WIM at TREC 2007

Jun Xu, Jing Yao, Jiaqian Zheng, Qi Sun, Junyu Niu
{062021158, 062021115, jqzheng, 052021188, jyniu}@fudan.edu.cn
Computer Science and Engineering Department of Fudan University,
Shanghai, China 200433

Abstract

This paper introduced the four tracks that WIM-Lab Fudan University had taken part in at TREC 2007. For spam track, a multi-centre model was proposed considering the characteristics of spam mails in contrast of traditional 2-class classification methodology, and the incremental clustering and closeness-based classification methods were applied this year. For enterprise track, our research was mainly focused on ranking functions of experts and selecting correct supporting documents regarding to a given topic. For legal track, the effects of word distribution model in query expansion and various corpus pre-processing methods were mainly evaluated. For genomics track, three score methods were proposed to find the most relevant text snippets to a given topic. This paper gives an overview of the methods employed for each sub tasks, and compares the results of each track.

1. Introduction

For spam track, in real world there could be several different genres in the spam category. There was a great diversity within the single spam class, while at the same time they might be similar to some certain kinds of ham mails. For this reason, in addition to the traditional 2-class classification methodology, we proposed a multi-centre model and applied the incremental clustering and closeness-based classification method to the Spam Track tasks this year. We applied character-level model and online linear classifier as our basic classification method, then we adopted a multi-centre model to treat the mails that were equivocal to the basic classifier. This method was evolved from the idea once used in [7] for expanding a credible negative sample set from the unlabeled training corpus. We developed this idea to make it compatible to the spam-filtering task.

For enterprise track, we participated in both tasks for the track. A new enterprise corpus was introduced in the track this year, CSIRO repository instead of W3C repository, which makes great difference in the tasks and the topics for the tasks were provided by science communicators which are of great real meaning. However, chances always come from the challenges. Some new methods are used to accomplish the tasks.

For legal discovery track, our team submitted 5 runs finally (1 manual + 4 autos). The objective of Legal Track is to evaluate the efficacy of automated support for review and production of electronic records in the context of litigation, regulation and legislation. The corpus consists of 650 xml files, 60G+ after unzipped. Each file contains multiple documents. In the document, 61 types of xml leave nodes form all textual information. Since legal corpus is converted from OCR format by program automatically, corpus may contain lots of meaningless text blocks and latent data inconsistency, which makes it a challenge for lawyers to lookup related documents supporting their quoting. The query of each topic is give by xml format as well. Participating teams can build queries in any way they like, using materials provided in the complaint, the production request, the boolean query, and any external resources that they have available.

For genomics track, the system was required to extract the relevant passages of text that answers the topic questions [10]. It's similar to the task of Genomics Track 2006 except the question types. Our group submitted three runs based three different score models. Three methods had the same process which extracted the concepts for followed scoring process.

2. Spam Track

2.1. Overview

Many traditional spam filters regarded anti-spam challenge as a 2-class classification problem, yet there was a potential premise for us to use common 2-class classification methods to solve this problem. The premise was that samples in the same class were quite similar while samples in different classes had a distinct difference. Common 2-class classification methods could somehow extract some implicit features to predict whether a mail was ham or spam. But empirically the border between spam and ham was not so clear even for manual recognition. And there was also a great diversity within the single spam or ham class. For instance, there could be several different genres in spam, such as automated, list, newsletter, phishing, sex, virus and etc. Various kinds of spam mails were quite different to each other, while at the same time they might be similar to some certain kinds of ham mails.

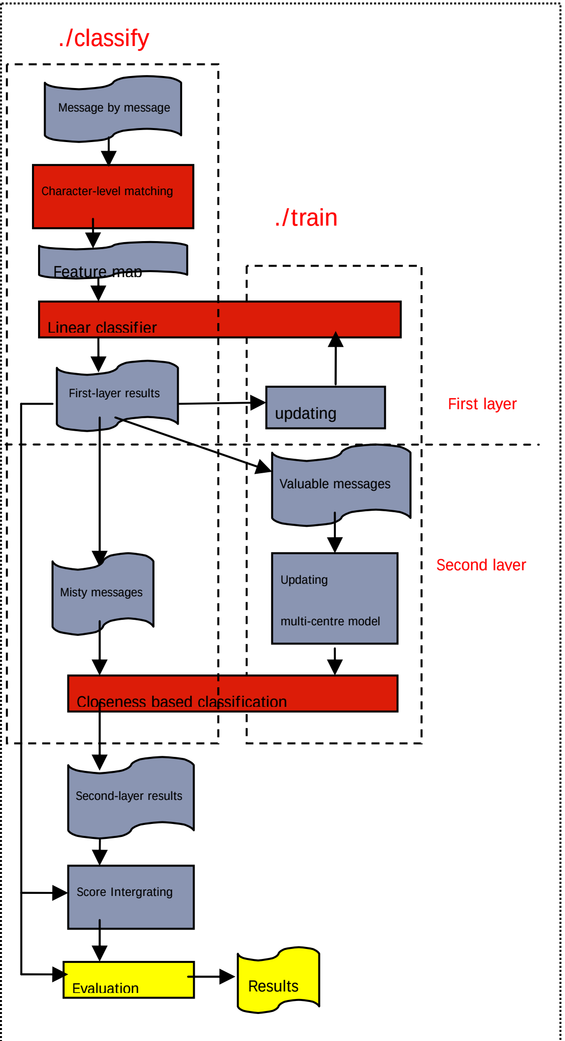
For this reason, in addition to the traditional 2-class classification methodology, we proposed a multi-centre model and applied the incremental clustering and closeness-based classification method to the Spam Track tasks this year. And we tried to evaluate whether this method could make some further improvements to the precision of the spam filters.

We applied character-level model and online linear classifier as our basic classification methods, which were inspired by IJS [5] and Tufts [6] in last two years' tasks. Then we adopted a multi-centre model to treat the mails that were equivocal to the basic classifier. This method [7] was evolved from the idea once used for expanding a credible negative sample set from the unlabeled training corpus, and worked well on the TREC 2005 Genomics Corpus. We developed this idea to make it compatible to the spam-filtering task.

2.2. Methods

The "FDW" filter we submitted had a two-layer structure. We used character-level model and linear classifier as the first-layer filter. During the "classify" period, each mail processed through the first-layer filter would be given a spamminess to denote the likelihood to be a spam or not. Mails with a relatively extreme spamminess score would skip the second-layer filter and were put into the spam or ham class directly, while those with moderate scores would go to the next-layer filter. The second-layer filter used a multi-centre model. We applied a closeness-based text classification method on it to get an adjustment score of the mail. Finally, we integrated the scores of the two layer filters together to determine if the incoming mail was a spam.

During the "train" period, the parameters of the first-layer filter were adjusted according to the online linear classifier mechanism. We choose part of the mails by the spamminess score given by the first-level filter, and then use incremental clustering method to establish and update a multi-centre model for the second-layer filter. The details of these methods were described in the following subsections.



System Architecture

2.2.1.Character-Level Matching and Linear Classification

Character-level spam filters had the advantage to avoid the vulnerability of the key-word escape attacking by spammers compared with those bag-of-words filters. [5] and [6] had shown this advantage on the basis of the experimentation in last two years' Spam Track. We designed our first-layer filter referring to the inexact word matching idea by [6], but made some changes.

The first difference was that we used a fuzzy weighting strategy to denote the similarity of the character-level matching instead of giving a binary score to each explicit string feature, since one certain string in the feature space might have tens of transformed forms by the inexact matching method, and from the empirical view most of them would have a relation with the target string but weaker than the exact matching. The related weighting method was once proposed in [9]. The method introduced a decay factor $\lambda \in (0,1)$ to weight the presence of a certain feature in a text. The weight of each dimension is λ n, where n is determined by the similarity between the incoming text and the string in the feature space. In our system, as we only considered one-character obfuscation as [6] did. We simplify our fuzzy weighting strategy as follows:

matching condition	weighting score
exact matching	1
inexact matching	λ
no matching	0

The second difference was that we used a fixed length of string feature space instead of limiting of the string length in an optional range. This would save some computational cost and empirically too short length of string would do little benefit but bring noise to the filter.

After the character-level feature mapping, we applied the Perceptron classifier to do the first-layer filtration.

2.2.2. Incremental Clustering and Closeness Based Classification

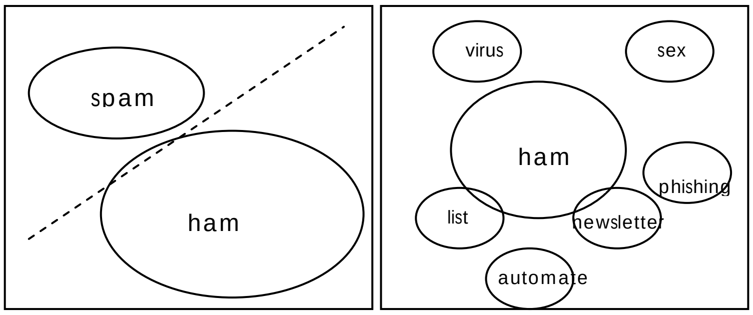
There were two hypotheses for us to think of using the multi-centre model in our system.

The first hypothesis was that in one hyperspace, the spam and the ham were not distributed as two single clusters. By practical experience, we found that there were a variety of types of mails in the spam category in the real world. Each type of mails had their own characteristics and was quite different from each other type. So we suggested a model that spams were represented in scattered clusters while hams still in one single cluster. Later we would evaluate if this model would be more reasonable to represent the spam and ham distribution by experiments.

The second hypothesis was that for a practical spam filter, a low hm% misclassification rate was more important than sm%. So a misty mail would be judged spam only when it was a credibly spam sample in the multi-centre model.

For these two hypotheses, we applied incremental clustering and closeness based classification methods to our system.

The similar methods were originally proposed by [7], and had produced effective results for the



classification task on TREC 2005 Genomics Corpus. The scenario of the method was that there was a corpus with some pre-labeled credible positive samples and only a few credible negative samples extracted by the Rocchio classifier, and then it was needed to expand the negative sample set to facilitate further classification. The methods were overviewed as follows.

Firstly it used the k-means algorithm to divide the previously-produced credible negative sample sets into k clusters to generate a multi-centre model. Next a closeness-based classification method is applied to expand the negative sample sets. We denoted the k clusters of the credible negative set with $N_1, N_2, ..., N_k$, and denoted the centers of them as $C_1, C_2, ..., C_k$.

The closeness of each cluster represents the average similarity of the samples within a cluster, it was represented as:

$$Cl(N_i) = \frac{1}{|N_i|} \sum_{d_j \in N_i} S(d_j, C_i)$$

The average closeness of the negative sample set (including k clusters) was:

$$Cl(N_{negative}) = \frac{1}{k} \sum_{i=1}^{k} Cl(N_i)$$

And the difference between the similarity within the negative set and the similarity between the positive and negative sets was defined as:

$$Df = Cl(N_{negative}) - \frac{1}{k} \sum_{i=1}^{k} \frac{1}{|N_i|} \sum_{d_j \in N_i} S(d_j, C_{positive})$$

For an incoming sample text d, there were two requirements need to be filled to be regarded as a negative sample:

$$\max_{i=1...k} \{S(d, C_{N_i}) - S(d, C_{positive})\} > Df$$

$$\max_{i=1..k} \{S(d, C_{N_i})\} > Cl(N_{negative})$$

The closeness based method mainly solved two problems. One was that it was compatible to the situation that the samples in the hyperspace are not distributed centralized and well-proportioned. The second point was that it made the expanding negative samples credible and left those misty samples untreated.

These two points were actually suitable to the situation we met in the anti-spam scenario. While in our hypotheses proposed above, the spam mails were distributed in several scattered clusters, and moreover, we needed to be more careful to judge a mail as spam than ham in order to avoid the risk of losing important messages.

Here was an analogy between the scenarios in the Closeness based method to expand negative sample sets [7] and in the Spam Track.

Closeness based method to expand negative sample sets [7]	Our System
Credible Negative samples	Known Spam mails
Labeled Positive samples	Known Ham mails
Unhandled samples needed to be judged if it belonged to the expanding negative sets	Incoming messages to be judged as ham or spam
Negative examples were not distributed well-proportioned and needed to be clustered	Spams were multi-centered
Only credible negative texts would later be included in the negative sets	Low hm% is more important than sm% in real world

As for the common benefits the idea of [7] would bring, we inherit the idea in our second-layer filter but made some changes to make it adaptive in our system. The main challenge we face was the computational cost of the system since an online spam filter has many limits on time and space requirements.

For the “train” aspect, the main task was to establish a multi-centre model and update the related information of each clusters when new mails coming. As the mails were coming in a stream form and considering the time limit for the system, it was unacceptable to apply a k-means or some other static clustering methods to the system. We adopt two measures to solve this problem. First we limit the number of mails to be trained in the second-layer filter. We set a score range *r* and only the mails whose first-layer filter score were accepted by the range were regarded as valuable to be trained in the next-layer filter. Secondly we applied an incremental clustering method to build and update the multi-center model.

The pseudo code was as follows.

```
Second-layer Training
1.  if (message.first_layer_score ∈ r)
2.          if (message.judge=ham)
3.          add message to ham cluster
4.          update Cpositive, Df
5.          end if
6.   else if (message.judge=spam)
7.          if (distance(message,nearest cluster centre Ci)<t)
8.                  add message to the specific cluster Ni
9.                  update $C_i$, $Cl(N_i)$, $Cl(N_{negative})$,Df
10.             end if
11.             else if (distance(message,nearest cluster centre)>=t)
12.                     create new cluster $N_{k+1}$
13.                     update $C_{k+1}$, $Cl(N_{k+1})$, $Cl(N_{negative})$,Df
14.             end if
15.         end if
16.     end if
```

Considering the computational cost, we changed the calculation of some attributes into an approximate form to make it adaptive for incremental computation.

$$C_{new} \approx [C_{old} \times (|N|-1) + d] / |N|$$

$$Cl(N_i)_{new} \approx [Cl(N_i)_{old} \times (|N_i|-1) + S(d, C_{N_{new}})] / |N_i|$$

$$Df \approx Cl(N_{negative}) - \frac{1}{k}\sum_{i=1}^{k} S(C_i, C_{positive})$$

And the parameter *t* also played an important role to control the number of clusters and the effect of the second-layer filter.

During the experiments we found that the computation of closeness *Cl* and difference *Df* would bring little benefit to this filtering scenario, so a more simplified method was used in our submitted versions. We only computed and updated the cluster centre *C* of each cluster during the clustering. And the filtering results were determined by:

$$\max_{i=1 \ldots k}\{S(d, C_{N_i}) - S(d, C_{positive})\}$$

Finally we integrated this score and the first-layer score together with certain weights to work out the final spamminess score.

2.3. Systems submitted for Trec 2007

2.3.1. Active Learning Mechanism

This year we didn't put much emphasis on the active learning mechanism. We proposed a naïve mechanism as follows: We tried to train the mails as early as possible, and we assumed that mails with extreme spamminess score were credible for training. To implement this mechanism under the Spam Track Framework, we simply gave those mails with moderate spamminess scores "Label N" labels, and gave those mails with extreme spamminess scores "Label B" labels.

2.3.2. System Configuration

This year we submitted four filters to participate in the Spam Track. The configurations of the filters were as follows:

Filters/Methods	Closeness Based Clustering	Inexact Matching	Fuzzy Weighting	Upper Limit of Characters to Process
Fdw1	Yes	Yes	Yes	5000
Fdw2	No	Yes	Yes	5000
Fdw3	Yes	Yes	Yes	3000
Fdw4	No	Yes	No	5000

2.4. Results and Future Works

Here is the (1-ROCA)% statistics of our submitted filters on this year's tasks:

Tasks/Filters	Fdw1	Fdw2	Fdw3	Fdw4	Median	Best
Trec07p-full	0.0198	0.0195	0.0157	0.0109	0.03	0.003
Trec07p-delay	0.0223	0.0159	0.0367	0.0229	0.1	0.01
Trec07p-partial	0.1066	0.0921	0.1109	0.1151	0.1	0.03
Trec07p-active1000	0.0641	0.0881	0.1629	0.2029	0.1	0.01
Mrx3-immediate	0.0155	0.0147	0.0154	0.0255	0.1	0.003
Mrx3-delay	0.0747	0.0751	0.1062	0.1258	0.3	0.03

From the comparison between Fdw2 and Fdw4, we found the fuzzy weighting method could make some improvements to the filter. From the comparison between Fdw1 and Fdw3, we found in intermediate tasks Fdw3 performs as well as or even better than Fdw1, while in delayed and active learning tasks, Fdw1 has some obvious advantages. Yet how the maximum length for each mail to be processed could affect the filter performance was still not determined by our experiments. And from the comparison between Fdw1 and Fdw2, which actually shows the filter performance with multi-centre model and without multi-centre model. They both perform well, but we haven't found obvious improvements by applying the model to the filter.

So far we still have several problems to solve in our future work. As for the multi-centre model itself, we think it has some reasonable factors to solve the spam filtering scenario; this is an attempt during the beginning phase of our work and we would try to improve our methods more considerately. And another crucial point in our further work would be how to control the computational cost of the spam filtering method. As the mail streams arrived continuously, it is a realistic problem to control the expanding scale of temporal and spatial cost, especially for the multi-centre and clustering related methods. So we would focus the study on making our spam filtering methods more adaptable to huge volume stream data in real world.

3. Enterprise Track

3.1. Candidate profiling and searching

For the expert search task, as no candidate list was provided, the first thing we did was to recognize emails using the pattern "first.last@csiro.au" as expert identifiers and found candidates' full names in the context of emails. However, we got almost 4000 candidates and the list seemed too big. As key people were CSIRO staff members who were the correct key contacts for this topic e.g. the project leader according to the guideline, we filtered the candidate list by the rule that the candidate should be contact on some project at least once. In addition, to evaluate the relationship between expert and page, we also gave higher weight if the candidate appeared in the document as a contact for he was more responsible for the document than other candidates. Since the tie between the email and the name can not be completely accurate. We separately calculated the score for each candidate's identifier on each topic, one for the candidate's name and another for his email. We added the two scores in some proportion to get the candidate's final score.

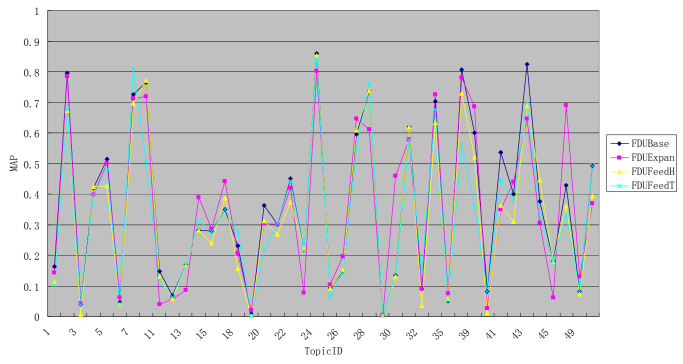
3.2. Document grouping and categorizing

The enterprise track had two big changes from last year: new corpus and no candidate list for the expert search task. However, some example pages were listed for each topic to simulate the pages which were often clicked according to the click log. We found that the most example key pages were of same kind: home project pages which were preferred for the task. So our system made feedback runs based on the page structure to find the same kind pages as example key pages. We analyzed the page, got out the id from html elements, for example <table id="expertTable" ….> ….</table>, and made all ids stable and hash them up. Using the hash value as the key in the dictionary, every single page had the hash value as the recognizing of the "type" of it. When we found those pages, we gave those pages higher weight to improve the rank of the documents relevant to the topic.

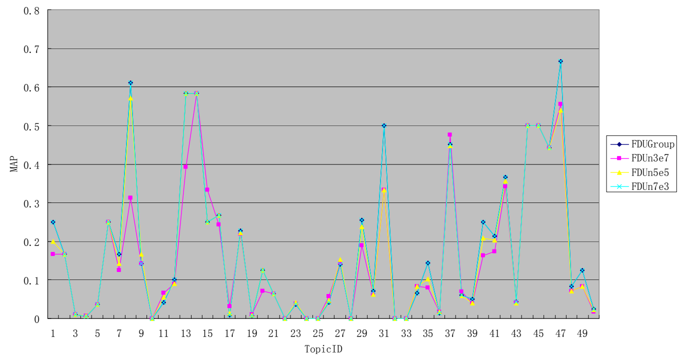
3.3. Overview, strategies and runs

Document Search Task was the first enterprise track experiment involving document search over a full crawl. As our users were science communicators, retrieved documents should be those that help them create an overview page in the given topic area, which was quite different from the general document search task [1]. These "key" pages would tend to be authoritative pages such as project homepages and documents dedicated to the topic, rather than pages that made passing mention of the topic. For document search, we submitted four runs. First, FDUBase was query only run. We used Lemur as the search engine to index and query the topics. We added the score of the document by analyzing the pages which linked to this document. Second, we use auto term expansion mehod in FDUExpan run. By analyzing the narrative field, we automatically chose candidate expansion terms for topics and calculate the frequencies of the terms in the first five documents in the search result by Lemur. We used those high frequency terms as query terms for the second time search. Then we added the score of the document by analyzing the pages which linked to this document. Third, we used html tag as classification evidence in FDUFeedT run. We used FDUBase as the basic result. Then we analyzed the structure of each document and if the document had similar structure as the documents in the page field, we added the score of the document. At last, we took advantage of hits score as classification evidence for FDUFeedH run. We used HITS algorithm to judge the quality of the document and regard documents of the same quality as a category. We used FDUBase as the basic result. If the document was in the same category as the documents in the page field, we added the score of the document.

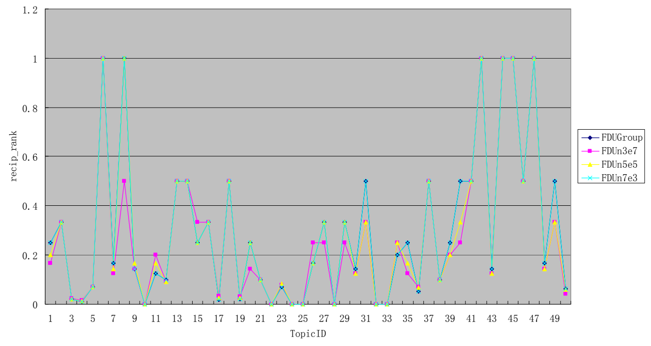
For expert search, we also submitted four runs. First, FDUn5e5 1, which gave the portion that name: 50% and email 50%. We detected email addresses and relevant full names automatically from the corpus. We also filtered the candidate list and remain those who were probably contacts on some projects. We calculated two scores for each candidate, one for the candidate's name and another for his email. We added the two scores by 50% and 50%. Second, FDUn3e7 3 which gave name 30% and email 70%. We detected email addresses and relevant full names automatically from the corpus. We also filtered the candidate list and remain those who were probably contacts on some projects. We calculated two scores for each candidate, one for the candidate's name and another for his email. We added the two scores by 30% and 70%. Third, FDUn7e3 4, which gave name 70% and email 30%. We detected email addresses and relevant full names automatically from the corpus. We also filtered the candidate list and remained those who were probably contacts on some projects. We calculated two scores for each candidate, one for the candidate's name and another for his email. We added the two scores by 70% and 30%. Finally, FDUGroup 2, which used group search. We detected email addresses and relevant full names automatically from the corpus. We also filtered the candidate list and remain those who were probably contacts on some projects. We divided the corpus to several groups according to the document structure. We gave different weights for the different kinds of the documents when calculating each candidate's score on the topic.



MAP Result for Document Search Task



MAP Result for Expert Search Task



RR Result for Expert Search Task

3.4. Conclusion

From above illustrations we can conclude that: to conquer the new task we were using several new approaches such as expert profiling and document categorizing. We mainly illustrated the technical issues we faced after an introduction of expert search. Also we explained where our final result was coming from and how we chose our model and organized the system. The results show that the "group" method did not improve the document search result effectively. It is mainly because the example pages given do not cover all kinds of pages required by users. FDUn7e3 run preformed best in all of the runs which shows the importance of name in expert finding. Although the email is more correct, names are more helpful in person's expertise judgment.

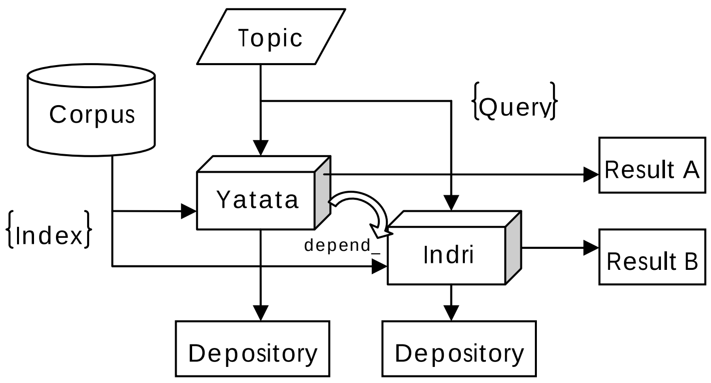
In the further work, we will pursue the study of finding similar-structure documents to improve the document search results. We will focus on the methods to extract similar-structure documents more accurately. Furthermore, we will pay great attention to the improvement of expert profiling model to get more information about the experts besides their expertise.

4. Legal Track

4.1. Task Introduction

WIM team participated in the main task of TREC 2007 Legal Discovery Track, and submitted 5 runs finally (1 manual + 4 autos). The objective of Legal Track is to evaluate the efficacy of automated support for review and production of electronic records in the context of litigation, regulation and legislation. The corpus consists of 650 xml files, 60G+ after unzipped. Each file contains multiple documents. In the document, 61 types of xml leave nodes form all textual information. Since legal corpus is converted from OCR format by program automatically, corpus may contain lots of meaningless text blocks and latent data inconsistency, which makes it a challenge for lawyers to lookup related documents supporting their quoting. The query of each topic is give by xml format as well. Participating teams can build queries in any way they like, using materials provided in the complaint, the production request, the boolean query, and any external resources that they have available.

4.2. System Overview



Before building search platform, we found there were many duplicated steps between pro-processing corpus and building query. So we built a unified framework to prepare the process of indexing and searching. The system framework lists as above.

In the framework, considering the efficiency, we employed Indri 2.3 (also known as next generation of lemur) as the main engine to index the whole corpus and returned the basic query results. The management module was called Yatata coded by java. Yatata was developed by WIM team for this year's legal track. It was an independent mini search engine fully implementing B-tree dictionary, core inverse table, query purser and candidate documents ranking. The reason why we took effect to develop a private search engine is that the experiment of many new methods could not be implemented in existed engine framework. But finally we only apply it as management module to do assistant work for Yatata's low performance when the scale of corpus extends to 50GB+. To make it clear, we made experiments between Indri and Lucene on the indexing efficiency using their default parameter settings. Indri took 48 hours to finish indexing the whole corpus while Lucene took 26 hours to finish 1/26 of them. For the reason that java is mush slower than native code in I/O operation we applied Yatata as the assistant of Indri.

In index step, we make two copies of depositories. First one was indexed by Indri directly without any pro-processing. The second one processed by Yatata, which removed "stop words" and "meaningless words", took word relevance statistics, constructed distribution model (abbreviated to DM, which will be described in next section) and finally input to Indri as another parallel depository.

In the step of searching, we generated different runs based on depository A or B. Required by the virtual court scenario, legal track preferred recall to precision in the production of documents. Most of teams in last year took the step of query expansion in their system. We tentatively handled the query expansion by applying DM built in the step of indexing by Yatata. Next section is the detail about DM application.

4.3. Methodology

Similar to idea of scoring and term weighting[11], we calculated the distribution of each word in whole corpus as the background model. In background model, each word had a mapped float value BM_w representing how frequently it appeared in the whole corpus. This mapping information was maintained by Yatata. When one topic was submitted to Yatata, by measuring the distribution of words in fields of <request text>, <instruction>, <definition>, <complaint> related to each topic, Yatata generated topic-specified distribution models for each topic. In this model a float value TM_w was mapped to each word appeared in query expression. Here is the expansion formulation:

$$foreach\ w \in TM \begin{cases} add(w) = true & if\ \frac{TM_w}{BM_w} > avg(TM,BM) + \sigma(TM,BM) \\ add(w) = false & otherwise \end{cases}$$

In above formulation, avg(TM,BM) denotes for the average value of the division TM_i/BM_i for each word i in TM. Similarly σ(TM,BM) denotes for standard deviation of the value TM_i/BM_i. DM takes the factor into consideration that more frequently word appeared in specific topic relatively, the more important it related to current topic.

On opposite sides of expansion, we tentative carried out the method of query shrink as follow:

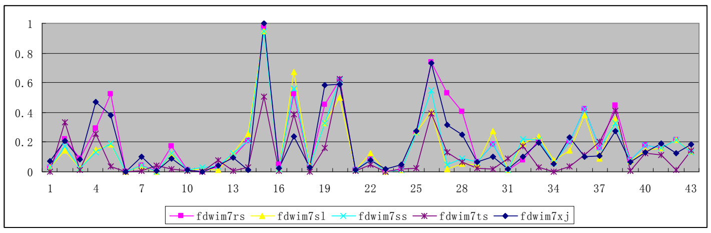
$$foreach\ w \in TM \begin{cases} del(w) = true & if\ \frac{TM_w}{BM_w} < avg(TM,BM) - \sigma(TM,BM) \\ del(w) = false & otherwise \end{cases}$$

Among the five runs submitted by WIM, four of them were generated by applying DM. and the query of last one is built by manual as baseline. Here is the list of all runs.

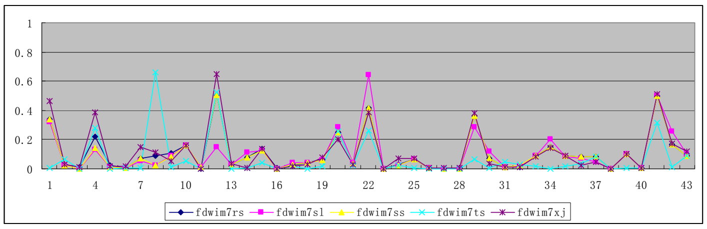
Run ID / Type	fdwim7ts	fdwim7rs	fdwim7ss	fdwim7sl	fdwim7xj
auto/manual	auto	auto	auto	auto	manual
pre-process corpus	no	no	no	yes	no
DM to expansion	no	no	yes	no	no
reduced by DM	no	yes	no	no	no

4.4. Result Analysis

All runs submitted by WIM only took use of textual information. Since final scores were generated by pooling algorithm, we selected est_PB and est_RB as the main evaluation. Here is the illustration of the comparison among the five run iteratively by topics. The baseline fdwim7xj approached to median score of whole runs in Legal Track 2007.



The estimated precision @ B



The estimated recall @ B

From the illustration, we can learn that different methods would cause great performance fluctuation. Since Legal Track 2007 gave the scores by topics, in order to make the comparison clear, we used the aggregate voting function. For each topic, if the run ranked at 1st, the bonus was 4 points; and the 2nd with bonus 3 points, and etc. We reached the result in evaluation est_PB that fdwim7rs > fdwim7xj > fdwim7sl > fdwim7ss > fdwim7ts, and the same as est_RB.

To conclude, only fdwim7rs was above the median score, while other three runs were under performance. This result leaded to the conclusion that:

a) Using DM to query shrink could harvest better performance

b) Query expansion implemented DM seems to reduce the original precision and recall. In another word, the text information existed in fields of <instruction>, <definition>, <complaint> may be useless in enhancing retrieval performance.

c) Indri prefers raw materials to pre-processed material during the step of indexing.

The future work can be focused on handling data inconsistency in legal corpus. Perhaps many runs of our team suffered low recall scores by unsuited pre-processing.

5. Genomics Track

5.1 Overview

For the TREC 2007 Genomics Track, the system was required to extract out the relevant passages of text that answers the topic questions [10]. It's similar to the task of Genomics Track 2006 except the question types. We group submitted three runs based on three different score models. Three methods have the common process which extracts the concepts for followed scoring process.

5.2 Relevant Concept Extraction

A topic question is an information need unit, in which some key biological entities can catch the leading need. The relevant concept extraction is based on the heuristic method that the frequent biological entities occurring around those key ones in the text should have some association with them. So we find the relevant concepts from the context of those key entities. Concretely, we retrieval top 1000 sentences for each topic question by language model ranking strategy, and remove these sentences that don't contain any of those key biological entities. We think of the top most frequent biological entities as relevant concepts according to the remaining sentence snippets. The extracted concepts are used to expand the corresponding topic question.

5.3. Score Models

We employ three methods to score sentences: 1) sentence language model; 2) context language model; 3）boost co-occurring method.

The first one, sentence language model, is our baseline method, which looks at each sentence in the corpus as a document. Then the language model with a linear smoothing is used to score and rank them according to the expanded query.

The context language model definitely involves the context component. That's, the context text can increase the score of a sentence to some extent according to the text relevance with the given topic question. We score the sentence by employing following context language model:

$$P(A \mid S_{j,k}, Q) = \prod_{i=1}^{n} p(c_i \mid S_{j,k})^{\alpha} p(c_i \mid D_k)^{(1-\alpha)}$$

$$\underline{\underline{rank}} \sum_{i=1}^{n} (\alpha \log p(c_i \mid S_{j,k}) + (1-\alpha) \log p(c_i \mid D_k))$$

The third method scores sentence according to two basic components: baseline score and boosted score derived from the co-occurring of different type of biological entities. The baseline score is calculated from the following formula:

$$\text{BScore} = \sum_{i=1}^{n} \frac{C_i}{SLen} C_i$$

The boosted score is calculated as follows:

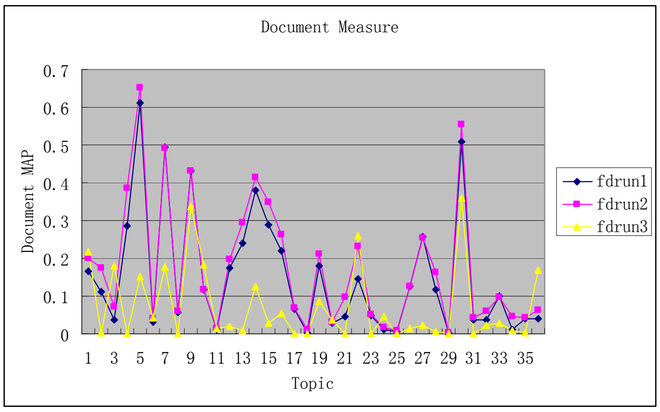
BoostedScore = 2 × min(rEntityNum, lEntityNum) × min(maxlweight,maxrweight)

The final score of a given sentence is the sum of these two parts. That is:

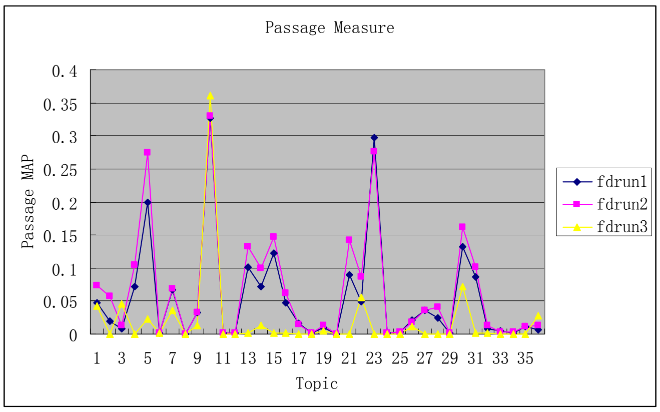
SentenceScore = BScore + BoostedScore

5.4. Results

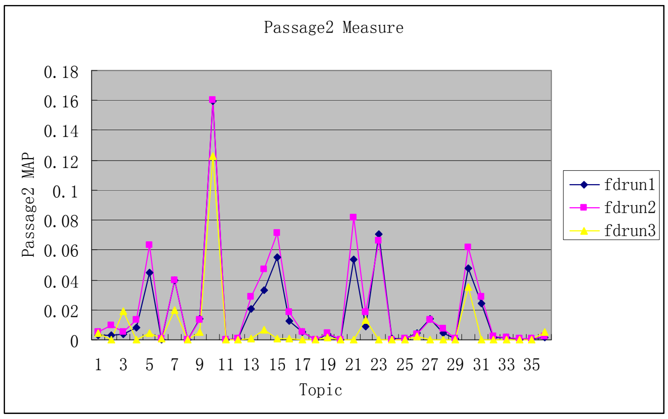
The following figures give the result of each run regard to each performance measure method:



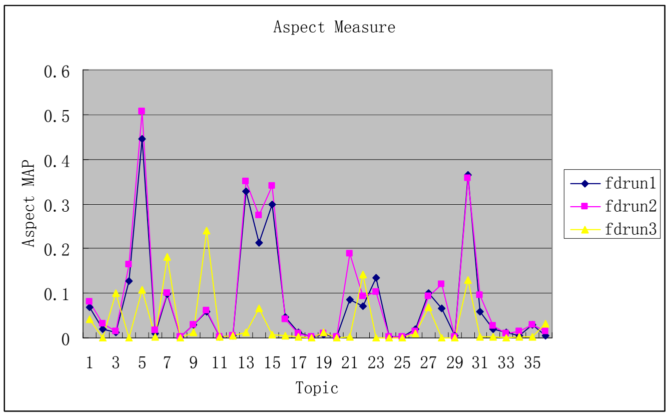
Document MAP of each topic for three submitted runs



Passage MAP of each topic for three submitted runs



Passage2 MAP of each topic for three submitted runs



Aspect MAP of each topic for three submitted runs

For each measure, the overall performance of fdrun2 is better than other two runs. Two reasons support the success of the second method: Firstly, the concepts, instead of key words of questions, adopted in the method guarantee a better recall; Secondly, the contexts of each underlying answer sentence boost the precision of answers to each question.

Reference

[1] "TREC-2007 Enterprise Track Guidelines", http://www.ins.cwi.nl/, 2007

[2] Junyu Niu, Chen Lin, "WIM at TREC Enterprise Track", In: Proceedings of 15th Text Retrieval Conference (TREC 2006), 2006.

[3] Jing Yao, Jun Xu, Cheng Jin, Junyu Niu, "Role Centralized Modeling for Expert Search in Enterprise Corporation", In: Proceedings of The 7th International Conference on Advanced Language Processing and Web Information Technology, 2007.

[4] K. Balog, L. Azzopardi, and M. de Rijke. "Formal models for expert finding in enterprise corpora", In SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference, 2006.

[5] Spam Filtering using Character-level Markov Models: Experiments for the TREC 2005 Spam Track, Andrej Bratko and Bogdan Filipic

[6] Spam Filtering using Inexact String Matching in Explicit in Explicit Feature Space with On-Line Linear Classifiers, D. Sculley, Gabriel M. Wachman, and Carla E. Brodley

[7] A Closeness-based Semi-supervised Text Classification Method, Zheng Haiqing, Lin Chen and Niu Junyu

[8] WIM at TREC 2005, J Niu, L Sun, L Lou, F Deng, C Lin, H Zheng, X Huang

[9] Text Classification using String Kernels, Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini and Chris Watkins

[10] TREC 2007 Genomics Track Protocol. http://ir.ohsu.edu/genomics/2007protocol.html

[11] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. An Introduction to Information Retrieval. Cambridge University Press, Cambridge, England