Meiji University Web, Novelty and Genomics Track Experiments

Tomoe Tomiyama, Kosuke Karoji, Takeshi Kondo, Yuichi Kakuta
and Tomohiro Takagi
Department of Computer Science, Meiji University
{tomiyama, karoji, t-kondo, kakuta, takagi}@cs.meiji.ac.jp

Akiko Aizawa
National Institute of Informatics
akiko@nii.ac.jp

Teruhito Kanazawa
KYA group Corp.
tkana@kyagroup.com

1 Introduction

We participated in Novelty track, the topic distillation task of Web track and ad hoc task of Genomic Track. Our main challenge is to deal with meaning of words and improve retrieval performance.

2 Conceptual Fuzzy Sets

To make computers understand a language, the most difficult problem is the ambiguity of the language. In Information Retrieval, a presence of multisense words causes retrieval performance decrement. According to the theory of *meaning representation of use*, proposed by Wittgenstein, the various meanings of a word can be represented by other words, and when a decentralized knowledge representation is formed, the meaning of the word is clear.

For instance, "typhoon" can be expressed as "violent tropical storm" or "tropical cyclone". "cyclone" can in turn be expressed by other words. In summary, the concept of the word "typhoon" can be expressed by other words ("violent", "tropical", and, "storm").

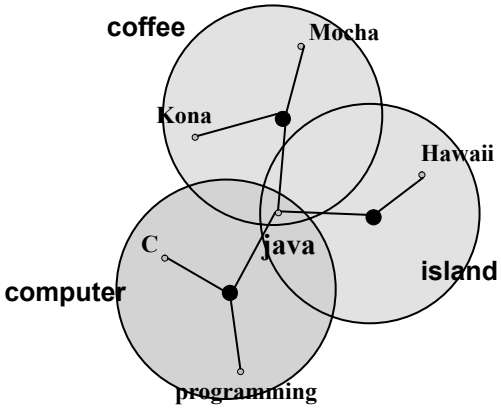
Conceptual Fuzzy Sets (CFSs) has been proposed to deal with the ambiguity problem [i] [ii]. In CFSs, to represent of meaning of words, prototype concepts are represented by the activity values of words, and CFSs use overlap of the activity values of the prototype concepts. The concept of an arbitrary input is represented by overlapping the distributions of the activity values of prototype concepts.

2.1 Example of CFSs

Let us consider a concrete example of a CFS in Figure 2.1. The noun "java" has three senses: "coffee", "island", and "programming language". However, if "C" and "Java" appear together in the same document, the context is recognized as "computer". Therefore, "Java" probably means "programming". In the same way, if the context is "island", "Java" probably means "island". If the context is "coffee", "java" probably means "coffee". The words related to the context, such as "Mocha", "Kona", and "coffee" had high activity values. Thus, if the context can be specified by words, the ambiguity of the meanings of words can be eliminated.

Input

Mocha	0
Kona	0
coffee	1
java	1
island	0
C	0
programming	0
computer	0
Hawaii	0



Output

Mocha	0.59
Kona	0.53
Coffee	0.84
java	0.63
Island	0.26
C	0.18
Programming	0.11
Computer	0.13
Hawaii	0.22

Figure 2.1: Example of CFSs.

3 Web Track

We submitted five runs for the mixed query task and one manual run for the query classification task. We had two main challenges, which were the query expansion using CFS and the document modification by RS model. The RS model is described in 3.2, and CFS is described in 3.3.

3.1 Search Engine System Overview

R^2D^2 is designed as a full-text retrieval system based on the vector space model[iii].

Formal definition of the vector space model is the following. The query $\mathbf{Q}$ consists of searching terms $\{q_1, q_2, \cdots, q_m\}$. The similarity between the query $\mathbf{Q}$ and the document d_j is defined as:

$$S(\mathbf{Q}, \mathrm{d}_j) = \sum_{i=1}^{m} w(i, j),$$

$$w(i, j) \equiv f_T(j, i) \cdot f_D(i).$$

f_T: factor based on the term frequency in a document.

f_D: factor based on the document frequency containing the term.

In R^2D^2, f_T and f_D are calculated using the RS model described in the following section.

3.2 The RS model

We have proposed a method named the Relevance-based Superimposition (RS) model to solve the semantic ambiguity problem in information retrieval. A query usually provides only a very restricted means to represent the user's intention. Query expansion is a method for semantic disambiguation on query issuing phase. It includes index terms related to the original query expression, thus assisting novice users who have limited vocabulary in the target field. However, it is difficult to choose terms that represent the user's intention automatically and carefully. Therefore, pragmatically effective retrieval can only be achieved by adjusting many parameters depending on the database[iv]. Document feature vector modification is one of the methods that use information extracted from the documents for semantic disambiguation in index generation phase. We believe it achieves higher recall without losing precision of retrieval, because documents usually have much more information than a query.

The RS model is designed using the document feature vector modification approach This model partitions the documents so that the relevant documents dealing with the same topic fall into the same cluster. However, the idea is different from the traditional cluster-based methods[v,vi] in which the document clusters are usually mutually exclusive. These methods assume that documents can be classified into orthogonal topics; however, it is natural to assume that a document can belong to several topics. This difference in assumptions will reflect on the retrieval. The details of the RS model has already been reported in [vii] and [viii]. We have evaluated the effectiveness of this model using TREC San Jose Mercury consisting of news articles and NTCIR 1/2 test set consisting of scientific papers. The experimental results showed that the RS model improves the average precisions by 7%, which can be considered significant (5–10% is generally required for significant improvement[ix]).

3.2.1 Model overview

The RS model is designed using the document feature vector modification approach, as described in Figure 3.1. This model partitions the documents so that the relevant documents dealing with the same topic fall into the same cluster. However, the idea is different from the traditional cluster-based methods in which the document clusters are usually mutually exclusive. These methods assume that documents can be classified into orthogonal topics; however, it is natural to assume that a document can belong to several topics. This difference in assumptions will reflect on the retrieval.

Let us define the RS model formally. In the RS model, each document is represented by a feature vector. Term frequencies are often used as the features. Suppose that a document database contains a set of documents $\{\mathsf{d}_1, \mathsf{d}_2, \cdots, \mathsf{d}_n\}$ and their feature vectors are $d_1, d_2, \cdots, d_n$.

In the RS model, documents in the database form clusters $C_1, C_2, \cdots, C_m$ which represent topics.

Note that a document may be contained in more than one cluster in the RS model, whereas clusters in other methods are often mutually exclusive. At this point, we must decide what type of relevance we will use to make clusters. The principle of the RS model is independent of the source

of relevance information, and our choice will depend on the type of database and the types of elements in it. For instance, the following elements included in the database can be candidate sources for relevance information and used for document clustering:

- keywords given by the authors or automatically extracted;
- references, hyperlinks;
- bibliographic information, such as author name, publication date, and journal title.

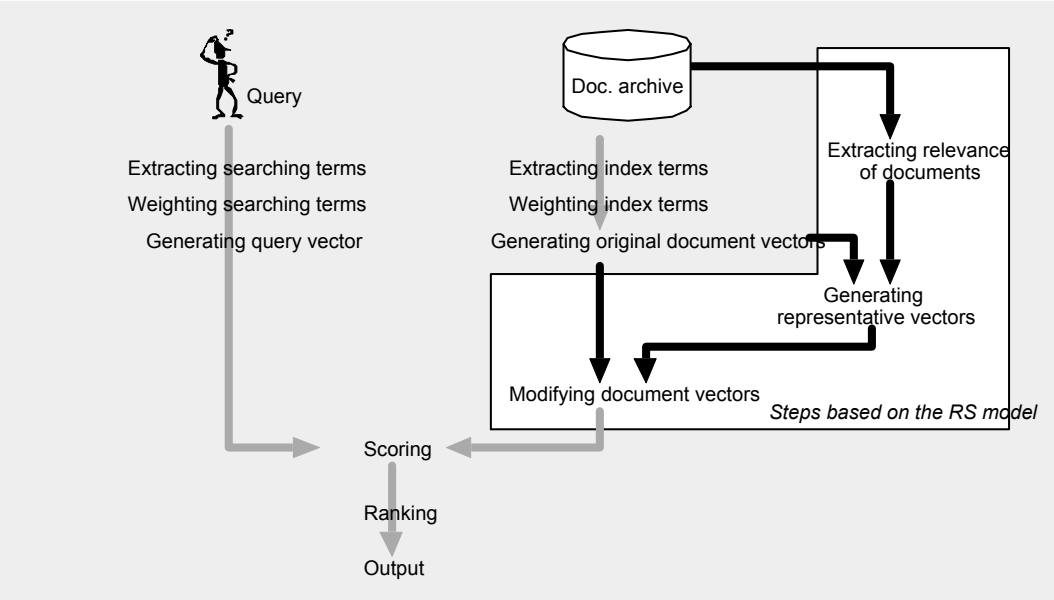


Figure 3.1: The process flow of the RS model approach.

3.2.2 Representative vector generation

When clusters representing topics are given, the document feature vector is modified in two steps: (1) representative vector (RV) generation for each cluster, and then, (2) feature vector modification by RVs. We can design a statistical method so that the RV can be considered to accurately represent overall characteristics of the documents that belong to the same cluster. Next, the modification method should properly perform the superimposition of features represented by RVs so that the topics of each cluster are reflected in the modified document feature vectors, thereby reducing the ambiguity of retrieval caused by expressional mismatches between the query and the documents.

RV r of cluster C is constructed from the feature vectors of the documents in C. Currently, we have tested five types of representative-vector-generator (RVG) functions, derived from the α-family distributions[x], where the i th component r_i of RV r is defined as follows:

$$r_i \equiv \left(\frac{1}{|C|} \sum_{d \in C} d_i^{\frac{1-\alpha_1}{2}} \right)^{\frac{2}{1-\alpha_1}}.$$

here d_i denotes the i th component of the feature vector of document d and $|C|$ denotes the number of documents contained in cluster C. The dimension of RV is equal to that of document feature vectors. We empirically evaluated an appropriate value for α_1 and identified α_1=-3 as the most effective

parameter[xi]. When α_1=-3, the variation of the α-family distributions is called 'root-meansquare':

$$r_i \equiv \sqrt{\frac{1}{|C|}\sum_{d\in C} d_i^{\,2}} \ .$$

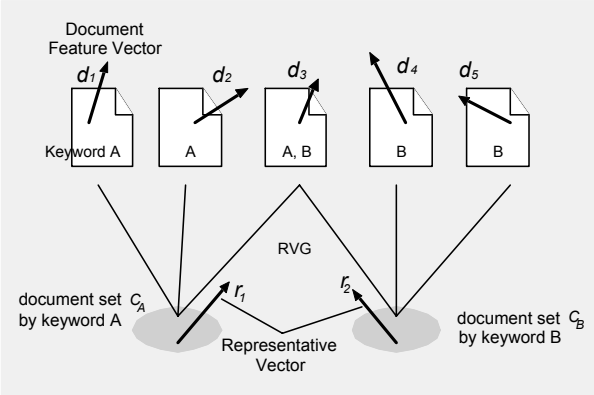


Figure 3.2: Representative vector generation.

3.2.3 Document feature vector modification

The second step is modification of the document feature vector using the RVs of the clusters to which the document belongs.

We assume that important index terms for a document d are any terms that occur frequently in any cluster to which d belongs, as well as other terms occurring frequently in d itself. This characteristic is considered to be 'conjunctive'. We now propose the document-feature-vector-modifier (DVM) function. Let $\mathsf{D(d)}$ be the set of clusters to which the document d belongs. Let $\mathsf{S(d)}$ denote the set of RVs that belong to the clusters in $\mathsf{D(d)}$. Then the i th component s_i of the vector of D (d) can be defined as follows:

$$s_i \equiv \left(\frac{1}{|S(\mathsf{d})|} \sum_{\mathbf{r}\in S(\mathsf{d})} r_i^{\frac{1-\alpha_2}{2}} \right)^{\frac{2}{1-\alpha_2}}$$

Let $(d_1, d_2, \cdots, d_I)$ represent the feature vector of document d and let $(S_1, S_2, \cdots, S_I)$ represent the vector of the cluster set D (d). The modified document feature vector d' is then defined as $(f_s(d_1, s_1), f_s(d_2, s_2), \cdots, f_s(d_I, s_I))$, where f_s is the superimposing function defined as:

$$f_s(x, y) \equiv \left\{ \frac{1}{2} \left(x^{\frac{1-\alpha_3}{2}} + y^{\frac{1-\alpha_3}{2}} \right) \right\}^{\frac{2}{1-\alpha_3}} .$$

We have evaluated some members of the α-family distributions for s and f_s, and identified

α_2=-3 (root-mean-square) and α_3= -∞(maximum) as the most effective parameters, respectively [xi]. We used α_1=-3 for r , α_2=-3 for S and α_3= -∞ for f_s in the following evaluation.

3.2.4 Automatic Keyword Extraction

In the previous section, we described how to make document clusters using the well-chosen keywords given by the authors of the documents. However, we must also consider archives where no explicit keywords are given for clustering. There are two possible answers: one is automatic unsupervised keyword extraction and the other is to find another clue of relevance. We investigated the former approach in the evaluation. Details are described in [xii].

3.3 Query expansion using Conceptual Fuzzy Sets (CFS)

We used CFS to expand queries. When users want to search for something, their requests are not always clear. When users input query into search engines, they are imaging a concept related to the query terms. Traditional search engines do not retrieve based on concepts, but based on input query terms, so such systems may return results that are different from users' concepts. To solve this problem, our system represented users' queries as concepts. In CFS, the concepts are represented by some words and their degree of relationship. To construct CFS, we need a dictionary in which the concepts are represented by some words and their degree of relationship.

3.3.1 Dictionary

A dictionary contains knowledge that is necessary to achieve CFS. It is defined as a set of prototype vectors. A prototype vector is represented by words and their weights. We automatically built a dictionary from HTML documents in the .GOV. To build the dictionary, we classified the documents using RS Model as described in the section 3.2. A generated document cluster can be considered to have meanings. The centered vector of each cluster was used as the prototype vector. As a result, we got a dictionary consisting of 73,827 prototype vectors. We used about the top 150 words in each centered vector.

3.3.2 Query Expansion

We used the prototype vectors to expand input queries. Each prototype vector was activated depending on users' queries. Prototype vectors that were related to the query were activated strongly and prototype vectors that had none of the query terms were not activated. Overlapping activated prototype vectors generated new concepts, and the queries were represented by them. The expanded query vector is computed by summing weighted concept vectors:

$$S_i = \frac{V_q \cdot V_{C_i}}{\left|V_q\right|\left|V_{C_i}\right|} \cdot w_i$$

$$V_q{}' = \sum_i S_i \cdot V_{C_i}$$

where S_i is the activation value of the prototype vector C_i , V_q is the word vector that every query word's value is 1.0, V_{ci} is the word vector that represents the prototype vector C_i, and w_i is the weight based on the number of query terms that is contained in the prototype vector.

3.4 Experiments

We used document structures and out-degree reranking for all runs. The description of each run is shown in Table 3.1.

Table 3.1: The description of our runs.

Run ID	Query Classification	Document modification	Query Expansion
meijihil1			
meijihil2	O		
meijihil3	O	O	
meijihil4	O		O
meijihil5	O	O	O

3.4.1 Indexing

We used modified document vectors by RS model in *meijihil3, meijihil5*. In the other runs, we used ordinary tf-idf method.

3.4.2 Document scoring

First, our system classified queries into four types. If a query was the TD type, the query would be expanded by CFS. To score a document, we used content of the document, its structures, and link information. Summary of our searching procedure is shown in Figure 3.3.

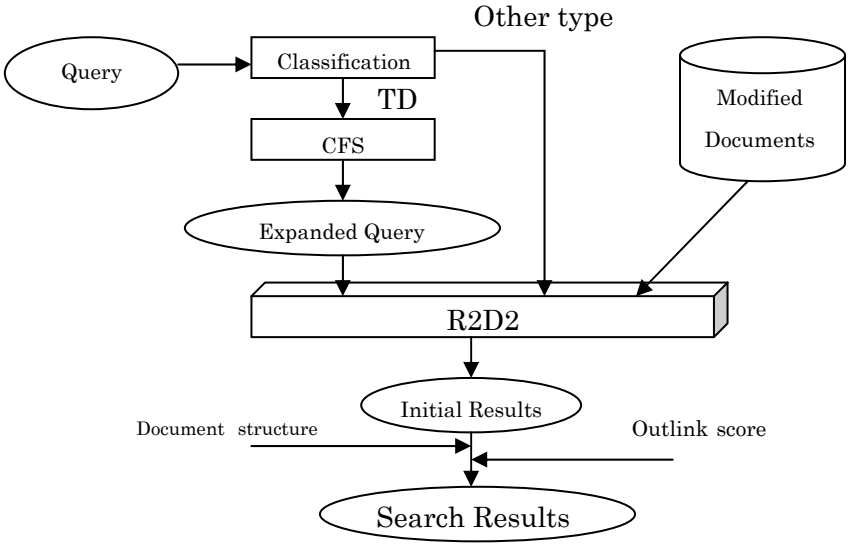


Figure 3.3: Our searching procedure.

3.4.3 Query classification

We classified queries into four types that are the topic distillation (TD) type, the named page (NP) type, the home page (HP) type and the unknown (UK) type in order to apply CFS for only TD type queries. One reason was that CFS was effective to only TD type queries on the experiments using TREC2003 data. Moreover, when users are searching for known items, the relevant documents are thought to include the query terms. Thus, we assumed that such queries would not need to be expanded. Consequently, we classified the queries into each type.

Our query classification procedure is as follows. First, we classified query into NP type, HP type and other type. This was based on appearing in the home page finding queries and the named page finding queries from last year's queries. Then, we classified other type query into TD type and UK type. The method was based on template-based phrase list, and used verbs and abbreviations list. As a result of classifying this year's query, TD type queries were 103, HP type queries were 32, NP type queries were 8 and UK type queries were 82.

3.4.4 Using document structure

We used three types of information in order to calculate the document structure score. These were anchor text of incoming links, URL length of the document, and title text of the document. The document structure score were added to the document-query similarity:

$$S_i = (Similarity(Query, D_i) + \alpha S_{ianchor}) \cdot \beta S_{iurl} + \gamma S_{ititle},$$

where S_i is the initial document score of D_i, $S_{ianchor}$ is the anchor text score of D_i and S_{iurl} is the URL length score of D_i, S_{ititle} is the title text score of D_i , α is the weight for $S_{ianchor}$, β is the weight for S_{iurl} , γ is the weight for $S_{ititle.}$ We empirically determined α, β and γ based on the data on TREC2003. The parameters for each document structure were decided according to each type as shown in Table 3.2.

Table 3.2: The parameter for score of each structure.

	Anchor text	url length	Title
TD	0.6	0	0
HP	0.5	0.8	0.5
NP	1.0	0	0.4
UK	0.5	0.2	0.8

3.4.5 Outlink score

To give the higher score homepage of the site, the system added outlink score to initial document score. The outlink score is based on the assumption that relevant homepages have links to many relevant pages. We defined it as follows:

$$S_i' = S_i + \sum_j^{N_i} S_j \left(\frac{\log N_i}{N_i}\right) \quad ,$$

where S_j is the score of initial search result in *document j* and N_i is the number of outlinks in *document j*. In our experiment on the TREC2003 Topic Distillation Task, this method was 36% more precise than baseline. However, on the TREC2003 NP/HP finding task, MAP did not improve much.

3.5 Result and discussion

Our official results are shown in Table 3.3. In the mixed query task, the best result was *meijihil1*. Our query classification method was failed. This method based on last year's data did not fit to this year's. 86 of 225 queries correctly classified.

Table 3.3: Official results.

Run ID	All topics (Normalized result)	Success @ 1	Success @ 10	TD/MAP	NP/MRR	HP/MRR
meijihil1	0.69	0.3644	0.7378	0.1099	0.6111	0.4728
meijihil2	0.68	0.3556	0.7156	0.1101	0.5841	0.4585
meijihil3	0.69	0.3689	0.7111	0.1148	0.5913	0.4568
meijihil4	0.56	0.3156	0.5600	0.0949	0.4339	0.4165
meijihil5	0.54	0.2933	0.5867	0.0712	0.4726	0.4258

From comparison *meijihil2* and meijihil3, we found that the RS Model improved the Suc@1 score, while it got the Suc@10 score worse. In traditional Information Retrieval, like TREC ad-hoc task and NTCIR, the RS Model made approximately opposite effects, which means improving recall precision in top ranking without decreasing relevant precision. We suppose the RS Model could retrieve some relevant documents that could not be retrieved by the general IR Model however it did not be adapted to this task efficiently.

3.6 Additional experiment

As shown in official results, the query expansion using CFS method did not performed well. In our experiments, we could expand only 78% of 75 TD queries, so we did not find the effect of CFS. Therefore, to expand all TD queries, we tested to expand all of 225 queries. The result was shown in the last line in Table 3.4.

Table 3.4: Additional experiment.

runID	TD/MAP	HP/MRR	NP/MRR
meijihil1	0.1099	0.6111	0.4728
cfs(additional)	0.1115	0.3530	0.2859

We found that CFS had bad influences in HP and NP type queries, while it slightly improved MAP in TD queries. Our approaches were effective to the topic distillation task on TREC2003, however the approaches did not perform well on this year's data. We assumed that there were many difficult queries in this year for CFS. For example, the query of "information security", "the arts in education" and so on could not expand well. Our future project is to build a more robust dictionary that can deal with such query terms.

4. NOVELTY TRACK

Our main challenge in Novelty Track is concept-based expansion of words and sentence using the CFSs. We thought that the concept-based expansion of words and sentence can represent meanings of the words and the sentence more exactly.

4.1. Concept-based Expansion

We used an RBF network to mount CFSs on the computer. Our CFSs, based on the RBF network, had the structure shown in Figure 4.1.

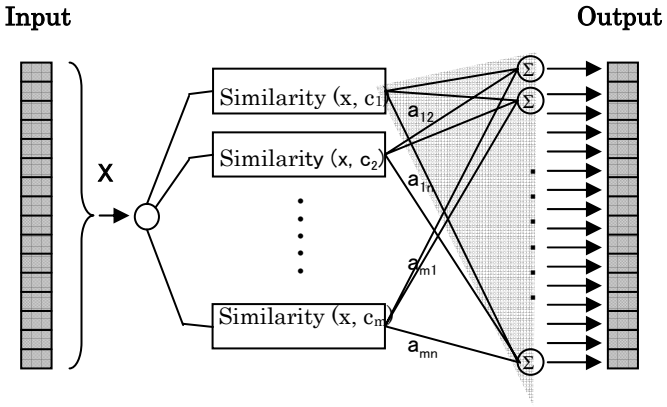


Figure 4.1: Structure of CFSs based on the RBF network.

There are the two required elements for the query expansion model using the CFSs.

(1) Prototype concepts, $\mathbf{c_i}$

(2) The degrees of relationships between each prototype concept and each word, a_{ij}

We approached the construction of $\mathbf{c_i}$ by clustering the documents in a corpus. The corpus is Reuters corpus[xiii] or Open Directory Project (ODP)[xiv]. Then, the degrees of relationship (weights) a_{ij} were learned using the least square minimum method. Supervisor data was needed to learn. It was made by using topics and tagged documents in the Reuters corpus. Each document was tagged to topic it belonged to, and the supervisor vector of each topic was made by using the tags. We used transposed matrix of $\mathbf{c}$ as the weights of the CFSs using the ODP. An arbitrary input, $\mathbf{x}$ was expanded as follows:

1. Calculate the similarities S_i, between input vector, $\mathbf{x}$ and each prototype concept, $\mathbf{c_i}$

$$S_i = \frac{\mathbf{x} \cdot \mathbf{c_i}}{|\mathbf{x}||\mathbf{c_i}|} \quad .$$

2. After expanding an arbitrary input, $\mathbf{x}$, using CFSs, the activity value, $f_j(\mathbf{x})$ of word, w_j is calculated by

$$f_j(\mathbf{x}) = \sum_i a_{ij} S_i \ .$$

4.2. Relevant Sentences Retrieval

4.2.1. Relevant Detection

We treated relevant detection as binary classification problem. Each topic which is made with the topic title, description, and narrative, was expanded using CFSs. We calculated Term-Frequency (TF) and Document-Frequency (DF) considering the sentence as a very short document. When classifying the sentences as relevant or non-relevant, a similarity between the topic and the sentences was calculated. The similarity is calculated by using vector space model (VSM). If the similarity exceeded a certain threshold, the sentence was classified as relevant.

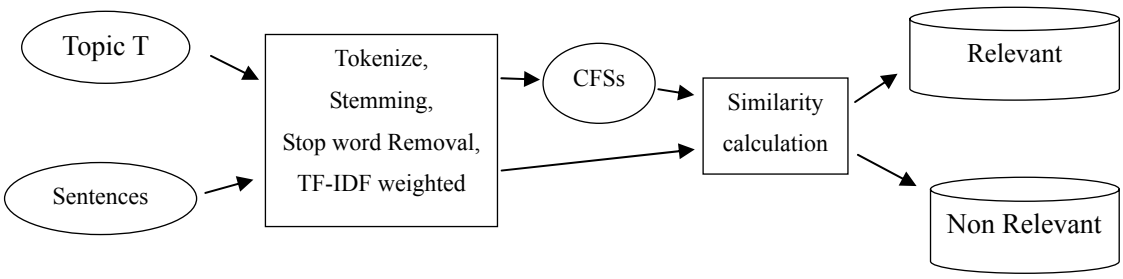


Figure 4.1: Architecture of Relevant Detection System.

4.2.2. Threshold Learning

In this system, we must set an appropriate threshold to distinguish relevant sentences from non-relevant ones. In Task1, The threshold was learned using the TREC2002 Novelty Track corpus. In Task3, The threshold of each topic was learned using the relevant sentences in the first 5 documents which were provided. If the first 5 documents include some non-relevant documents, the supervisor data decreases. Therefore, the supervisor data of all topics was used in the topic without (or small number) the supervisor data.

4.3. New Sentences Retrieval

We used three criteria in order to detect new sentences.

4.3.1. Sentence Weight Score

A new sentence which would be presented to a user should be itself important sentence in text. In order to detect new in news stream, to consider locally is more effective rather than to consider globally. Therefore we used sentence weight proposed by Zechner [xv], we improved it. The score can be calculated using *N-window-idf* which is document (sentence) frequency in past N sentences. By this means, weights of frequent words decrease; sentence weights represent local importance.

Therefore we represented Sentence Weight Score and *N-window-idf(t)* as follows:

$$N-window-idf(t)=\log\frac{N}{N-window-df(t)}.$$

$$SentenceWeightScore(s)=\frac{1}{Length(s)}\times\sum_i tf(t_i)\times N-window-idf(t_i).$$

where *Length(s)* is the length of a sentence. The score is canonicalized by the length. $tf(t_i)$ is the frequency of a word, t_i in the sentences. N is the window size. *N-window-df(t)* is the sentence frequency of a word, t_i in past N sentences.

4.3.2. Scarcity Score

An appearance of the word which has not appeared before may contribute to the novelty. So we used term frequency in past N sentences. The score is as follows:

$$ScarcityScore(s)=\frac{1}{Lenght(s)}\times\sum_i\frac{1}{N-window-tf(t_i)}.$$

where, *N-window-tf(t)* is the term frequency of the word, t_i in past N sentences.

4.3.3. Redundancy Score

A new sentence probably differs from the sentences judged to be new in the past. Concept-based expansion using the CFS is available. Therefore the formula is as follows:

$$RedundancyScore(s)=\max_{NovSi\in NoveltySentece} Similarity(NovSi,s).$$

where we used the cosine similarity as *Similarity*.

4.3.4. New Detection

We used the Support Vector Machine (SVM) for classification of new or non-new. We used the SvmLight program [xvi] to train the SVM models. In Task 1, 2, and, 3, the SVM model was trained using the TREC2003 Novelty corpus. In Task4 the SVM model was trained using the TREC2004 corpus. The time window N was experientially set to 200. The Novelty Detection system we constructed is shown in Figure 4.2.

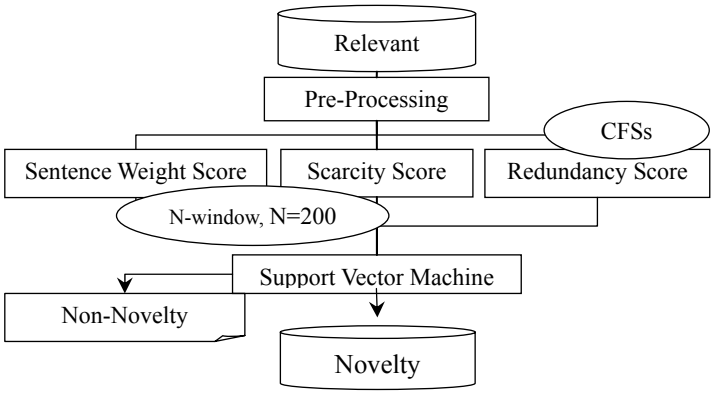


Figure 4.2: Architecture of Novelty Detection System.

4.4. Novelty Result

We submitted three to five runs to Task 1-4. Table 4.1 and 4.2 show our results.

Table 4.1: Official Results of Novelty Task1, 3.

	Run ID	CFSs type	Topic expansion	Sentence expansion	Relevant			New		
					P	R	F	P	R	F
Task1	MeijiHIL10	-	-	-	0.29	0.76	0.384	0.14	0.58	0.212
	MeijiHILcfs	Reurters	yes	no	0.24	0.92	0.357	0.12	0.75	0.194
	MeijiHILodp	ODP	yes	no	0.23	0.96	0.349	0.12	0.77	0.187
Task3	MeijiHIL3	-	-	-	0.31	0.62	0.376	0.14	0.41	0.190
	MeijiHIL3Tc	Reurters	yes	no	0.30	0.54	0.339	0.13	0.41	0.174
	MeijiHIL3STc	Reurters	yes	yes	0.27	0.62	0.339	0.12	0.46	0.166

Table 4.2: Official Results of Novelty Task2, 4.

	Run ID	CFSs type	N-window	Sentence Weight	Scarcity	P	R	F
Task2	MeijiHIL2WCS	Reurters	yes	yes	yes	0.49	0.85	0.608
	MeijiHIL2CS	Reurters	no	no	yes	0.42	1.00	0.580
	MeijiHIL2WRS	-	yes	yes	yes	0.48	0.93	0.619
	MeijiHIL2RS	-	no	no	yes	0.46	0.97	0.609
	MeijiHIL2WR	-	yes	yes	no	0.48	0.93	0.617
Task4	MeijiHIL4WRc	Reurters	yes	yes	yes	0.49	0.64	0.525
	MeijiHIL4RSc	Reurters	no	no	yes	0.40	0.97	0.544
	MeijiHIL4WRS	-	yes	yes	yes	0.54	0.51	0.492
	MeijiHIL4RS	-	no	no	yes	0.44	0.89	0.566
	MeijiHIL4WR	-	yes	yes	no	0.5	0.61	0.522

where the Run ID column in Table 4.1 and 4.2 shows the actual Run ID in TREC. The CFSs type column in Table 4.1 and 4.2 shows the corpus to construct the CFSs. "-" means that concept-based expansion using the CFSs was not available. The Topic expansion and Sentence expansion column

in Table 4.1 show whether the expansion using the CFSs applied or not. "yes" shows that the concept-based expansion was applied in Topic or Sentence. "no" shows that the concept-based expansion was not applied. The N-window, Sentence Weight, and Scarcity column show whether each algorithm was applied or not. The redundancy score used in all submission results.

4.5. Novelty Conclusions

Results show that our concept-based expansion using the CFSs was not working well in both Relevant and Novelty. Table 4.2 shows that N-window is useful in new detection. The difference between *MeijiHIL2WRS* and *MeijiHIL2WR* shows that slightly improve performance. To represent novel feature, we used three criteria, and to distinguish new sentences, we used SVM. The results showed that this approach was effective in Novelty Detection that is binary classification.

5. Genomics Track

This is the first year that our group participates in the Genomics track of the TREC. Here we report our system and a method on the ad hoc task. Our method is to use the model expressing the meaning of words to implement CFS.

5.1. The model expressing the meaning of words to implement CFS

Within one document, any words have only one meaning. we assumed. That is, we expected that the document expresses the meaning of words. Based on this assumption, we created the model.

Using this assumption, a meaning of words is expressed from relationship between vast quantities of words and documents. Specifically, first, we constructed bipartite graph to define relations between words and documents as shown in Figure 5.1. Second, we obtained N documents which are received highest relation value from input words. Finally, relation values of words were calculated from N documents. We regarded relation values of words as Semantic Vector of input words.

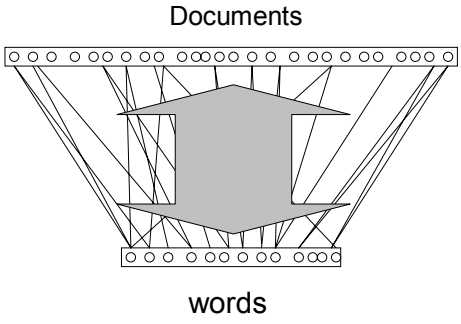


Figure 5.1: Bipartite Graph.

5.1.1. The calculation method of Semantic Vector

1 Define the function, *R*, which determines relation between word set, *W*, and document set, *D*,

$$R(t \in T, d \in D) = \begin{cases} 1 & (if \quad relation \quad between \quad t \quad and \quad d \quad exists) \\ 0 & (else) \end{cases}.$$

2 Define the function, *S*, which determines the degree of relationship between *W* and *D,*

$$S(t \in T, d \in D) = \frac{R(t,d)}{L} \log\left(\frac{L}{C(t)C(d)}\right)$$
$$L = \sum_{T,D} R(T,D), \quad C(t \in T) = \sum_{D} R(t,D), \quad C(d \in D) = \sum_{T} R(T,d)$$

.

3 Calculate *Rd* which is the relation value of documents from input words, *Ti* (⊂*T*),

$$Rd(d \in D) = Rd(d \in D, Ti \subset T) = \sum_{t \in Ti} S(t,d).$$

4 Obtain N documents, *Dn* (⊂*D)*, which have the highest relation value from *Rd*.

5 Calculate Semantic Vector, *SV*,

$$SV(t \in T) = SV(t \in T, Dn \subset D) = \sum_{d \in Dn} S(t,d).$$

5.1.2. Relation between documents and words which were used in the system

To get relation between documents and words, we used tf-idf value calculated from <ArticleTitle> and <AbstractText>. We normalized values which calculated based on tf-idf. And if values were over a certain threshold, we presupposed that relations between words and documents exist.

5.2. System

To weigh the model expressing the meaning of word to implement CFS, we used simple system in Figure 5.2.

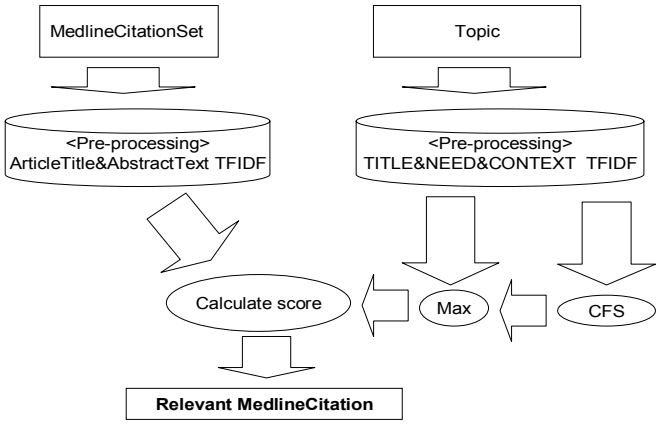


Figure 5.2: System of Genomics Track.

< Outline of the system >

- ✓ Used data field
  - ➢ MedlineCitation
    <ArticleTitle> and <AbstractText>.
  - ➢ Topic
    <TITLE>, <NEED>, and <CONTEXT>.
- ✓ Create word vector
  - ➢ MedlineCitation
    Word vector of MedlineCitation was created using tf-idf.
  - ➢ Topic
    We compared tf-idf and Semantic Vector, and selected the higher one.
- ✓ Similarity calculation
  - ➢ We used cosine measure for calculating relevancy score of MedlineCitation to Topic.

5.3. Conclusion

Results are shown in Table 5.1. To compare the results, we superimpose result which used only tf-idf. The system using the model failed to improve retrieval performance.

Table 5.1: Results.

	run	MAP	R-PREC
tf-idf	*unofficial*	0.2401	0.2883
CFS	MeijiHilG	0.0924	0.1409

References

[i] T. Takagi, A. Imura, H. Ushida, and T. Yamaguchi, Multilayered Reasoning by Means of Conceptual Fuzzy Sets, *International Journal of Intelligent Systems*, vol.11, pp. 97-111, 1996.

[ii] T. Takagi, A. Imura, H. Ushida, and T. Yamaguchi, Conceptual Fuzzy Sets as a Meaning Representation and their Inductive Construction, *International Journal of Intelligent Systems*, vol.10, pp. 929-945, 1995.

[iii] G. Salton, and M. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, 1983.

[iv] M. Mitra, A. Singhal, and C. Buckley, Improving Automatic Query Expansion, In *SIGIR '98*, pp. 206–214, 1998

[v] R. Burgin, The Retrieval Effectiveness of Five Clustering Algorithms as a Function of Indexing Exhaustivity, In *J. American Society for Information Science*, vol. 46, pp. 562–572, 1995.

[vi] M. Hearst and J. Pedersen, Reexamining the cluster hypothesis: Scatter/Gather on retrieval results, In *SIGIR '96*, 1996.

[vii] T. Kanazawa, A. Takesu, and J. Adachi, A Relevancebased Superimposition Model for Effective Information Retrieval, IEICE Transactions, E83-D(12):2152–2160, Dec. 2000.

[viii] T. Kanazawa, A. Aizawa, A. Takasu, and J. Adachi, The Effects of the Relevance-based Superimposition Model in Cross-Language Information Retrieval, In *Proc. 5th European Conference on Research and Advanced Technology for Digital Libraries*, pp. 312–324, Darmstadt, Sept. 2001.

[ix] E. Voorhees, Variations in Relevance Judgements and the Measure of Retrieval Effectivenss, In

SIGIR '98, pp. 315–323, 1998.

[x] Y. Hayashi, On a New Data Model suitable for Intellectual Accesses by Personal Preference, In *IPSJ SIG Notes*, 98-DBS-116(2), pp. 381–388, July 1998.

[xi] T. Kanazawa, A. Takasu, and J. Adachi, Effect of the Relevance-based Superimposition Model on Information Retrieval, In *IPSJ Database workshop 2000 (IPSJ SIG Notes)*, 2000-DBS-122, pp. 57–64, Iwate, July 2000.

[xii] T. Kanazawa, A. Takasu, and J. Adachi, Improving the Relevance-based Superimposition model for IR with automatic keyword extraction, In *RIAO 2004*, May 2004.

[xiii] Reuters Corpus @ NIST or Reuters Corpus
http://trec.nist.gov/data/reuters/reuters.html
http://about.reuters.com/researchandstandards/corpus/

[xiv] http://dmoz.org/

[xv] K. Zechner, Fast Generation of Abstracts from General Domain Text Corpora by Extracting Relevant Sentences, In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pp. 986-989, 1996.

[xvi] T. Joachims, *Making large-Scale SVM Learning Practical*, In *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges and A. Smola (eds.), MIT Press, pp. 169–184, 1999.