PAPER *Special Issue on Text Processing for Information Access*

Effects of Structural Matching and Paraphrasing in Question Answering

Tetsuro TAKAHASHI[†], **Kozo NAWATA**[†], **Kentaro INUI**[†], ***and*** **Yuji MATSUMOTO**[†], ***Nonmembers***

SUMMARY In this paper, we propose an answer seeking algorithm for question answering that integrates structural matching and paraphrasing, and report the results of our empirical evaluation conducted with the aim of examining effects of incorporating those two components. According to the results, the contribution of structural matching and paraphrasing was not so large as expected. Based on error analysis, we conclude that structural matching-based approaches to answer seeking require technologies for (a) coreference resolution, (b) processing of parse forests instead of parse trees, and (c) large-scale acquisition of paraphrase patterns.

key words: *question answering, structural matching, paraphrasing, paraphrase space*

1. Introduction

Question answering is a specific task of language understanding, which may act as a good benchmark to approach *deep* processing toward language understanding. A tempting but probably hasty approach would be to attempt fully conceptual matching between questions and documents. Such a system would derive conceptually represented information from question and target documents and analyze them to infer the answer. Such an approach would, however, entail obvious problems: above all, (a) the overhead of the development and maintenance of the conceptual representation for open-domain natural language documents, and (b) the lack of robustness of state-of-the-art language understanding technologies.

Given this background, it is worthwhile to seek a compromise between fully conceptual and shallow *bag-of-words* matching. A feasible option is structural matching at the level of syntactic structures (or dependency structures). The previously proposed methods being concerned, most of them rely on a scoring function based on bag-of-wards similarity or string matching, whereas one can find only a very limited number of attempts in which structural information is intensively used for matching [7], [8], [11]. Furthermore, even in the latter exceptional attempts, effects of applying structural matching to answer seeking have never been empirically evaluated. Considering this context, in this paper, we discuss the potentialities of structural matching for question answering focusing the following issues.

- For question answering, strict structural matching is not adequate because a given question is unlikely to be structurally identical with a passage that includes an answer candidate (simply *passage*, hereafter). We thus need to seek a method of *soft* matching — more specifically, a method to evaluate the degree of structural similarity that suits the purpose of answer seeking. At the same time, we also need to consider computational overheads because we may need to carry out structural matching hundreds of times to search a single passage for an answer.
- Language contains redundancies. The same piece of information can often be linguistically expressed by more than one expression. For example, the information that '*the name of John F. Kennedy's father is Joseph*' can also be realized by, for example, '*John F. Kennedy, . . ., his father, Joseph P. Kennedy*', '*John F. Kennedy — son of Joseph Patrick Kennedy*', or '*Joseph named his second son John Fitzgerald Kennedy*'. Structural matching may fail to detect the identity between the information conveyed by such paraphrases. The second issue is therefore how to identify diverse paraphrases for answering questions.

For the first issue, we extend Collins's Tree Kernel [1] to formulate a new algorithm for soft structural matching. We present it in Sect. 2. For the second issue, we explore possibilities of incorporating paraphrase generation into question answering. We briefly explain it in Sect. 3. While these two components are both expected to contribute to the approximation of conceptual matching, combining them is also problematic. Addressing this issue, we present an answer seeking algorithm in Sect. 4. Based on the setting described in those sections, we then report our empirical evaluation and discuss the issues we encountered in Sections 5 and 6 focusing on effects of structural matching and paraphrasing in question answering.

2. Soft structural matching

As the basis of our soft structural matching algorithm, we adopted the Tree Kernel method proposed by Collins and Duffy [1] for the following reasons:

- It is designed to quantify the degree of similarity between a given pair of trees, which already partly fits our purpose.
- It detects partial matches of subtrees.
- It is computationally efficient.

To adapt Tree Kernel to question answering, however, further extensions are necessary.

Manuscript received November 30, 2002.
Manuscript revised March 7, 2003.
Final manuscript received May 2, 2003.
[†]Nara Institute of Science and Technology

2.1 Tree Kernel

Collins first defined the inner product between a pair of trees as the number of common subtrees included in both trees. The inner product of two trees thus indicates to which degree they structurally overlap, which can potentially be used as the score of structural matching. Tree Kernel is a computationally efficient method for calculating inner products.

In the Tree Kernel method, a tree T is represented as an n-dimensional vector $\mathbf{h}(T) = \{h_1(T), h_2(T), \ldots, h_n(T)\}$, where the i'th element $h_i(T)$ counts the number of occurrences of the i'th subtree of T. The inner product between two trees is given by

$$\begin{aligned} K(T_1, T_2) &= \langle \mathbf{h}(T_1), \mathbf{h}(T_2) \rangle \\ &= \sum_i h_i(T_1) h_i(T_2). \end{aligned} \quad (1)$$

Note that naive computation of this formula (i.e., summing over the counts of an exponential number of subtrees) would be intractable. The solution proposed by Collins is as follows.

Equation (1) can be transformed to Eq. (2).

$$K(T_1, T_2) = \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} C(n_1, n_2) \quad (2)$$

where N_i is the set of nodes in tree T_i, and $C(n_1, n_2)$ is the number of common subtrees rooted at n_1 and n_2. The $C(n_1, n_2)$ can be calculated recursively as follows:

- If the production (CFG rule) expanding node n_1 is not the same as the production expanding n_2, then $C(n_1, n_2) = 0$.
- If the production expanding n_1 is the same as that of n_2, and n_1 and n_2 are both pre-terminals, then $C(n_1, n_2) = 1$.
- Otherwise,

$$C(n_1, n_2) = \prod_{i=1}^{nc(n_1)} (1 + C(ch(n_1, i), ch(n_2, i))) \quad (3)$$

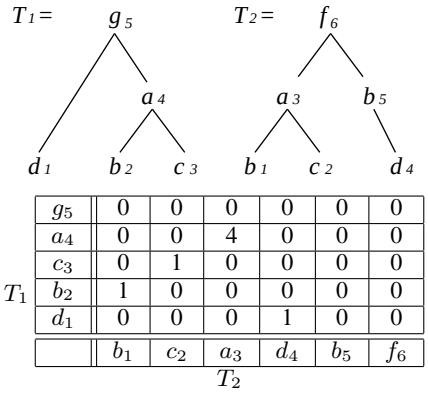
where $nc(n_1)$ is the number of non-terminal children of n_1, and $ch(n_j, i)$ is the i'th child node of n_j.

The $K(T_1, T_2)$ can be calculated in $O(|N_1||N_2|)$ time. Note that this computational order is as small as that for the inner product of simple bag-of-words vectors.

Let us give an example, for the trees in Fig. 1. Alphabetical labels denote node types, while suffix numbers denote token identifiers. The above algorithm fills the table as in Fig. 1 in the left-to-right and bottom-to-top order. The final result $K(T_1, T_2)$ is given by summing up all the counts in the table.

2.2 Adapting Tree Kernel to question answering

Now we extend the original Tree Kernel method to adapt it to structural matching for question answering.



T_1 \ T_2	b_1	c_2	a_3	d_4	b_5	f_6
g_5	0	0	0	0	0	0
a_4	0	0	4	0	0	0
c_3	0	1	0	0	0	0
b_2	1	0	0	0	0	0
d_1	0	0	0	1	0	0

Fig. 1 Node-wise similarity $C(n_1, n_2)$ for $n_1 \in T_1$ and $n_2 \in T_2$

Let us first see an example. Assume we are now trying to match question (4)[†] with passage (5).

(4) 聖火リレーは最初に ⟨PLACE⟩ のオリンピックで行なわれた。
(The Olympic Torch Relay was first introduced in the Olympic Games in ⟨PLACE⟩)

(5) . . . 長野オリンピックの事務局によると聖火リレーはベルリンオリンピックではじめて行なわれた。. . .
(. . . According to the Nagano Olympic secretariat, first Olympic Torch Relay was conducted for the first time at the Berlin Olympic Games. . . .)

(5) has at least two answer candidates: "長野 (Nagano)" and "ベルリン (Berlin)". Since both appear near the keywords, a bag-of-words model may not select *Berlin* correctly. Our aim is to develop a model that chooses correct answers even in such ambiguous cases.

First, we replace Eq. (3) to Eq. (6):

$$C(n_1, n_2) = sim(n_1, n_2) \prod_{k \in ch(n_1)} \prod_{l \in ch(n_2)} (1 + C(k, l)) \quad (6)$$

where $ch(n)$ denotes the set of the children of node n, k and l denote a child node of n_1 and n_2, respectively. $sim(n_1, n_2)$ is a function that gives the degree of similarity between nodes n_1 and n_2 ranging between [0,1]. This extension enhances the flexibility of structural matching in the following sense:

- While the original Tree Kernel applies only to ordered trees, Eq. (6) allows us to treat *unordered* trees, in which the order of siblings is not cared. This enables us to use the dependency tree representation to represent questions and passages as in Fig. 2, which is advantageous in structural matching particularly for free-word-order languages such as Japanese.
- The factor $sim(n_1, n_2)$ enables the incorporation of se-

[†]Here, (4) is assumed to be a sentence obtained by paraphrasing of the original question sentence. ⟨PLACE⟩ denotes a question variable representing the question word "どこ (where)".

mantic similarity measurements between nonidentical words. In Fig. 2, for example, it is reasonable to count the match between "最初に (first)" and "はじめて (for the first time)".

Second, in question answering, a passage that covers larger subparts of a given question is preferable. At the same time, on the other hand, even if an identical common subtree occurs repeatedly in a passage, we want to avoid counting them redundantly. For example, in the above example, the word "オリンピック (Olympic)" may appear again and again in the passage as in Fig. 2. If we added such repeated occurrences to the score, the system would choose longer passages, which is not beneficial. The original function Eq. (2), however, counts such occurrences redundantly. We solve this problem by replacing Eq. (2) with:

$$K(T_1, T_2) = \sum_{n_1 \in N_1} \max_{n_2 \in N_2} C(n_1, n_2), \tag{7}$$

where T_1 is assumed to be a question and T_2 a passage. Equation (7) means that we select the best match with the highest score for each node of a question and only sum up the scores of those matches.

Third, the goal of structural matching is to find the element that corresponds best to the question word, such as *who* or *when*, of a given question. The similarity function Eq. (6) is, however, not very helpful for finding the correct match for a question word, because a question word usually appears in a leaf position, whereas an inter-node similarity score given by Eq. (6) only reflects the similarity between the subtrees rooted at a given pair of nodes. For example, again in Fig. 2, Eq. (6) has no context information for judging whether the question word node ⟨PLACE⟩ matches better with "長野 (Nagano)" or "ベルリン (Berlin)" because ⟨PLACE⟩ is a leaf. To solve the problem, we use Eq. (8), instead of Eq. (6), to seek the best correspondence for a question word. Equation (6) counts the number of the common subtrees rooted at the given nodes, whereas Eq. (8) counts the number of all the common subtrees *including* them.

$$\begin{aligned} &C(n_1, n_2) = C_{bu}(n_1, n_2) \times C_{td}(n_1, n_2) \qquad (8)\\ &C_{bu}(n_1, n_2) = \\ &\ sim(n_1, n_2) \textstyle\prod_{k \in ch(n_1)} \prod_{l \in ch(n_2)} (1 + C_{bu}(k, l)) \\ &C_{td}(n_1, n_2) = \\ &\quad \left(C_{td}(p(n_1), p(n_2)) \times \tfrac{C_{bu}(p(n_1), p(n_2))}{C_{bu}(n_1, n_2)} \right) + 1 \end{aligned}$$

Here $p(n)$ denotes the parent node of n. C_{bu} corresponds to the original C given by Eq. (6), which counts the number of common subtrees rooted at n_1 and n_2 in the bottom up manner. On the other hand, C_{td} counts the number of common subtrees that has n_1 and n_2 as a leaf in the top down manner. For example, going back to Fig. 1, $b_2 \in T_1$ and $b_1 \in T_2$ have three common subtrees, $\{a, b, c\}$ $\{a, b\}$ and $\{b\}$ as a leaf. Thus, $C_{td}(b_2, b_1) = 3$. On the other hand, $b_2 \in T_1$ and $b_5 \in T_2$ have no common subtree except a single node tree $\{b\}$; thus, $C_{td}(b_2, b_5) = 1$. The total number of common subtrees is the combination of subtrees counted in C_{bu} and those in C_{td}, which can be calculated as the product of C_{bu} and C_{td}.

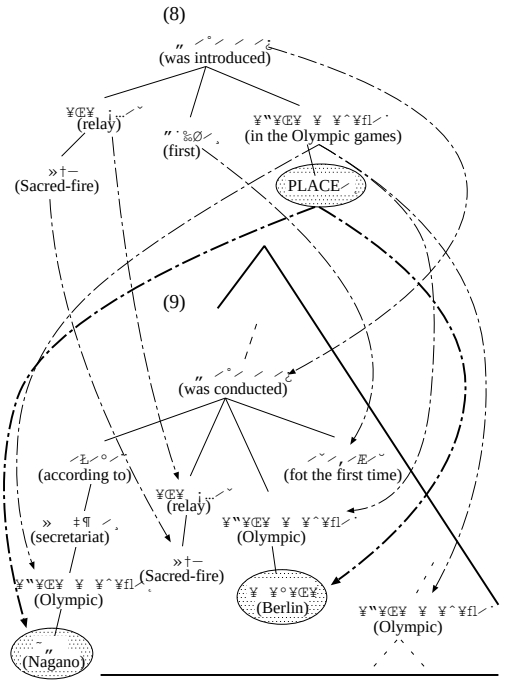


Fig. 2 Dependency trees of (4) and (5)

3. Paraphrasing for question answering

For the paraphrasing process, we use our lexico-structural paraphrasing engine KURA[14]. For a given source sentence, KURA generates possible paraphrases by using a rule-based syntactic transfer.

When we started to develop our question answering system, several types of paraphrasing rules for general purposes had already been implemented on KURA. We added seven types of paraphrasing rules specialized for question answering to it, obtaining a rule set as summarized in Table 1. The rule classes marked with an asterisk are the newly added ones. The second column shows the number of rules of each class.

Examples of paraphrases are as follows.

- Put a relative clause and matrix clause in two separate sentences
  日本の代表的な国語辞典として 知られる 広辞苑の第五版が１１月１１日に刊行される。
  → 広辞苑は日本の代表的な国語辞典として 知られる。広辞苑の第五版が１１月１１日に刊行れる。
- Simplify a functional expression (e.g. "in order to")
  宮崎駿 に対する 評価は高い。
  → 宮崎駿 への 評価は高い。
- Replace a noun with its gloss
  夏目漱石の 遺作 は「明暗」だ。
  → 夏目漱石の 死後に残した未発表の作品 は「明暗」だ。
- Remove a cleft construction from a sentence
  ペルー軍兵士に殺害されたのは PERSON だ。

Table 1 Paraphrasing rules

Rule class	Number
relative clause	8
adverbial clause	18
sahen-verb to *wago*-verb	6642
verb alternation	34
idiomatic phrase	3942
functional expression	261
noun to synonym	3633
cleft sentence	25
* interrogative to declarative	62
* noun to gloss	45565
* appositive	25
* coordination	18
* copula	10
* compound noun	13
* newspaper-specific	29

→ PERSON はペルー軍兵士に殺害された。

- Transform an interrogative form into a declarative form
  夏目漱石の名作は何ですか
  → 夏目漱石の名作は THING だ。
- Decompose an appositive relation into a copula "A is B" and two parallel instances of the sentence (one for A and one for B)
  デビッド・スミス容疑者は、コンピュータウィルス「メリッサ」を作った。
  → デビッド・スミス容疑者は、コンピューターウイルスを作った。デビッド・スミス容疑者は、「メリッサ」を作った。「メリッサ」は、コンピューターウイルス だ。
- Decompose a coordinate structure by repeating the sentence for each member of the coordination
  景気後退などの影響を受け、ハウステンボス、志摩スペイン村 なども入場者が減っている.
  → 景気後退などの影響を受け、ハウステンボス も入場者数が減っている。景気後退などの影響を受け、志摩スペイン村 なども入場者数が減っている。
- Transform "A is B" to "B is A"
  夏目漱石の「坊っちゃん」は 名作 だ。
  → 夏目漱石の 名作 は「坊っちゃん」だ。
- Make the relation in a compound noun explicit by creating a relative clause
  日本製アニメ、続々と米国に上陸。
  → 日本で作られたアニメ、続々と米国に上陸。

4. Answer seeking by applying structural matching and paraphrasing

We use structural matching as an approximation of conceptual matching. Namely, the score of the structural matching of a given question-passage pair is considered as a rough approximation of the likelihood that the node corresponding to the question variable is the correct answer. Obviously, this approximation is unprecise in many cases because structural matching does not take paraphrases into account. To resolve this problem, we propose a straightforward method to combine structural matching with paraphrasing, where paraphrasing is responsible for making structural matching more closely approximate to conceptual matching.

Given question q and a set of passages $\mathcal{P}$ that may include the answer for q (see Sect. 5 for passage retrieval), we call the procedure *SeekAnswerCandidates* (see Fig. 3) to obtain the n-best answer candidates. For each passage p in $\mathcal{P}$, this procedure generates a paraphrase space between q and p to seek better structural matches as illustrated in Fig. 4. Here a paraphrase space is the search space consisting of paraphrases generated from either a question or a passage. Since it can be intractably large, we restrict the paraphrase generation in a greedy search-like manner as described in Fig. 3 (*SearchParaphraseSpace*).

SeekAnswerCandidates$(q, \mathcal{P})$ {
  $\mathcal{A} \leftarrow \{\}$
  for each $p \in \mathcal{P}$ do
    $\mathcal{A} \leftarrow \mathcal{A} \cup SearchParaphraseSpace(q, p)$
  return $ChooseNBest(\mathcal{A})$ }

SearchParaphraseSpace(q, p) {
  $Q, P, A \leftarrow \{\}$
  while not $TerminateCondition(A)$ do {
    $Q \leftarrow Q \cup Paraphrase(q)$
    $P \leftarrow P \cup Paraphrase(p)$
    $(q, p; a) \leftarrow ClosestPair(Q, P)$
    $A \leftarrow A \cup (q, p; a)$ }
  return A }

$ChooseNBest(\mathcal{A})$: a function that returns the n-best answer candidate string according to the structural matching score
$TerminateCondition(A)$: a boolean function that checks if the improvement of the best structural matching score in A is saturated
$Paraphrase(p)$: a function that returns a set of p's paraphrases generated by an application of a single paraphrasing rule
$ClosestPair(Q, P)$: a function that returns the best structural matching pair and the corresponding answer candidate string.

Fig. 3 The answer-seeking algorithm

5. Experiments

5.1 Test data

For empirical evaluation, we used the QAC Formal Run test collection provided at the NTCIR workshop 3 [3]. More specifically, we used a set of question-document pairs consisting of a question and a document that is manually verified to include the correct answer, in order to concentrate our attention on the effects of structural matching and paraphrasing but not on errors in document retrieval. For the 200 questions of the Formal Run test data, we obtained 1706 question-document pairs in total. Hereafter, we call each of those pairs a "*case*". We then removed from the 1706 cases those in which the system failed to retrieve passages and used the remaining 1542 cases as the test data.

5.2 Implementation of three methods

For the purpose of comparison, we implemented three versions of algorithms:

- **The baseline method (BL)**: seeks answers based only on bag-of-words similarity and keyword proximity.
- **The structural matching-based method (SM)**: seeks

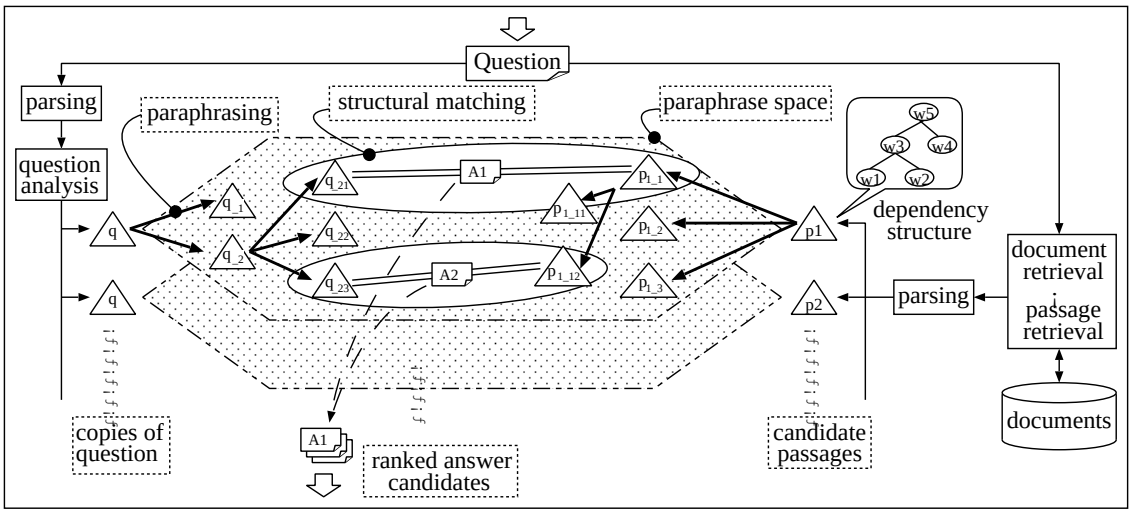


Fig. 4 System overview

answers by applying the structural matching algorithm presented in Sect. 2 but not paraphrasing.

- **The structural matching-based method with paraphrasing (SMP)**: seeks answers by applying both structural matching and paraphrasing as described in Sect. 4.

In the implementation, we developed all the component with the Formal Run test collection kept virtually unseen.

The BL method has three steps. For each case,

1. *Question analysis:* Analyze a given question sentence to extract a set of keywords based on a set of heuristics analogous to those presented in [10]. Also determine the expected answer type by simple pattern matching.
2. *Document Analysis:* Annotate the document with NE tags using an NE chunker [15], and also with semantic category tags in terms of the *Goi-Taikei* Japanese dictionary [6].
3. *Answer seeking:* Let all the noun phrases included in the document be answer candidates, and choose five best-ranked candidates using simple proximity-based heuristics as follows:
   - Prefer a candidate whose semantic category matches the expected answer type.
   - Prefer a candidate whose neighborhood includes more keywords.

The SM method has four steps.

1. *Question analysis:* Same as above.
2. *Document Analysis:* Same as above.
3. *Passage retrieval:* Tokenize passages using a window, and scores all the passages included in the document using proximity-based heuristics analogous to those above.
4. *Answer seeking:* For each of the ten best-ranked passages, apply the structural matching algorithm described in Sect. 2, and score each answer candidate using the scoring function given by Eq. (8).

Table 2 The performance of the three methods

	BL	SM	SMP
MRR	0.58	0.422	0.423
Rank 1	754	591	578
Rank 2	191	86	128
Rank 3	77	23	34
Rank 4	58	10	15
Rank 5	52	2	5
no answer	410	830	782
total	1542	1542	1542

The SMP method is the same as the SM method except that the answer seeking step is done by searching paraphrase spaces as described in Sect. 4.

5.3 Results

The performance of the three methods is summarized in Table 2, where a row labeled "Rank n" gives the number of cases where the correct answer was given in the n-th rank by each method.

6. Analysis

6.1 Effects of applying structural matching

Among the 410 cases where the BL method failed to answer, the SM method found the correct answer in 56 cases. Thus these were cases where the structural matching worked effectively. However, among the 754 cases where the BL method found the correct answer at the first rank, the SM method missed it in 257 cases. So, first of all, we analyzed these 257 cases to see why the incorporation of structural matching degraded the performance. As a result, we found two major reasons; one is that structural information was useless in many cases due to scattering keywords, and the other is that the present structural matching algorithm was

too sensitive to errors of sentence analysis. We elaborate each here.

6.1.1 Scattering keywords

First, structural information turned out to be unhelpful for answer seeking in 224 cases out of the 257 cases. The following case is a typical example.

(9) Q. 〈ＰＥＲＳＯＮ〉は、菅原道真/*Michizane Sugawara*$_{(1)}$と　誕生日/ *birthday*$_{(2)}$ が　同じ/*same*$_{(3)}$ 首相/*prime minister*$_{(4)}$ だ。*Which prime minister was born on the same day as the politician and scholar Michizane Sugawara?*

A. **小渕恵三/*Keizo Obuchi***$_{[\mathrm{answer}]}$首相$_{(4)}$ へ太宰府から「梅の使節」。天満宮の祭神、菅原道真$_{(1)}$は６月２５日生まれと伝えられ、誕生日$_{(2)}$が同じ$_{(3)}$という 首相 $_{(4)}$ は「親しみを感じます」とニッコリ。

Here, in sentence (9A), no explicit dependency relation can be found between the correct answer "小渕恵三/*Keizo Obuchi*" and the two important keywords "菅原道真/*Michizane Sugawara*$_{(1)}$" and "誕生日/*birthday*$_{(2)}$". Nevertheless, a human can find the answer correctly probably because she understands that "誕生日 $_{(2)}$ が同じ $_{(3)}$ / *his birthday is the same*" means his birthday is the same *as Michizane Sugawara*, and that the second appearance of "首相/*prime minister*$_{(4)}$" refers to the answer NP "小渕恵三/*Keizo Obuchi*".

As exemplified by this example, question keywords often appeared in positions syntactically isolate from the answer. Furthermore they tended to be scattered beyond sentence boundaries. In such cases, structural matching has no effectiveness, without deeper analysis of discourse including anaphora and ellipsis resolution. And even those cases, a simple proximity-based scoring method has a chance to find the answer (Remember that the current structural matching algorithm does not take the factor of proximity into account). An easygoing solution would be to integrate a proximity-based scoring function with a structure-based scoring function. We believe, however, that it must be more important to seek a way to incorporate a coreference resolution component, and see the effects. The scope of our future work definitely includes the latter direction.

6.1.2 Fragileness against sentence analysis errors

Second, our error analysis also revealed that the present structural matching algorithm was weak against errors of question analysis and dependency structure analysis.

The present structural matching algorithm requires that a given question sentence tree should include a question variable. The SM method thus inevitably fails in answer seeking if the question analysis process fails to create a question variable. This happened in 19 cases of the above 257 cases. The BL method, on the other hand, still has

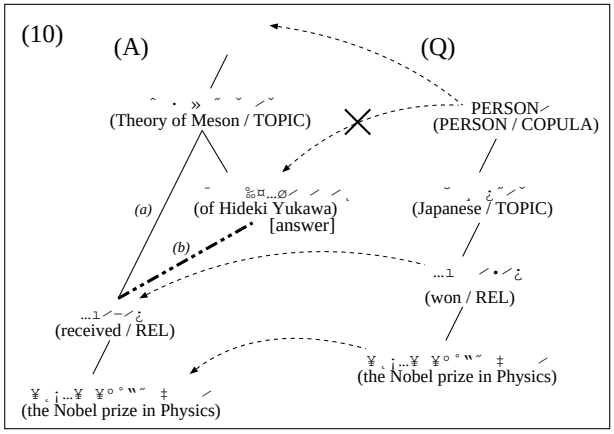


Fig. 5 Example of an essential parsing error

enough chances to find a correct answer even if no information about the expected answer type is available, because it is still able to rely on proximity information.

The other 14 cases of the 257 cases were related to the problem of dependency parsing. A half of the 14 cases can be attributed to parse errors. More importantly, however, the other half of the cases indicate that the current structural matching method is still too strict to be tolerant of *structural arbitrariness*.

Figure 5 shows an example of such a case. In the figure, (10Q) gives a question tree and (10A) gives a portion of a sentence including the answer "湯川秀樹 (Hideki Yukawa)". The figure shows that the VP "受けた (received)" was interpreted as a modifier of "中間子理論は (Theory of Meson/TOPIC)". But it also makes equally good sense to interpret the same VP as a modifier of "湯川秀樹 (Hideki Yukawa)" as indicated by edge (b). In the latter case, the SM method would have successfully found the correspondent of the question variable "PERSON". This sort of arbitrariness appeared innegligibly often, and the BL method tuned out to be much robuster against it.

A possible solution to this problem is to introduce redundancy into parsing, namely to represent multiple plausible parses as a parse forest, which is then given to the answer seeking process. In that case, an important issue would be how to reduce the computational cost of applying structural matching and paraphrasing to a parse forest. This direction will also be in the scope of our future work.

6.1.3 Finding k-best answer candidates

The present structural matching algorithm seeks best-matching correspondence in a deterministic manner. This means that the algorithm outputs only the best answer candidate for a given question-passage pair. In Fig. 4, for example, one obtains the best answer candidate A_1 from the pair of $q_{\text{-}21}$ and $p_{1\text{-}1}$, but not any other candidate from the same pair. Therefore, the SM and SMP methods may not find as many answer candidates as five. In fact, the average number of output candidates for a case was 1.35 in the SM

method and 2.14 in SMP. On the other hand, the BL method always found more than five candidates because it ranked all the noun phrases included in a given passage. This difference is strongly unfavorable for SM and SMP when one compares them with BL according to the MRR measure as shown in Table 2, and should be regarded as one of the drawbacks of the present algorithm for structural matching. If we tried to obtain k-best candidates in a naive method, the computational cost would increase from $O(|N_1||N_2|)$ to $O(k|N_1||N_2|)$. Reducing this computational cost will also be in the scope of our future work.

6.2 Effects of applying paraphrasing

Let us move on to see the effects of paraphrasing. Among the 830 cases that the SM method failed to answer, the SMP method newly found the answer in 58 cases. On the other hand, among the 712 cases where the SM method found the answer, the SMP method missed it in only 10 cases. This indicates that the application of paraphrasing did contribute to answer seeking positively. We must note, however, that the effect was small.

Investigating more deeply why the effect was so small, we found that, among the 1542 cases in hand, the system only generated 4254 paraphrases in total that gained a structural matching score. This means that the paraphrasing component contributed unexpectedly little to the answer-seeking process. The main reasons are as follows:

- Paraphrasing rules were sometimes not applied due to parsing errors. Our syntactic transfer-based algorithm for paraphrasing may have been too sensitive to parsing accuracy.
- If the keywords associated with a question are scattered over different sentences in a given passage, the currently implemented paraphrasing rules are almost helpless. This is because, so far, we have no rule that can aggregate such scattered keywords into a single sentence. Our analysis supports, however, that, if we could have resolved coreferences (including ellipses) beforehand, the results would have changed in many cases.
- The coverage of the implemented paraphrase patterns was still too small for practical use. For example, the current paraphrasing knowledge does not include such a *near-paraphrase* pattern as "$X_{[hum]}$ が $Y_{[movie]}$ を監督する (X directs Y)" → "X が Y で知られる (X is known for Y)". It is interesting to empirically investigate how effectively existing methods for paraphrase acquisition [9], [13] resolve this coverage problem.

The current SMP method has another important drawback, namely the computational overhead of paraphrasing. As stated above, the system generated only 4254 paraphrases for the 1542 questions that gained a structural matching score. To find those effective paraphrases, however, the system generated 74340 paraphrases in total. This means that almost 95% of the paraphrases were generated in vain just for probing search spaces. Clearly, we need a more sophisticated way of controlling paraphrase generation.

7. Related work

The information of syntactic structures is used in several existing question answering systems. Harabagiu et al. [4] use dependency structures to derive logical forms for justification of answers. During the transformation into a logical form, syntactic information is somehow abstracted. In contrast, we directly use structural information without abstracting. Ittycheriah et al. [7], Murata et al. [11], and Kiyota et al. [8] use fragmental information of syntactic structures as a feature of their machine learning approach, while we used entire structures. More importantly, none of them empirically evaluated the effects of the use of structural information, which is the issue we addressed in this paper.

Paraphrasing has also been an issue in the question-answering research community. Among all, perhaps, the search-based answer seeking algorithm we examined in this paper is most similar to the model proposed by Murata and Isahara [12], in which the system is supposed to paraphrase both a question and target document repeatedly so as to maximize the similarity between them. Murata, however, did not report the results of a large-scale experiment using a large set of paraphrase patterns. This is also the same case with other similar work done by, for example, Dumais et al. [2] and Lin and Pantel [9]. In contrast, we empirically examined the reality using a reasonably large set of paraphrase patterns, and revealed several practical problems we should address in future work.

Ravichandran and Hovy [13] propose a method for acquiring paraphrase patterns for question answering from Web text. Following their work, Hermjakob et al. [5] report that using those paraphrase patterns achieved considerable improvements when using the web as information source, but did not work effectively when the information source was limited to a closed document collection. They have not specified the reason, but we guess that they also confronted the same problems we explored in this paper.

8. Conclusion

In this paper, we proposed a way of extending Collins's Tree Kernel to adapt it to question answering, formulating a new algorithm for structural matching. We also proposed a greedy search-based algorithm for answer seeking that integrates structural matching and paraphrasing. We then reported the results of our empirical evaluation that examined the effects of incorporating these two components. According to the results, the contribution of structural matching and paraphrasing was unexpectedly small. Our error analysis revealed that the proposed method encountered several problems to overcome including (a) coreference resolution, (b) processing of parse forests instead of parse trees, and (c) large-scale acquisition of paraphrase patterns. We believe that those problems are not specific to the algorithm at hand, but rather be general problems one has to address whenever

she attempts to apply structural matching and/or paraphrasing to question answering.

Acknowledgments

We thank Hiroyasu Yamada (JAIST) for allowing us to use his NE tagging tool, Masao Utiyama (Communications Research Laboratory) for his IR engine ruby-ir, and the NTT Communication Science Laboratories for their case frame dictionary and thesaurus, which we used for paraphrase generation. We also thank the anonymous reviewers for their suggestions and encouraging comments.

References

[1] M. Collins and N. Duffy, "Convolution kernels for natural language", Proc. Neural Information Processing Systems (NIPS 14), pp. 625–632, 2001.

[2] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng, "Web question answering: Is more always better?", Proc. The 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002), pp. 291–298, 2002.

[3] J. Fukumoto, T. Kato, and F. Masui, "Question answering challenge (qac1): Question answering evaluation at ntcir workshop 3", Proc. Working Notes of the Third NTCIR Workshop Meeting: QAC1, pp. 1–10, 2002.

[4] S. Harabagiu, D. Moldovan, M. Pasca, M. Surdeanu, R. Mihalcea, R. Girju, V. Rus, F. Lactusu, P. Morarescu, and R. Bunescu, "Answering complex, list and context questions with lcc's question-answering server", Proc. The Text REtrieval Conference (TREC), pp. 355–361, 2001.

[5] U. Hermjakob, A. Echibahi, and D. Marcu, "Natural language based reformulation resource and web exploration for question answering", Proc. The Text REtrieval Conference (TREC), 2002.

[6] S. Ikehara, M. Miyazaki, S. Shirai, A. Yoko, H. Nakaiwa, K. Ogura, Y. Ooyama, and Y. Hayashi, ed., *Goi Taikei*-A Japanese Lexicon (in Japanese), Iwanami Publishing, 1997.

[7] A. Ittycheriah, M. Franz, and S. Roukos, "IBM's statistical question answering system–TREC-10", Proc. The Text REtrieval Conference (TREC), pp. 258–264, 2001.

[8] Y. Kiyota, S. Kurohashi, and F. Kido, ""dialog navigator" : A questions answering system based on large text knowledge base", Proc. The 19th International Conference on Computational Linguistics (COLING 2002), pp. 460-466, 2002.

[9] D. Lin and P. Pantel, "Discovery of inference rules for question answering", Proc. Natural Language Engineering, vol.7, no.4, pp. 343–360, 2001.

[10] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Goodrum, R. Girju, and V. Rus, "Lasso: A tool for surfing the answer net", Proc. The Text REtrieval Conference (TREC), pp. 175–184, 1999.

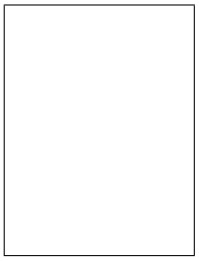
[11] M. Murata, M. Utiyama, and H. Isahara, "Qestion answering system using similarity-guided reasoning (in japanese)", Proc. Information Processing Society of Japan NL-135, pp. 181–188, 2000.

[12] M. Murata and H. Isahara, "Universal model for paraphrasing: Using transformation based on a defined criteria", Proc. The Natural Language Processing Pacific Rim Symposium (NLPRS) Workshop on Automatic Paraphrasing: Theories and Applications, pp. 47–54, 2001.

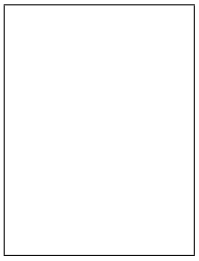
[13] D. Ravichandran and E. Hovy, "Learning surface text patterns for a question answering system", Proc. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 41–47, 2002.

[14] T. Takahashi, T. Iwakura, R. Iida, A. Fujita, and K. Inui, "Kura: A revision-based lexico-structural paraphrasing engine", Proc. The Natural Language Processing Pacific Rim Symposium (NLPRS) Workshop on Automatic Paraphrasing: Theories and Applications, pp. 37–46, 2001.

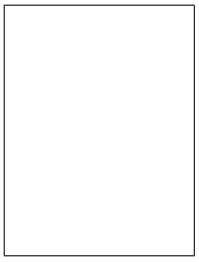
[15] H. Yamada, T. Kudo, and Y. Matsumoto, "Japanese named entity extraction using support vector machine (in japanese)", Proc. Transactions of IPSJ, Vol. 43, No. 1, pp. 44–53, 2002.



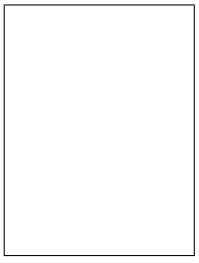
Tetsuro Takahashi He is currently a Student at Nara Institute of Science and Technology. He received his B.E. and M.E. degrees in information science from Kyushu Institute of Technology in 2000 and 2002. His main research interests are question answering, paraphrasing and knowledge acquisition for these techniques.



Kozo Nawata He is currently a Student at Nara Institute of Science and Technology. He received his B.E. degrees in information science from Kyushu Institute of Technology in 2002. His main research interests are question answering, paraphrasing and knowledge acquisition for these techniques.



Kentaro Inui He is currently an associate professor of Information Science at Nara Institute of Science and Technology. He received his doctorate in information engineering from Tokyo Institute of Technology in 1995. He has then experienced a research associate at Tokyo Institute of Technology, an associate professor at Kyushu Institute of Technology, and a research fellowship of Japan Science and Technology Corporation. His main research interests are natural language understanding and generation.



Yuji Matsumoto He is currently a Professor of Information Science at Nara Institute of Science and Technology. He received his M.S. and Ph.D. degrees in information science from Kyoto University in 1979 and in 1989. He joined Machine Inference Section of Electrotechnical Laboratory in 1979. He has then experienced an academic visitor at Imperial College of Science and Technology, a deputy chief of First Laboratory at ICOT, and an associate professor at Kyoto University. His main research interests are natural language understanding and linguistic knowledge acquisition.