Question and Answering System based on Predicate-Argument Matching

Daisuke Kawahara Nobuhiro Kaji Sadao Kurohashi
Graduate School of Information Science and Technology, University of Tokyo
7-3-1 Hongo Bunkyo-ku, Tokyo, 113-8656, Japan
{kawahara, kaji, kuro}@kc.t.u-tokyo.ac.jp

Abstract

This paper presents a Question and Answering system based on predicate-argument matching. Predicate-argument structures are utilized to capture more semantics in questions than a set of keywords. In addition to the predicate-argument matching, relation type matching is also used to handle expressions which are not mediated by verbs.

Keywords: *predicate-argument structure, case analysis, relation type.*

1 Introduction

"Question and Answering" (Q&A) is a task to obtain appropriate answers for given domain independent questions written in natural language from a large document collection. Many of existing Q&A systems are based on information retrieval techniques, that is, keywords techniques. These systems cannot capture semantics in questions, such as relations between words.

Our system is based mainly on matching of **predicate-argument structures**. A predicate-argument structure describes what kinds of nouns a verb is related to. Since this is a unit of an event or an action and captures more semantics than keywords, it is promising that answers can be extracted more precisely. This structure is like a logic form used by Harabagiu et al. [1]. To extract predicate-argument structures from sentences, we employ case and ellipsis analysis based on a large case frame dictionary constructed automatically [2].

In addition to predicate-argument structures, we use **relation types** which describe relations between nouns. These are used to handle expressions which are not mediated by verbs, namely cannot be represented as predicate-argument structures. For instance, a relation type is used to deal with "*A no imouto B wa* · · ·" 'B, A's sister, · · ·', which cannot be transformed into a predicate-argument structure.

The outline of our system is illustrated in Figure 1. **Answer type extractor** recognizes what kind of

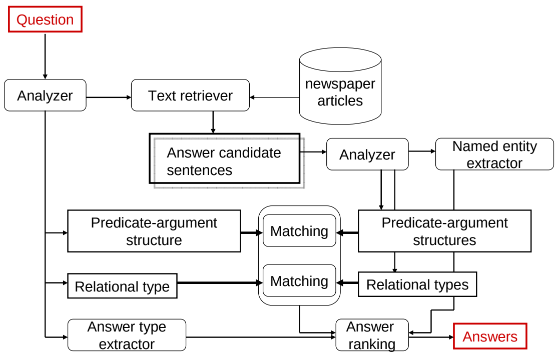

Figure 1. The outline of our system

answer a question requires. **Text retriever** returns sentences which match a question. **Named entity extractor** recognizes named entities in the retrieved sentences. **Answer ranking** module orders answers which are output by matching of predicate-argument structures and relation types.

2 Answer type extractor

An answer type means what kind of answer a question requires. This is extracted from an interrogative expression of a question. For example, if there is an interrogative pronoun *dare* 'who' in a question, an answer type of this question is decided as "person". These correspondences are written by hand. We handle the following six answer types: person, location, organization, time, number, and thing. Examples of them are shown in Table 1. When interrogative expressions are"*nan-nin*" 'how many persons' or "*nan-nen*" 'what year', suffixes of answers such as *nin* 'person' or *nen* 'year' are also indicated. These information is utilized for answer ranking as described in Section 7.

Table 1. Examples of answer type

interrogative expression	answer type
dare 'who'	person
doko 'where'	location, organization
itsu 'when'	time
donokurai 'how much'	number
nan 'what'	thing
nan-nin 'how many persons'	person (suffix=*nin*)
nan-nen 'what year'	time (suffix=*nen*)

3 Text retriever

We employ a text retriever which is utilized by *Dialog Navigator* [3]. The text retriever accepts a question, then returns articles in order of matching score between the question and each article. This score becomes high when an article preserves syntactic dependencies in the question. For instance, when there is a dependency "*teiri wo* ··· *toku*" 'resolve ··· the theorem' in a question, an article that has the same dependency is ranked higher than an article that has both *teiri* 'theorem' and *toku* 'resolve' separately. The maximum number of returned articles is set to 20 because of temporal restriction. This threshold was determined by a preliminary experiment not to deteriorate the accuracy of the system.

From returned articles, our text retriever extracts paragraphs which include one of the content words in the question. We call each sentence of these paragraphs an **answer candidate sentence**.

4 Named entity extractor

We apply named entity extraction to answer candidate sentences. Named entities in each sentence are classified into person, location, organization, time or number. Our named entity extractor is based on a handmade suffix dictionary in which there are rules of correspondence between suffixes and named entity classes. For example, "··· *daitoryo*" '··· president' is recognized as person, and "··· *kawa*" '··· river' is recognized as location.

Named entity information is used in the rules of relation types as shown in Section 6, and in the matching with answer types in the answer ranking described in Section 7.

5 Predicate-argument structures

This section describes how to extract predicate-argument structures of a question and its answer candidate sentences, and explains the matching method of them.

5.1 Extraction of predicate-argument structures

We apply case and ellipsis analysis[2] to a question and its answer candidate sentences to extract predicate-argument structures. Let us consider the following answer candidate sentence.

Fermat *no saisyu teiri wo toita nowa* Andrew Wiles *da*. (1)
(Andrew Wiles resolved the Fermat's last theorem.)

The result of case and ellipsis analysis to this sentence is the following predicate-argument structure of *toku* 'resolve'.

Andrew Wiles:*ga* *teiri*:*wo* *toku* (2)
(Andrew Wiles:nom theorem:acc resolve)

If there are more than one verb in a answer candidate sentence, predicate-argument structures are extracted from all these verbs without modal verbs.

There are many sentences that express the same meaning. For example, the following sentences have the same meaning as the above sentence.

Fermat *no saisyu teiri wa* Andrew Wiles *ni tokareta*. (passive form of (1))

Fermat *no saisyu teiri wo toita jinbutsu wa* Andrew Wiles *da*.

Andrew Wiles *ga* Fermat *no saisyu teiri wo toita*.

(Andrew Wiles resolved the Fermat's last theorem.)

Each *toku* in these sentences has the same predicate-argument structure shown in (2), because our case and ellipsis analyzer normalizes these expressions.

In case of a question, predicate-argument structures can be extracted similarly.

Fermat *no saisyu teiri wo toita nowa dare desuka*. (3)
(Who resolved the Fermat's last theorem?)

The result of case and ellipsis analysis to this sentence is the following predicate-argument structure of *toku* 'resolve'.

X:*ga* *teiri*:*wo* *toku* (4)
(X:nom theorem:acc resolve)

An interrogative pronoun in a question is replaced by "X". In this question, *dare* 'who' is replaced by "X". "X" is a wildcard, which corresponds with any expressions in the matching of predicate-argument structures. If there are more than one verb in a question, only the final verb without modal verbs is handled.

Table 2. Examples of relation types

example	relation type
Natsume Souseki "*Sanshiro*"	"*Natsume Souseki*":author *Sanshiro*:writing
A *no hahaoya*, B	A:child B:mother
America *daitoryo no* Clinton	America:organization *daitoryo*:position Clinton:person
Bill Clinton (52)	"Bill Clinton":person 52:age
Tokyo daigaku (*Bunkyo-ku*)	"*Tokyo daigaku*":organization *Bunkyo-ku*:location
uridashi kakaku no 370 yen	"*uridashi kakaku*":price name "370 yen":price
Conbini, Lawson	*Conbini*:appositive Lawson:appositive
muteiden dengensouchi (UPS)	"*muteiden dengensouchi*":full name UPS:abbreviation

5.2 Matching of predicate-argument structures

We match a predicate-argument structure of a question with those of answer candidate sentences. Answers are extracted from the pairs of predicate-argument structures which have at least one same case component. "X", replaced by an interrogative pronoun, can be correspond to any case component whose case marker is the same as that of "X". Answers are extracted from the case components corresponding with "X".

When answering the question (3), the pair of (2) and (4) has the same verb *toku* 'resolve' and the same case component *teiri*:*wo* 'theorem:acc', and "Andrew Wiles:*ga*" corresponds with "X:*ga*". Consequently, "Andrew Wiles" is extracted as an answer.

If the verb of (1) is *syoumei-suru* 'prove', the following predicate-argument structure is acquired.

Andrew Wiles:*ga* *teiri*:*wo* *syoumei-suru* (5)
(Andrew Wiles:nom theorem:acc prove)

In this case, "Andrew Wiles" can be an answer, because (4) and (5) have the same case component *teiri*, though their verbs are different.

6 Relation types

This section describes how to extract relation types of a question and its answer candidate sentences, then shows the matching method of them.

6.1 Extraction of relation types

We apply pattern matching approach to a question and its answer candidate sentences to extract relation types. We made 120 matching patterns by hand using a thesaurus and named entity classes. Examples of relation types are shown in Table 2.

Let us consider the following answer candidate sentence.

$\cdots$ *niwa, Natsume Souseki "Sanshiro" ga aru.* (6)
(There is *Natsume Souseki* "*Sanshiro*" ...)

The pattern matching produces the following relation type.

Natsume Souseki:author *Sanshiro*:writing (7)

We call the components that constitute relation types, such as "*Natsume Souseki*:author", **relation components**.

The pattern matched by the above example is as follows.

$\underline{<\text{person}>}_1$ "$\underline{.*}_2$" $\rightarrow$ 1:author 2:writing

where <person>is a named entity class, and .* matches any expressions.

We can extract a relation type from a question similarly.

Natsume Souseki no meisaku wa nan desuka. (8)
(What are the masterpieces of *Natsume Souseki*?)

From this question, the following relation type is extracted.

Natsume Souseki:author X:writing (9)

In the same way as the predicate-argument structures, an interrogative pronoun is replaced by "X". In this sentence, "*nan*" 'what' is replaced by "X". The pattern matched by this example is as follows.

$\underline{<\text{person}>}_1$ *no* [work] *wa* ... $\underline{.*}_2$ *da*.
$\rightarrow$ 1:author 2:writing

where [work] means a semantic node in a thesaurus.

6.2 Matching of relation types

We match a relation type of a question with those of answer candidate sentences. Answers are extracted from the pairs of relation types which have more than one same relation component which does not include "X". "X" is handled in the same way as matching of predicate-argument structures. Answers are extracted from the relation components corresponding with "X".

When answering the question (8), the pairs of (7) and (9) have the same relation component "*Natsume Souseki*:author", and "*Sanshiro*:writing" corresponds with "X:writing". Consequently, "*Sanshiro*" is extracted as an answer.

7 Answer ranking

Matching of predicate-argument structures and relation types produces several answers. It is necessary to filter and order these answers to make the final answers for each task of Question and Answering Challenge.

At first, we check the consistency between an answer type and named entity classes of answers. For instance, if an answer type is <person>, answers which are not tagged as <person> by the named entity extractor are filtered out. Only if an answer type is <thing>, the condition is different, and answers tagged as <person>, <time>, or <number> are filtered out.

Secondly, answers are ordered by the number of corresponding case and relation components in the matching. The more the number of corresponding components are, the higher the answer is ranked. When verbs are different in the matching of predicate-argument structures, an answer extracted from this matching is ranked lower.

Note that filtered out answers are added again to the end of the answers of only task 1, because they do not deteriorate the score of task 1.

8 Result of the formal run and discussion

Our system participated in task 1 and task 2 of the formal run of Question and Answering Challenge. The score of task 1 was 0.155 in MRR, and the score of task 2 was 0.123 in average of F-measure (AFM).

Our scores are not so good. This is because the kinds of relation types and the patterns of each relation type are not sufficient. We should elaborate these by investigating more Q&A examples.

On the other hand, the accuracy of the matching of predicate-argument structures is not bad. One reason is that the text retriever can retrieve sentences including answers very precisely. However, there is still room for improvement. For example, the following question could not be resolved by our system.

Mos Burger *wo sougyou-shita no wa dare desuka.* (10)
(Who founded Mos Burger, the hamburger shop?)

From this question, the following predicate-argument structure was extracted.

X:*ga* Mos Burger:*wo* *sougyou-suru* (11)
(X:nom Mos Burger:acc begin)

The following answer candidate sentence, which has the answer (*Sakurada Satoshi*), could be retrieved.

Mos Burger *no sougyou-sya de, 60 sai de nakunatta Sakurada Satoshi* · · · (12)
(*Sakurada Satoshi*, who is the beginner of Mos Burger and died when he was 60 years old, · · ·)

From this sentence, no predicate-argument structure of *sougyou-suru* 'begin' was extracted, because *sougyou* is a part of the noun phrase "sougyou-sya" 'beginner'. We must extend our system to handle such nominals as verbs.

9 Conclusion

In this paper, we presented a Q&A system based on predicate-argument matching. In addition to the predicate-argument matching, relation type matching is also utilized. Our system participated in Question and Answering Challenge, and our scores were not so good. For future work, we will investigate more Q&A examples and elaborate the patterns of relation types, and refine and extend the matching of predicate-argument structures.

References

[1] S. M. Harabagiu, M. A. Pasca, and S. J. Maiorano. Experiments with open-domain textual question answering. In *Proceedings of the 18th International Conference on Computational Linguistics*, 2000.

[2] D. Kawahara and S. Kurohashi. Fertilization of case frame dictionary for robust Japanese case analysis. In *Proceedings of the 19th International Conference on Computational Linguistics*, 2002.

[3] Y. Kiyota, S. Kurohashi, and F. Kido. "Dialog Navigator": A question answering system based on large text knowledge base. In *Proceedings of the 19th International Conference on Computational Linguistics*, 2002.