NLPR at TREC 2003: Novelty and Robust

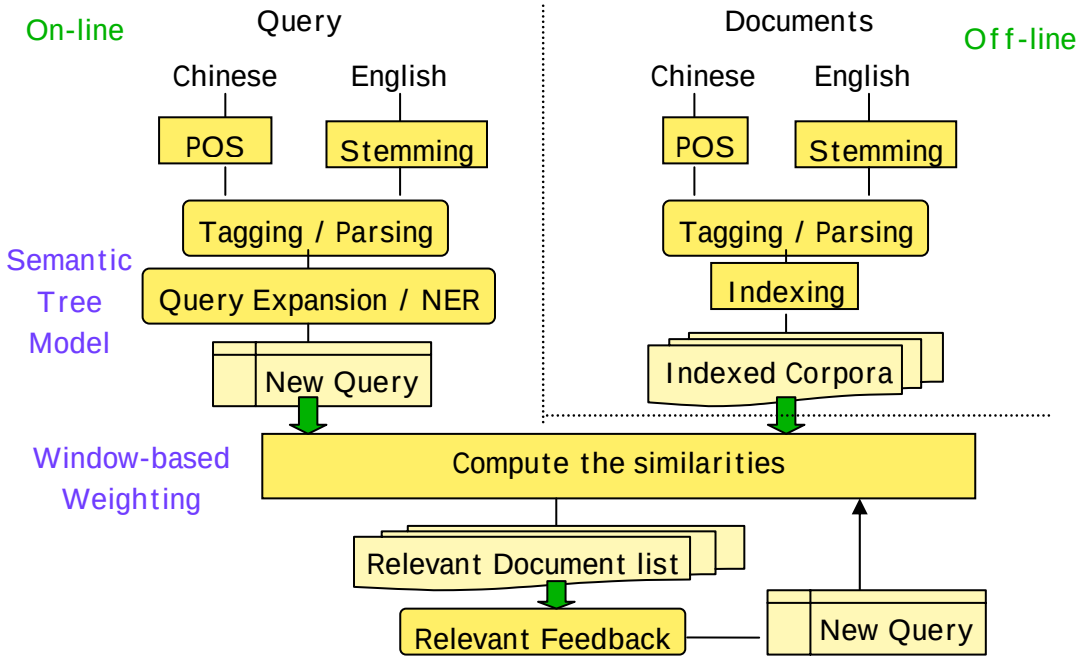
Qianli Jin, Jun Zhao, Bo Xu
National Laboratory of Pattern Recognition,
Institute of Automation, Chinese Academy of Science
Beijing, China 100080
{ qljin, jzhao, bxu }@nlpr.ia.ac.cn

It is the first time that the Chinese Information Processing group of NLPR participates in TREC. Our goal in this year is to test our IR system and get some experience about the TREC evaluation. So, we select two retrieval tasks: Novelty Track and Robust Track. We build a new IR system based on two key technologies: Window-based weighting method and Semantic Tree Model for query expansion. In this paper, the IR system and some new technologies are described first, and then some detail work and results in Novelty and Robust Track are listed.

1 IR System and new technologies

1.1 The Architecture of IR system

Our IR system is both for English Retrieval and Chinese Retrieval. Some maim parts of the system are shown in the following.



There are many modules in the system, such as POS, stemming, tagging, NER, Query Expansion and etc. Most of them are traditional and common, except that there are two new technologies: Window-based Weighting and Semantic Tree Model. In the following two parts, they are detail introduced.

1.2 Window-based Weighting

The key algorithm of an IR system is similarity computing between queries and documents. Till now, the most popular algorithm is the inner product of vectors, and the vectors can be built by using weighting technologies, such as binary weight, tf-idf, query expansion, relevant feedback and etc. In other words, most of the existing algorithms are based on vector computing. However, this method usually gets limited precision, because sometimes, a vector can not represent a query properly. For instance, if we have the following query and two documents:

Query: "**Can radio waves from radio towers or car phones affect brain cancer occurrence?**"

Document A: "John claimed his brain cancer was caused by the wave from his cellular phone. That claim, put forth in a lawsuit, has no basis in accepted scientific fact."

Document B: "I was listening to the radio, when the tower collapsed. I ran several blocks before my brain kicked in, and saw that another wave of people started running towards a police car."

We definitely know that Document A is relevant to the Query, while Document B is not. But if binary word vector model is used, the similarity value between Document B and the query is larger than the one between Document A and the query. We also try some other weighting technologies such as tf-idf, query expansion and etc., but find that in this case, based on vector computing, Document B seems more "relevant" to the query than Document A.

In order to solve this problem, we develop two key notions as follows:

1. *Query words appearing closely in the document provide more contributions to the similarity value than the ones appearing separately. The closer the query words in a document, the larger the similarity value between the query and the document.*

2. *Some query words, like named entities and baseNP are called "Core Words", while the other words are called "Surrounding Words". "Core Words" are much more important than "Surrounding Words", and should have special status in the retrieval processing (i.e. having larger weights).*

Based on the above two key notions, we developed three window-based models for the application of information retrieval. They are called "Simple Window-based Model", "Dynamic Window-based Model" and "Core-window-based Model", from the simplest model to the most complex one.

1.2.1 Model One: Simple Window-based Model

As our first key notion, the closer the query words in a document, the larger the similarity value between the query and the document. So, we introduce a window in the retrieval processing. When the query words co-occur in the window, a larger similarity weight is provided. First we put all the words of the document into the word

sequence orderly, like the Figure 1. Each sub-sequence with *d* continual words is included in a *d*-width window.

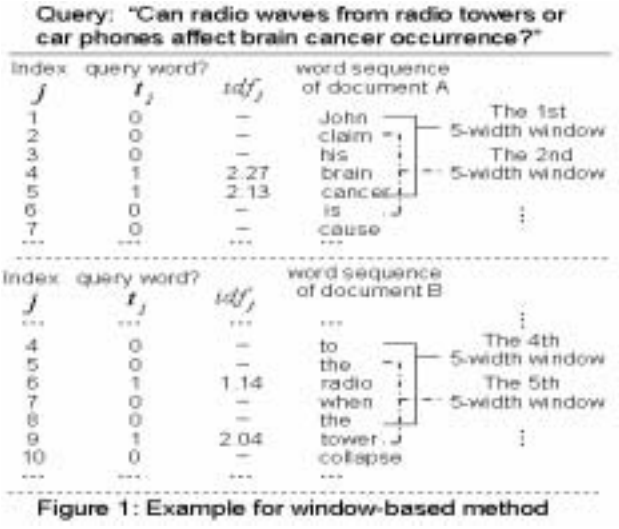


Figure 1: Example for window-based method

Let N denote the number of the words in the whole sequence, and *d* denote the width of the window. Then the similarity value ($Sim1(q,d)$) between query and document can be represented as follows:

$$Sim1(q,d) = \sum_{i=1}^{N-d} SWin(i, i+d) \qquad .....(1)$$

$$SWin(i,i+d) = [\sum_{j=i}^{i+d} t_j * idf_j] * [\sum_{j=i}^{i+d} t_j] .....(2)$$

where $SWin(i,i+d)$ denotes the similarity value between the query and the *d*-width window from the *i*th word to (*i*+*d*)th word in the whole sequence in the Simple Window-based Model. t_j is the binary signal of the *j*th word in the whole sequence. Here, t_j is equal to ONE if the *j*th word is a query word, otherwise, it is equal to ZERO. And idf_j is the inverse document frequency of the *j*th word in the sequence. The final similarity value between a query and a document is the sum of the similarity values of all the windows. The similarity value in a single window, represented as Formula 2, consists of two items. The first one $[\sum_{j=i}^{i+d} t_j * idf_j]$ is just like the traditional tf-idf method. And the second one $[\sum_{j=i}^{i+d} t_j]$ provides more weight, when more than one query word appeared in the corresponding window.

1.2.2 Model Two: Dynamic Window-based Model

In Simple Window-based Model, we give larger weight to the window, which includes more than one query word. But what is the distribution of these query words in the window? They can be separate or conjoint. If these query words in the window are conjoint, they maybe form a phrase. As we all know, phrases usually are less ambiguous than words. So, we should give the conjoint query words larger weight than the separate query words in the window. Another problem in Model One is that it is difficult to decide the width of window in real applications. A fixed window width cannot be suitable for all queries. In order to solve the above two problems, Model Two is proposed, which is called Dynamic Window-based Model. In the new Model, a dynamic window width called "TightWin" is developed to modify the original fixed window.

Define: TightWin is the smallest window width, which can overlay all the query words in the original window.

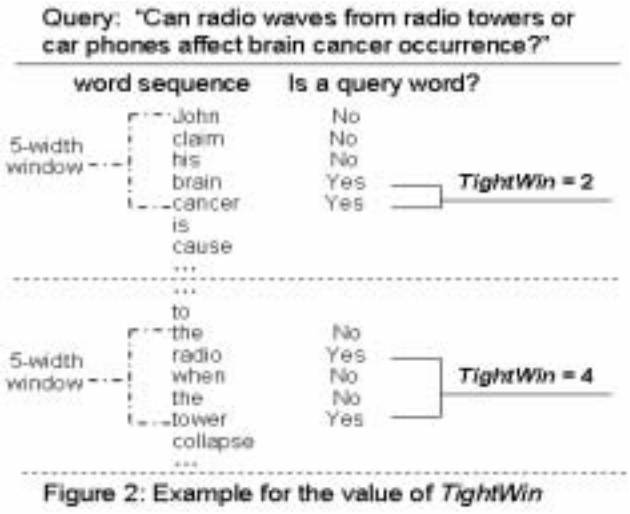


Figure 2: Example for the value of *TightWin*

In Figure 2, we give several examples about the value of TightWin. If the query words distribute separately in the original window, the value of TightWin is large. And if they are conjoint, the TightWin is small. So, we should give a large weight when TightWin is small.

Let N denote the number of the words in the whole sequence, and *d* denote the width of the window. Then the similarity values between query and document in Model Two ($Sim2(q,d)$) can be represented as follows:

$$Sim2(q,d) = \sum_{i=1}^{N-d} DWin(i, i+d) \qquad .....(3)$$

$$DWin(i,i+d) = SWin(i,i+d) * (\frac{[\sum_{j=i}^{i+d} t_j]}{TightWin})^p = [\sum_{j=i}^{i+d} t_j * idf_j] * [\sum_{j=i}^{i+d} t_j] * (\frac{[\sum_{j=i}^{i+d} t_j]}{TightWin})^p .....(4)$$

where $DWin(i,i+d)$ denotes the similarity value between the query and the *d*-width

window from the *i*th word to (*i*+*d*)th word in the whole sequence in Dynamic Window-based Model. t_j is the binary signal of the *j*th word in the whole sequence. Here, t_j is equal to ONE if the *j*th word is a query word, otherwise, it is equal to ZERO. And idf_j is the inverse document frequency of the *j*th word in the sequence. *TightWin* is defined above, and *p* is a parameter, which is larger than zero. Compared with Model One, Model Two has an additional item $(\frac{[\sum_{j=i}^{i+d} t_j]}{TightWin})^p$, which provides adjustment to the original fixed window. The conjoint query words provide more contributions to the final similarity value.

1.2.3 Model Three: Core Window-based Model

In the above two models, when query words appear closely in the document, they will be given larger weight. In some cases, it may bring some problems. Take a look at the above example query again.

> "**Can radio waves from radio towers or car phones affect brain cancer occurrence?**"

When the query words "radio waves" and "brain cancer" appear closely in a document, we can say that this document is most likely relevant to the query. But, when the query words "car phone" and "affect" appear closely in a document, we are not sure whether it is relevant. So, based our second key notion, we parse the query sentence and classify the query words into two groups. They are "Core Words" and "Surrounding Words" defined as follows.

Define:

(1) The query words, which represent the main meaning of the query, such as baseNP and Named Entities, are called "Core Words".
(2) The query words, which are not core words, are called "Surrounding Word".
(3) A window is called "Active Window", if and only if it includes Core Words.

Obviously, Core Words are much more important than Surrounding Word. So, Active Window should have larger weight than the common window.

Let N denote the number of the words in the whole sequence, and *d* denote the width of the window. Then the similarity value between query and document in Model Three ($Sim(q,d)$) can be represented as follows:

$$Sim3(q,d) = \sum_{i=1}^{N-d} CWin(i, i+d) \qquad .....(5)$$

$$CWin(i,i+d) = DWin(i,i+d) * [\sum_{j=i}^{i+d} t_j^*] = [\sum_{j=i}^{i+d} t_j * idf_j] * [\sum_{j=i}^{i+d} t_j] * (\frac{[\sum_{j=i}^{i+d} t_j]}{TightWin})^p * [\sum_{j=i}^{i+d} t_j^*]^m ...(6)$$

where $CWin(i,i+d)$ denotes the similarity value between the query and the *d*-width window from the *i*th word to (*i*+*d*)th word in the whole sequence in Core Window-based Model. t_j is the binary signal of the *j*th word in the whole sequence. Here, t_j is equal to ONE if the *j*th word is a query word, otherwise, it is equal to ZERO. t_j^* is another binary signal of the *j*th word in the whole sequence for Core Words. t_j^* is equal to ONE if the *j*th word is a Core Word, otherwise, it is equal to ZERO. And idf_j is the inverse document frequency of the *j*th word in the sequence. *TightWin* is defined in 1.2.2, and *m* is a parameter, which is larger than zero. Compared with Model Two, Model Three has an additional item $[\sum_{j=i}^{i+d} t_j^*]^m$, which focuses on the Core Words in the window. Only the active window has contributions to the final similarity value. The more the core words in the window, the larger the similarity value.

Detail evaluations of window-based weighting are included in the **Reference [1]**.

1.3 Semantic Tree Model for Query Expansion

The key problem of query expansion is to compute the similarities between terms and the original query. In other words, the original query can be regard as a point in the semantic space, and the goal of query expansion is to select some additional terms, which have the closest meaning to the point. So, as the first step, like most of the former methods, we need to compute the prior similarities between the terms. And we use Term Similarity Trees to represent and estimate the similarities between terms, which can cluster the terms according to their meaning. Then, we use the TSTM to expand queries.

1.3.1 Grow Term Similarity Trees based on prior similarities between terms

1.3.1.1 Build elements of Term Similarity Tree

Let $PSim(q, p)$ denote the prior similarity between the term *q* and *p*. For a given term *q*, use a tree to represent the sorted similarities in descending order between *q*

and all the other words, like the left part of the following Figure.

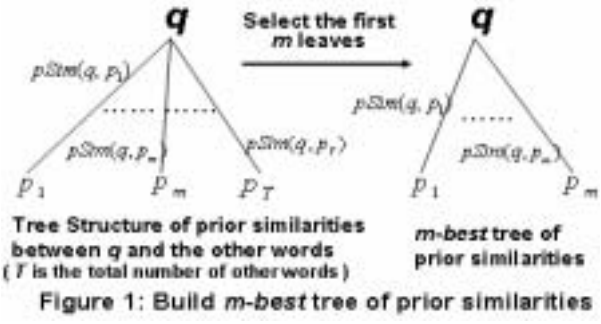


Figure 1: Build *m-best* tree of prior similarities

Note that the weight of the branch between q and p_m is $PSim(q, p_m)$, and:

$$PSim(q, p_1) \geq ... \geq PSim(q, p_m) \geq ... .$$

Then, we keep the first *m* leaves and discard the rest to build *m-best* tree of prior similarities, like the right part of the Figure. There are several methods for computing the prior similarities between terms. Here, we use normalized local co-occurrence algorithm to estimate them.

1.3.1.2 Grow Term Similarity Tree

Let $Q = (q_1, q_2, ..., q_i, ..., q_K)$ denote the original query including *K* terms, where q_i is the *i*th term in query. Then, we grow the Term Similarity Tree of query *Q* as $TSTM(Q, v, m)$, where *v* denotes the expanded level and *m* indicates that each element in term similarity tree is an *m-best* tree of prior similarities. The term similarity tree of query *Q* is shown in the following Figure.

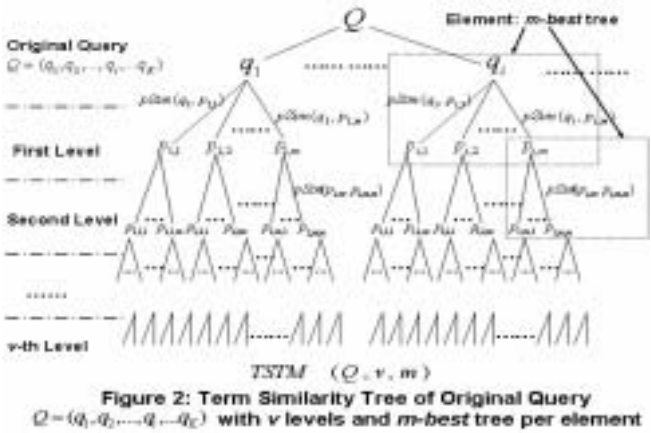


Figure 2: Term Similarity Tree of Original Query $Q = (q_1, q_2, ..., q_i, ..., q_K)$ with *v* levels and *m-best* tree per element

Each part framed by a quadrangle in the Figure denotes an element of TSTM, which is an *m-best* tree of prior similarities. Using the multi-level term similarity tree, we can easily compute the semantic similarity between two terms (one query term and one other term), no matter whether they co-occur in the training corpora. Note that each term in the query, like q_i, has its own sub-tree, whose root is the query-term itself. We define:

(a) A path between the query-term q_i and its leave-term *p* is the route from the root node q_i to the leave node *p*.
(b) The weight of a path between the query-term q_i and its leave-term *p* is the product of all the branch weights (prior similarity) on the path from q_i to *p*.
(c) The shortest path between the query-term q_i and its leave-term *p* is the path between q_i and *p*, which has the largest weight.
(d) The similarity between the query-term q_i and its leave-term *p* is defined as the weight of their shortest path.

For instance, we can compute similarity between two terms q_1 and $p_{1,i,j}$ as:

$$\begin{aligned} &Sim\ (q_1, p_{1,i,j}) = weight\ -of\ -shortest\ -path\ (q_1, p_{1,i,j}) \\ &= PSim\ (q_1, p_{1,i}) \times PSim\ (p_{1,i}, p_{1,i,j}) ..... \quad (7) \end{aligned}$$

1.3.2 Query Expansion based on TSTM

Let $Q = (q_1, q_2, ..., q_i, ..., q_K)$ denote the original query including *K* terms, where q_i is the *i*th term in query. The $TSTM\ (Q, v, m)$ is the term similarity tree of query *Q*.

The term *w* is expanded to query *Q*, when *w* satisfies two conditions illustrated as the following two formulae:

$$\begin{cases} Sim\ (Q, w) = \sum_{i=1}^{K} Sim\ (q_i, w) \geq cv \\ Overlay\ (TSTM\ (Q, v, m), w) \geq percent \times K \end{cases} \quad ..... \quad (8)$$

where $Sim(Q, w)$ is the similarity between the query *Q* and the term *w*. $Sim(q_i, w)$ is the similarity between the term q_i and *w*, which can be estimated as formula (7). And *cv* is the threshold value of similarity.

$Overlay(TSTM(Q, v, m), w)$ denotes the occurrence times of term *w* in the sub-trees of $TSTM(Q, v, m)$. It means that in the total *K* sub-trees of *Q*'s term similarity tree, how many sub-trees include (overlay) the term *w*. *percent* is the threshold value of overlay degree.

The first discrimination function in formula (8) is used to estimate the similarities between the term *w* and the query terms $q_1, q_2, ..., q_i, ..., q_K$ respectively. And the second discrimination function in formula (2) is used to estimate the similarity between the term *w* and the whole query *Q*. The query Q can be regard as a point in

the semantic space, so we need to know whether the term *w* is close in meaning to this point, not just close to the independent meaning of each q_i.

Detail evaluations of Semantic Tree Model for query expansion are included in the **Reference [2]**.

2 Novelty Track

The Novelty Track is designed to investigate systems' abilities to locate relevant and new information within a set of documents relevant to a TREC topic. The goal of the track is to find out relevant/new sentences, instead of documents.

2.1 Relevant

Considering sentences have few words than documents, query expansion is much more important. So, we use the following process to deal with relevant retrieval:

A) Two Stages Query Expansion. In the first stage, we use Term Similarity Model to expand queries. In the second stage, we use Relevant Feedback to modify and expand queries again to improve the retrieval result. Usually, top 20% sentences are used for relevant feedback, after the first retrieval.

B) We use two different methods to compute similarities between queries and sentences. The first is the traditional tf-idf method. And the second is window-based method to ensure that the closer the query words in sentences, the higher the similarity value. Actually, this method is the expansion of N-gram model (because window-based method does not require the query words appearing directly continual).

C) We use an existing method called 'pivoted document length normalization' to normalize sentence length. (see the **Reference [3]**)

D) The similarity values of different topics are usually different. The main reason is that queries have different length and query words have different characteristics (i.e. idf). So, it's unreasonable to use a simple and fixed threshold for every topic. Here, we developed one dynamic threshold for one topic, based on the probabilistic characteristics of similarity values between this topic and sentences.

2.2 Novelty (new)

Having relevant sentences, we have another task to filter out repeated information. We define a value called 'New Information Degree'(NID) to present whether a sentence includes new information related to the former sentences. If the value of NID is big, this sentence is reserved, or it will be discarded. There are two different ways to

define NID of the latter sentence related to the former sentence.

$$NID\_1 = 1 - \frac{\text{Sum the 'idf' value of words appeared in both sentences}}{\text{Sum the 'idf' value of words appeared in latter sentences}}$$

$$NID\_2 = 1 - \frac{\text{The number of matched bi-gram word sequences}}{\text{Total number of bi-gram word sequences in latter sentences}}$$

Usually, if the value of NID is bigger than 10%-20%, the latter sentence will be considered useful (including new information).

2.3 Official Results

Dyn: Dynamic Threshold
Sta: Static Threshold
Win: Core Window-based weighting method
RF: Relevant Feedback
Leng: Length Normalization
Tf-idf: tf-idf weighting
NID_1 / NID_2: defined above in 2.2
QE: Query Expansion

Task 1 Table

ID TAG	Algorithms	Relevant Results Average F Measure	Novelty Results Average F Measure
NLPR03n1w1	Dyn-Win-RF	0.510	0.425
NLPR03n1f1	Tf-idf-leng-RF	0.477	0.399
NLPR03n1f2	Tf-idf-leng-RF	0.407	0.349
NLPR03n1w2	Dyn-Win-RF	0.391	0.325
NLPR03n1w3	Dyn-Win-RF	0.330	0.279

Task 2 Table

ID TAG	Algorithms	Average F Measure
NLPR03n2d1	NID_1, Dyn	0.807
NLPR03n2s1	NID_1, Sta	0.819
NLPR03n2d2	NID_2, Dyn	0.808
NLPR03n2s2	NID_2, Sta	0.817
NLPR03n2d3	NID_1+2, Dyn	0.803

Task 3 Table

ID TAG	Algorithms	**Relevant** Average F	**Novelty** Average F
NLPR03n3d1	RF, Win, leng, NID_2, Dyn	0.687	0.518
NLPR03n3s1	RF,Win, leng, NID_1, Sta	0.677	0.532

NLPR03n3d3	RF,Win, leng, NID_1, Dyn	0.674	0.509
NLPR03n3d2	QE,RF,tf-idf, leng, NID_2,Dyn	0.618	0.472
NLPR03n3s2	QE,RF, tf-idf, leng, NID_1,Sta	0.624	0.489

Task 4 Table

ID TAG	Algorithms	Average F Measure
NLPR03n4d1	NID_1,Dyn	0.775
NLPR03n4s1	NID_1, Sta	0.789
NLPR03n4s2	NID_1+2, Sta	0.794
NLPR03n4s3	NID_2, Sta	0.796
NLPR03n4d2	NID_2. Dyn	0.773

Form the above results, we find that the technologies of window-based weighting method and relevant feedback are useful. In task 2,3,4, we all get very big values of average F measure, but in task 1, it is small. The reason is that in task 1, we use TREC Novelty 2002 data as our training corpora, which have quite few relevant sentences. However, the data for 2003 have many relevant sentences (even more than 50% for some topics), so we should use small thresholds. The big thresholds from the training corpora (2002 data) make good precision and poor recall (also poor F measure). In task 2-4, we use a part of the 2003 data as the training corpora and get better F values.

3 Robust Track

As a mature IR system, the robustness is quite important. The goal of the Robust Track is to improve the consistency of retrieval technology by focusing on poorly performing topics. So, in this track, we improve our system based on the following two points:

1) Considering that even the worst topic should have an acceptable result, a robust algorithm of similarity should be produced to improve precision for each topic.
2) Though at most 1000 documents will be accepted by NIST per topic, we suppose that most users only look through the former part of retrieval result (Maybe 2 or 3 screens). So, for each run, we submit several dozens of retrieved documents. We should make sure that the true relevant documents have the top similarity values.

3.1 Processing

In order to improve the robustness described above, we use a combined algorithm, including four technologies. The processing of our system is as follows:

A) Query expansion based on Term Similarity Trees. It is described in the first part of this paper and also in the Reference [1].
B) Window-based weighting methods for computing similarities. It is described in the first part of this paper and also in the Reference [2].
C) Length normalization (see the Reference [3])

D) Dynamic Threshold. The similarity values of different topics are usually different. The main reason is that queries have different length and query words have different characteristics (i.e. df). So, it's unreasonable to use a simple and fixed threshold for every topic. Here, we developed one dynamic threshold for one topic, based on the probabilistic characteristics of similarity values between this topic and documents.

3.2 Official Results

ID Tag	Algorithms	Retrieved documents	Average Precision (non-interpolated)	Number of topics with no relevant in top 10
NLPR03vb25	Dynamic Window-based weighting	25	0.1516	7%
NLPR03vb10	Simple Window-based weighting	10	0.1055	7%
NLPR03w16	Core Window-based weighting	16	0.1153	10%
NLPR03vb50	Dynamic Window-based weighting	50	0.1770	7%
NLPR03w49	Core Window-based weighting	49	0.2434	10%

From the above table, we can see that we get a low value of average precision. The reason is that for each topic, only several dozens of retrieved documents are returned, not 1000. We suppose that most users only look through the former part of retrieval result and a robust IR system should ensure that users can find relevant documents at the top 10 or 20. Based on the experiments, the window-based methods, especially core window based method, outperform most traditional tf-idf weighting method. Detail evaluation and discussion are given in the paper of reference [1].

Reference

[1] Qianli Jin, Jun Zhao, Bo Xu, *Window-based Method for Information Retrieval*, 2004, The First International Joint Conference on Natural Language Processing

[2] Qianli Jin, Jun Zhao, Bo Xu, *Query Expansion based on Term Similarity Tree*, 2003 International Conference on Natural Language Processing and Knowledge Engineering, IEEE.

[3] Amit Singhal, Chris Buckley, Mandar Mitra, *Pivoted Document Length Normalization*, SIGIR 1996.