TREC-10 Experiments at CAS-ICT: Filtering, Web and QA

Bin Wang, Hongbo Xu, Zhifeng Yang, Yue Liu, Xueqi Cheng, Dongbo Bu, Shuo Bai
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
{wangbin,hbxu,zfyang,yliu,cxq,bdb,bai}@ict.ac.cn
http://www.ict.ac.cn/

Abstract

CAS-ICT took part in the TREC conference for the first time this year. We have participated in three tracks of TREC-10. For adaptive filtering track, we paid more attention to feature selection and profile adaptation. For web track, we tried to integrate different ranking methods to improve system performance. For QA track, we focused on question type identification, named entity tagging and answer matching. This paper describes our methods in detail.

Keywords: TREC-10, Filtering, Web track, QA

1. Introduction

CAS-ICT took part in the TREC conference for the first time this year. Among the total six tracks of TREC-10, we choose three of them: Filtering, Web and QA.

For filtering track, we undertook the adaptive filtering subtask. Our model is still based on vector representation and computation. A topic-term relevance function is defined to guide feature selection. For profile adaptation, we use a Rocchio-like algorithm. Four runs have been submitted for evaluation: three of them are optimized for T10U measure, another one for T10F measure. We use very simple optimization methods in our experiments and we do not use any other resource except the new Reuters Corpus.

For web track, we undertook the ad-hoc subtask. Our system is based on a general-purpose search engine developed by us alone. We try to improve system performance by integrating different ranking methods. Query expansion technology is used to modify the initial query. The *PageRank* algorithm is investigated in our experiments. Four runs have been submitted and two of them use hyperlink information.

For QA track, we undertook the main subtask. We first use SMART search engine to retrieve a set of documents from the TREC data sets. At the same time, a question analyzer is used to analyze the given 500 questions of TREC-10 and generates the question types and keyword lists. Then we use GATE to analyze the top 50 retrieved documents and extract the named entities from them. Finally, an answer extractor extracts the relevant answers from the named entities. Three QA runs have been submitted for evaluation.

2. Filtering

In the filtering task, we undertook the adaptive filtering subtask, which we think, is more interesting and realistic than the other two subtasks.

2.1 Problem Description

This year the filtering task has 84 topics, which are exactly the categories in Reuters Corpus.

The total documents for this task (new Reuters Corpus) are divided into two parts: 23,307 documents for training (training set) and the remaining about 783,484 documents for testing (testing set). All the documents are Reuters everyday news, dating from August 1996 to August 1997. Two measures are given for adaptive filtering: T10U and T10F, the former is a linear utility measure and the latter is a kind of F-measure. In the adaptive task, only two positive samples in training set are given for each topic, the goal is to retrieve relevant documents one by one from the coming testing documents stream and get maximum T10U or T10F value at the same time.

2.2 System Description

Our adaptive filtering system consists of two components: the initializing component and the adaptation component. The former is used to get the initial data through training and the latter is to adapt these data when retrieving testing documents.

2.3 Training

In training procedure, we first process the training set for basic term statistics, this includes term tokenization, stemming and frequency counting. Then we can select terms from the positive and some pseudo-negative samples. After topic processing, we can get the initial profile vector by summing up the topic and feature vectors with different weight. Finally, we can compute the similarity between the initial profile vector and the positive documents to set the initial threshold.

2.3.1 Training set processing

In this step, all the training documents are processed. First we tokenize each document into single words, then eliminate the stop words and some other words with low frequency in the training set, and then we stem each word using the Porter Stemmer(*http://www.cs.jhu.edu /~weiss/*). Finally, we count each word's frequency(*TF*) within each document and the word's document frequency(DF) across the training set. When processing the training texts, we only use the <*title*> and <*text*> fields. Thus each document can be represented with its term frequency vector. Meanwhile, we can get the *IDF* statistics of the training documents. Since we can't use the *IDF* statistics of testing set, we use in the following steps the *IDF* statistics of the training document in term weighting. Ideally, we can update the IDF statistics when retrieving documents from the testing documents stream. But [16] has indicated that doing so does not seem to improve the overall filtering performance. So we use the same *IDF* statistics of the training documents all over our experiments.

2.3.2 Topic set processing

This year, each topic consists of three short parts: the <*num*> field, the <*Reuters-code*> field and the <*title*> filed. The <*title*> field includes only one or two words. We can't get more information from such kind of topics than from the topics of TREC-8 or TREC-9, which are described with more words. Of the three parts, we regard the <*title*> field as the most important. Though the <*Reuters-code*> fields may provide some information about relationships between different topics, we do not use them at all. Processing the topics is very simple, we only extract and stem the <*title*> field words to construct the vector for each topic.

2.3.3 Term selection

To reduce the computation complexity, we apply a method for feature selection. Here the features are all terms, each term is a word.

For each topic, we have two positive samples. So for all 84 topics, we have 168 positive samples. Thus, of the 168 samples, each topic has two positive samples, and we can suppose the other 166 samples are *negative* to the topic. Because one document may be relevant to more topics, our supposition is not very correct. But we try it for lack of information.

We define a word-topic correlation function as:

$$Cor(w_i, T_j) = \log\left(P(w_i \mid T_j) \big/ P(w_i \mid \neg T_j)\right) \tag{2.1}$$

Where $P(w_i|T_j)$ means the probability that word w_i exists in the relevant documents of topic T_j. On the contrary, $P(w_i \mid \neg T_j)$ means the probability that word w_j exists in non-relevant documents of topic T_j. For each topic, we compute the *Cor* value of each word in the positive two samples and choose the words with high *Cor* values as the features. Here we use maximum likelihood estimation. We compute the frequency that a word exists in the two positive documents as the estimation of $P(w_i|T_j)$, and the frequency that a word exists in the other "*negative*" documents as the estimation of $P(w_i \mid \neg T_j)$. If the estimation of $P(w_i \mid \neg T_j)$ is equal to zero, we give the *Cor* a big value.

After getting the feature words for each topic, we combine them to construct one feature space; the topic vector and the feature vector must be mapped into this space.

2.3.4 Profile initialization

For each topic, the profile vector(denoted as $\vec{P}$) is the weighted sum of the topic vector(denoted as $\vec{T}$) and the feature vector(denoted as $\vec{F}$), which is the sum of the two positive documents vectors. The formula is:

$$\vec{P} = \alpha * \vec{F} + \beta * \vec{T} \tag{2.2}$$

In our experiments, we set α=1,β=2 to give prominence to the topic words. So far, each component of the vectors is represented with *TF* values. Then we change it by multiplying with its *IDF* coefficient.

2.3.5 Similarity computation

To compute the similarity between a topic profile($\vec{P}_i$) and a document($\vec{D}_j$), we use the vector *cosine* similarity formula:

$$sim(\vec{P}_i, \vec{D}_j) = \cos(\vec{P}_i, \vec{D}_j) = \frac{\vec{P}_i \bullet \vec{D}_j}{|\vec{P}_i| \times |\vec{D}_j|} \tag{2.3}$$

Each component of the vectors is represented with *TFIDF* value. Here we use $TF_i * \log(1 + \frac{N}{DF_i})$ formula. We also try other formulae in our experiment, but the results are almost the same.

2.3.6 Initial threshold setting

We do not have good idea to set the initial threshold. According to our understanding, we believe we can't use the training documents to train the initial threshold because they are prepared for batch filtering task, not for adaptive task, we cannot use the relevance information of the other documents in training set except the two positive ones. Thus we have to use a very simple method, for each topic, we choose a small fixed value as the initial threshold which is smaller than the similarity between the initial profile vector and the two positive samples.

2.4 Adaptation

For each topic, after initializing the profile and the threshold, we can scan documents one by one from the testing set. If the similarity between the profile and the document is higher than the threshold, the document is retrieved and meanwhile the system can tell you the document is really relevant or not. With this information, some kind of adaptation may need to take to improve system performance. The adaptation may include threshold updating or profile updating.

2.4.1 Threshold adaptation

In TREC-10, two measures are defined to measure the performance of an adaptive filtering system. One is T10U, which is a linear utility; another is T10F, which is a kind of F-value. The goal of the adaptive filtering system is to get maximum T10U or T10F.

For T10U, we have two goals: one is to avoid negative T10U as far as we can, another is to improve the precision while the recall can't be greatly reduced. We only apply method for the former goal in our experiments.

For T10F, we also have two goals: one is to avoid retrieving zero documents, another is also to improve the precision while the recall can't be greatly reduced. We only apply method for the former goal, too.

2.4.2 Profile adaptation

After retrieving more and more relevant or non-relevant documents, we can get more useful information and understand the user's interest better. Thus we can adapt each profile vector, which represents each user's interest. Our profile adaptation includes positive adaptation and negative adaptation. For positive adaptation, we add the document vector of the positive documents to the old profile vector. For negative adaptation, we subtract the document vector of the negative documents from the old profile vector. When retrieving the *n+1* th document D_{n+1}, we can adapt the *n*th profile to the *n+1* th profile according the following formula:

$$\vec{P}_{n+1} = \begin{cases} \vec{P}_n + \alpha * \vec{D}_{n+1} & \text{if } D_{n+1} \text{ is relevant} \\ \vec{P}_n - \beta * \vec{D}_{n+1} & \text{otherwise} \end{cases} \tag{2.4}$$

Thus after retrieving *n+1* documents, all the retrieving relevant documents make the positive set denoted as $\{D^+\}$, the other documents set is $\{D^-\}$. Then the new profile vector become

$$\vec{P}_{n+1} = \vec{P}_0 + \alpha * \sum_{D_i \in \{D^+\}} \vec{D}_i - \beta * \sum_{D_i \in \{D^-\}} \vec{D}_j$$
$$\text{here } \{D^+\} \cup \{D^-\} = \{D_1, D_2, ..., D_{n+1}\}, \{D^+\} \cap \{D^-\} = \phi \tag{2.5}$$

Formula (2.5) is some kind of the Rocchio[19] algorithm except one point: we do not compute the centroid vector of the positive set or negative set and regard it as one vector. In other words, we pay more attention to the retrieving documents than the initial profile vector. Furthermore, we investigate the values of α and β. We found without negative feedback, the result is worse. In our experiments, we set α=1, β=1.1 or 1.3.

2.5 Evaluation Results and Analysis

We have submitted four adaptive filtering runs: one for T10F optimization, three for T10U optimization. Because we do not use complex optimization method, the results of the 4 runs are similar to each other. The evaluation results are shown in Table 2.1.

Table 2.1 shows that in each run, the results of about 2/3 of all topics are better than the medians, and most of the remaining results are worse. Unfortunately, about 1/3 of the worse results are worst results and most of the worst results are zero. Thus the overall performance of our runs is not high.

Run ID	MeanT10SU	T10SU vs. median(topic nums)			MeanT10F	T10F vs. median(topic nums)		
		>(Best)	=	<(Worst/Zero)		>(Best)	=	<(Worst/Zero)
ICTAdaFT10Ua	0.204	55(8)	1	28(11/11)	0.368	59(3)	2	23(8/7)
ICTAdaFT10Ub	0.205	51(0)	3	30(12/12)	0.366	55(0)	4	25(7/6)
ICTAdaFT10Uc	0.207	52(3)	2	30(11/11)	0.354	51(1)	4	29(12/9)
ICTAdaFT10Fa	0.206	55(4)	1	28(14/14)	0.387	62(1)	1	21(6/5)

Table 2.1 ICT adaptive filtering runs in TREC-10

We focused on the worst zero results and the best results. We found, on one hand, most of the zero results belong to "small" topics, which have small amount of relevant documents in the whole testing set and we called them "hard" topics. On the other hand, most of the best results belong to "big" topics. That is to say, our method is somehow fitful for "big" topics but not very fitful for "small" topics. This may result from two reasons: First, our feature selection method cannot find the best features of these "hard" topics from only two positive samples. Second, our optimization method is too simple to satisfy different cases.

In the future work, we will pay more attention to three aspects. For feature selection, we will try more effective methods. For optimization, we will try more complex methods. For profile adaptation, we may add feature reselection module.

3. Web Track

3.1 System Description

This year, the goal of the main web track is to retrieve the most relevant documents for each topic in the topic set (501~550) from the WT10G collection. Our system is based on a general-purpose search engine which we have been developing and improving since 1998. The system consists of four basic components: indexer, query generator, search agent, and search server. The indexer scans all documents of the WT10G and generates full text indexes and term statistics. The query generator analyzes the TREC topics and generates real queries. The search agent submits real queries to search server and shows the return results in visible format. The search server receives queries, searches documents and returns results to the search agent

3.2 Searching Process

3.2.1 Query construction

Query generator reads TREC topics, extracts valuable terms and submits them to the search system. Therefore, the initial query for each topic is a set of some useful terms of the topic.

At this stage all stop words are removed. In addition to the basic stop words, we have also removed some other stop words that carry little information in the topics, such as *find*. While processing each topic, we only use the <*title*> field.

3.2.2 Initial retrieval

For each initial query, the search system retrieves some documents and ranks them using formula (3.1)[7]. Some top ranked documents are returned as the initial search results.

$$S_{q,d} = \frac{\sum_{t \in q \cap d} (1+\log f_{t,d}) * \log(1+\frac{N}{f_t})}{\sqrt{\sum (1+\log f_{t,d})^2} * \sqrt{\sum \log^2 (1+\frac{N}{f_t})}} \qquad (3.1)$$

Where $S_{q,d}$ is the similarity of document d and query q, f_t is the number of documents in which term t occurs in WT10G, $f_{t,d}$ is the frequency term t occurs in document d (within-document frequency), and N is the total number of documents in WT10G.

3.2.3 Query expansion

After first retrieval we can get many ranked documents. Then we can regard some top ranked documents as *relevant*. All the terms in these documents are weighted according to formula (3.2)[1][8].

$$TSV = r * W_t = r * (\frac{k_5}{k_5+\sqrt{R}} \log(k_4^{'} \frac{N}{N-n} + \frac{n}{N-n}) + \frac{\sqrt{R}}{k_5+\sqrt{R}} \log \frac{r+0.5}{R-r+0.5}$$
$$- \frac{k_6}{k_6+\sqrt{S}} \log \frac{n}{N-n} - \frac{\sqrt{S}}{k_6+\sqrt{S}} \log \frac{s+0.5}{S-s+0.5}) \qquad (3.2)$$

Where *TSV* means *Term Selection Value* that is used to rank terms. N is the number of documents in WT10G, n is the number of documents in which term t occurs, R is the number of relevant documents, r is the number of relevant documents which contain term t, S is the number of non-relevant documents in WT10G, s is the number of non-relevant documents which contain term t in WT10G. $k_4^{'}$, k_5, k_6 are parameters.

Some top terms with their *TSVs* are selected for the new query vector construction.

3.2.4 Second retrieval

In this stage, the new query is submitted to the search system and new results are retrieved which are ranked by formula (3.3).

$$w_{q,d} = \sum_{t \in q} w_t \frac{(k_1+1) * f_{d,t} * b_{d,t}}{k_1 * [(1-b) + b * \frac{L_d}{avr\_L_d}] + f_{d,t}} * \frac{(k_3+1) * f_{q,t}}{k_3 + f_{q,t}} \qquad (3.3)$$

Where L_d is the length of document d, avr_L_d is the average document length, $b_{d,t}$ is within-document importance of term t in document d, which includes multiple factors such as the frequency t occurs in document title, bolded text, and hyperlink text. k_1,k_2, k_3 are parameters.

3.3 Applying Link Analysis Technology in Web track

We also investigate link analysis technology that is used to rank web pages using link information. In our experiments, we mainly use an improved *PageRank* algorithm.

3.3.1 Basic *PageRank* algorithm

Brin & Page[5] suggested a link based search model called *PageRank* that first evaluated the importance of each web page based on its citation pattern. The *PageRank* algorithm re-ranks the retrieved pages of a traditional search scheme according to their *PageRank* values.

In this approach, a web page will have a higher score if more web pages link to it and this value will increase if those web pages' scores increase. The *PageRank* value of a given web page t, denoted as Pr*(t)*,can be iteratively computed according to formula (3.4)[5].

$$\Pr(t) = (1-d) + d * \sum_{i=1}^{m} \frac{\Pr(t_i)}{c(t_i)} \quad (3.4)$$

Where $t_1, t_2, ..., t_m$ are the web pages which link to page *t*, *d* is a parameter (set to 0.85 as suggested by [5]) and $c(t_i)$ is the number of outgoing links for page t_i. For simplification, all the pages which link to page *t* are called the *pre-set* of *t*, denoted as *pre-set(t)*,and all the pages which page *t* links to are called the *post-set* of *t*, denoted as *post-set(t).*

3.3.2 Implementation of the *PageRank* algorithm

From formula (3.4) we can see that computing *PageRank* is very simple in itself. But because the numbers of web pages is usually very large (in WT10G the number is 1,692,097) and the number of hyperlinks is even larger, the iteration process may be time-consuming. In order to solve this problem, we do some useful pretreatment.

First, we try to get rid of all the noisy hyperlinks that have little information before iteration. Meanwhile, we mark the web pages that have empty *pre-set* and do not participate in the iteration.

Second, we pretreat the *post-sets* of all the web pages. From formula (3.4) we can see that, for web page *t* and one page in its *pre-set*, t_j, if $c(t_j)$ is sufficiently large, the value of $\Pr(t_j)/c(t_j)$ will be very small, intuitively speaking, that means the influence of webpage t_j to webpage *t* is very small in the web graph. Thus for web pages like t_j, we don't let them only simply take part in the iteration. A threshold is set in advance, if one page's *post-set* has bigger size than the threshold, we assign a fixed value as the *PageRank* for this page. In this way we can greatly reduce the iteration cycle in our experiments.

3.3.3 Convergence properties

According to the probabilistic meanings of formula (3.4), after each time of iteration, the sum of all pages' *PageRank* should be equal to 1 after standardization. To avoid that the *PageRank* values are too small and incomparable, we introduce a new standardization method.

In addition, we use formula (3.5) to determine whether we should stop or not after the *i+1* th iteration.

$$\underset{t}{Max}(\Pr(t)^{(i)} - \Pr(t)^{(i+1)}) - \underset{t}{Min}(\Pr(t)^{(i)} - \Pr(t)^{(i+1)}) < \delta \quad (3.5)$$

In order to determine whether the iteration should be stopped or not, we consider not only the decreasing tendency of all the web pages, but also the low changeability between iterations. In our experiments, after 25 times' iteration we get a relatively steady convergence results.

3.3.4 Integrating ranking methods

We integrate the above ranking methods into formula (3.6) to get the integrated rank of one page:

$$W_d = W_{dc} + W_{dl} = W_{dc} + \frac{W_{dc}}{\log \frac{PR_{\max} * k}{PR_d}} \quad (3.6)$$

Where W_d is the final weight of a page (document), W_{dc} is the content-based document weight computed by formula (3.3), W_{dl} is the link-based document weight. PR_d is the *PageRank* value of document *d* computed by formula (3.4), PR_{max} is the maximum *PageRank* value among all the documents, *k* is a constant greater than 1.

3.4 Evaluation Results

We have submitted four runs to NIST. The results are listed in Table 3.1.

Technology(Run id)	Average Precision (non-interpolated) over all relevant docs	P@20 docs	R-Precision
Baseline(ICTWeb10n)	0.0860	0.1450	0.1244
Query Expansion(ICTWeb10f)	0.0464	0.0620	0.0657
Link Analysis(ICTWeb10nl)	0.0860	0.1410	0.1147
Query Expansion &Link Analysis(ICTWeb10nfl)	0.0464	0.0620	0.0657

Table 3.1 Web track results

Table 3.1 shows that our query expansion method leads to performance degradation. The reason may be that we first use such a complex query expansion method in our system and some of the parameters need to be revised in future tests.

From the above table, we have also found that the results integrated with link analysis have little difference with the benchmark. We believe the reason is not algorithm itself but the small size of the web pages set. In our experiments, we use WT10G which is a close set that doesn't link to outside, thus many informative hyperlink are not included. If the experimental data size is large enough, the results should be very good. The *PageRank* method offers an approach that evaluates the web pages objectively.

Our future work includes three aspects: First, we will try different probabilistic retrieval models; second, we will try alternative feedback methods; third, we will try new connectivity computation methods.

4. Questing Answering track

4.1 System Description

Our TREC-10 question answering system consists of four basic components: IR search engine, question analyzer, named entity tagger and answer extractor. Figure 4.1 illustrates the whole architecture.

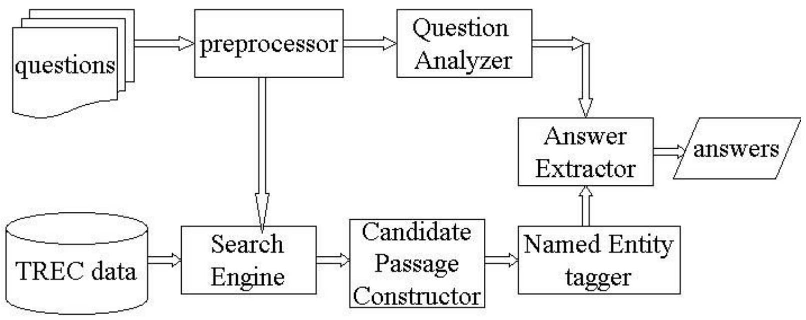

Figure 4.1: Architecture of the ICT TREC-10 QA System

We first use an IR search engine to retrieve a set of documents from the TREC data sets. At the same time, a question analyzer analyzes the given 500 questions of TREC-10 and generates the question type and keyword-list for each question. Then we use a named entity tagger to analyze the top 50 documents retrieved by the search engine and extract the named entities from them. Finally, an answer extractor determines the relevant answers from the named entities using the question type and keyword-list.

4.2 SMART Search Engine

As we know, SMART(*ftp://ftp.cs.cornell.edu/pub/smart*) is an implementation of the vector-space model of information retrieval proposed by Salton dating back in the 60's. The primary purpose of SMART is to provide a framework in which one conducts information retrieval research. It is (as we heard) distributed for research purpose only. Since TREC-QA allows us to use search engine freely, we choose SMART as our IR search engine because it is easy to use. To meet the need of our QA system, we add some new components into SMART. We also use the feedback function of SMART to generate a set of retrieved documents, based on which we make a run ICTQA10c. But this run becomes the worst of all our three runs, which proves we failed in applying the feedback. This indicates that we need to do further research on feedback, such as using the LCA (Local Context Analysis) feedback technique[14].

4.3 Question Analyzer

Main answer and question types what we can extract are listed in table 4.1.

Answer type	Question type	Example
PERSON	Who/Whom/What-person/Which-person	Mr. Mulroney
LOCATION	Where/What-location/Which-location	Orange County
ORGANIZATION	What-org/Which-org/Who	Penn Central Corp.
MONEY	How much	$28.5 million
PERCENT	How much/What percentage	4%
DATE	When/What date	Aug. 6, 1945
NUMBER	How many	Five
DURATION	How long	20 years
DISTANCE	How long/how far/How tall	10 miles
AREA	How large/How big	100 square kms
MEASUREMENT	How heavy/How fast	4 tons
CURRENCY	What currency	Dollar
NATIONALITY	What nation/What language	Icelandic
REASON	Why/How	-
NAME	What/Which is,are NP	small brain defect typical victims
No Answer	All	NIL

Table 4.1 Answer and question types

We determine the question type based on two rule sets. First rule set is keyword-based, which consists of some patterns as follows:

Who : PERSON

Where : LOCATION

What day : DATE

How many : NUMBER

Such patterns determine the question type only according to the interrogative of the question. This kind of rules can be applied to the PERSON, LOCATION, DATE, NUMBER types and so on. The second rule set is template-based. Some examples of this kind of rules are as follows:

What do,does,did @2 cost? : MONEY

What person @3? : PERSON

Which person @3? : PERSON

What is,was @1's birthday? : DATE

What is,are,was,were @1's employer? : ORGANIZATION

Such rules are useful to determine those questions led by "What", whose question types are hard to determine only by keyword-based rules.

4.4 Named Entity Tagger

We use GATE(*http://gate.ac.uk*) as our Named Entity tagger. GATE is developed by Sheffield NLP group. It can extract the named entities of PERSON, LOCATION, MONEY, PERCENT, DATE, ORGANIZATION and so on. But it has some obvious shortcomings, for example, it can't extract the pure NUMBER and most MEASUREMENT entities. Furthermore, many entities are incorrectly identified in GATE. Therefore, we revise the basic GATE system. First, we add several new components to identify the NUMBER and MEASUREMENT entities. Second, we modify GATE to improve the tagging correctness, mainly for the MONEY and PERCENT entities. As to some abstract question types, such as REASON(why), MEANING(what), METHOD(how) and so on, we apply some rules based on certain features to tag a snippet from the candidate passage as the candidate answer.

Before identifying the named entities, we need to construct candidate passages using the top 50 documents retrieved by SMART search engine. The algorithm[10][11][15] is as follows:

Step 1:Parse the sentences of the documents.

Step 2:Retrieve the sentences that contain keywords in the question.

Step 3:Construct a candidate passage every two sentences. If one sentence is long enough, it becomes a candidate passage itself.

Step 4:Assign each candidate passage a initial score equal to the score ranked by SMART search engine, i.e. *score(P)=IR(D)*, *D* is the document where the candidate passage *P* lies.

Step 5:Add the *idf* value of all matched keywords contained by a candidate passage to its score.

Step 6:Calculate the number of matched keywords(*count_m*) in each candidate passage *P*. Add *0* to *score(P)* if the number of matched keywords is less than the *threshold*. Otherwise, add *count_m* to *score(P)*. The *threshold* is defined as follows:

threshold=count_q if *count_q<4*;

threshold=count_q/2.0+1.0 if *4<=count_q<=8*;

threshold=count_q/3.0+2.0 if *count_q>8*;

here *count_q* is the number of keywords in the question.

Step 7:Calculate the size of matching window, then add *40*count_m/size(matching_window)* to *score(P)*. The size of matching window is defined as the number of keywords in the candidate passage between the first matched keyword and the last one.

Step 8:Re-rank the candidate passages by their final scores and output the top *10* or *20* passages to the Named Entity tagger.

4.5 Answer Extractor

The answer extractor compares the question type with each named entity in candidate passages. If a candidate passage contains a named entity matching the question type, we add *100* to the score of the passage. When there is more than one matched named entity, we only count once.

After the process above, we re-rank the candidate passages according to the named entity and question types, then output the top 5 named entities as the final answers. If the question type is unknown, we intercept a snippet with the largest density of keywords in the candidate passage to answer the question.

4.6 Results and Analysis

There are three subtasks in TREC-10 QA: main, list and context. We only participate in the main task and submit three runs in the 50-byte category. ICTQA10a uses the top 20 candidate passages for each question. Both ICTQA10b and ICTQA10c use only top 10 candidate passages for each question, the difference is that ICTQA10c uses the feedback of SMART in the IR phase. The evaluation results are presented in table 4.2 and 4.3. Table 4.2 shows the results in strict evaluation while table 4.3 does in lenient evaluation. (MRR means "Mean Reciprocal Rank".)

ICTQA10b is better than ICTQA10a, which shows that more candidate passages can't guarantee to generate better results.

Task	Run	Correct # (strict)	Correct % (strict)	MRR(strict)	Correct # of final answer	Correct % of final answer
main	ICTQA10a	63	12.8%	0.090	26	8%
main	ICTQA10b	67	13.6%	0.100	34	10%
main	ICTQA10c	56	11.4%	0.077	20	6%

Table 4.2 Strict performance in TREC-10

Task	Run	Correct # (lenient)	Correct % (lenient)	MRR(lenient)	Correct # of final answer	Correct % of final answer
main	ICTQA10a	74	15.0%	0.102	26	8%
main	ICTQA10b	76	15.4%	0.109	34	10%
main	ICTQA10c	66	13.4%	0.089	20	6%

Table 4.3 Lenient performance in TREC-10

Table 4.4 shows some statistical results by question types. Our system is pleasant for the NATIONALITY, DURATION, CURRENCY, LOCATION and No Answer question types, but disappointing on the DATE, PERSON, MONEY and REASON question types, though these question types are easy to determine. The main reason is that some bugs exist in our ranking strategy when there are too many candidate named entities matching these question types. We try to find more detailed ranking strategies to solve this problem in the future work. We also try to introduce some syntactic and semantic parsing technology to solve other problems, especially the NAME(*What is NP?*) question type, which we badly handle in TREC-10.

Question type	# of question	Correct #	Correct %	MRR
PERSON	53	5	9.43%	0.04
LOCATION	29	10	34.48%	0.24
MONEY	2	0	0	0
PERCENT	5	1	20%	0.20
DATE	33	6	18.18%	0.16
NUMBER	13	3	23.1%	0.13
DURATION	4	3	75%	0.75
MEASUREMENT	30	7	23.33%	0.217
CURRENCY	5	2	40%	0.40
NATIONALITY	2	2	100%	0.625
REASON	4	0	0	0
NAME	130	18	13.85%	0.08
No Answer	49	15	30.61%	0.306

Table 4.4 Lenient performance for each question type

5. Conclusion

This year we participate in the TREC conference for the first time, the main goal is to understand the process and the ideas of the TREC conference. We spend much more time on this goal than we do in system construction. We think we have achieved this goal though our results are not so satisfactory.

Before attending TREC-10, we have had some experiences in Chinese information processing. After attending TREC-10, we have got some experiences in English information processing and we will try to apply these useful experiences in our future work.

Acknowledgements

This research is supported by the national 973 fundamental research program under contact of G1998030413. We give our thanks to all the people who have contributed to this research and development, in particular Yanbo Han, Li Guo, Qun Liu, Hai Zhuge, Zhihua Yu and Lei Cheng.

References

[1] Ogawa, Y., Mano, H., Narita, M., Honma, S. Structuring and Expanding Queries in the Probabilistic Model. In *The Ninth Text REtrieval Conference (TREC 9),2000.*

[2] O. Yasushi, M. Hiroko, N. Masumi, H. Sakiko. Structuring and expanding queries in the probabilistic model. In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[3] S.E. Robertson, S. Walker. Okapi/Keenbow at TREC-8. In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[4] S.Brin and L.Page. The anatomy of a large scale hypertextual web search engine. In *The 7th WWW Conference,1998.*

[5] Lawrence Page,Sergey Brin,Rajeev Motwani,Terry Windograd. The Pagerank citation ranking:

Bring order to the web. *Stanford Digital Libraries working paper, 1997-0072.*

[6] J.Kleinberg. Authoritative sources in a hyperlinked environment. *Proc 9th ACM-SIAM SODA,1998.*

[7] Ian H. Witten, Alistair Moffat, Timothy C. Bell. *Managing gigabytes: Compressing and indexing documents and images, 2nd ed, 1994.*

[8] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford. OKAPI at TREC-3, In *The Third Text REtrieval Conference (TREC 3),1994*.

[9] Ellen M. Voorhees, Dawn M. Tice, The TREC-8 Question Answering Track Evaluation, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[10] Amit Singhal, Steven Abney, Michiel Bacchiani, Michael Collins, David Hindle and Fernando Pereira, AT&T at TREC-8, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[11] Toru Takaki, NTT DATA: Overview of system approach at TREC-8 ad-hoc and question answering, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[12] Rohini Srihari and Wei Li, Information Extraction Supported Question Answering, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[13] John Prager, Dragomir Radev, Eric Brown, Anni Coden and Valerie Samn, The Use of Predictive Annotation for Question Answering in TREC-8, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[14] Jinxi Xu and W. Bruce Croft, Query Expansion Using Local and Global Document Analysis, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[15] Xiaoyan Li and W. Bruce Croft, Evaluating Question-Answering Techniques in Chinese, Computer Science Department University of Massachusetts, Amherst, MA , *2001.*

[16] Chengxiang Zhai, Peter Jansen, Norbert Roma, Emilia Stoica, David A. Evans, Optimization in CLARIT TREC-8 Adaptive Filtering, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[17] Lide Wu, Xuanjing Huang,Yikun Guo, Bingwei Liu, Yuejie Zhang, FDU at TREC-9: CLIR, QA and Filtering Tasks. In *The Ninth Text REtrieval Conference (TREC 9),2000.*

[18] Stephen Robertson, David A. Hull, The TREC-9 Filtering Track Final Report. In *The Ninth Text REtrieval Conference (TREC 9),2000.*

[19] Rocchio, J. J. Relevance Feedback in Information Retrieval. In *The SMART Retrieval system, Prentice-Hall, Englewood NJ. 1971, 232-241.*