# FDU at TREC2002: Filtering, Q&A, Web and Video Tasks

*Lide Wu, Xuanjing Huang, Junyu Niu, Yingju Xia, Zhe Feng, Yaqian Zhou*

*Fudan University, Shanghai, China*

This year Fudan University takes part in the TREC conference for the third time. We have participated in four tracks of Filtering, Q&A, Web and Video.

For filtering, we only participate in the sub-task of adaptive filtering. A novel method is presented, in which a winnow classifier from the description and narrative fields is constructed, and then utilized to assist our previous adaptive filtering system.

A novel approach to confidence sorting, which is based on Maximum Entropy, is proposed in our Question Answering system. The rank of individual answer is determined by several weighted factors, and the confidence score is the product of the exponent of the weights of every factors. The weight of every factor is assigned during the training of previous questions.

To return highly relevant key resources for web retrieval, we modified our original search system to make it return higher precision result than before. First, we proposed a novel search algorithm to get a base set of highly relevant documents. Then special post-processing modules are used to expand and re-sort the base set.

This year we tried a fast manifold-based approach to face recognition in the Video Search Task. It can be used when there are only few different images of a specific person and runs fast. Experiment shows that applying this step will make the face recognition 5-fold faster and with almost no decreasing of performance.

## 1. Filtering

Our research focuses on how to make use of the narrative and description fields of each topic. Experiment results have shown that these two fields are very important and the proper exploitation of them can enhance the filtering system's performance greatly.

In this section, we will first introduce the training stage of our adaptive filtering system in details, and then the adaptive filtering stage; some experiment results are also given.

### 1.1 Training of adaptive filtering

Figure 1.1 shows the architecture of the training stage, which includes topic processing, feature vectors extracting and initial threshold setting.

#### 1.1.1 Topic processing

By examining the words in the description and narrative of each topic carefully, we believe that smart use of these words will bring notable gain. Therefore, we construct a winnow classifier [Littlestone88] from these two fields to assist our adaptive filtering system.

First, we remove the function words from these two fields. Here function words including stop word and those words such as "relevant", "irrelevant", "documents" which bear no content in the topic description. Then we initialize a winnow classifier for each topic using the remaining words and assigning each word with equal weight. After that, we adjust the winnow item's weight during training. The adjustment procedure can be described as below:

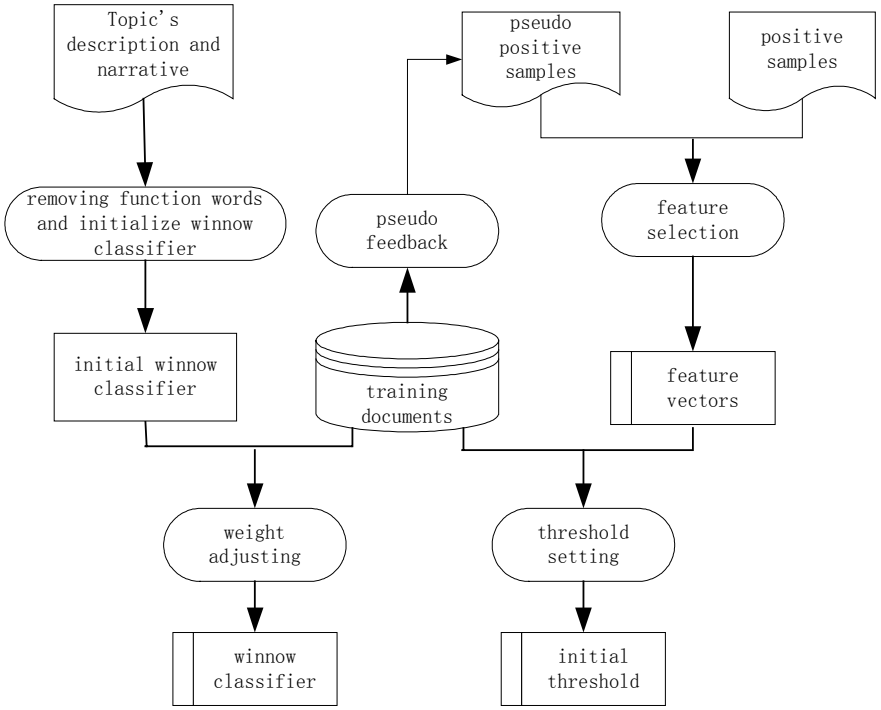


Figure 1.1 Architecture of training stage

If one of the positive samples of each topic (3 documents per topic) has not been retrieved by the winnow classifier, we promote weights of the words occur in this positive sample by a coefficient of 1.5.

If an irrelevant document has been retrieved by the winnow classifier, we demoted the words occur in this document by a coefficient of 0.8.

After that, we set threshold for each winnow classifier through the whole training set.

### 1.1.2 Feature selection

Since the total number of all words is very large and it costs much time in similarity computation, we decide to select some important words from them. First, we carry out morphological analysis and stopword removing. Then we compute the *logarithm Mutual Information* between remaining words and topics:

$$\log MI(w_i, T_j) = \log\left( P(w_i \mid T_j) \Big/ P(w_i) \right) \qquad (1.1)$$

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. $P(w_i)$ and $P(w_i|T_j)$ are both estimated by *maximal likelihood 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. Logarithm Mutual Information is not only used as the selection criterion, but also as the weight of feature words.

### 1.1.3 Similarity Computation

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

$$Sim(d_i, p_j) = Cos\theta = \sum_k d_{ik} * p_{jk} \Big/ \sqrt{(\sum_k d_{ik}^2)(\sum_k p_{jk}^2)} . \quad (1.2)$$

Where, $p_j$ is the profile 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: $d_{ik} = 1 + \log(tf_{ik} * avdl / dl)$ , where $tf_{ik}$ is the frequency of the kth word in the ith document , *dl* is the average number of different tokens in one document, *avdl* is the average number of tokens in one document.

### 1.1.4 Creating initial profile and Setting initial threshold

Each topic's feature vector is the weighted sum of feature vector from positive (relevant) documents and feature vector from pseudo relevant documents with the ratio of 1: 0.25. The pseudo relevant documents are acquired during pseudo feedback procedure, which uses the similarity as selection metric. Those documents that have highest similarity and do not occur in the positive documents are regard to be pseudo relevant.

After combining the positive and pseudo-positive feature vectors, we obtain the initial profile and then set the initial thresholds to get the largest value of T11SU or T11F.

## 1.2 Adaptive stage

Figure 1.2 shows the architecture for the adaptive stage. While filtering the input document stream, we first use the winnow classifier to make the initial decision. If winnow classifier has retrieved a document, the system will compute the similarity between this document and the feature vector and make final decision based on the threshold. For each document retrieved, we will see whether it is relevant and do some adaptation accordingly. The adaptations including adjusting the weight of winnow classifier and modifying the feature vectors and threshold. The threshold-adjusting algorithm is the heuristic algorithm we presented at TREC10 [Wu01]. The winnow's weight adjusting is the same as described in 1.1.1.

## 1.3 Effect of Winnow classifier and analysis

To see the effect of winnow classifier, we investigate the words in both winnow classifier and the vectors gotten from training set and find that only about 36% of the words of winnow classifier has occurred in the vectors. Therefore, there must exist many documents on that winnow classifier and VSM will make different decisions. If these documents happen to be irrelevant, combining winnow classifier with normal VSM will enhance the system's precision. We have done several experiment and found that Winnow classifier's recall is very high (0.5137 in training set and 0.4360 in testing set) but it's precision is very low (0.1406 in training set and 0.0823 in testing set). So combining winnow classifier with VSM will lower the system's recall, but will enhance the system's precision at the same time. Since the system's performance is the function of precision and recall and emphasize on precision, combine normal VSM with winnow classifier will enhance system's performance ultimately. The experiment results have confirmed this assumption. Table 1.1 is the experiment results of filtering system using winnow classifier and Table 1.2 without winnow classifier. We can see the efficiency of using winnow classifier.

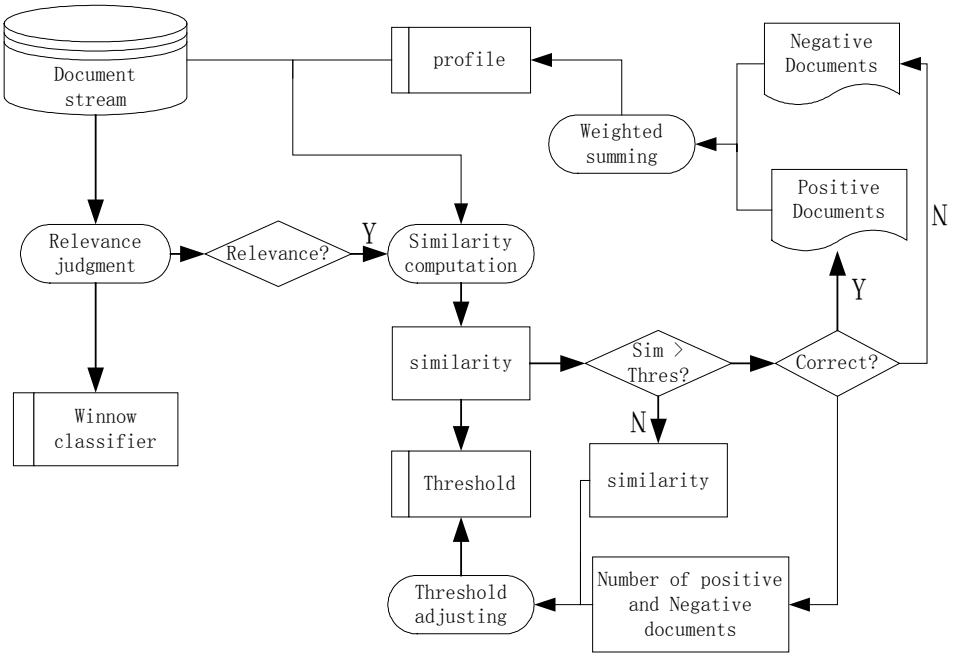


Figure 1.2 Architecture of adaptation stage

Table 1.1 Experiment results of filtering system based on normal VSM

| VSM | Recall | Precision | T11F | T11SU |
|---|---|---|---|---|
| Total | 0.2026 | 0.1544 | 0.1271 | 0.1841 |
| R101 ~ R150 | 0.3159 | 0.2668 | 0.2204 | 0.2287 |
| R151 ~ R200 | 0.0894 | 0.0421 | 0.0337 | 0.1394 |

Table 1.2 Experiment results of filtering system combine normal VSM with winnow classifier

| Winnow | Recall | Precision | T11F | T11SU |
|---|---|---|---|---|
| Total | 0.1390 | 0.2456 | 0.1771 | 0.2334 |
| R101 ~ R150 | 0.2240 | 0.4511 | 0.3189 | 0.3657 |
| R151 ~ R200 | 0.0541 | 0.0401 | 0.0354 | 0.1011 |

We also conducted experiments to see the effect of dynamic adjusting winnow's weights during adaptive filtering. Table 1.3 and Table 1.4 show the efficiency of adjusting winnow's weights. We also have try the method that add the words of winnow classifier to topic vectors and try different way to assign weight to words, but the results are not satisfactory.

Table 1.3 Experiment results of filtering without adjusting the winnow weight

| static winnow | Recall | Precision | T11F | T11SU |
|---|---|---|---|---|
| Total | 0.1387 | 0.2471 | 0.1741 | 0.2484 |
| R101 ~ R150 | 0.2257 | 0.4461 | 0.3064 | 0.3556 |
| R151 ~ R200 | 0.0517 | 0.0481 | 0.0417 | 0.1412 |

Table 1.4 Experiment results of adjusting the winnow weight while filtering

| Dynamic winnow | Recall | Precision | T11F | T11SU |
|---|---|---|---|---|
| total | 0.1382 | 0.2573 | 0.1788 | 0.2735 |
| R101 ~ R150 | 0.2278 | 0.4595 | 0.3116 | 0.3717 |
| R151 ~ R200 | 0.0485 | 0.0550 | 0.0460 | 0.1754 |

# 2. Question Answering

Since the TREC evaluation for Question Answering begins four years ago, quite a few sites joined in this task. The deep NLP method get succeed during the first three years [Harabagiu99], while the shallow method [Soubbotin01] get even greater success in last year's evaluation. Their success propels us to focus on the shallow NLP method.

Our QA system can be divided into four modules: pre-processing and indexing (the offline model), question analysis, sentence searching, and answer finding. Moreover, the last module can be divided into two sub-modules: answer extracting & meriting, and confidence calculation & sorting. Among them, we pay much attention on the second and last modules, which can be shown in the next figure.

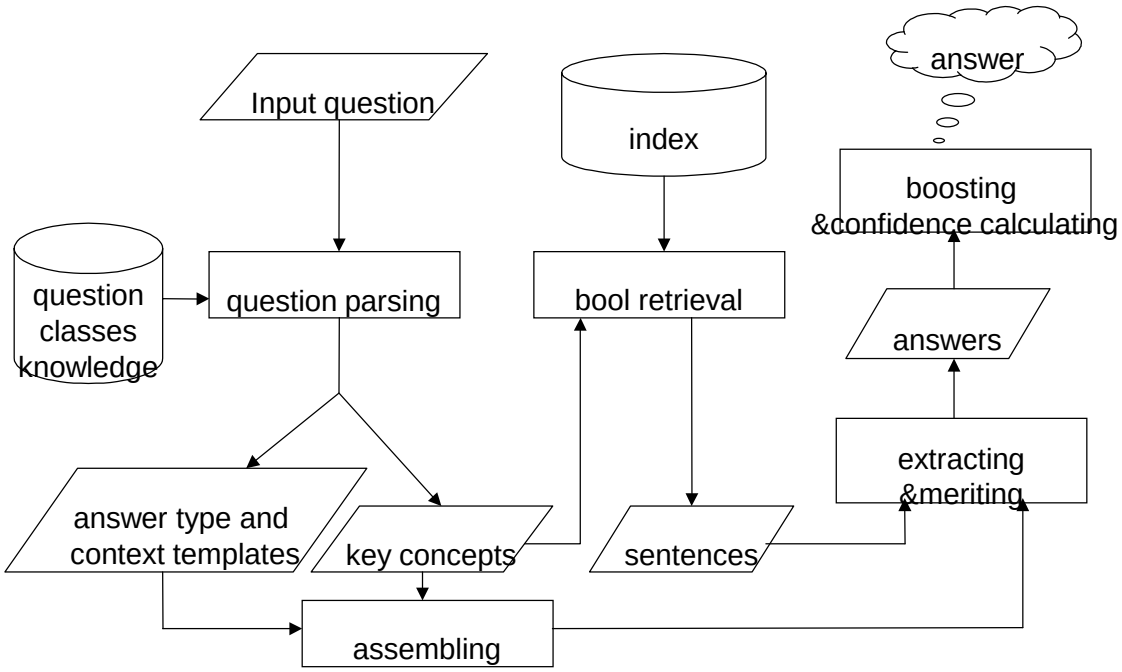


Figure 2.1 QA online modoles

From left to right: question analysis, sentence searching, answer finding

## 2.1 Building the QA knowledge base

The QA task is open domain. Quite a few participants use the open domain knowledge base such as WordNet to build their systems, while the QA task needs additional knowledge. How can the system deduce from the WordNet that, the answer of the question "*Who invented telephone*?" is a person name? There should

be some knowledge base to tell the system that such kind of question's answer should be person name. Thus, almost every system has integrated more or less special knowledge for the QA task.

Our knowledge base includes about 80 question classes, which can be used in the Question Analysis model and Answer Extracting model. Each class includes three parts: question patterns, answer types and context templates. Following is an example:

*<QuestionClass ID=302>*

*<Pattern ID=1> Who; <KeyConcept ID=1 Type=VBD>; <KeyConcept ID=2 Type=NP> ;QUERY; </Pattern>*

*<Pattern ID=2> Who;VBZ11;NP1; QUERY;</Pattern>*

*<Answer ID=1 Weight=1>  PersonName </Answer>*

*<Context ID=1 Weight=2> A;VBD1;NP2; </Context >*

*<Context ID=2 Weight=1.5> NP2;VBN1;by;A; </Context >*

*<Context ID=3 Weight=1.8> A;Comma;NP2;POS;NN1; </Context >*

*<Context ID=4 Weight=1.8> A;VERB_BE;NP1;of;NP2; </Context >*

*<Context ID=5 Weight=1> NP2;VERB_BE;VBN1;by;A; </Context >*

*<Context ID=7 Weight=1> A;POS;VBG1;IN;NP2; </Context >*

*<Context ID=8 Weight=1> A;IN;VBG1;NP2;</Context >*

*<AnyOrder ID=1 Weight=0.3> A;VBD1;NP2 </ AnyOrder>*

*</QuestionClass>*

One question class may include one or more patterns, and these patterns determine whether a question belongs to this class or not. The question pattern may include normal words and key concepts. Every key concept is combined with ID and Type, which identify and recognize the concept.

The answer types are common, such as base NP, person name, location, time, etc.

The context template is quite like those of [Soubbotin01], but we use concept matching instead of its pure string matching method. We also divide the template into two types of strict order and lenient order.

## 2.2 Question Analysis

The pre-processing and indexing model is general. We analysis the morphology and delete the stop words based on the POS tagging. Only the nouns, verbs, adjectives, and adverbs are indexed.

We divide the key word into two categories: words that must appear and optional words, which are abbreviated as MA-Keywords and OA-Keywords. The OA-Keywords are replaceable words such as verbs. Others are all MA-Keywords.

When a question is presented, its question class is examined. If one question class's question pattern can match the question, the system classifies the question into this question class. Then the question will be divided into several parts by the question pattern. The system only regard the words belong to key concepts as the candidate keywords, while the others will be discarded as stop words. We then use some structures to discard other stop words, such as "the A of B" and if A is the hypernym of B, then A is the stop word. For example, the phrase "the state of Alaska", only "Alaska" will be tagged as keywords. Then we discard the words except noun, verb, adjective and adverb. Finally, we divide the remaining words into MA-keywords and OA-Keywords by the above principle. Then the system acquires the corresponding context patterns and answer types.

## 2.3 Answer extraction and merit

There is a change in this year's task. Instead of five 50 bytes (or 250 bytes) answer gobbets, every question should return an exact answer. This change requires us paying more attention to answer itself.

In order to get a high precision and a moderate recall, we limit the answer from both the inside and outside pattern of the answer. We divide the context template and answer type into two categories: the strict and the lenient. When assembling the context template and answer type, every combination is allowed, except both lenient.

Each term in the context template, including the answer, is a concept. System first locates all the concepts in the coming sentence, then checks if the sentence matches the pattern, and extract the answer if success.

One answer's score equals the summation of the score of context template, answer type, and sentence match score and concepts match score.

$$S_{ans}=S_{ct} + S_{at} + S_{sen}+ \sum S_c$$

Where, $S_{ans}$ is the score of the candidate answer, $S_{ct}$ is the score of context template, $S_{at}$ is the score of answer type, and $S_c$ is score of the concepts.

Concept match is based on keyword matching:

$$S_c =\sum S_{mk} / \text{total keywords in the concepts of the question}$$

$$S_{sen}=\sum S_{mk} / \text{total keywords in the question}$$

Where, $S_{mk}$ is the score of **m**atched **k**eywords. If two keywords are matched by original or derivation form, $S_{mk}$ is set to 1.0. If keyword in the question is the hypernym of the keyword in the sentence, , $S_{mk}$ is set to 0.8. Otherwise, the concept cannot be matched.

## 2.4 Confidence

As known to all, another dramatic change in this year's task is that the submitted questions should be ranked by confidence. This means that we should be aware of what we have submitted. We use the ME model to determine every answer's confidence[Berger96].

According to our maximum entropy based sorting method, the rank of different answer is determined by several weighted factors, and the confidence score is the product of the exponent of the weights of every factors. The weight of every factor is assigned during the training of TREC-10 questions.

As for the non-nil answers, five factors are considered, which are:

- Score (The score of $S_{ans}$ is dependent on the question, and cannot be used to compare the confidence of different questions directly.)
- Step (The answer can be extracted in step1 or step2. Step1 use strict answer patterns, while step2 use lenient patterns, so step1 is better.)
- BestCount (For each question, a lot of answer snippet can be found, some of which are the same. We use boosting algorithm to find the best answer, and the best answer can be find in several places, the number of which is the BestCount.)
- BestCount/totalCount (totalCount is the total of possible answer snippets.)
- Answer Type (Person, Place, Time, etc.)

As for the nil answers, five factors are considered, which are:

- The number of key words (Key words are those words which are important to answer finding in the question. They are extracted in our question analysis module.)
- The number of key words which cannot be matched between questions and TREC corpus.
- The total number of question words (key words as well as optional words)
- The number of all the question words, which cannot be matched between questions and TREC corpus.
- Answer Type

Following are some of our experiment results, where the training data are the 500 questions of TREC-10, and the test data are the 500 questions of TREC-11.

Sorted by question id: 0.315 (this is the baseline)

Four factors (open test, answer type is not considered): 0.461

Five factors (open test): 0.434

Four factors (closed test): 0.498

Five factors (closed test): 0.489

Therefore, our maximum entropy method does significantly better than the baseline method. In addition, only consider the first four factors is better.

# 3. Web Retrieval

This is year is the second year that we attend TREC Web task. We have submitted five runs for the topic distillation task: fduwt11b0, fduwt11t1, fduwt11o1, fduwt11t2, fduwt11o2. Detailed information of each run is given in the following table.

Table 3.1 Web runs submitted

| Run ID | Information used |
|---|---|
| fduwt11b0.submit | Use basic search |
| fduwt11o1.submit | Use title and link structure to expand and re-sort the basic search result |
| fduwt11t2.submit | Use title and anchor text to expand and re-sort the basic search result |
| fduwt11t1.submit | Use title and link structure to expand and re-sort the basic search result |
| fduwt11o2.submit | Use title, anchor text and link structure to expand and re-sort the basic search result |

## 3.1 System Architecture

The Topic Distillation task of this year requires the retrieval system return “key resources” in the .GOV corpus for certain queries. Compared with last year’s relevance retrieval task, the amount of “key resources” the system should return is not as many as relevant documents in older task. However, the quality of the key resources is more important than that of older task. Considering this specialty, we modified our original search system to let it return high precision result. The main idea does not change a lot; we use the basic search algorithm to get a set of relevant documents. Then special post processing modules are used to expand and re-sort the base set using title, link structure and anchor text information. The final result set is a set of desired key resources.

The process of indexing includes:

- Transform HTML files in the corpus into plain text. At the same time of transformation, we use special module to extract information of links with anchor text for constructing link database later.

- Index the plain text corpus for basic search algorithm
- Index the link database

The process of searching includes:

- Make morphology analysis of queries.
- Use basic search algorithm to get base set of relevant documents.
- Use post-processing modules to do expansion and re-sorting upon the base set, get the final result of “key resources”.

## 3.2 Improved kernel search algorithm

This year we have made some modifications on the kernel search algorithm based on “shortest extend”. The goal is to improve the precision of the first retried documents. We believe this will make it easy for the post-processing module to expand and re-sort.

We modified the score equation of calculating the shortest extend. The purpose is trying to avoid the situation that extents of queries longer than two words will get extremely low scores.

For shortest extent (p, q), its the score I(p, q) is

$$I(p,q) = \begin{cases} \left(\dfrac{K}{q-p+1}\right)^{a} & \text{若 q – p + 1 >= K} \\ 1 & \text{若 q – p + 1 <= K} \end{cases} \qquad (3.1)$$

in which, we set ‘a’ to1, K=16 * (length(Q) – 1), length(Q) represents the number of the word in the query Q.

We noticed that the average length of document in the .GOV corpus is longer than that in older corpus. Thus, we altered the method of getting document score. The basic idea is to lower the score of the document if the document length exceeds the average document length. The equation is show as below:

$$S(D) = \sum_{(pi,qi)\in D} I(p_i,q_i) * w1(p_i,q_i)$$
$$\text{if } (\text{length(D)} > \text{AVG\_DOC\_LEN}) \qquad (3.2)$$
$$\text{S(D)} = \text{S(D)} * \frac{\text{AVG\_DOC\_LEN}}{\text{length(D)}}$$

in which, length(D) represents the number of the word in document D. AVG_DOC_LEN represents the average length of documents in the corpus, also counted in word number.

## 3.3 Post process module

We use two ways to expand and re-sort the base set of relevant documents. Details are given in the following.

1) Use structured information to improve expand base set

Words in different parts of a document have different importance. For example, Words in title are more important than words in content; words of large font are more important than words of small font; words of bold or underlined font are more important than words of normal font. In this year’s web track, we used title

information to improve our retrieval result.

We increase the score of a document if the title of this document includes retrieval words. The increment coefficient is between 0 and 1:

$$score\_title_d=(\frac{count\_of\_match}{count\_of\_retrieval\_words})^{\gamma} \quad (3.3)$$

in which, $\gamma$ is a parameter greater than 1.

2) Use breadth-first traverse algorithm to find out key resource

By breadth-first traverse algorithm, we access all pages of a domain from the homepage of the domain. Then we construct a tree, the root of which is the homepage of the domain.

Our retrieval procedure has two steps:

Step 1: Retrieve a batch of documents using our text-only search engine. Every document retrieved is given a score. This score of document d is called $score\_text_d$.

Step 2: Calculate key resource score of a document score_KR

$$score\_KR_i=\sum_{j\in subtree(i)} \frac{score\_text_j}{\gamma^{\Delta level(i,j)}} \quad (3.4)$$

in whick, $\gamma$ is parameter greater than 1.

3) Synthesize score_ text, score_title, score_KR

$final\_score_d=\alpha * score\_text_d+\beta * score\_title_d+\gamma * score\_KR_d$

then we re-sort documents on final_score, rettun the top n documents as our result.

Because the task of “Topic Distillation” is a totally new task, we cannot use data in the past task for training. We have made a human tagging system to let several students tagging our experiments result at the same time. In this way, we can get some feedback from human.

Experiment results show that the above algorithm can lead to satisfactory results. The average P@10, P@20, P@30 of 49 queries are 0.45, 0.31, 0.26 respectively, while the average median precision is 0.11. 0.09, 0.08.

# 4. Video Track

On Video Track of this year, we paticipated in Shot Segmentation, Feature Extraction and Search task.

## 4.1 Shot Segmentation

This year we use most parts of TREC-10 Shot Segmentation System [Wu01]. FFD (*Frame-to-Frame Difference*) calculated by Luminance Difference and Color Histogram Similarity are used to detect the Shot Changes. We use two thresholds $\theta_C$ and $\theta_G$, which are calculated automatically according to the FFD value histogram in 500 frames, to detect if there is a clear FFD value change caused by Shot Changes. Then Flashlight Detection and Motion Detection are applied for candidate Shot Changes to remove the false alarms of Cut and Gradual. The parameters used in the system are trained and adjusted based on the TREC-10 Video Library. According to the performance on TREC-10 Video Library, we selected the system parameters to generate the submissions.

We add Fade In/Out Detection into our system this year although Shot Segmentation task does not include

it. In our submissions, Run02, Run09 and Run10 include Fade Detection. In our system, Fade In/Out Detection is applied to all candidate Gradual Changes. If a black screen chain exists in the candidate duration, we think it is a Fade. Otherwise, it will be labeled as Dissolve. In order to detect the black screen and get the accurate boundary of Fade, a frame is split into several blocks whose size is 8×8 (pixels). *Maximum Luminance* $Lum_{max}(n)$ and *Black Value* $Black(n)$ are calculated for each frame *n* in the candidate duration.

$$Lum_{max}(n) = \sum_{block_k \in B_{max}} Luminance(n,k)$$
$$B_{max} = \{block_{i1}, \cdots, block_{i10}\} \text{ are set of ten blocks with maximum Luminance value} \quad (4.1)$$

$$Lum_{min}(n) = \sum_{block_k \in B_{min}} Luminance(n,k)$$
$$B_{min} = \{block_{i1}, \cdots, block_{i10}\} \text{ are set of ten blocks with minimum Luminance value} \quad (4.2)$$

$$Black(n) = \frac{Lum_{max}(n)}{10} \times \frac{Lum_{max}(n)}{Lum_{min}(n)+1} \quad (4.3)$$

Then system finds four points in the candidate duration:

- Black Screen Start Point: the first frame *n* satisfies $Lum_{max}(n) < th_{black}$.
- Black Screen End Point: the first frame *n* satisfies $Lum_{max}(n) \geq th_{black}$.
- Black Decrease Point: the frame *n* which has the maximum Black Value between Gradual Start Frame and Black Screen Start Point.
- Black Increase Point: the frame n which has the maximum Black Value between Black Screen End Point and Gradual End Frame.

If Black Screen Start Point is equal to the Gradual Start Frame, we regard it Fade In. It starts from Black Screen Start Point and ends at Black Increase Point. On the contrary, if Black Screen End Point is equal to the Gradual End Frame, a Fade Out, which starts from Black Decrease Point and ends at Gradual End Frame, is detected. The third condition is that the black screen chain located in the middle of candidate duration. At this time, a Fade Out followed by a Fade In will be detected. It starts at Black Decrease Point and ends at Black Increase Point.

Evaluation shows that our system has a good balance between precision and recall. Comparing F-Value, the rank of our best result for all the changes, Cut Changes and Gradual Changes is 3, 3 and 9 (out of 54 systems). On Gradual Accuracy, frame-recall of our system is better than frame-precision. Comparing with other submitted systems, our system located at the middle on Gradual Accuracy.

## 4.2 Feature Extraction

According to the FE Task of this year, we developed a new Video Feature Extraction System. It consists of five sub-systems: Outdoor / Indoor Detection, Cityscape / Landscape Detection, Face / People Detection, Text Detection and Speech / Music / Monologue Detection. In each sub-system, a value calculated by whatever methods and features is used for ranking. In the following part, we call it “Ranking Value”. Evaluation shows that our system works well on these features: Cityscape, Landscape, Indoor and Music.

### 4.2.1 Outdoor / Indoor Detection

In our Outdoor / Indoor System, we use K-Nearest Neighbor Classifier. Considering the difference between Outdoor and Indoor, we select Color Histogram and Edge Direction Histogram as the feature [Szummer98]. A 512-bin color histogram is calculated in Color Space T=R-B. The Edge Direction is obtained by calculating the ratio of the grades on vertical direction and horizontal direction at every edge point. The edge point is got by Canny Edge Detector. In the Edge Direction Histogram, all the directions are separated as 12 bins.

The training set includes 600 indoor images and 1300 outdoor images. They are selected from Feature Dev Set. For each shot in Feature Test Set, we only apply the classifier to the selected keyframe for a shot. In Run01, the keyframe are divided into 4×4 blocks. The histogram distance is calculated for each block. After summing with weights, we will get the distance for whole keyframe. On the contrary, we only calculate the distance on whole keyframe in Run02. In each run, the Ranking Value is minimum distance between keyframe and the training images.

### 4.2.2 Cityscape / Landscape Detection

Similar with Outdoor / Indoor System, we also use K-Nearest Neighbor classifier in Cityscape / Landscape Classification. However, we only select Edge Direction Histogram as the feature in this system [Vailaya98].

The training set includes 410 cityscape images and 250 landscape images. They are also selected from Feature Dev Set. We think all the cityscape and landscape image should be outdoor image at first. Therefore, the outdoor Ranking Value calculated in Outdoor / Indoor Detection will multiply with the maximum distance output by Cityscape / Landscape Classifier. The product is the Ranking Value in this system. In the submissions, Run01 use the Ranking Value of Outdoor/Indoor Detection Run01 and Run02 use the Ranking Value of Outdoor/Indoor Detection Run02.

### 4.2.3 Face / People Detection

This system uses the same idea of TREC-10. The method consists of three steps: Skin-Color based Segmentation, Motion Segmentation, and Shape Filtering. Considering the difference between the TREC-10 Video and TREC-11 Video, some new training data selected from Feature Dev Set are added into the Skin-Color template.

We use the following equation to calculate the Ranking Value for each shot:

$$RankingValue_{face} = \frac{\#of\ \ frames\ contain\ \ face}{\#of\ \ frames\ in\ a\ shot} \qquad (4.4)$$

$$RankingValue_{people} = \frac{\#of\ \ frames\ contain\ two\ or\ more\ \ faces}{\#of\ \ frames\ in\ a\ shot} \qquad (4.5)$$

### 4.2.4 Text Detection

Same with last year's work, there are still three main parts in our Video Text Detection System: Text Block Detection, Text Enhancement and Binarization. In order to reduce the false alarms, we combine Neural

Network [Li00] as a preprocessing in our Text Block Detection.

System applies 2-level Harr Wavelet Decomposition on the image. The image will be split as four sub-bands at each level: HH, HL, LH and LL. For each N×N window, we can calculate three features:

$$E(I) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} I(i, j) \tag{4.6}$$

$$\mu_2(I) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} (I(i, j) - E(I))^2 \tag{4.7}$$

$$\mu_3(I) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} (I(i, j) - E(I))^3 \tag{4.8}$$

Totally, we can get 24 (2 [level] ×4 [sub-band] × 3 [feature] = 24) features. Eight features are selected as the input of the Neural Network.

The output of Neural Network is a confidence between 0 and 1. The more the confidence approaches 1, the more possible the window is classified as text. A threshold is used to determine whether it is a text or non-text. We select three-layer BP Neural Network as a classifier to identify text regions. Bootstrap method is used in training. The training data come from Feature Dev Set and TREC-10 Video Library.

For each shot, the system processes only one frame in every ten. On each processed frame, we use a small window (16×16 pixels) to scan the image and classify each window as text or non-text using trained neural network. When scanning the image, we move the window 4 pixels at a time. If a window is classified as text, all the pixels in this window are labeled as text. Those pixels which are not covered by any text window are labeled as non-text. Then we generate a binary map from original image. The following Text Block Detection is only applied in the region labeled as text. The experiments show that the text detection precision increases about 30% while the recall almost does not decrease.

We average the output values generated by Neural Network for all the small windows in one frame. This average value is the confidence that a single frame contain text region. We select the maximum frame confidence of all the frames in a shot as the ranking value.

### 4.2.5 Speech / Music / Monologue Detection

Speech/Music Classification is applied on the 1-second window. The features we used include: Mean and Covariance of Zero-crossing Rate, High Zero-crossing Rate Ratio, Mean and Covariance of Short Time Energy, Low Short Time Energy Ratio, Noise Frame Ratio, Mean and Covariance of Brightness, Spectral Flux, Spectral Roll-off Point, Mean and Covariance of LPC, Mean and Covariance of MFCC, Mean and Covariance of Pitch, Mean and Covariance of Band Spectrum, Mean and Covariance of Band Width [Lu01][ Scheirer97].

Nearest Neighbor Model and Gaussian Mixture Model are trained by TREC-10 Videos. Applying these trained models on 1-second window, we can get the type of each window. In our submission, Run01 uses NN Model and Run02 uses 16-mixture GMM Model.

The Ranking Value of speech, music and monologue are calculated by:

$$Ranking\ Value_{speech} = \frac{\#of\ \ windows\ whose\ type\ is\ speech}{\#of\ \ windows\ in\ a\ shot} \tag{4.9}$$

$$Ranking\ Value_{music} = \frac{\#of\ \ windows\ whose\ type\ is\ music}{\#of\ \ windows\ in\ a\ shot} \qquad (4.10)$$

$$Ranking\ Value_{monologue} = RankingValue_{speech} \times RankingValue_{face} \qquad (4.11)$$

## 4.3 Search

We have submitted four runs in Search Task. Considering the difficulty of search topics, not all of the topics are processed in each run.

The whole architecture of the Search system is almost same with last year [Wu01]. However, there are some improvements in Face Recognition and Object Search.

### 4.3.1 Face Recognition

In TREC-11 Search Task, there are four topics concerning about a certain people. Face Recognition is the basis for such topics.

The eigen-face and Fisher method are commonly used in face recognition especially when the person set is completely known. However, in most cases of video retrieval the complete person set cannot be acquired. This year we tried a fast manifold-based approach to face recognition in TREC-11 Search Task. It can be used when there are only few different images of a specific person and runs fast.

Let $X_1, X_2, \cdots, X_n \in R^d$ be the vectors of images of a specific person. Usually *n* is small and the dimension *d* is very big, i.e. $n << d$

Denote $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, $Y_i = X_i - \overline{X}, 1 \le i \le n$ and $S$ the subspace spanned by $\{Y_i\}$. The set $M = \{X : X = \overline{X} + Y, Y \in S\}$ is called manifold.

Denote $S^{\perp}$ the orthogonal complement subspace of *S*. It is easy to see that if $X \in M, (X - \overline{X}) \in S$ i.e. $P_{S^{\perp}}(X - \overline{X}) = \Phi$, or $\left\| P_{S^{\perp}}(X - \overline{X}) \right\| = 0$, where $P_S(X)$ is the projection of *X* on *S*.

Based on observation above, we scan the input image with sub-windows of different positions and levels of the image pyramid. Then the distance between the scanning window and the mean of samples in $S^{\perp}$ is calculated. The sub-window with the minimum distance may include the person we search for if the minimum distance is below a threshold.

Experiment shows this approach works quite well even in case where there are only few samples. However, it is quite time-consuming.

To speed up, we collect a large number of non-face samples $U_1, U_2, \cdots, U_m$, where *m* is very big. Denote $V_i = P_{S^{\perp}}(U_i - \overline{X}), 1 \le i \le m.$ Note that if *X* is a face then $P_{S^{\perp}}(X - \overline{X}) \approx \Phi$ and the dimension of

$S^{\perp}$ is *d-n* and very big.

We do the principle component analysis with $V_1, V_2, \cdots, V_m$ and get the first few principle components $Z_1, Z_2, \cdots, Z_k$, where *k* is small. Denote *Z* the subspace spanned by $Z_1, Z_2, \cdots, Z_k$.

Then based on the note above, we can filter out the input image *X*, if $\left\| P_Z\left(X - \overline{X}\right)\right\| > th$. Since *k* is much smaller than *d-n*, this filtering step is much faster. In our experiment, applying this step will make the face recognition 5-fold faster and with almost no decreasing of performance.

### 4.3.2 Color Histogram Comparison

Color Histogram similarity is used to compare the Image Example and Key Frame of each shot. It will provide us the similarity on the image between Image Example and Key Frame. We calculate the histogram in RGB and YUV space. During the calculation and comparison, two modes are used:

- Whole Image Mode: For both Key Frame and Image Example, the histogram is calculated on the whole image.
- Block mode: For Key Frame, we split it into several blocks with different size. The histogram is calculated on each block. For Image Example, the histogram is calculated on the whole image. Then the histogram comparison is processed between the histogram of each block and image example. The maximum similarity will be selected as the final similarity.

In searching, Block mode is used on the topics which is concerning about a certain object. Such as parrots, butterfly and so on.

### 4.3.3 Searching

For each topic, we combine the similarities come from different modules. Such as Face Recognition, Text Recognition, Color Histogram Comparison, ASR Text etc.

In our submission, Sys1 only use the information get by our own search modules. There is no ASR Text and Feature Extraction results are used. However, Feature Extraction Confidence is useful for some topics. For example, Face Confidence is useful to certain people searching and Cityscape Confidence is useful for Topic 86. So in Sys2 and Sys3, we combined feature extraction confidence into the searching. Sys2 use our own Feature Extraction results and Sys3 use the reference Feature Extraction results provided by IBM and MediaMill. In Sys4, we combine the ASR Results provided by LIMSI. We select some keywords manually for each topic. These selected keywords are used for Text Retrieval on ASR Results.

NIST's evaluation shows that our searching system is not efficient in several topics. In the future work, we should pay more attention on Image Similarity calculation.

## ACKNOWLEDGMENTS

This research was partly supported by NSF of China under contracts of 69935010 and 60103014, as well as the 863 National High-tech Promotion Project of China under contracts of 2001AA114120 and

2002AA142090. We are thankful to Lin Mei, Yuefei Guo, Sitong Huo, Yi Zheng, Xin Li, Kaijiang Chen, Xiaoye Lu, Jie Xi, He Ren, Li Lian, Wei Qian, Hua Wan, Tian Hu, Jiayin Ge, Jian Gu, Danlan Zhou, Zhiyan Tang, Xipeng Qiu, Lan You, Lin Zhao, Rongrong Wang, Min Jin, Jiawei Rong, Wanjun Jin for their help in the implementation.

## Reference

[Berger96] Adam L. Berger, Stephen A. Della Pietra, Vincent J. Della Pietra, *A Maximum Entropy Approach to Natural Language Processing*, ACL'96

[Harabagiu99] Sanda Harabagiu and Dan Moldovan. *FALCON: Boosting Knowledge for Answer Engines.* Proceeding of The Eighth Text Retrieval Conference, November, 1999

[Li00] Huiping Li, *Automatic Processing and Analysis of Text in Digital Video*, Technical Report of Center for Automation Research, University of Maryland College Park, December 2000

[Littlestone88] N.Littlestone, *Learning quickly when irrelevant attributes abound: a new linear threshold algorithm*, Machine Learning, 2: 285-318, 1988

[Lu01] Lie Lu, Hao Jiang and Hongjiang Zhang, *A Robust Audio Classification and Segmentation Method*, Proc. of the 9th ACM International Multimedia Conference and Exhibition, pp. 103-211, 2001

[Scheirer97] E. Scheirer and M. Slaney, *Construction and Evaluation of a Robust Multifeature Music/Speech Discriminator*, Proc. of ICASSP'97, vol. II, pp 1331-1334. IEEE, April 1997

[Soubbotin01] M. M. Soubbotin, S.M. Soubbotin. *Patterns of Potential Answer Expressions as Clues to the Right Answers,* Proceeding of The tenth Text Retrieval Conference, November, 2001

[Szummer98] Martin Szummer, *Indoor-Outdoor Image Classification*, IEEE International Workshkop on Content-based Access of Image and Video Databases, in conjunction with ICCV'98, Jan. 1998

[Vailaya98] Aditya Vailaya, Anil Jain, HongJiang Zhang, *On Image Classification: City Images vs. Landscapes*, Pattern Recognition, Vol.31, pp1921-1936, Dec. 1998

[Wu01] Lide Wu et al., *FDU at TREC-10: Filtering, Q&A, Web and Video Tasks*. Proceeding of The tenth Text Retrieval Conference, November, 2001