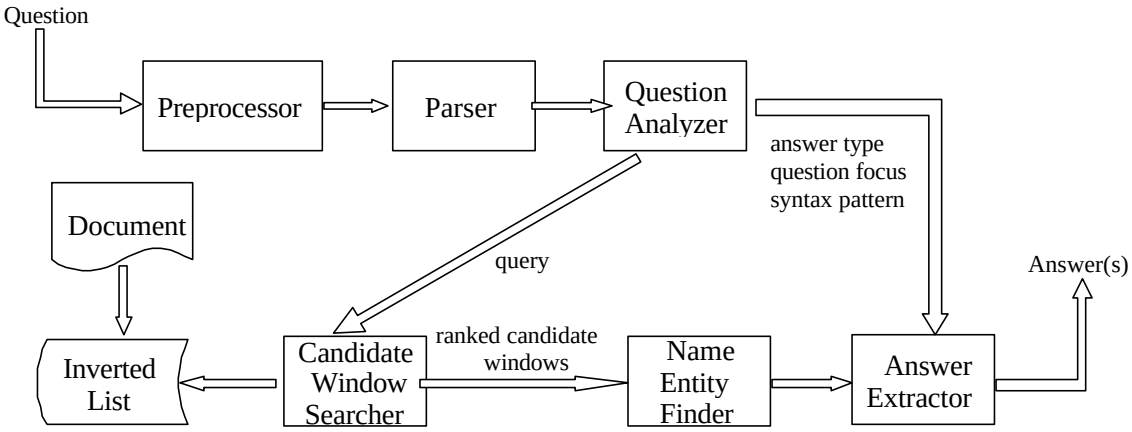
Question Answering is an interesting challenge for NLP researchers because it requires a combination of many traditional NLP techniques, such as tokenization, parsing, named entity identification and retrieval.

The next section introduces Fudan TREC-9 question answering system. It is followed by the detailed discussion of three main components. Followed are the evaluation results. Finally we will discuss the future prospects of our system.

3.1 Overview of Fudan Question Answering System

Similar to other systems[5], our system consists of three components: *Question analyzer*, *Candidate Window Searcher* and *Answer extractor*. The architecture is illustrated by Figure 3.1.

Figure 3.1: Architecture of the Fudan QA Systems



Initially, with a question parser and semantic mapping chart, we process the given questions and extract useful information: answer type, question focus and the syntax pattern of question. Furthermore, the Question Analyzer generates a set of query terms.

Each document retrieved by our ranked Boolean search engine is divided into segments of about 4k-byte. Each candidate segment is assigned with a score according to its similarity to the query generated by question analyzer.

The top-ranked segments are then passed to the Answer Extractor. A named entity finder based on HMM model[6] and a syntax parser based on chart algorithm are involved in this procedure. The extraction and ranking of the final answer are based on some empirical feature matching.

3.2 The Question Analyzer

The *Question Analyzer* attempts to excavate all available information inside the given question and generate a query for search engine.

In order to extract the real answer from the tremendous collection of documents, it's very important to know what the question is asking for. Fortunately, quite a few questions request certain type of answer. For example, for the question "*Who invented the paper clip?*", a person name is needed. We can either judge the question's answer type directly by its interrogative (who, where, when), or by semantic mapping of other words in question (e.g. how much, what city, which year, etc.). The semantic class of the answer type is listed in table 3.1. Cooperating with the named entity finder, it does much help to locate and score answer in our QA system.

Answer Type	*Question Type*	*Example*
PERSON	who/what-who/ which-who	Hugo Young
LOCATION	where/what-where/ which-where	China
ORG	who/what-who/ which-who	Phoenix Suns
MONEY	how much/ how many money	Pounds 12m
PERCENTAGE	how much/ what-percentage	0.10%
DATE	when/what-date/ which-date	10 Feb 1994
TIME	what time	6:33 a.m.
DURATION	how long	9 1/2-month
LENGTH	how long	147 feet
SIZE	how large	1.5 million acres
NUMBER	how many	562

Table 3.1: Answer types of question

Not all questions can provide obvious clue of their goals. Some questions, which start from *what* and *which*, are ambiguous and scarcely say anything about specific answer type. We solve it by defining a concept named *question focus*.

Question focus can be interpreted as the most important part of question, which distinguishes the question from others at the most. It may be a word or a sequence of words. Low frequency word and proper noun/phrase is commonly chosen as question focus. For example, in the question "*What culture developed the idea of potlatch?*", the question focus is *potlatch*. Question focus makes it easier to filter the irrelevant document and locate the exact answer.

The *syntax pattern* of question is generated by question parser. The purpose of parsing is to predict the possible syntax structure of answer sentence. Take the question "*What is a caldera?*" for an example, the possible answer sentence may be like "*Caldera is ...* ", "*Caldera, ...* " or "*... known as Caldera.*".

Finally, the question analyzer produces a set of query term and sends them to the search engine. Each term of the query comprises three fields:

a. *Query Term*, word or phrase extracted from the question.
b. *Term Rank*, calculated by the term's syntax role in question and the word frequency.
c. *Search Mode*, the suitable searching method for this query term.

3.3 Candidate Window Searcher

Among the large document collection, we try to find some segments of information that may be relevant to the question in order to restrict the scope for further processing. In this phase, we search the entire corpus for the query and generate N best candidate windows, from which we will extract answer to each question.

Our search engine makes use of the Boolean retrieval model, which is modified to suit for the QA task. Firstly, we define four kinds of search modes, named "Single Word Search", "Common NP Search", "Proper NP Search" and "Quoted Part Search" respectively.

Single Word Search is used to search the query term, which has only one word. It aims at finding all the occurrences of the word in the corpus.

Common NP Search is just like operator "OR" in Boolean Information Retrieval in that the words being searched need not co-occur with each other. It is often used to search people's name, which often occurs partially in the corpus.

Proper NP Search is somewhat like operator "OR" in Boolean Information Retrieval, except it discards sentences that only contain familiar query words that can be found in dictionary. Therefore, the remaining sentences just contain OOD query words, such as named entity. For example, for the query term "*Star Trek*", the search engine will only retrieve sentences that contain the word "*Trek*". And the sentence contains both of the words "*Star Trek*" will be ranked higher than that contains "*Trek*" only. It makes intuitive sense that the word "*Star*" occurs too often in the corpus to depict the information need.

Quoted Part Search performs just like operator "EXACT MATCH" in Boolean Information Retrieval. It is used to search quoted name, such as name of films or books. It not only requires query words to co-occur, but the order of query words to be matched exactly in the corpus as well.

Then, we create N-best window ranked by their window scores.

For every matched sentence among the corpus, we scan forward and backward within the same articles to get a candidate window with the size of no more than 4k bytes. That is, we try to locate all 4k-byte windows containing one or more matched sentences. However, these matched sentences are included in only one of those candidate windows, no overlap is allowed.

The windows are scored by the formula given below:

$$WS_i = \sum_{t=1}^{k} \left(\frac{2 * mc_t}{c_t} + \frac{msc_i}{sc_i} \right) * w_{qs}^t \qquad (3.1)$$

Where, mc_t is the number of terms in each query that locate in the window, while c_t is the total number of terms in each query. msc_i is the number of total matched sentences in the window, while sc_i is the total number of sentences containing in the window. The former factor indicates the coverage of

the query terms for the candidate window, whereas the latter favors candidate window with more occurrences of query terms. *k* is the number of query term for the question. w_{qs}^{i} is the weight of the *i*th query term. We assign weights to each query term according to its search mode and rank.

Sort all the windows by its score and select top N best window for further processing. In our experiments, we use 2k windows for every question at most.

3.4 Answer Extractor

The *Answer Extractor* identifies and extracts answers from the candidate windows. Each candidate window first passes through a named entity finder, which identifies names of person, location and organization, monetary units, dates, time, etc. By use of the answer type and question focus, all possible answers are located within the candidate window. For each possible answer, a 250-byte-long section in the candidate window named *answer-window* is then created. We evaluate each answer-window using the following four scores:

1) Matched_queries-score: Compute a match-score for each query term and sum them all. The match-score of query term is determined by its search mode, the match degree and the distance to answer-window of each match situation in the candidate window.
2) Query_coverage-score: Assign a coefficient to each matched query term in the answer window and cumulate them.
3) Syntax_pattern-score: If certain sentence in answer-window satisfies any predictive syntax pattern of the question, a correspondent score will be assigned to it.
4) Consistent_question_part-score: If certain part of question is found consistent in the answer-window, a score determined by the number of words in that part will be computed. It's a useful feature especially for back-formulation questions or coincident back-formulation situation.

The final score for a given answer-window is computed as:

final-score = k_1 * *matched_queries-score* +
k_2 * *query_coverage-score* +
k_3 * *syntax_pattern-score* +
k_4 * *consistent_question_part-score*

where, the weight vector (k_1,k_2,k_3,k_4) depends on the question feature, table 3.2 shows our empirical weight vector values:

Question feature	(k_1,k_2,k_3,k_4)
Num of query term=1	(8,8,16,4)
Num of low-freq word=0	(8,4,16,12)
Otherwise	(8,8,8,8)

Table3.2. Weight vector for final-score of answer-window

3.5 Evaluation

In this section, we describe the performance of our system. The system is evaluated by the Mean Reciprocal Answer Rank (MRAR):

$$MRAR = \frac{1}{n}\sum_{i=1}^{n}(1/rank_i). \qquad (3.2)$$

We submitted two runs in the 50-byte category and two runs in the 250-byte category. The first two runs are generated by using the top 100 candidate windows. The next two runs are by processing only top 24 candidate windows. The strict evaluation results are presented in Table 3.3.

The accuracy (measured by the percentage of questions correct) of our system fluctuates on various answer type. It is pleasant on questions demanding for PERSON (58%) and LOCATION (55%), but disappointing on DATE (35%) and NUMBER (25%). It is mainly because we concentrated on training the statistical model and worked little on rule-based identification, which is relatively simple but more useful on number-relevant named entity.

Run	*Category*	*Number / Percentage of questions correct*	*MRAR*
FDUT9QS1	50-byte (1)	200 / 29%	0.192
FDUT9QL1	250-byte (1)	313 / 46%	0.339
FDUT9QS2	50-byte (2)	187 / 27%	0.195
FDUT9QL2	250-byte (2)	288 / 42%	0.319

Table3.3. Performance in TREC-9

On the training corpus of TREC-8, our system did best while using top 24 candidate windows. But in TREC-9, the 250-byte run using top 100 candidate windows (FDUT9QL1) does better than that of top 24 one (FDUT9QL2). We presume it is caused by the variation on question style. The question of TREC-9 is shorter than that of TREC-8 on average. And some new question structure is too unfamiliar for us.

4. Discussion and Future Work

It is our first time to take part in TREC. Attending TREC-9 provides us further understanding of NLP technology. We have accumulated such knowledge resources as bilingual dictionary and Chinese synonym dictionary. We have also designed several NLP tools during this period, such as named entity finder, query translator, parser and search engine.

Although moderate performance has been achieved in our three systems, we still have a lot of things to do in the future. First, we need to enrich our knowledge resources, especially in English. We need to acquire knowledge from different domains and employ a comprehensive machine dictionary (e.g. WORDNET or HowNet) for semantic analysis. Currently, our three systems are developed almost independently. Next time, we will try to implement techniques developed for one system to another. For example, *feature selection* of filtering system can also play important role in the search engine. And *relevance feedback* in search engine is quite similar to adaptation in filtering.

Finally, we hope to apply the ideas and notions learned from TREC to corresponding tasks of our native language.

ACKNOWLEDGMENTS

This research was partly supported by NSF of China under contracts of 69873011 and 69935010, and 863 High Technology Project of China under contract of 863-306-ZD-02-02-4. We are thankful to Yaqian Zhou, Kaijiang Chen, Li Lian and Wei Qian for their help in the implementation of corpus and topic processing, syntactic parser and HMM based English named entity finder.

Reference

1. Kenney Ng. A maximum likelihood ratio information retrieval model. In Proceedings of the 8h Text Retrieval Conference TREC-8,1999
2. Wu Li-de, et. al, Large Scale Chinese Text Processing, Fudan University Press, 1997
3. C. Buckley, G. Salton, J. Allan, Automatic Retrieval With Locality Information Using SMART, Proceedings of the 1st Text REtrieval Conference (TREC-1), NIST Special Publication 500-207, 1992
4. Fabrizio Sebastiani, *Machine Learning in Automated Text Categorization*, Technical Report B4-31, Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, IT, 1999
5. Dan Moldovan, Sanda Harabagiu, et. al, LASSO: A Tool for Surfing the Answer Net. In Proceedings of the Eighth Text Retrieval Conference, 1999
6. D.M. Bikel, S. Miller, R. Schwartz and R. Weischedel, *Nymble: a High-Performance Learning Name-finder.* Proceedings of the Fifth Conference on Applied Natural Language Processing, Association for Computational Linguistics, pp. 194-201, 1997