JAVELIN III: Answering Non-Factoid Questions in Japanese

Hideki Shima and Teruko Mitamura

Language Technologies Institute
School of Computer Science
Carnegie Mellon University

Abstract

In this paper, we describe our adaptation of the JAVELIN system to Japanese question answering for the NTCIR-6 QAC track. To establish a baseline Japanese-to-Japanese non-factoid question answering system, we performed the minimum extensions to our factoid question answering system. The answer boundary recognition task was simplified by introducing a "One Sentence Assumption" so that answer extraction task only deals with ranking of answer candidates in one sentence level. In the end, the performance of our machine learning-based sub-modules was affected by a scarcity of training data; nevertheless, our system performed close to the average accuracy in the number of questions correctly answered.

Keywords: *Complex Question Answering, Information Retrieval.*

1. Introduction

JAVELIN is a modular question-answering architecture[1] originally implemented as an English monolingual system. The architecture is language-independent, and we have been extending the system to for multilingual use[3]. In preparing for the NTCIR-6 QAC-4 task, we made further extensions to JAVELIN so that it can accept Japanese questions and return longer (non-factoid) answers. The questions for QAC-4 included non-factoid questions, such as *why*-type questions, definition questions, and *how* questions.

The JAVELIN system is composed of four main modules: Question Analyzer (QA), Retrieval Strategist (RS), Information eXtractor (IX) and Answer Generator (AG). The QA module is responsible for parsing the input question, choosing the appropriate answer type, and producing a set of keyterms where alternate forms are expanded. Subsequently, the RS module formulates a query and finds relevant documents. The IX module produces a ranked list of answer candidates from the relevant documents. The AG module is responsible for merging similar candidate answers. The overall architecture and data flow is illustrated in Figure 1.

Our goal was to make the minimum necessary extensions to our baseline architecture for factoid QA. The Javelin QAC-4 system was extended from the Japanese-to-Japanese monolingual factoid system (used for CLQA-2) in the following ways:

- Questions: There were 30 questions for training, and 100 questions for the formal run. The expected answers are not factoids.
- QA: The set of answer types does not include named entities.
- RS: The corpus (Mainichi 1998-2001) contains an additional 2 years of material.
- IX: The answer boundaries go beyond the phrasal level. We used task-specific features to train the extractor.
- AG: Answer validation is not performed; the AG module merges duplicate answers only.

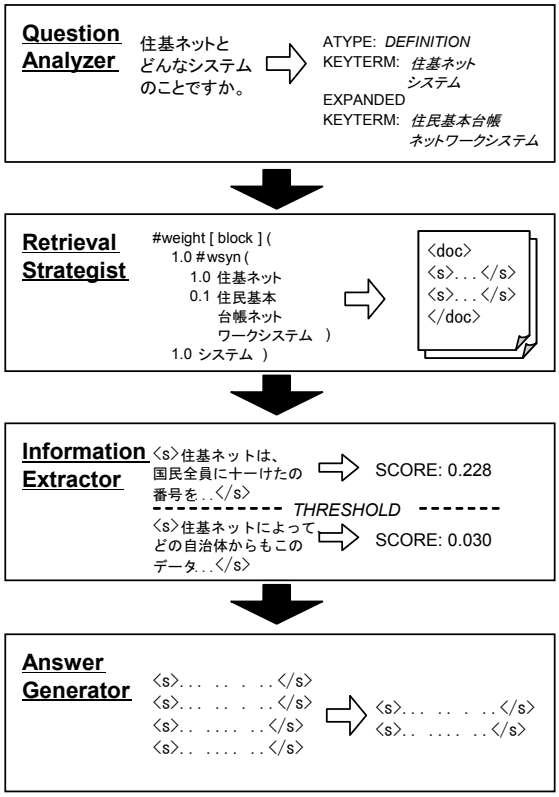

Figure 1. Run time data flow in JAVELIN for non-factoid QA

2. Corpus Preprocessing

We preprocessed the corpus in order to annotate it with useful information for the RS and IX modules. To support multiple overlapping annotations over the documents, we used the UIMA framework[9] and Annotations Database (ADB)[1], which support stand-off document annotation.

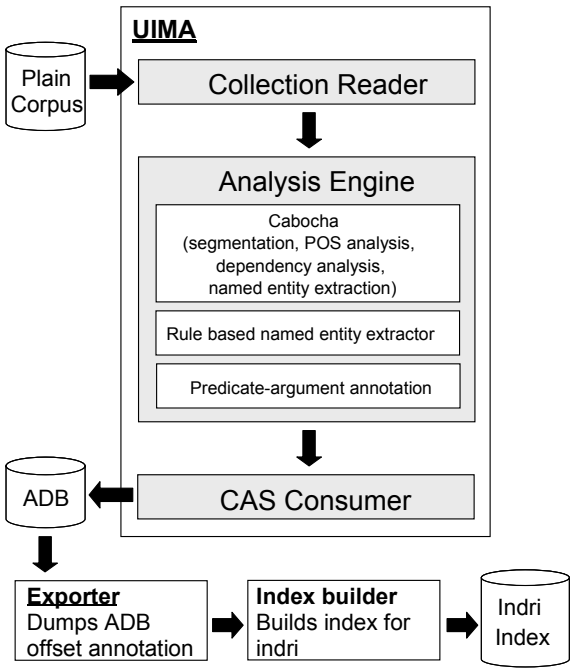

Figure 2. Batch time corpus preprocessing in Javelin III

Figure 2 illustrates the data flow during preprocessing. A UIMA Collection Reader first reads plain texts from the corpus, and passes the data to the Analysis Engine where multiple NLP analyses are performed. The major part of the analysis is done by the CaboCha[8] tool and the MeCab[5] tool, provide morpheme-level text chunks along with parts-of-speech, named entities, and non-projective dependencies among *bunsetsu*[2]. To add additional granularity in named-entity recognition, we run a rule-based named entity extractor. Simple predicate-argument analysis is performed using the dependency structure provided by CaboCha. The annotations are processed by the CAS Consumer, which stores the annotations in the ADB. The IX module accesses the ADB to retrieve the corpus analysis at run time. Finally, keywords and annotated metadata are indexed for run-time use by the Indri search engine.

3. Question Analyzer

The Question Analyzer is responsible for extracting a representation of the user's information need from the input question. The resulting analysis is used by all the subsequent modules, and thus high accuracy, especially in key term extraction, is indispensable. For the QAC-4 track, we developed a module to handle Japanese non-factoid questions for the first time.

The following subsections describe the key term extraction and answer type classification approaches used in question analysis.

3.1. Key Term Extraction and Expansion

We make use of morphological information produced by MeCab[5] to extract noun phrases and verbs. We extract terms in different overlapping spans; in other words, we entire Noun-Noun compounds as well as each noun individually.

We assigned heuristic scores to extracted key terms. Quoted text, Named Entities and Noun phrases consisting of Noun-Noun compounds have the highest score of 1. Verbs and Nouns including those decomposed from Noun phrases are assigned a score of 0.5.

We performed a careful analysis on the 30 training questions and found that there can be some vocabulary mismatches between questions and the corpus documents. Examples are shown in Table 1.

Table 1. Examples from training questions where term expansion may help

Original Term	Alternate Forms
住基ネット	住民基本台帳ネットワークシステム
五輪	オリンピック
原発	原子力発電所

To deal with the vocabulary mismatch problem, we expand terms and use the WSYN (weighted synonym operator) to retrieve documents in the RS module. To expand terms, synonym/alias dictionaries are created from Wikipedia and Eijiro [3] (English-to-Japanese dictionary).

From Wikipedia[4], we extracted 184,166 redirections from titles linked between articles. Wikipedia's redirections are very straightforward to obtain, and the coverage seems to be good.

From Eijiro, we assumed that the target translation words are synonyms to each other, but this approach carries the risk of treating unrelated words as synonyms, for instance, merging translations of "financial bank" and "river bank" together.

3.2. Answer Type Classification

Before designing the system, we considered the following questions: what are the answer types that characterize non-factoid questions, how we can model them, and how can we make use of them during run-time question answering? After examining the 30 training questions, we formulated the answer types in Table 2. In Table 2, the first column holds an answer type, the

[1] http://guadalajara.lti.cs.cmu.edu/annotations_db/

[2] the smallest meaning-bearing components in a Japanese sentence

[3] http://www.eijiro.jp/

[4] http://www.wikipedia.org/

second column holds the frequency of that type in the training questions, the third column hold the frequency in the formal run questions, and the fourth column provides an example (glossed in English).

Table 2. Answer types and their frequencies in the training set and formal run

Answer Type	$\#_T$	$\#_F$	Example (glossed)
REASON	10	29	Why X?, What is the reason for X?
DEFINITION	8	22	What is X?
METHOD	6	7	How do you do X?
RESULT	2	14	What is the consequence of X?
DEGREE	2	2	How much was the damage of X?
CONDITION	1	2	What is required to do X?
PERSON_BIO	1	1	Who is X?

The frequencies of the answer types in the formal run question set does not sum to 100, since there are 23 questions which were not covered by the types found in the training questions (e.g., *"What are the differences between X and Y"*.)

The types we defined might be too fine-grained considering the size of the training data, given that we used a machine learning-based approach to classify answer types. See our CLQA-2 paper[4] for the detailed implementation.

4. Retrieval Strategist

The Retrieval Strategist is responsible for retrieving relevant text from the corpus, given key terms from the Question Analyzer. The scores of key terms and their alternate forms are used as weights at retrieval time by a weighted synonym operator.

We preprocessed and indexed the corpus just like we did for the NTCIR-6 CLQA-2 task[4] where the only difference is the size of the corpus; CLQA-2 and QAC-4 used Mainichi 1998-1999 and Mainichi 1998-2001 respectively.

5. Information Extractor and Answer Generator

The Information Extractor is responsible for extracting the Answer Candidates (ACs), scoring them and returning a ranked list of ACs. We are interested in finding the exact ACs from retrieved documents. To simplify the design of the system, we did not use the web as an external resource to find ACs. Unlike factoid QA, where the answer units are words or phrases, we are expected to return answers in any meaningful chunk (i.e. phrase, sentence, multiple sentence, etc.). What makes this more complicated is that recognizing the answer boundary may involve advanced (e.g. syntactic and semantic) analysis, as compared to the task of named entity recognition (which almost all systems use in factoid QA). Therefore, AC extraction for non-factoid QA is very challenging.

To cope with this challenge, we came up with a heuristic *One Sentence Assumption* to assume that the answer is always within one sentence. By introducing the assumption, the AC extraction problem is simplified to a sentence ranking problem.

5.1. Machine Learning Approach for AC Scoring

When scoring an AC, we wish to incorporate multiple useful sources of information in calculating the likelihood that a candidate is correct. We use Maximum Entropy to model the distribution of correctness of ACs, given arbitrary features that may help to predict that an AC is correct.

In judging correctness, we assume that an AC is correct when it contains or is contained by one of the pre-defined answers. With this correctness criterion, we followed the steps below to construct learning examples:

1. Given a question, obtain the answer bearing documents and extract the sentences.
2. For each answer candidate (i.e. virtually all sentences in retrieved documents), extract associated features from the sentence itself, the parent passage, the parent document and the question.
3. For each pair consisting of an answer candidate and its set of features, label the pair with a +1 if the AC is correct, or -1 otherwise.

As training data, we were given 30 questions and their answers pooled from the participants. We finally obtained 741 positive examples and 16,638 negative examples for training.

5.2. Determining the AC Cutoff Threshold

Lack of official evaluation metrics made it difficult for us to tune the system. Since the QAC-4 task did not pre-define how many top *N* answer candidates should be submitted per question, we set a minimum threshold on the score (i.e. estimated probability from MaxEnt models) to prune ACs.

Unless we know what metric to maximize/minimize, we will not be able to optimize the system's performance. We learned the cutoff threshold by assuming that the average F1 score over the training questions is the target function to maximize. For each question, we calculated the F1 score as follows:

$$R = \frac{\#of\ correct\ AC\ returned}{\#of\ all\ answers}$$

$$P = \frac{\#of\ correct\ AC\ returned}{\#of\ AC\ returned}$$

$$F1 = \frac{2RP}{(R+P)}$$

5.3. Features

Features we used for AC scoring are summarized in Table 3. Feature values can take both binary and non-negative numeric values. Note that italicized feature names are variables.

Table 3. Features used for AC scoring

Feature Name	Description
RANK-OF-RETRIEVAL	Normalized rank of a document retrieved
KEYTERM	# of key terms found in an AC
EXPANDED-KEYTERM	# of expanded key terms found in an AC
POSITION-IN-DOCUMENT	Distance of an AC from the beginning of the document
TENSE-MATCHED	1 if the tense of the question and answer bearing sentence matched
SENTENCE-LENGTH	A value from the normal distribution p.d.f over the correct sentence length
HAS-SUBJ	1 if the Subject found in the AC.
HAS-PRON	1 if a pronoun found in the AC.
ATYPE-AND-*CUETYPE*	1 if an answer type and a cue under a certain type fired together
LIST-CUE	1 if a cue for bulleted lists and numbered lists found
PAREN	1 if a cue for parenthesis found
PARAGRAPH-HEAD	1 if the AC is at the beginning of the paragraph
ADJACENT_WORD	1 if the key term and a certain kind of words (e.g. particles, symbols, kanji/hiragana/katakana e.t.c.) are located adjacent to each other.

In order to exploit the answer types obtained in Question Analysis, we would like to link them to some types associated with the AC. To identify instances of complex answer types, we used hand-crafted cues which tend to appear in sentences of a certain type. Figure 3 exemplifies some cues for the *REASON* type.

理由, 原因, 事由, 因果, 因由, 訳, 因子, 動機, 背景, 結果, 理屈, 由来, 根拠, 名目, 意図, 所以, 目的, 起因, 影響, 続いて, 故に, いきさつ, こと(により|より|によって|から|で), というのも, なので, ため, ので, だから, からだ, このように, したところ

Figure 3. Example cues for REASON type

Some features are designed to identify an AC that is less likely to be the answer. For example, SENTENCE-LENGTH returns a value close to 0 if the AC is too short or too long.

5.4. De-duplication by Answer Generator

In factoid QA, the role of our AG was to perform answer merging, filtering and validation. In QAC-4, however, we used the AG just to merge the duplicate ACs. Merging multiple similar (non-identical) ACs may require better understanding of the text through advanced techniques such as semantic analysis or text summarization.

6. Results

Human judgment was performed to label four levels of correctness. Refer to [6] for the details of the human judgment evaluation metrics.

Table 4 presents the formal human judgment results on all the returned answers from our system. In theory, 400 is the maximum number of answers (the top 4 answers per question, multiplied by 100 questions). Our system returned 24 answers labeled with *Level A* which are close to the median and mean of all 14 systems, but returned as many as 310 totally incorrect answers which were judged as *Level D*.

In Table 5, the human judgment results are presented on a per-question level. For 19 questions, our system was able to return at least one correct answer judged as *Level A*, which is equal to the median and better than the average over all systems.

7. Issues

For a system which uses machine learning-based techniques for a certain subtask, scarcity of training data is a crucial issue. The number of training questions (30) is apparently too small for a training/development set. On another note, there seems to be a discrepancy between the training and test set in terms of answer type distribution, which also affected our performance. In the test set, there were question types that did not appear in the training set. Acquiring similar distributions of attributes in training and testing is an issue for applying machine learning techniques.

The threshold we used for answer cutoff was too lenient, resulting in too many ACs being returned as answers. On the formal run questions, the maximum number of ACs returned for a particular question was 138; the minimum was 0; and the average was about 27. In a real-world application, users might not be interested in retrieving this many results.

8. Conclusions and Future Work

Since we wanted to establish a baseline system for a task where a lot of challenges exist, we made a minimal set of extensions to our existing factoid QA system. We introduced strong assumptions and did not use the web as a resource, in an attempt to keep the system simple. As a consequence, we achieved average performance among the submitted systems.

Now that we have established a baseline system, we will develop more sophisticated improvements in the future. For example, in answer type analysis, we intend to classify multiple binary features of questions, instead of picking out just one "A-type category". Consider the question "*Why do some people dislike X*". We may want to make use of multiple features (such as *REASON* and *OPINION*) at the same time.

The *One Sentence Assumption* seems to have worked fairly well on the formal run dataset. After its answer set was delivered, we analyzed it and found that all 100

questions included at least one answer where the length is shorter than a sentence. To calculate the ratio of multiple-sentence level answers, we put all answers from each question into a pool and obtained the result which is very small, i.e. 6.7% (=72/1071). Given what we have found above, we conclude that our assumption is useful as long as we have the current evaluation criteria.

It would improve the utility of the system to return answers with more appropriate length, if doing so is possible with acceptable accuracy. One possibility is to take a sequential tagging approach, treating the answer boundary detection problem as a text segmentation problem. Such an extension will lead to better accuracy in automatic evaluation based on well-known metrics from other fields, such as COAP or Co-Occurrence Agreement Probability[7] for text-segmentation.

Acknowledgment

This work was supported in part by the Disruptive Technologies Office (DTO)'s Advanced Question Answering for Intelligence (AQUAINT) Program. We thank Yotaro Watanabe, Jeongwoo Ko, Justin Betteridge, Matt Bilotti, Eric Riebling and Andy Schlaikjer for their assistance in extending the QAC JAVELIN system. We also thank Eric Nyberg for his help in the final preparation of this paper.

References

[1] E. Nyberg, R. Frederking, T. Mitamura, M. Bilotti, K. Hannan, L. Hiyakumoto, J. Ko, F. Lin, L. Lita, V. Pedro, and A. Schlaikjer. JAVELIN I and II Systems at TREC 2005, In *Proceedings of TREC*, 2005.

[2] Ellen M. Voorhees. Trec 2005 question answering track guidelines. In *Proceedings of the 14th Text REtrieval Conference*, November 2005 (TREC 2005), 2005.

[3] F. Lin, H. Shima, M. Wang, and T. Mitamura. CMU JAVELIN System for NTCIR CLQA1. In *Proceedings of the Fifth NTCIR Workshop*, Tokyo, Japan, December, 2005.

[4] T. Mitamura, F. Lin, H. Shima, M. Wang, J. Ko, J. Betteridge, M. Bilotti, A. Schlaikjer and E. Nyberg. Javelin III: Cross-Lingual Question Answering from Japanese and Chinese Documents. In *Proceedings of the Sixth NTCIR Workshop*, Tokyo, Japan, May, 2007.

[5] T. Kudo, K. Yamamoto, and Y. Matsumoto. Appliying Conditional Random Fields to Japanese Morphological Analysis. EMNLP, pp. 230-237, July, 2004.

[6] J. Fukumoto, T. Kato, F. Masui, T. Mori. An Overview of the 4th Question Answering Challenge (QAC-4) at NTCIR Workshop 6. In *Proceedings of the Sixth NTCIR Workshop*, Tokyo, Japan, May, 2007.

[7] D. Beeferman, A. Berger, and J. Lafferty. Statistical models for text segmentation. Machine Learning, special issue on Natural Language Learning, C. Cardie and R. Mooney eds., 34(1-3), pp. 177-210, 1999.

[8] T. Kudo and Y. Matsumoto. Japanese Dependency. Analysis using Cascaded Chunking. In *Proceedings. of the 6th Conference on Natural Language Learning*. (CoNLL). pp.63-69, 2002.

[9] D. Ferrucci and A. Lally. UIMA: an architectural approach to unstructured information processing in the corporate research environment. Natural Language Engineering 10. No. 3-4, 327-348. 2004.

Table 4. Formal human judgment evaluation based on all answers returned

System	All answers	A	B	C	D	No answer
LTI JAVELIN	377	24	30	13	310	1
Median of 14 systems	337.5	24.5	33	13.5	248	0
Mean of 14 systems	302.6	23.9	38.6	15.6	226.9	8.6

Table 5. Formal human judgment evaluation based on question

System	Answered Qnum	Correct Qnum	Wrong Qnum	Including A	Including B	Including C	Including D	No answer
LTI JAVELIN	99	42	57	19	23	10	96	1
Median of 14 systems	100	46	43.5	19	25.5	10	87.5	0
Mean of 14 systems	91.4	42.6	48.8	17.9	25	12.9	84.1	8.6