English Sentence Retrieval System Based on Dependency Structure and its Evaluation

Yoshihide Kato
Graduate School of International Development
Nagoya University
Furo-cho, Chikusa-ku, Nagoya,
464-8601 Japan
yosihide@gsid.nagoya-u.ac.jp

Seiji Egawa
Graduate School of Information Science
Nagoya University
Furo-cho, Chikusa-ku, Nagoya,
464-8601 Japan

Shigeki Matsubara
Information Technology Center
Nagoya University
Furo-cho, Chikusa-ku, Nagoya,
464-8601 Japan

Yasuyoshi Inagaki
Faculty of Engineering
Toyohashi University of Technology
1-1 Hibarigaoka, Tempaku-cho, Toyohashi, Aichi,
441-8580 Japan

Abstract

This paper proposes a system of retrieving English sentences by utilizing linguistically structural information. The user's query consists of a sequence of keywords. The system automatically identifies dependency relations between occurrences of the keywords in sentences and classifies the sentences according to those dependency relations. The system provides a simple and intuitive keyword-based interface and realizes linguistically sophisticated sentence retrieval. We conducted an experiment to evaluate the accuracy of the system. The results demonstrate that the system provides high accuracy.

1. Introduction

Sentence retrieval systems help non-natives to write papers in English. These systems are useful for English learners, translators and so on. Many systems have been presented so far.

Most systems provide keyword-based search functionality (for example, Tonguen[6] and WebCorp[1]). They are simple and intuitive, but not linguistically sophisticated enough to utilize language structure.

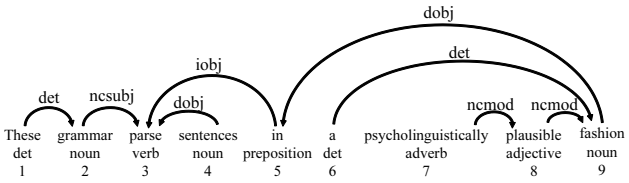
On the other hand, Corley et al.[2] and Resnik et al.[5] have presented syntax-based search systems, Gsearch and Linguist's Search Engine (LSE), respectively. The Gsearch receives a phrase structure pattern and context free grammar from the user. The system parses sentences by using the given grammar, and returns the sentences whose parse trees match the given pattern. In the LSE, the user first gives an example of sentences which s/he needs. The system parses the example sentence by using a parser and returns the parse tree. The user edits the parse tree to specify a structural pattern. Finally, the system returns the sentences whose parse trees match the pattern.

These systems can retrieve sentences by utilizing syntactic structure. However, the interfaces are not simple. The user needs to build a grammar or to specify a structural pattern. It is a hard task for naive users.

This paper presents a sentence retrieval system which provides both a keyword-based interface and syntax-based search functionality. The user's query consists of a sequence of keywords. The system automatically identifies dependency relations (modifier-modifee relations) between the occurrences of the keywords in sentences, and classifies the sentences according to the identified dependency relations. The classification assists the user to obtain the sentences which s/he needs. The user needs neither to build a grammar like the Gsearch nor to edit a structural query like the LSE. We conducted an experiment to evaluate the accuracy of the classification. The results demonstrate that the classification is performed with high accuracy.

This paper is organized as follows: Section 2 gives an overview of our system. Section 3 proposes an algorithm of identifying dependency structure patterns, which is the basis of our system, and Section 4 describes a sentence clas-

[1] http://www.webcorp.org.uk/



det	:determiner	ncmod	:modifier
ncsubj	:subject	iobj	:preposition modifying verb
dobj	:object		

Figure 1. Dependency structure for "These grammars parse sentences in a psycholinguistically plausible fashion."

sification based on the algorithm. Section 5 reports experimental results. In Section 6, we concludes this paper.

2. An Overview of Our System

Our proposed system has the following three features:

1. keyword-based interface: The user's query consists of a sequence of keywords.

2. dependency-based retrieval: The resulting sentences should include the all keywords and there should be dependency relations between the occurrences of the keywords.

3. sentence classification: The system classifies the resulting sentences according to the dependency relations holding between the occurrences of the keywords.

The first feature realizes an intuitive interface between the system and its user. We expect that the second and third features make the retrieval effective. In the rest of this section, we describe the basic concept of the second and third features.

As an example, let us consider the following query:

```
parse sentence in -n          (1)
```

Our system accepts words or part-of-speech (POS) tags as keywords. `-n` means a noun word. For query (1), our system returns sentences such as the following:

(a) These grammars *parse sentences in* a psycholinguistically plausible *fashion*.

The sentence has the dependency relations between the occurrences of the keywords (see Figure 1).

On the other hand, the system discards from the search results the following sentence which has no dependency relations between the occurrences of the keywords:

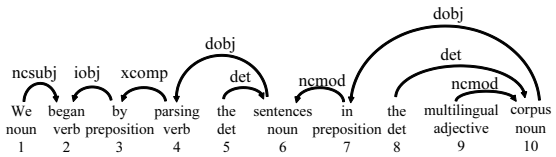


Figure 2. Dependency structure for "We began by parsing the sentences in the multilingual corpus."

(b) We want to map phrase structure trees to *parses* which predict the words of the *sentences in* the multilingual *corpus*.

This sentence is probably irrelevant to the user intention. The second feature means that the system discards this type of sentence from the search results.

Let us consider another sentence:

(c) We began by *parsing* the *sentences in* the multilingual *corpus*.

This sentence also has dependency relations between the occurrences of the keywords (see Figure 2). That is, sentence (c) is also included in the search result. However, the dependency relations are different from that in sentence (a). In order for the user to easily obtain sentences which s/he needs, the system should display these sentences separately. This is why our system has the third feature.

3. Identifying Dependency Structure Pattern

This section proposes a method of identifying dependency structure patterns. The method is the basis of our system. It receives a keyword sequence and a sentence annotated with dependency structure, and returns a *dependency structure pattern* which represents dependency relations holding between the occurrences of the keywords in the sentence. As an example, assume that query (1) and the sentence with the dependency structure shown in Figure 1 are given. For this inputs, the method generates a pattern shown in Figure 3.

The method is an extended version of the method proposed in the literature[4]. It is augmented with types of dependency relations.

3.1. A Method of Identifying Dependency Structure Patterns

Our proposed method identifies dependency structure patterns which are formed by occurrences of keywords in

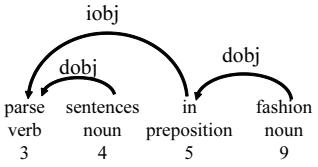


Figure 3. An example of dependency structure pattern for `parse sentence in -n`.

a sentence. If the occurrences of the keywords have no dependency relations, the method fails to identify the pattern.

The inputs are as follows:

query: $q_1 \cdots q_m$ ($q_1, \ldots, q_m$ are keywords)

sentence: $s = w_1 \cdots w_n$ ($w_1, \ldots, w_n$ are pairs of words and POS tags)

dependency structure: D

where D is a set of dependency relations between the words in s. If w_j modifies w_k and a relation r holds between w_j and w_k, (j, k, r) is an element of D. We write $j \xrightarrow{r} k$ for (j, k, r).

We define a dependency structure pattern as a 5-tuple $d = (h, D_L, D_R, T_L, T_R)$, where h is a word position, D_L and D_R are lists of dependency structure patterns and T_L and T_R are lists of relations. h is called *head* of d. The dependency structure pattern d represents that each head of the dependency structure pattern in D_L modifies h from left. Similarly for D_R, from right. The i-th element r_i in T_L means that the relation r_i holds between the head of the i-th element in D_L and the head of d. Similarly for T_R.

Our method generates dependency structure patterns from a query $q_1 \cdots q_m$ by applying the following operations in a bottom-up fashion.

initialization: For each $q_i (1 \leq i \leq m)$ and each $w_j (1 \leq j \leq n)$, if q_i matches w_j, then generate the dependency pattern $(j, \varepsilon, \varepsilon, \varepsilon, \varepsilon)$. ε means null list.

combining: Let $d = (h, D_L, D_R, T_L, T_R)$ and $d' = (h', D'_L, D'_R, T'_L, T'_R)$ be dependency structure patterns for $q_i \cdots q_j$ and $q_{j+1} \cdots q_k$, respectively. Let $rm(d) < lm(d')$ where $rm(d)$ and $lm(d)$ are the rightmost and leftmost positions in d, respectively. If $h \xrightarrow{r} h' \in D$ and $D'_R = \varepsilon$, then generate a dependency structure pattern $(h', dD'_L, D'_R, rT'_L, T'_R)$ for $q_i \cdots q_j q_{j+1} \cdots q_k$ (see Figure 4(a)). If $h' \xrightarrow{r} h \in D$, then generate a dependency structure pattern $(h, D_L, D_R d, T_L, T_R r)$ for $q_i \cdots q_j q_{j+1} \cdots q_k$ (see Figure 4(b)).

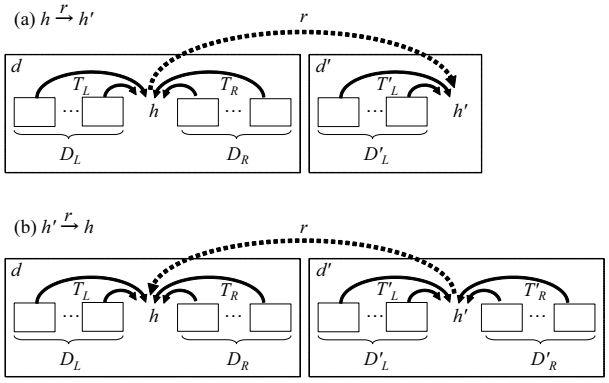


Figure 4. Combining.

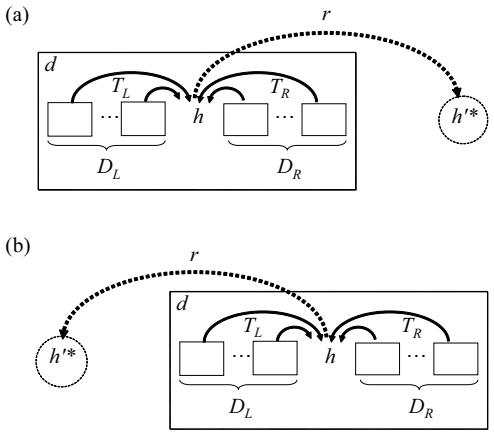


Figure 5. Interpolation.

By applying the combining operation to the query, all dependency structure patterns that directly connect the occurrences of the keywords in the sentence s can be generated.

3.2. Interpolation

In some cases, the user may not intend that some keywords in the query directly modify the other keywords. To process such queries robustly, the interpolation operation is used:

interpolation: Let d be a dependency structure pattern for $q_i \cdots q_j$ whose head is h. For h' and r such that $h \xrightarrow{r} h' \in D$, if $rm(d) < h'$, then generate a dependency structure pattern $(h'^*, d, \varepsilon, r, \varepsilon)$ for $q_i \cdots q_j$ (see Figure 5 (a)). If $h' < lm(d)$, then generate a dependency structure pattern $(h'^*, \varepsilon, d, \varepsilon, r)$ for $q_i \cdots q_j$. A symbol $*$ means that h' is introduced by interpolation (see Figure5 (b)).

The operation can generate dependency structure patterns which include words not occuring in the query. However, unrestricted application of the interpolation operation generates dependency structure patterns for any sentences where all the keywords occur. It is inconsistent with the concept of our system described in Section 2. To avoid useless application of the operation, we introduce a cost defined by the number of occurrence of $*$ in the dependency structure pattern. The algorithm generates no dependency structure patterns whose costs are greater than a threshold.

3.3. Algorithm

Figure 6 illustrates the algorithm of identifying dependency structure patterns. θ is the threshold of cost. $D_{i,j,c}$ is used for recording the dependency structure patterns with cost c for $q_{i+1} \cdots q_j$.

input: query $q_1 \cdots q_m$,
sentence $w_1 \cdots w_n$,
dependency structure D

initialization:
for $i = 1$ **to** m **do**
 for each j s.t. $w_j = q_i$ **do**
  push $(j, \varepsilon, \varepsilon, \varepsilon, \varepsilon)$ to $D_{i-1,i,0}$;

for $cost = 0$ **to** θ **do**
 combining:
 for $k = 2$ **to** m **do**
  for $j = k - 1$ **down to** 1 **do**
   for $i = j - 1$ **down to** 0 **do**
    for $c = o$ **to** $cost$ **do**
     for each $d = (h, D_L, D_R, T_L, T_R) \in D_{i,j,c}$
      $d' = (h', D'_L, D'_R, T'_L, T'_R) \in D_{j,k,cost-c}$
      s.t. $rm(d) < lm(d')$ **do**
      if $h \xrightarrow{r} h' \in D \wedge D'_R = \varepsilon$ **then**
       push $(h', dD'_L, D'_R, rT'_L, T'_R)$ to $D_{i,k,cost}$;
      if $h' \xrightarrow{r} h \in D$ **then**
       push $(h, D_L, D_R d', T_L, T_R r)$ to $D_{i,k,cost}$;

 interpolation:
 for $j = 1$ **to** m **do**
  for $i = j - 1$ **down to** 0 **do**
   if $i \neq 0 \vee j \neq m$ **then**
    for each $d = (h, D_L, D_R, T_L, T_R) \in D_{i,j,cost}$ **do**
     for h' s.t. $h \xrightarrow{r} h' \in D$ **do**
      if $h' > rm(d)$ **then**
       push $(h'^*, d, \varepsilon, r, \varepsilon)$ to $D_{i,j,cost+1}$;
      else if $h' < lm(d)$ **then**
       push $(h'^*, \varepsilon, d, \varepsilon, r)$ to $D_{i,j,cost+1}$;

return $D_{0,m,0} \cup \cdots \cup D_{0,m,cost}$;

Figure 6. An algorithm of identifying dependency structure patterns

3.4. Examples

3.4.1 An example of combining operation

As an example, let us consider query (1) and the sentence with the dependency structure shown in Figure 1.

The first keyword `parse` matches the third word in the sentence, so the following dependency structure pattern is generated by the initialization:

$$(3, \varepsilon, \varepsilon, \varepsilon, \varepsilon) \tag{2}$$

Similarly, the following dependency structure pattern is generated for the keyword `sentence`:

$$(4, \varepsilon, \varepsilon, \varepsilon, \varepsilon) \tag{3}$$

For the keyword `in` ,

$$(5, \varepsilon, \varepsilon, \varepsilon, \varepsilon) \tag{4}$$

is generated. For the keyword `-n`,

$$(2, \varepsilon, \varepsilon, \varepsilon, \varepsilon) \tag{5}$$

$$(4, \varepsilon, \varepsilon, \varepsilon, \varepsilon) \tag{6}$$

$$(9, \varepsilon, \varepsilon, \varepsilon, \varepsilon) \tag{7}$$

are generated.

For the adjacent patterns (2) and (3), there exists a dependency relation between their heads, that is, $4 \overset{dobj}{\rightarrow} 3$ holds. Therefore, these patterns are combined and the following pattern is generated for `parse sentence`:

$$(3, \varepsilon, (4, \varepsilon, \varepsilon, \varepsilon, \varepsilon), \varepsilon, dobj) \tag{8}$$

Furthermore, since $9 \overset{dobj}{\rightarrow} 5$ holds between adjacent patterns (4) and (7), the following pattern is generated for `in -n`.

$$(5, \varepsilon, (9, \varepsilon, \varepsilon, \varepsilon, \varepsilon), \varepsilon, dobj) \tag{9}$$

By applying the combining operation to patterns (8) and (9),

$$\begin{aligned}(3, \quad & \varepsilon, (4, \varepsilon, \varepsilon, \varepsilon, \varepsilon) \cdot (5, \varepsilon, (9, \varepsilon, \varepsilon, \varepsilon, \varepsilon), \varepsilon, dobj), \\ & \varepsilon, dobj \cdot iobj)\end{aligned} \tag{10}$$

is generated.

On the other hand, for the sentences where some occurrences of keywords do not directly modify the other (for instance, sentence (b) in Section 2), the algorithm generates no dependency structure patterns.

3.4.2 An example of interpolation operation

Let us consider another example of query:

```
parse sentence fashion
```

and the sentence in Figure 1. If we set the threshold of cost to 0, no dependency structure patterns are generated. If the threshold is 1, a dependency structure pattern is generated by the following steps. For the keyword `fashion`, dependency structure pattern (7) is generated. Since $9 \overset{dobj}{\rightarrow} 5$ holds, the following dependency structure pattern is generated for `fashion` by applying the interpolation operation to pattern (7):

$$(5^*, \varepsilon, (9, \varepsilon, \varepsilon, \varepsilon, \varepsilon), \varepsilon, dobj) \quad (11)$$

The cost of this pattern is 1. Next, for this pattern and pattern (8), the dependency structure pattern in Figure 3 is generated by the combining operation.

This example shows that the interpolation operation allows the generation of the dependency structure pattern in the case where some occurrences of keywords do not directly modify the other occurrences.

4. Classifying Sentences

Given a query, our system identifies dependency structure patterns for sentences in a sentence collection and classifies the sentences according to the identified dependency structure patterns. If sentences have an identical dependency structure pattern, they are classified as a same class.

As an example, let us consider query (1). For the query and the following sentences, the identical dependency structure pattern shown in Figure 3 is identified:

- These grammars *parse sentences in* a psycholinguistically plausible *fashion*. (=sentence (a))
- Our method can still *parse* large *sentences in* a reasonable *amount* of time.

These sentences are included in a same class.

On the other hand, for the following sentences, the dependency structure pattern shown in Figure 7 is identified:

- We began by *parsing* the *sentences in* the multilingual *corpus*. (=sentence (c))
- We *parse* all the *sentences in* the domain document *collection*.

These sentences are included in the other class.

This example shows that dependency structure pattern identification enables the system to classify sentences according to the relation holding between keywords.

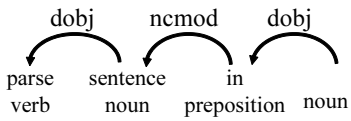


Figure 7. Another example of dependency structure pattern for `parse sentence in -n`.

5. Implementation and Evaluation

We developed an English sentence retrieval system based on the algorithm described in Section 3. The system works on a web server. It has a collection of English sentences which were taken from papers in the area of computer science.

To evaluate our system, we conducted an experiment. We collected queries and judged which sentences are relevant to the queries. By using this test collection, we measured recall and precision of our system.

5.1. Building an English Sentence Collection

We built an English sentence collection by extracting sentences from PDF formatted papers in the area of computer science. The procedure is as follows:

1. Convert PDF formatted papers to XML formatted files by using pdf2html[2]. XML formatted files have the information of text and its layout.
2. Extract paragraphs by using the method in the literature [3], which analyzes the layout of a document.
3. Annotate sentences with dependency structures by using the dependency parser RASP [1].

We obtained 185,488 sentences.

5.2. Test Collection

To evaluate our system, we constructed a test collection, which consists of the following data:

1. Queries
2. Sentences sampled from the sentence collection
3. Relevance judgements

[2] http://pdftohtml.sourceforge.net/

Table 1. Distribution of queries.

query length (word)	# of queries	# of queries for which relevant sentences were sampled
1	42	11
2	128	34
3	95	27
4	26	9
5	11	1

Generally speaking, whether a sentence is relevant to a query depends on the user's need. So the sentence relevances were assessed by subjective judgement.

The subjects made queries and assessed sentence relevances. The relevances are binary. To motivate the subjects to make queries, we gave questions about English composition to the subjects. Because it is impractical to assess the relevances for all sentences in the sentence collection, we sampled about 20 sentences for every query and the subjects judged the relevances. Sentences were sampled through the following procedure:

1. Extract all sentences which include all keywords in the query.
2. Classify the extracted sentences by the sentence classification described in Section 4.
3. Randomly sample sentences from every class. The total of sampled sentences are about 20.

Sampled sentences are shuffled and shown to the subjects. The information about dependency structure were removed from the sentences. This process is automatically executed on a web-based interface. The subjects assess the relevances of these sentences.

To evaluate the influence of the interpolation operation, we set the threshold of cost to 0 or 1 for each query.

The subjects are 7 students in computer science. We obtained a total of 302 queries. The average length of the queries is 2.5 words. Table 1 shows the distribution of the queries. For 68 queries, no sentences were sampled. For 58 queries, the subjects did not judge the relevances of the sampled sentences. For 94 queries, the subjects judged that no sentences were relevant. For 82 queries, the subjects judged that some relevant sentences existed. We measured precision and recall for 71 queries in these queries whose lengths are greater than 1.

5.3. Evaluation

By using the test collection described in the previous section, we measured recall and precision of our system.

Table 2. Precision and recall.

	precision(%)	recall(%)	F-value
baseline1	44.2	100.0	0.613
baseline2	100.0	45.9	0.629
our system	69.0	77.9	0.732

We define recall and precision as the following. Let d_1, ..., d_l be dependency structure patterns and S_{d_i} be a set of sentences for which the dependency structure pattern d_i are identified. Let C be a set of sentences relevant to the query.

Before giving the definition, let us consider an ideal classification. If the classification is ideal, there exists $S^* \in \{S_{d_1}, \ldots, S_{d_l}\}$ such that $S^* = C$. By using precision and recall measures, we judge whether a classification has such a class.

The definition is as follows:

1. For each S_{d_i}, calculate the precision P_{d_i} and the recall R_{d_i} of the class S_{d_i}.
2. Select S_{d_i} which maximizes F-value $\frac{2P_{d_i}R_{d_i}}{P_{d_i}+R_{d_i}}$. We define the precision and the recall of classification $\{S_{d_1}, \ldots, S_{d_l}\}$ as that of S_{d_i}.

P_{d_i} and R_{d_i} are defined as follows:

$$P_{d_i} = \frac{|S_{d_i} \cap C|}{|S_{d_i}|}$$

$$R_{d_i} = \frac{|S_{d_i} \cap C|}{|C|}$$

5.4. Results

Table 2 shows the precision and the recall of our proposed classification. As a baseline for the experiment, we consider the cases where sentences are not classified (baseline1) and where each sentence is in a class. The F-value of our system is higher than the baseline. This result shows the effectiveness of the sentence classification.

5.5. Evaluating the influence of interpolation

We assume that dependency relations exist between the occurrences of keywords in relevant sentences. If the assumption is true, dependency structure patterns with cost 0 must be generated for relevant sentences. To prove the assumption, we investigated the relation between cost and number of relevant sentences. Table 3 shows the result. This result is for 23 queries where dependency pattern identification with cost threshold 1 were executed. The result shows that dependency structure pattern with cost 0 are identified

Table 3. Relation between cost and number of sentences relevant to queries

the cost of generated dependency structure pattern	the number of relevant sentences	total
0	76	193
1	51	353

for many relevant sentences. χ^2-test revealed that there was significant difference ($p < 0.01$) between cost 0 and 1.

On the other hand, for a few of the relevant sentences, dependency structure patterns with cost 1 were generated. This means that the interpolation is useful for processing queries robustly.

6. Conclusion

This paper has presented an English sentence retrieval system based on dependency structure. The system finds sentences in which dependency relations hold between the occurrences of keywords. A subjective experiment demonstrated the effectiveness of the system.

Our proposed dependency structure pattern identification relies on the parsing results of a dependency parser. This means that a misanalysis of the parser may lead to incorrect identification. In order to solve this problem, we will investigate a method to identify dependency structure patterns by utilizing only reliable parsing results.

Acknowledgment

This research was partially supported by the Grant-in-Aid for Scientific Research (B) (No. 20300058) of JSPS.

References

[1] T. Briscoe, J. Carroll, and R. Watson. The second release of the RASP system. In *Proceedings of COLING/ACL 2006 Intaractive Presentation Sessions*, pages 77–80, 2006.

[2] S. Corley, M. Corley, F. Keller, M. W. Crocker, and S. Trewin. Finding syntactic structure in unparsed corpora: The gsearch corpus query system. *Computers and the Humanities*, 35(2):81–94, 2001.

[3] Y. Ishitani. Logical structure analysis of document images based on emergent computation. *IEICE Transactions on Information and Systems*, E88–D(8):1831–1842, 2005.

[4] Y. Kato, S. Matsubara, and Y. Inagaki. A corpus search system utilizing lexical dependency structure. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 2269–2272, 2006.

[5] P. Resnik and A. Elkiss. The linguist's search engine: An overview. In *Proceedings of the 43rd ACL Interacrive Poseter and Demonstration Sessions*, pages 33–36, 2005.

[6] K. Tanaka-Ishii and Y. Ishii. Multilingual phrase-based concordance generation in real-time. *Information Retrieval*, 10(3):275–295, 2007.