Passage-Level Document Retrieval Using Lexical Chains

Hajime Mochizuki[†], Makoto Iwayama[‡], Manabu Okumura[†]
[†]School of Information Science
Japan Advanced Institute of Science and Technology
{motizuki,oku}@jaist.ac.jp
[‡] Central Research Laboratory
Hitachi Ltd.
iwayama@harl.hitachi.co.jp

Abstract

The importance of document retrieval systems which can retrieve relevant documents for user's needs is now increasing with the growing availability of full-text documents. In the traditional document retrieval, each document is treated as a single unit. However, since long documents tend to contain various topics, the passage-level retrieval has been received more attention in the recent document retrieval. The passage can be considered as a sequent part of the document which contains a content related to a content of the query. In the passage retrieval, it is a problem how to decide the passages. It is considered that the passages which form coherent semantic units can effectively improve the performance. Furthermore, it may be also effective that the size and position of passage can be flexibly changed depending on the query and the document. In this paper, we describe a better method for calculating the passages that uses lexical chains. We will also present a passage-level document retrieval method which improves the performance.

1 Introduction

Document retrieval systems try to identify and retrieve relevant documents in response to a query. The relevance of documents to the query depends on the similarity between the documents and the query. The similarity is usually calculated on the basis of importance of the terms in the document that match the terms in the query. The importance of a term is often calculated on the basis of the *term frequency (tf)* and the *inverse of document frequency (idf)*.

In the traditional document retrieval, a document is treated as a single unit, and the importance of a term is calculated from the whole document. However, since real documents, especially longer documents, contain various topics, they tend to consist of multiple units that deal with different topics. The difference of topics tends to appear as the difference of terms that occur. In the part of a document that deals with a topic, the set of terms which relate to the topic tend to occur. Therefore, in several cases, a set of terms frequently occur in one part of a document but rarely occur in other parts. Such a document should not be treated as a single unit, and it is better to calculate importance of a term from the part of the document. From this background, passage-level retrieval has received more attention in the recent document retrieval(Salton, Allan, and Buckley 1993; Callan 1994; Hearst and Plaunt 1993; Knaus, Mittendorf, and Schäuble 1994; Moffat, Sackes-Davis, Wilkinson, and Zobel 1994; Kaszkiel and Zobel 1997; Melucci 1998).

Passage retrieval systems treat a document as a set of passages and calculate the similarity of each passage to a query. The systems can not only retrieve passages themselves as a result but also retrieve documents in order of the similarity of the passages. Since two kinds of the similarity can be calculated between documents and a query, one from the whole document and the other from the passages, it is possible to combine the two retrieval methods for improving the performance.

In the passage retrieval, it is a problem how to decide passages. It is important because the performance of the retrieval will improve if good passages are extracted and used. A passage can be defined as a continuous part of a document in general. We think it would be more suitable to extract from a document as a passage the coherent semantic unit which has the content related to a given query. Furthermore, it may be effective that the size and position of a passage can be flexibly changed depending on a query and a document.

In this paper, we describe a better method for calculating passages dynamically that uses lexical chains(Morris and Hirst 1991). Lexical chains are sequences of words that are in lexical cohesion relation with each other, and tend to indicate fragments of a document that form a semantic unit. Using lexical chains would enable to calculate more appropriate passages and similarity than previous approaches to the passage retrieval. We will also present a passage retrieval method which improves the performance.

In the next section, we first review previous passage retrieval methods. In Section 3, we describe how to calculate lexical chains. In Section 4, we explain the passage extraction method and retrieval method that is based on the lexical chains. In Section 5, we present the experimental procedure to investigate the effectiveness of our method and the results obtained from the experiment.

2 Passage-level document retrieval

The unit of passages can be classified into three types: a unit based on textual discourse units such as chapters, sections, and paragraphs of a document(Salton et al. 1993; Salton, Allan, Buckley, and Singhal 1994; Moffat et al. 1994; Wilkinson 1994; Mittendorf and Schäuble 1994; Knaus et al. 1994), a unit based on a prefixed or variable length window(Callan 1994; Kaszkiel and Zobel 1997; Melucci 1998), and a semantic unit based on coherence or cohesion of a topic(Hearst and Plaunt 1993).

Passage extraction based on discourse units is the method which uses the text structure that the author determines. Therefore, it is relatively easy to extract this type of passages, and it can be done at the indexing time. However, in Japanese, it is often pointed out that semantic boundaries do not always coincide with formal boundaries(Tokoro 1987). In such a case, using discourse units might cause the degrade of the retrieval performance. Furthermore, since this type of passages are extracted beforehand, they do not depend on a query(Callan 1994; Melucci 1998).

Passage extraction based on prefixed or variable length windows is the method which uses the window that has the highest similarity with a query as a passage. When a query is inputted, each document is scanned by moving a window, and the similarity of the window to the query is calculated. Therefore, in this method, passages can be decided based on a query. However, it is a problem that we have to decide an effective window size for improving the performance. In practice, the window size is usually decided depending on the size and type of documents. Furthermore, this passage extraction method also has the problem that the extracted passages do not reflect semantic boundaries in a document.

Passage extraction based on semantic units is the most desirable method, because this type of passages reflect a semantic content in a document. The passage extraction method can be classified into two. The first method is that a document is first divided into discourse segments in the preprocessing, and the segments are candidates of passages. In this method, passages can be decided at the indexing time. The second method is that as a passage, the part of a document is extracted that is related to a given query dynamically after the query is inputted.

Hearst and Plaunt(Hearst and Plaunt 1993) propose the former method for extracting passages. They first divide a document into fixed length blocks and calculate strength of lexical cohesion(Halliday and Hasan 1976) between adjacent blocks as semantic similarity and merge the related blocks. The merged blocks are considered as discourse segments in the document and used as passages. However, since this method extracts passages without taking into account a query, it has the same problem as the case of discourse units. Furthermore, the better performance of text segmentation cannot be currently obtained, though there have been many works on it(Kozima 1993; Hearst 1994; Litman and Passonneau 1995; Mochizuki, Honda, and Okumura 1998).

Without rigidly dividing a document, we can extract topical units in the document by applying shallow processing. As the latter method for extracting semantic passages, we present a method that extracts passages that are related to a query in a document dynamically by using lexical chains(Morris and Hirst 1991).

In the next section, we explain the method for calculating the lexical chains that we use for extracting passages.

3 Calculating lexical chains

Lexical chains are sequences of words which have semantic relation with each other. The 'semantic relation' between words can be computed by the following three methods: the same words, the words in the same class in a thesaurus, and the words that tend to co-occur frequently documents. We explain the three methods in the following subsections. In the preprocessing, we perform morphological analysis(Matsumoto et al. 1997), and select only nouns, verbs, and adjectives in a document as terms for lexical chains.

3.1 Lexical chains based on repetition

A succession of terms can be considered as a chain when the terms have a same surface form.

3.2 Lexical chains based on thesaural relation

This type of lexical chains is based on thesaurus classification, such as WordNet(Fellbaum 1998). A succession of terms can be considered as a chain when the terms belong to the same class in a thesaurus. Since there are ambiguous words which belong to multiple classes in the thesaurus, we use Okumura and Honda's algorithm(Okumura and Honda 1994) that disambiguate word senses in calculating lexical chains. We use as the Japanese thesaurus Kadokawa Ruigo Shin Jiten(Oono and Hamanishi 1981).

3.3 Lexical chains based on co-occurrence relation

This type of lexical chains is based on the degree of similarity between words that can be calculated by using co-occurrence information between words in a corpus. The score of semantic similarity between words X and Y is calculated by the cosine distance, as in the following formula (1):

$$sim(X,Y) = \frac{\sum_{i=1}^{n} x_i \times y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \times \sqrt{\sum_{i=1}^{n} y_i^2}}, \tag{1}$$

where x_i and y_i are the term frequency of words X and Y in document i respectively, and n is the total number of documents in the corpus. This method can be considered to calculate the similarity score between words, based on the degree of their co-occurrence in the same documents. We use a set of newspaper articles for a year as the corpus for calculating the similarity score.

Using this similarity score, we construct clusters of words. In calculating the similarity score between clusters C_i and C_j, we use the shortest distance method:

$$SIM(C_i, C_j) = \max_{X \in C_i, Y \in C_j} sim(X,Y). \tag{2}$$

Sequences of words in the same clusters are then regarded as lexical chains. The method of constructing lexical chains proceeds as follows: calculate the similarity score among words in each sentence of a document, construct and try to merge clusters of words in each sentence while the similarity score between them exceeds the threshold, and then try to merge the clusters for sentences again while the similarity score exceeds the threshold, and finally obtain the clusters of words(lexical chains) for the document.

3.4 Removing unimportant chains

Since some chains are unimportant in the lexical chains calculated above, we remove them in the next step. In the case where occurrences of terms are sparse in a lexical chain, we think the chain is unimportant and should be removed. Consider, for example, Figure 1. Chains C and E can be considered to be unimportant and should be removed.

In a lexical chain, there exist parts where terms do not occur. Such parts can be called gaps. We think the topic which relates to the lexical chain is not described in the range of gaps. Therefore, if the length of a gap exceeds a certain length, it is better to divide a lexical chain into subchains. For example, chain B in Figure 1 has two gaps, from the 14th word to the 23rd and from the 28th to the 36th.

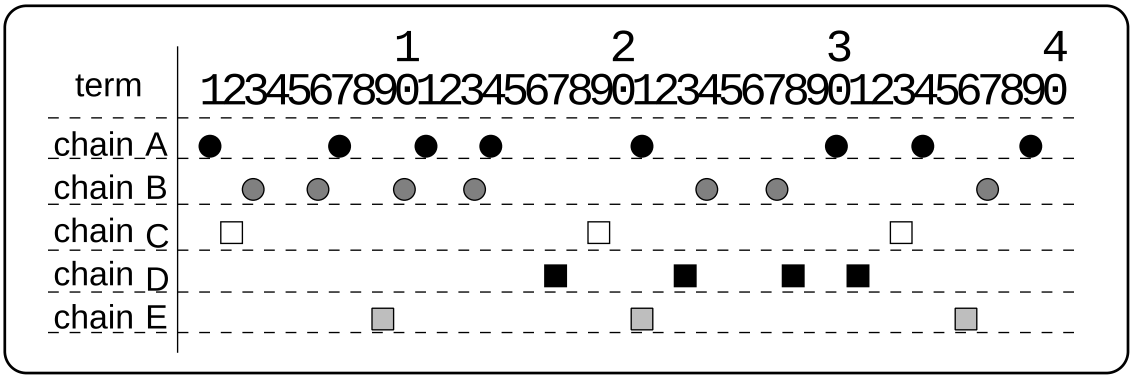

Figure 1: An example of lexical chains

To remove unimportant lexical chains, we provide the following two constraints using the thresholds of the gap and chain length:

- threshold of the gap length:

  If the length of a gap in a chain exceeds the threshold of the gap length, the chain is divided into subchains at the point of the gap.

- threshold of the chain length:

  If the length of (sub)chains do not exceed the threshold of the chain length, the chain is removed as unimportant.

Consider chain B in Figure 1 again. Chain B is divided into the subchains B1, B2, and B3 as in Figure 2 by the threshold of the gap length. Next, subchains B2 and B3 are removed as unimportant by the threshold of the chain length. As a result, only subchain B1 in chain B remains.

Chains C and E in Figure 1 are also divided into subchains by the threshold of the gap length, and all subchains are removed as unimportant by the threshold of the chain length.

4 Passage extraction and retrieval using lexical chains

In this section, we describe a method of weighting and indexing lexical chains, and explain the passage extraction and retrieval method that is based on the lexical chains. Our passage extraction and retrieval method is based on the following steps. First, the ranges and weights of lexical chains that are related to each term in the input query are obtained for each document. Next, passages are constructed by using the overlapped lexical chains for each document. Lastly, the similarity between the query and each passage is calculated by adding the weight of lexical chains in the passage.

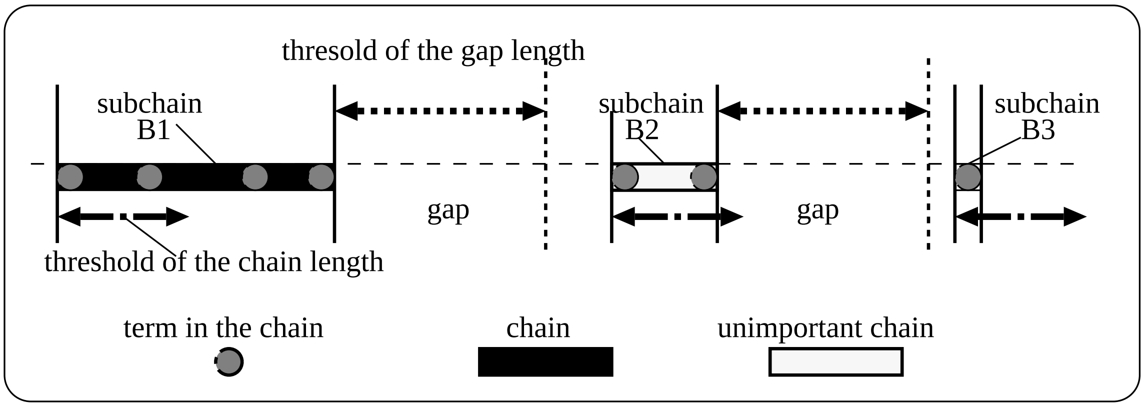

Figure 2: Removing unimportant chains

4.1 Weighting lexical chains

To calculate the weight w_{C_d}, of a lexical chain C_d in a document d, we define the following formulae based on the standard $tf * idf$ measure. We assume that a chain is more important if it contains more terms and it seldom occurs in other documents in the document set.

For the lexical chains that are based on repetition or thesaural relation, we use the following formula:

$$w_{C_d} = \mid C_d \mid \times log(N/n_{C_d}), \tag{3}$$

and for the chains that are based on co-occurrence relation of terms, we use the following formula:

$$w_{C_d} = \mid C_d \mid \times log(N/\max_{c \in C_d} n_c), \tag{4}$$

where $\mid C_d \mid$ is the total number of terms in chain C_d, N is the total number of documents in the document set, n_{C_d} is the number of documents in which lexical chain C_d occurs, and $max_{c \in C_d} n_c$ is the maximum of the number of documents where a term in lexical chain C_d occurs. Since in the case of co-occurrence based lexical chains, the terms in a lexical chain are different for each document, the exact number of documents where lexical chain C_d occurs cannot be calculated. Therefore, we use $\max_{c \in C_d} n_c$ instead of n_{C_d} as an approximation.

4.2 Indexing lexical chains

We make the index of lexical chains to retrieve chains against a given query and to get the ranges and weights of lexical chains for each document. The record of the index has the following form:

(a term in the chain, document ID, chain ID[range of the chain], weight).

An example of the index is shown in Figure 3.

(star,00001, A1[1-36], xxxx)
(galaxy,00001, A1[1-36], xxxx)
(star, 00001, A2[93-116],yyyy)
(sun, 00001, A2[93-116],yyyy)
(moon, 00001, A2[93-116],yyyy)

Figure 3: An example of chain index

Using the index in Figure 3, chain A1 and A2 in document 00001 can be retrieved by term 'star', and it is found that the range of chain A1 is from the 1st word to the 36th and the range of chain A2 is from the 93rd word to the 116th in document 00001.

4.3 Extracting passages

By using the index in the last section, passages which have strong relation to a query can be extracted by the following steps:

1. Nouns, verbs and adjectives are chosen as query terms from the query.
2. The index is searched for against each query term, and the ranges and the weights of the lexical chains that are related to each query term are obtained.
3. The ranges of passages are fixed for each document which include lexical chains matched to the query terms. The range of a passage is fixed by merging lexical chains that are overlapped with each other that is related to the query for a document.
4. The similarity between each passage and the query is calculated.

The score(similarity) of a passage is calculated based on the following factors:

- the scores of lexical chains in the passage,
- the degree of the overlap of lexical chains in the passage.

The score is calculated by the following three steps. First, the similarity cw_k, between query term q_k and lexical chain C_k is calculated by formula (5):

$$cw_k = (tf_{q_k} \times log(N/n_k))^2 \times w_{C_k}, \tag{5}$$

where tf_{q_k} is the frequency of term q_k in the query, N is the total number of documents in the document set, n_k is the number of documents that term q_k occurs , and w_{C_k} is the weight of lexical chain C_k. Next, cw_k is divided by the length of the lexical chain cl_k, and the score $cw_{k,j}$ of each position j in chain C_k is obtained (formula (6)).

$$cw_{k,j} = cw_k/cl_k \tag{6}$$

Lastly, the score is calculated by the sum of score $cw_{k,j}$, weighted to the overlap of the chain in each position, from the beginning point of a passage to the ending point in formula (7):

$$sim(Q, P_i) = \sum_{j=begin_i}^{end_i} (\sum_{k=1}^{|Q|} cw_{k,j}) \times (\mid q_j \mid^2 / \mid Q \mid^2), \tag{7}$$

where Q is a query, P_i is a passage, $begin_i$ and end_i represent the starting and ending point of P_i, respectively. $\mid Q \mid$ is the number of query terms. $\mid q_j \mid$ is the number of lexical chains which cover position j in passage P_i.

Consider, for example, Figure 4. Three query terms match lexical chains A1, A2, and B1, B2, and C1, C2, C3, respectively. Passages are then fixed as the maximal fragments that are covered by overlapped lexical chains for query terms. Therefore, in Figure 4, three passages can be extracted based on the three query terms.

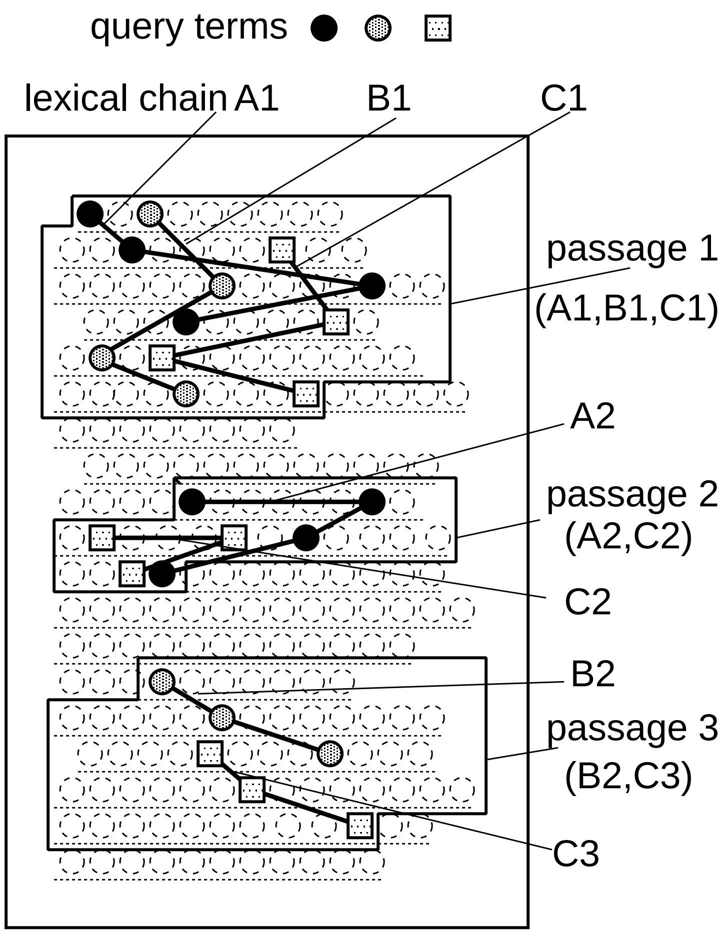

Figure 4: An example of passages in a document

5 Experiments

To evaluate the effectiveness of our method, we make an experiment using the Japanese test collection $BMIR-J2$(Kitani et al. 1998). The test collection BMIR-J2 consists of 5080 articles from the Mainichi newspaper and 50 queries[1]. The relevance of documents is judged on two levels 'A' and 'B'. The documents judged as 'A' are those that are strongly related to the query. The documents judged as 'B' are those that are partly related to the query. We adopt the documents judged as 'A' as the relevant documents in the experiment.

Since we think better comparison can be made for the passage retrieval if we use only longer documents, we select and use 904 documents that have more than 1600 bytes in 5080 documents. We also select only the queries which have more than five relevant documents in the 904 documents, since using queries that have fewer relevant documents might make experiments less reliable. The statistics of the test collection that we use is listed in table 1.

number of documents	904
number of queries	8
average number of answers	6.5
average bytes of documents	1785.8

Table 1: The statistics of test collection

In the experiment, the performance of our method is compared with the following three retrieval methods: the traditional keyword retrieval, the passage retrieval based on paragraphs, and the passage retrieval based on fixed length windows. We also make a comparison between the case where each passage retrieval method is used alone and the case where each passage retrieval method is combined with the keyword retrieval method.

5.1 Retrieval methods to be compared

We use the following retrieval methods for comparison with our method.

- Keyword Retrieval
  The weight of term t in a document is calculated by normal $tf * idf$ (formula (8)), and the document is expressed as the vector D with the weight of terms:

$$w_t = tf_t \times log(N/n_t), \tag{8}$$

  where tf_t is the frequency of term t in a document, N is the total number of documents, and n_t is the number of documents in which term t occurs.

[1]BMIR-J2 was constructed by the SIG Database Systems of the Information Processing Society of Japan, in collaboration with the Real World Computing Partnership.

The similarity between query vector Q and document vector D is calculated by the following formula:

$$sim(Q,D) = \sum_t (tf_{q_t} \times log(N/n_t))^2 \times w_t, \tag{9}$$

where tf_{q_t} is the frequency of query term q_t in the query.

- Passage Retrieval based on Paragraphs

  In this method, the document title and paragraphs are treated as a single passage. Similar to the case of keyword retrieval, the weight of each term in a passage is calculated by formula (8), and formula (9) is used to calculate the similarity between a query and each passage, except that tf_t means the number of term t which occurs in a passage, N is the total number of passages, and n_t is the number of passages in which term t occurs. There are 8.8 passages per document in average and 28.2 terms per passage in our test collection.

- Passage Retrieval based on Windows

  In this method, a passage is made by the fixed length window of the size l terms. Like Callan(Callan 1994), the scanning by a window begins from the first position of the term matching the query to the end of the document, while the window moves with $\frac{l}{2}$. The similarity is calculated at each window position.

  Similar to the case of keyword retrieval, the weight of each term in a window is calculated by formula (8), and formula (9) is used to calculate the similarity between a query and each window, except that tf_t means the number of term t which occurs in the window. In the experiment that we will make, we change the window size from 20 words to 300 words in units of 20 words.

- Passage Retrieval based on Lexical Chains

  This is the method that we described in the previous section. If more than two passages are extracted, we choose the best passage which has the maximum similarity:

$$sim(Q,D) = \max_i sim(Q,P_i), \tag{10}$$

  where $sim(Q,P_i)$ is calculated by formula (7).

  As mentioned in section 3.4, the thresholds of the gap length and the chain length are fixed to select important lexical chains. In the experiment, we change both thresholds in four ways of $1/4, 1/8, 1/16,$ and $1/32$ of the number of terms in the document, and the experiment is made by using the combination of $4*4(=16)$.

  As mentioned in section3.3, the threshold of co-occurrence score is used for term clustering in the calculation of co-occurrence based lexical chains. In the experiment, this type of lexical chains is calculated with the thresholds of $0.2, 0.25, 0.3, 0.35,$ and 0.4.

- Combination of Passage and Keyword Retrieval

  In the experiment, the keyword retrieval is combined with the passage retrieval based on paragraphs, windows, and lexical chains, respectively. We adopt a simple combination method as follows, though there are a variety of combination methods.

  In combining two methods, the similarity is calculated as the sum of the similarity in two methods, which are normalized by each maximum similarity (formula (11)):

$$sim(Q, D) = \frac{ksim}{\max_i ksim_i} + \frac{psim}{\max_i psim_i}, \tag{11}$$

  where $ksim$ and $psim$ are the similarity between the query Q and the document D in the keyword retrieval and the passage retrieval respectively. $Max_i ksim_i$ and $\max_i psim_i$ are the maximum similarity against the query Q in the keyword retrieval and in the passage retrieval respectively.

5.2 Experiments

The experiment is made by using four methods explained in the last subsection with the following combinations.

1. Keyword retrieval (**document**)
2. Passage retrieval based on paragraphs (**formpara**)
3. Passage retrieval based on fixed length windows (**window**)
4. Passage retrieval based on lexical chains
   with the information of
   (4-a) repetition (**repetition**)
   (4-b) thesaural relation (**thesaurus**)
   (4-c) co-occurrence score(**cooccurrence**)
5. Combination of 1 and 2 (**formpara_doc**)
6. Combination of 1 and 3 (**window_doc**)
7. Combination of 1 and 4
   (7-a) 1 and (4-a) (**repetition_doc**)
   (7-b) 1 and (4-b) (**thesaurus_doc**)
   (7-c) 1 and (4-c) (**cooccurrence_doc**)

We use two measures *Recall* and *Precision* for evaluation. *Recall* is the quotient of the number of correctly identified documents by the total number of correct documents. *Precision* is the quotient of the number of correctly identified documents by the number of retrieved documents.

We evaluate the performance for each query with the top M documents in the ranking. In the experiment, we change M from 2 to 26 in units of 2 documents. The average of recall and precision for all queries are calculated and plotted at each M.

Figure 5 shows the results of each passage retrieval alone and keyword retrieval, and Figure 6 shows the results of the combined methods. As we explained in the last section, we change the values for various thresholds in the experiment, and fix the value for each threshold as the one that yields the best average precision for each M. Figure 5 and 6 show the results with the fixed values for those thresholds. Table 2 shows the value of each threshold which yields the best average precision. Table 2 also shows the average and standard deviation of the passage size for each method.

	co-occurrence	repetition	thesaurus	window	paragraph
threshold of co-occurrence score	0.25	-	-	-	-
threshold of chain length	8	32	32	-	-
threshold of gap length	4	8	8	-	-
window size	-	-	-	260	-
average passage size	96.0	81.9	64.9	260	43.4
standard deviation	52.3	68.4	46.5	-	19.3

Table 2: Best parameter values

5.3 Results

In Figure 5, comparing the performance among **document** and passage retrievals, the results show that the keyword retrieval (**document**) is the best and **window** is much the same as **document**, and **cooccurrence** is the best in the other passage retrieval methods, though its performance is still lower than **document**.

On the other hand, in Figure 6, comparing the performance among **document** and each combination of **document** and passage retrievals, the results show that the best performance can be yielded by the combination of **document** and the passage retrieval based on the lexical chains by term co-occurrence information (**cooccurrence_doc**). The results also show that **window_doc** is much the same as **document**, and in the other four methods their performance cannot be higher than the performance of **document**, though both recall and precision can be improved by combining with **document**.

Therefore, we can say that the highest precision and recall can be obtained by combining the keyword retrieval with the passage retrieval based on the lexical chains by term co-occurrence information (**cooccurrence_doc**). The improvement of the performance is especially remarkable with the higher ranking documents. With the top 10 documents, for example, recall and

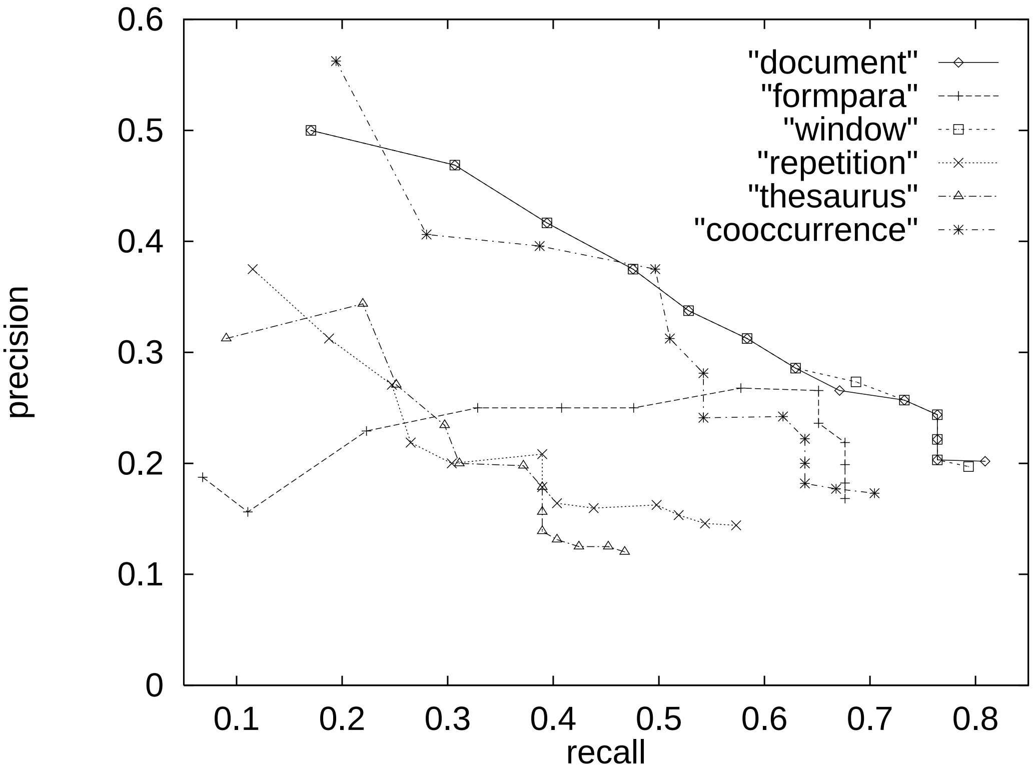

Figure 5: The performance with passage retrieval method

precision of **cooccurrence_doc** are 64% and 40% respectively, while recall and precision of **document** are only 53% and 34% respectively.

6 Conclusion

In this paper, we described a better method for calculating passages dynamically that uses lexical chains and also presented the passage retrieval method which improves the performance. The results from the experiments showed that combining the passage retrieval based on lexical chains by term co-occurrence information and the keyword retrieval outperforms others both in recall and precision.

There exist many options in our passage retrieval method to consider. In calculating co-occurrence score, for example, there exist many other methods for calculation. If there are multiple passages in a document, we are currently using the only passage with the highest similarity to a query. However, we can use the sum of the similarity of the top N passages instead of the only one. We think it is necessary to clarify what kinds of factors influence the performance.

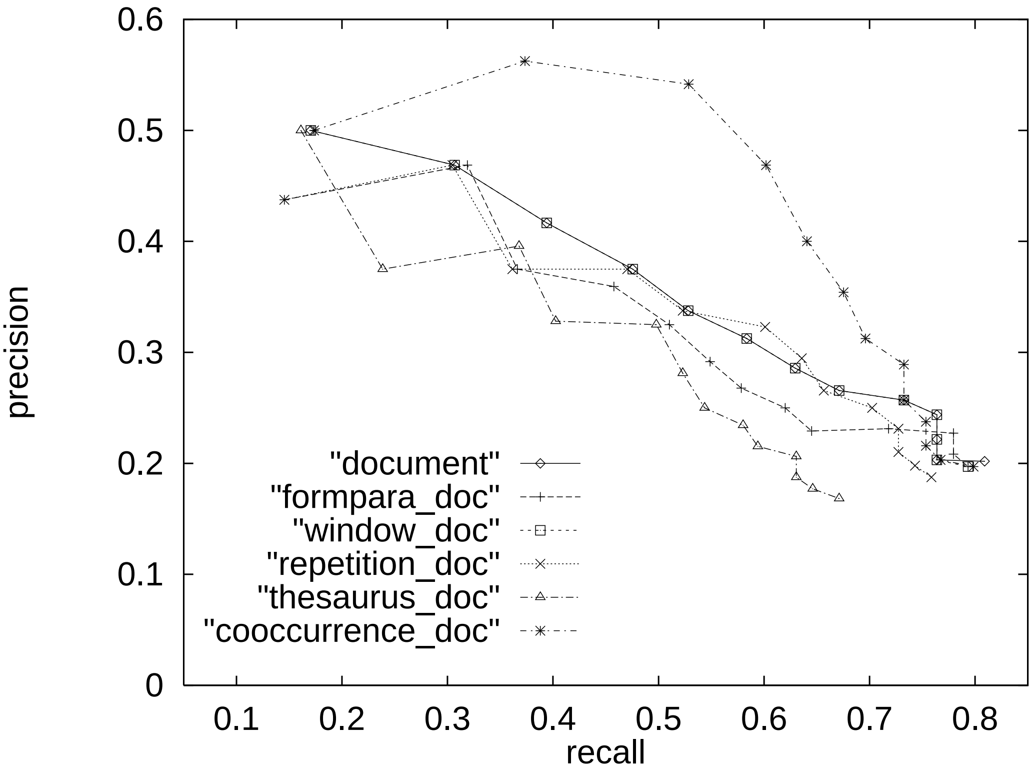

Figure 6: The performance with combined methods

Acknowledgment

The authors would like to express our gratitude to Dr. Akihiko Takano, Dr. Yoshiki Niwa, Dr. Shingo Nishioka and other members in the Central Research Laboratory, Hitachi co.,Ltd. for their precious suggestions to our work and allowing us to use their system for the word association calculation. We are also grateful to Kadokawa publisher for allowing us to use their thesaurus.

Reference

Callan, J. P. (1994). "Passage-Level Evidence in Document Retrieval." In *Proc. of 17th Annual International ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval*, pp. 302–310.

Fellbaum, C. (Ed.). (1998). *WordNet: An Electronic Lexical Database.* MIT Press.

Halliday, H., and Hasan, R. (1976). *Cohesion in English.* Longman.

Hearst, M. (1994). "Multi-Paragraph Segmentation of Expository Texts." In *Proc. of the 32nd Annual Meeting of the Association for Computati onal Linguistics*, pp. 9–16.

Hearst, M., and Plaunt, C. (1993). "Subtopic Structuring for Full-Length Document Access." In *Proc. of 16th Annual International ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval*, pp. 59–68.

Kaszkiel, M., and Zobel, J. (1997). "Passage Retrieval Revisited." In *Proc. of 20th Annual International ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval*, pp. 178–185.

Kitani, T., et al. (1998). "Japanese Test Collection for Evaluation of Information Retireval Systems, BMIR-J2." In *IPSJ SIG-DBS-144-3*, pp. 15–22. in Japanese.

Knaus, D., Mittendorf, E., and Schäuble, P. (1994). "Improving a Basic Retrieval Method by Links and Passage Level Evidence." In *Proc. of the third Text REtrieval Conference*, pp. 241–246.

Kozima, H. (1993). "Text segmentation based on similarity between words." In *Proc. of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 286–288.

Litman, D., and Passonneau, R. (1995). "Combining Multiple Knowledge Sources for Discourse." In *Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics*, pp. 108–115.

Matsumoto, Y., et al. (1997). *Japanese Morphological Analysis System Chasen version 1.5 Manual.* in Japanese.

Melucci, M. (1998). "Passage Retrieval: A Probabilistic Technique." *Information Processing & Management*, *34* (1), 43–63.

Mittendorf, E., and Schäuble, P. (1994). "Document and Passage Retrieval Based on Hidden Markov Models." In *Proc. of 17th Annual International ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval*, pp. 302–310.

Mochizuki, H., Honda, T., and Okumura, M. (1998). "Text Segmentation with Multiple Surface Linguistic Cues." In *Proc. of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, pp. 881–885.

Moffat, A., Sackes-Davis, R., Wilkinson, R., and Zobel, J. (1994). "Retrieval of Partial Documents." In *Proc. of the second Text REtrieval Conference*, pp. 181–190.

Morris, J., and Hirst, G. (1991). "Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text." *Computational Linguistics*, *17* (1), 21–48.

Okumura, M., and Honda, T. (1994). "Word Sense Disambiguation and Text Segmentation Based on lexical Cohesion." In *Proc. of the 15th International Conference on Computational Linguistics*, pp. 755–761.

Oono, Y., and Hamanishi, M. (1981). *Kadokawa Ruigo Shin Jiten*. Kadokawa. in Japanese.

Salton, G., Allan, J., and Buckley, C. (1993). "Approaches to passage retrieval in full text information systems.." In *Proc. of 16th Annual International ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval*, pp. 49–56.

Salton, G., Allan, J., Buckley, C., and Singhal, A. (1994). "Automatic Analysis, Theme Generation, and Summarization of Machine-Readable Texts." *SCIENCE 3 JUNE*, 1421–1426.

Tokoro, K. (1987). *Gendaibun Rhetric Dokukaihou*. Takumi. in Japanese.

Wilkinson, R. (1994). "Effective retrieval of structured documents." In *Proc. of 17th Annual International ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval*, pp. 311–317.