Applying Structural Matching and Paraphrasing

TAKAHASHI Tetsuro   NAWATA Kozo   INUI Kentaro
Graduate School of Information Science, Nara Institute of Science and Technology
Takayama, Ikoma, Nara, 630-0101, Japan
{tetsu-ta,kozo-n,inui}@is.aist-nara.ac.jp

KOUDA Shinya
Graduate School of Computer Science and System Engineering, Kyushu Institute of Technology
Iizuka, Fukuoka, 820-8502, JAPAN
s_kouda@pluto.ai.kyutech.ac.jp

Abstract

We discuss the potentialities of structural matching as a compromise between fully-conceptual deep processing approaches and shallow bag-of-words approaches to answer seeking. We extended Collins's Tree Kernel [1] to formulate a new algorithm for soft structural matching, and we also developed and tested a method for combing structural matching with paraphrasing. Unfortunately, we found no positive empirical evidence that either structural matching or paraphrasing contribute to question answering. We believe, however, that their usefulness should not be ruled out yet. We still need more deliberate experiments and analysis.

Keywords: *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 directly 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 context, we need to find a compromise between fully conceptual and shallow bag-of-words matching. A feasible option is structural matching at the level of syntactic structures (dependency structures). 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 almost never 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 realized by more than one language expression. For example, the information that *the name of John F. Kennedy's father is Joseph* can also be expressed by, for example, '*John F. Kennedy, …, his father, Joseph P. Kennedy*', '*John F. Kennedy (1917-1963) — 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 propose to extend Collins's Tree Kernel [1] to formulate a new algorithm for soft structural matching. The computational cost of the algorithm is $O(nm)$ for matching an n-word question and an m-word passage. We present the algorithm in Section 2.

For the second issue, we explore possibilities of incorporating of paraphrase generation into question answering. In our experiments we used our lexico-structural paraphrasing engine KURA[11]. We briefly explain it in Section 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 Section 4.

We finally report the results of an empirical evaluation in Section 5. The performance we have achieved so far is disappointingly poor. We discuss the reasons in Section 6.

2 Soft structural matching

As the basis of our soft structural matching algorithm, we adopted the Tree Kernel method proposed by Collins et al.[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. For further details, see our paper [10].

2.1 Tree Kernel

Collins et al. 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

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

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 et al. is as follows.

(1) can be transformed to (2).

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

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,

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

  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 (4). Alphabetical labels denote node types, while suffix numbers denote token identifiers. The above algorithm fills the table as in Table 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.

(4)

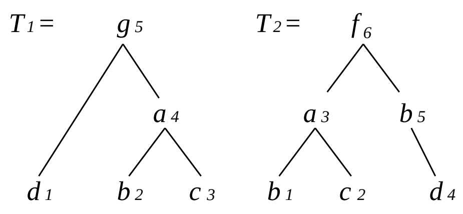

Table 1. Node-wise similarity matrix $C(n_1, n_2)$ for the trees in (4)

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
	b_1	c_2	a_3	d_4	b_5	f_6
			T_2			

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.

Let us first see an example. Assume we are now trying to match question (5) [1] with passage (6) .

(5) 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⟩)

(6) . . . 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, the first Olympic Torch Relay was conducted for the first time at the Berlin Olympic Games.) . . .

(9) has at least two answer candidates: “長野 (Nagano)” and “ベルリン (Berlin)”. While a bag-of-words model may not be able to select *Berlin* as the correct answer, we aim to develop a model that chooses *Berlin* with certainty it even in such an ambiguous case.

First, we modify formula (3) to (7):

$$C(n_1, n_2) = sim(n_1, n_2) \prod_{k \in ch(n_1)} \prod_{l \in ch(n_2)} (1 + C(k, l)) \quad (7)$$

where $ch(n)$ denotes the set of the children of node n, and $sim(n_1, n_2)$ denotes 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, formula (7) 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 Figure 1, 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 semantic similarity measurements between nonidentical words. In Figure 1, 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 Figure 1. If we added such repeated occurrences to the score, the system would choose longer passages, which is not beneficial. The original function (2), however, counts such occurrences redundantly. We solve this problem by replacing (2) with:

$$K(T_1, T_2) = \sum_{n_1 \in N_1} \max_{n_2 \in N_2} C(n_1, n_2). \quad (8)$$

where T_1 is assumed to be a question and T_2 a passage. The (8) 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 (7) 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 (7) only reflects the similarity between the subtrees rooted at a given pair of nodes. For example, again in Figure 1, (7) 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 (9), instead of (3), to seek the best correspondence for a question word. The (3) counts the number of the common subtrees rooted at the given nodes, whereas (9) counts the number of all of the common subtrees *including* them.

$$C(n_1, n_2) = C_{bu}(n_1, n_2) \times C_{td}(n_1, n_2) \quad (9)$$
$$C_{td}(n_1, n_2) = sim(n_1, n_2) \prod_{k \in ch(n_1)} \prod_{l \in ch(n_2)} (1 + C(k, l))$$
$$C_{td}(n_1, n_2) = \left(C_{td}(p(n_1), p(n_2)) \times \frac{C_{bu}(p(n_1), p(n_2))}{C_{bu}(n_1, n_2)} \right) + 1$$

3 Paraphrasing for question answering

For the paraphrasing process, we use our lexico-structural paraphrasing engine KURA[11]. For a given source sentence, KURAgenerates 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 161 paraphrasing rules to this rule set specialized for question answering. The current rule set is summarized in Table 2. The rule classes marked by * are the newly added ones. The second column shows the number of rules of each class, and the third column shows how many times the rules were applied in the 200 questions Formal Run (2000 pairs of questions and candidate passages in total).

- Relative clause
日本の代表的な国語辞典として知られる広辞

[1]Here, (5) is assumed to be a sentence obtained by paraphrasing the original question sentence. ⟨PLACE⟩ denotes a question variable representing the question word “どこ (where)”.

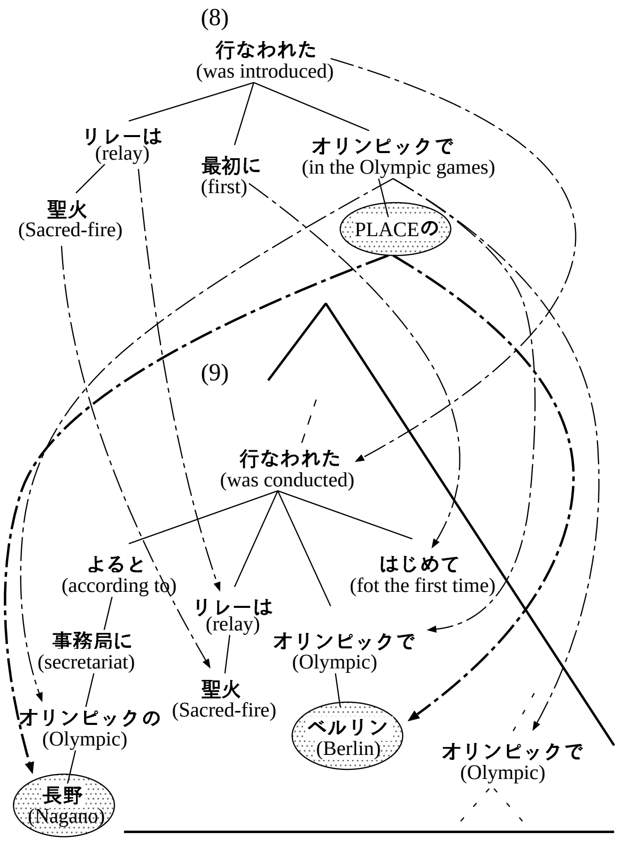

Figure 1. Dependency trees of (5) and (6)

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。広辞苑岩波書店の第五版が１１月１１日に刊行れる。

- Functional compound
  宮崎駿に対する評価は高い。→ 宮崎駿への評価は高い。

- Noun to 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。

- Ideomatic phrase
  道草を食う
  → 途中で他の事に時間を費やす

- Cleft sentence
  ペルー軍兵士に殺害されたのは PERSON だ。
  → PERSON はペルー軍兵士に殺害された。

- Appositive
  デビッド・スミス容疑者 (３０) 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（３０）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（３０）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。

Table 2. Paraphrasing rules

Rule class	Number	Applied
relative clause	8	3265
adverbial clause	18	754
sahen-verb to *wago*-verb	6642	4503
verb alternation	34	439
ideomatic phrase	3942	0
functional compound	261	471
noun to synonym	3633	4725
cleft sentence	25	475
* negative clause	62	3416
* noun to gloss	45565	345
* appositive	25	3133
* coordination	18	46
* copula	10	1162
* compound noun	13	115
* newspaper-specific	29	1351

- Coordination
  景気後退などの影響を受け、ハウステンボス (長崎県)、志摩スペイン村 (三重県)、レオマワールド (香川県) 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（長崎県）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（三重県）なども入場者数が減っている。景気後退などの影響を受け、レオマワールド（香川県）なども入場者数が減っている。

- Copula
  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。

- Compound noun
  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。

- Newspaper-specific
  異国で身を立てた我が子の雄姿に母は涙——第６７代横綱に昇進した。
  → 異国で身を立てた我が子の雄姿に母は涙。第６７代横綱に昇進した。

4 Answer seeking by structural matching and paraphrasing

We use structural matching as an approximation of conceptual matching. Namely, the structural matching score 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 imprecise in many cases because structural matching does not take paraphrases into account.

This section presents one straightforward method we have tested so far to combine structural matching with paraphrasing. In this method, 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 Section 5 for passage generation), we call the procedure *SeekAnswerCandidates* (see Figure 2) 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 Figure 3. 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 Figure 2 (*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.

Figure 2. The answer-seeking algorithm

5 Evaluation

5.1 System overview

In the present system, the overall question-answering process has three steps:

1. **Document retrieval:** The system first submits a set of the keywords contained in a given question to the IR tool [12] to retrieve the 20-best documents.

2. **Passage generation:** The system then summarizes the 20 retrieved documents by simple sentence extraction, and produces a set of passages. In this process, the system first use an named entity tagging tool named NExT [7] to annotate the retrieved documents with NE tags. Then, it ranks all the sentences included in the document set according to factors including the question type, question keywords, NE tags, etc. For each of the 10-best ranked sentences (key sentences) the system then generates a passage by conjoining the key sentence, its two adjacently preceding sentences , and the document title.

3. **Answer seeking:** Finally, the system searches the given the 10-best passages for answer candidates (see Section 4). For sentence parsing, we use the KNP parser[5]. We adopted dependency trees with their nodes the *Bunsetsu*-phrasal units to represent parsed questions and passages. The parameters for structural matching and for the paraphrasing rules were tuned manually by using the Dry Run data.

5.2 Results

In the Formal Run session held on April 22–26, 2002 our system was not able to produce a correct answer to any question. The precision and recall were thus both 0.0%. The primary reason was that the system simply did not work as intended due to implementation problems. In addition, the matching algorithm process was less flexible than described in Section 2; therefore, the system could in many case not even find one single answer candidate.

After submitting the Formal Run results, we fixed several implementation problems and also completed the implementation of the structural matching algorithm. We left the Formal Run data unseen throughout this refinement process. The final results are summarized in Table 3. In the rest of this paper, we discuss why the results attained after the refinement still remained equally poor. Hereafter, we refer to the run after the refinement simply as *the experiment*.

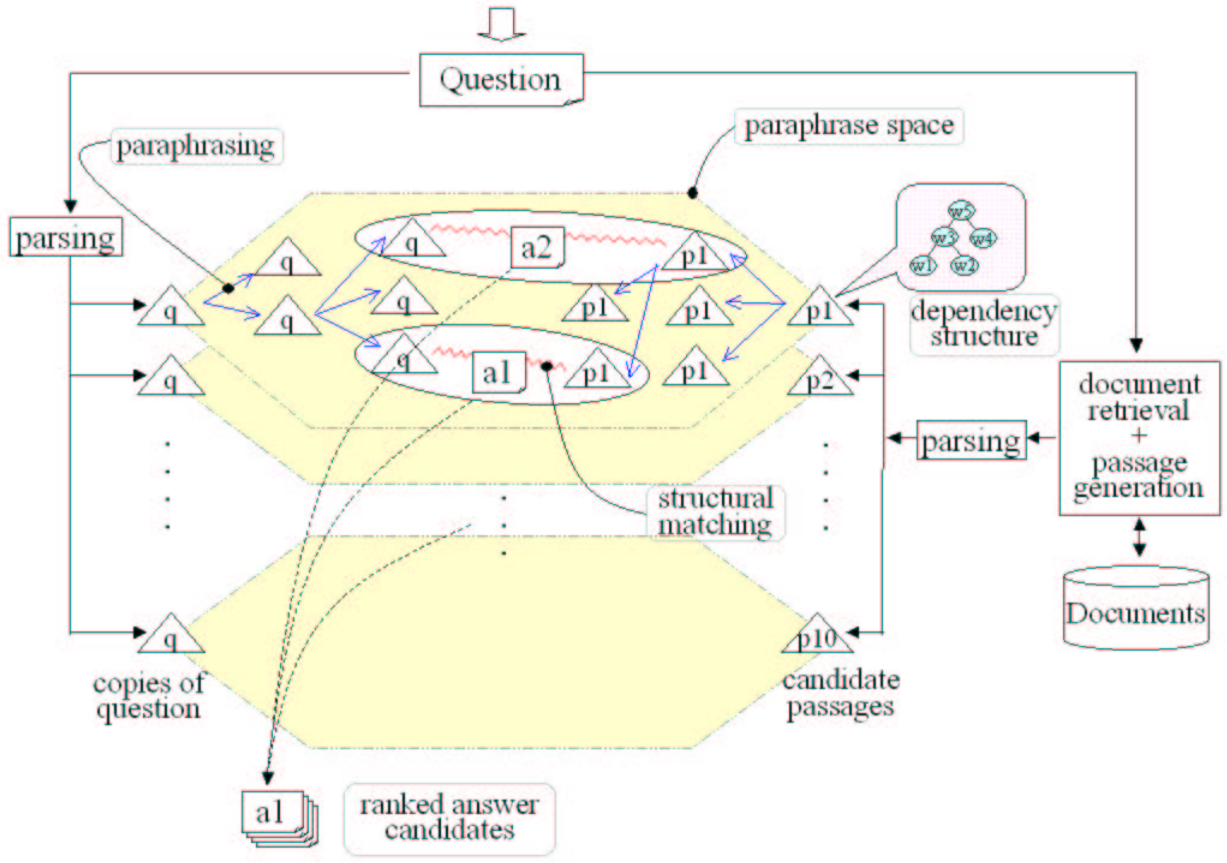

Figure 3. System overview

Table 3. Results for the Formal Run data

	Formal Run (April 26)	Final (August 15)
Correct	0	38
Recall	0.0	0.132
Precision	0.0	0.047
F-measure	0.0	0.070
MMR	0.0	0.116

6 Discussion

The results shown in Table 3 were extremely disappointing. This does not rule out, however, that structural matching or paraphrasing can contribute to question answering. We need to carefully investigate why the contribution was negligibly small. Our error analysis has so far revealed that the current system has problems both inside and outside the process of structure-based answer seeking.

6.1 Problems surrounding answer seeking

A major problem outside the structure-based answer seeking component is that the fundamental components for passage extraction, such as NE tagging and coreference generation, are too naive and fragile to bring out the potentialities of structural matching and paraphrasing. It was the case partly because we concentrated too much on developing the answer seeking procedure, including structural matching and paraphrasing, while leaving the other components almost untuned. Tuning is only a part of the problem, however.

Passage generation

One essential problem lies in passage generation. We only achieved a 55% precision for the 10 generation of best passages for the 200 Formal Run questions. Namely, there were only 110 questions where the 10-best passages returned by the passage generation module actually included at least one correct answer string. Furthermore, even in such a correct passage , the evidential information that is needed for answering was sometimes dropped. This was the case in 30 questions out of the above 110 questions. The precision of passage generation in the strict sense was merely 40%. Given that the precision of 10-best-*document retrieval* was 85% (89% for 20-best), the deterioration caused by the passage generation was serious.

This problem is closely related to the computational costs of structure-based answer seeking. As mentioned earlier, the reason we summarize documents before answer seeking is that structure-based answer seeking could be prohibitively costly if entire documents need to be searched.

Of course, the passage generation algorithm itself can be improved greatly because the current one is so naive. For example, exploring the possibilities of controlling sentence extraction interactively with answer seeking should be promising. We also need to reduce the computational costs of structure-based answer seeking (see Section 6.2).

NE tagging

Fortunately, since the NE tagging tool NExT was available for use, we did not have to spend time developing one ourselves. We have so far used NExT without any significant tuning or extension. However, since our structural matching algorithm relies heavily, possibly too heavily, on NE tags to find the bindings of question variables, the overall performance strongly depends on the NE class taxonomy and classification (tagging) performance.

In the experiment, out of the 2000 passages returned by the passage generation module for the 200 questions, 201 passages actually included an answer. Out of the 201 passages, the answer seeking procedure only found a correct answer for 44 passages. For the remaining 157 passages, 94 cases were related to NE-tagging errors.

6.2 Problems in answer seeking

Contribution of paraphrasing

For the above-mentioned 110 questions (999 candidate passages in total), where the 10-best passages included at least one occurrence of the answer string, the system only generated 128 paraphrases in total that gained a structural matching score (0.13 paraphrases per passage). This means that, in the experiment, the paraphrasing component contributed unexpectedly little to the answer seeking process. The main reasons are as follows:

- The paraphrasing rules were often not applied as a result of parsing errors. Our syntactic transfer-based algorithm for paraphrasing may be too sensitive to parsing performance.
- 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. If we could have applied a sophisticated coreference resolution (including ellipsis) beforehand, the results might have changed. How heavily we can rely on a coreference resolution to solve the problem remains unclear, however.

Contribution of structural matching

Given the current low performance, quantitatively evaluating the contribution of structural matching is not easy. Out of the above 44 passages where the answer seeking procedure was successful, the system could have found an answer for 41 passages, even if no structural information had been taken into account. Therefore, the contribution of structural matching was also very limited. The reason seems to be the same as the problem of paraphrasing, structural matching tends to function similarly to bag-of-words matching when keywords are scattered over different sentences.

Computational cost

Another obvious and serious drawback of our answer seeking method is the computational overhead of paraphrasing. As stated above, the system generated 128 paraphrases for the 110 questions that gained structural matching score. This number may seem negligibly small. To find the effective paraphrases, however, the system had to generate 12037 paraphrases in total. This means that almost 99% of the paraphrases were generated in vain just for probing search spaces. Clearly, we need a more sophisticated way of controlling paraphrase generation. Some goal-oriented mechanisms for paraphrasing-rule selection are worth considering.

6.3 Trainability

Parameter tuning

The behavior of the answer seeking process may significantly depend on the parameter settings of the structural matching. All the parameters have so far been set heuristically. We are in the process of developing a mechanism to tune them automatically by using question answering training data.

Acquisition of paraphrasing patterns

We may need a much larger number of paraphrasing rules to identify more diverse paraphrases. For this purpose, we also need to explore a way of acquiring paraphrasing patterns by aligning questions and answer passages.

7 Related work

The idea of using paraphrasing for question answering has been proposed [8, 2]. While their paraphrasing was done in surface level with small rule sets, we conducted paraphrasing in syntactic level with the large set of rules.

The information of syntactic structures is used in several existing question answering systems. Harabagiu et al.[3], for example, use dependency structures to derive logical forms which are used to justify 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.[4] use fragmental information of syntactic structures as a feature of their machine learning approach, while we see entire structures. Murata et al.[9] also used the dependency structure in Japanese to calculate the similarity between a question sentence and sentences which are expected to include the answer. They concerned binomial relationship between the nodes in a dependency tree in there calculation.

In comparison with those studies, our approach may be too rigid and less robust. Given the results, we need to seek more robust ways of both structural matching and paraphrasing.

8 Conclusion

In this paper, we proposed a method for combining structural matching and paraphrasing to realize structure-based answer seeking. So far we have found no positive evidence that either structural matching or paraphrasing contribute to question answering. We believe, however, that it is still too early to rule out the usefulness of the method. We need more deliberate experiments and analysis.

Acknowledgements

We thank Fumito Masui, Shinya Suzuki (Mie University) and Jun'ichi Fukumoto (Ritsumeikan University) for allowing us to use their NeXT NE tagging tool, and Masao Utiyama (Communications Research Laboratory) for his IR engine ruby-ir. We also thank Takaki Makino, Jun'ichi Tsujii and their colleagues at Tokyo University for helping us to use their typed-feature unification system, LiLFeS [6], for implementing our Q/A system and KURA, Sadao Kurohashi (Tokyo University) and Makoto Nagao (Kyoto University) for KNP, and the NTT Communication Science Laboratories for their case frame dictionary and thesaurus, which we used for paraphrase generation.

Finally, we thank Atsushi Fujita, Iida Ryu (Nara Advanced Institute of Technology) and Tomoya Iwakura (Kyushu Institute of Technology) for helping us use KURA.

References

[1] M. Collins and N. Duffy. Convolution kernels for natural language. In *Neural Information Processing Systems (NIPS)*, 2001.

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

[3] 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. In *the 2001 edition of the Text REtrieval Conference (TREC)*, 2001.

[4] A. Ittycheriah, M. Franz, and S. Roukos. Ibm's statistical question answering system–trec-10. In *the 2001 edition of the Text REtrieval Conference (TREC)*, page 258, 2001.

[5] S. Kurohashi and M. Nagao. Kn parser : Japanese dependency/case structure analyzer. In *the International Workshop on Sharable Natural Language Resources*, pages 48–55, 1994.

[6] T. Makino, K. Torisawa, and J. Tsujii. Lilfes—practical unification-based programming system for typed feature structures. In *the Natural Language Processing Pacific Rim Symposium (NLPRS)*, pages 239–244, 1997.

[7] F. Masui, S. Suzuki, and J. Fukumoto. テキスト処理のための固有表現抽出ツール *next* の開 発 (in japanese. In **言語処理学会 第 *8* 回年次大会 発表論文集**, page 176, 2002.

[8] M. Murata and H. Isahara. Universal model for paraphrasing: Using transformation based on a defined criteria. In *NLPRS'2001 Workshop on Automatic Paraphrasing: Theories and Applications*, 2001.

[9] M. Murata, M. Utiyama, and H. Isahara. Qestion answering system using similarity-guided reasoning. *Information Processing Society of Japan NL-135*, pages 181–188, 2000.

[10] T. Takahashi, K. Inui, and Y. Matsumoto. Methods for estimating syntactic similarity (in japanese). In *Information Processing Society of Japan NL-150*, pages 163–170, 2002.

[11] T. Takahashi, T. Iwakura, R. Iida, A. Fujita, and K. Inui. Kura: A revision-based lexico-structural paraphrasing engine. In *the Natural Language Processing Pacific Rim Symposium (NLPRS-2001) Workshop on Automatic Paraphrasing: Theories and Applications*, 2001.

[12] M. Utiyama and H. Isahara. Tools for exploring natural language. In *the Natural Language Processing Pacific Rim Symposium (NLPRS-2001)*, pages 779–780, 2001.