JAVELIN III: Cross-Lingual Question Answering from Japanese and Chinese Documents

Teruko Mitamura, Frank Lin, Hideki Shima, Mengqiu Wang, Jeongwoo Ko,
Justin Betteridge, Matthew Bilotti, Andrew Schlaikjer and Eric Nyberg
Language Technologies Institute
School of Computer Science
Carnegie Mellon University

Abstract

In this paper, we describe the JAVELIN Cross Language Question Answering system, which includes modules for question analysis, keyword translation, document retrieval, information extraction and answer generation. In the NTCIR6 CLQA2 evaluation, our system achieved 19% and 13% accuracy in the English-to-Chinese and English-to-Japanese subtasks, respectively. An overall analysis and a detailed module-by-module analysis are presented.

Keywords: *Multilingual Question Answering, Chinese, Japanese, Information Retrieval .*

1. Introduction

JAVELIN is a question-answering system with a modular, extensible architecture [1]. The JAVELIN architecture is also language-independent, and we have extended the original English version of JAVELIN for cross-language question answering between English and Chinese or Japanese. JAVELIN participated in four CLQA subtasks (J-J, E-J, C-C, E-C), for which we submitted a total of 11 official and 6 unofficial runs. Our best run for E-J achieved about 13% in answer accuracy and our best E-C run achieved 19% in answer accuracy.

We present the JAVELIN architecture and processing modules in Sections 2 through 7. Section 8 summarizes our corpus preprocessing strategies, and Section 9 presents our results from the formal evaluation. We list translation and language-specific research issues in Section 10 and conclude in Section 11 by describing future work.

2. JAVELIN-3 Architecture

The JAVELIN system is composed of five main modules: Question Analyzer (QA), Translation Module (TM), Retrieval Strategist (RS), Information eXtractor (IX) and Answer Generator (AG). Inputs to the system are processed by these modules in the order listed above.

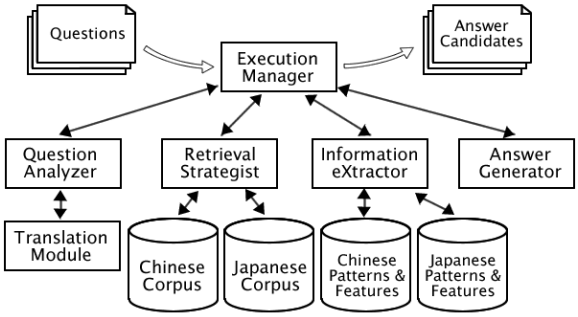

Figure 1: The JAVELIN Architecture

The QA module is responsible for parsing the input question, choosing the appropriate answer type, and producing a set of keywords. The TM module translates the keywords into task-specific languages. The RS module is responsible for finding relevant documents containing answers to the question using translated keywords. The IX module extracts answers from the relevant documents. The AG module normalizes the answers and ranks them in order of correctness. The overall architecture is shown in Figure 1.

3. Question Analyzer

The Question Analyzer (QA) is responsible for creating representation of the information need posed by the input question. Following is a description of the different subtasks performed by the QA.

Keyterm identification: First, the question is syntactically parsed, and named entities are tagged. Following this preprocessing, candidate keyterms are proposed by consulting a variety of sources. Syntactic category (POS) labels from the syntactic parse tree are used to select all nouns, verbs, adjectives and cardinal numbers as candidate keyterms. Terms recognized as named entities during preprocessing are also selected as separate keyterm candidates. Next, common noun phrases (as defined by a collection of multiple online dictionaries) as well as quoted terms become keyterm candidates. Finally, a non-overlapping set of keyterms is produced by unifying the list of possibly overlapping and duplicated candidate keyterms, using manually-assigned priorities for each source of evidence.

Question Classification: The QA categorizes the input question according to the expected type of the answer. Our current approach to question classification

completely replaces the approach we used at NTCIR-5, and makes use of both machine learning and manually-encoded rules. Both the trained classifier and the rule-based classifier rely on the same basic features to produce a hierarchical classification:

- lexical unigrams and bigrams
- question word (wh-word) and whether it modifies another word as a determiner
- main verb
- focus adjective (e.g. '*long'* in '*How long is the bridge?*')
- semantic type of the focus noun (e.g. the focus noun in '*Which town was attacked?*' is '*town*', which has a semantic type of CITY), determined using WordNet

For a given input question, the rule-based classifier produces a classification or a "don't know" value. In the cases where the rule-based classifier doesn't produce a classification, we use the value returned by a maximum entropy classifier [27] trained on question / answer type pairs from NTCIR-5 CLQA-1 as well as TREC 8-12.

For English-Chinese CLQA, we use a two-level answer type hierarchy containing a total of 54 categories, of which the top-level categories are *DATE, DURATION, ENTITY, LOCATION, MEASURE, MONEY, NUMEX, ORGANIZATION, PERCENT, PERSON, TITLE*. For English-Japanese CLQA, we use a slightly different two-level answer type hierarchy containing a total of 55, of which the top-level categories are *ARTIFACT, DATE, LOCATION, MONEY, NUMEX, ORGANIZATION, PERCENT, PERSON, TIME*.

Semantic Analysis: For English-Japanese CLQA, the Question Analyzer also analyzes the input question in terms of its semantic predicate-argument structure. We use ASSERT [2] to identify verbal predicates and their arguments and produce PropBank-style [3] argument labels. A significant problem arises, however, from the fact that no predicates for the verb 'be' are labeled in the PropBank corpus, which was used to train ASSERT. The subsequent inability of ASSERT to produce any results for sentences with 'be' as a main verb caused us to employ an additional tool for the semantic analysis of questions, many of which in the factoid domain contain 'be' as the only verbal predicate. The output of the Analyzer module of the KANTOO [4] machine translation system is thus consulted if ASSERT fails to recognize any semantic predicates in the question.

The set of semantic predicates recognized in the input question is also expanded before query formulation using an event ontology that defines *is-a*, *implies*, *inverse* and *reflexive* relations between verbs. For example, if the question is about entity A selling weapons to entity B (*sell(A,weapons,B)*), the system will also search for B buying weapons from A (*buy(B,weapons,A)*). The verbs *buy* and *sell* are encoded as *inverse* predicates. By definition, *inverse* predicates have essentially the same meaning as each other, but switch the *actor* (ARG0) and *destination/source* (ARG2) semantic roles.

4. Translation Module

To find answers to an English question in Chinese or Japanese document collections, the system first extracts keyterms from the English question and then passes the extracted keyterms along with their associated properties (such as its named entity type or part-of-speech) on to the Translation Module (TM). For every input English keyterm, TM returns a set of Chinese or Japanese translation candidates in ranked order of their translation score.

To acquire the set of translation candidates, TM uses different types of translation resources including Machine Readable Dictionaries (MRDs), Machine Translation systems (MTs), web-mining-based translators (WBMTs), and hand-built rule-based translators for dates and numbers. Every resource and every type of resource has its strengths and weaknesses. For example, MRDs are usually better for translating common nouns and verbs but have poor coverage of named entities, while WBMTs are good for translating popular named entities but do a poor job of translating common nouns and verbs. TM uses different resources for translating the keyterm based on whether it is a common noun, verb, proper name, numerical expression, or other types of word or phrase.

To rank the keyterm translation candidates, TM assigns translation candidates a score using co-occurrence statistics of the source keyterm (in this case English) and the target candidate translation (in this case Chinese or Japanese) found in HTML pages on the Web. The co-occurrence information is obtained by using a search engine, and the correlation statistic is calculated using chi-square (see Fig. 2). At the end TM returns the set of translation candidates in ranked order of their chi-square score.

$$\chi^2(X,Y) = \sum_{X\in\{0,1\}}\sum_{Y\in\{0,1\}} n\frac{(\Pr(X,Y)-\Pr(X)\Pr(Y))^2}{\Pr(X)\Pr(Y)} = \frac{n(ad-bc)^2}{(a+b)(a+c)(c+d)(b+d)}$$

Figure 2: Equation for chi-square statistic: *a* is the number of web pages containing both the source keyterm and the target translation candidate, *b*, only source, *c*, only target, *d*, neither.

The translated keyterm candidates, along with their ranking, are used to formulate a query for retrieving relevant blocks from the document collections. The ranking assigned to each translation candidate is used to boost its confidence score when formulating the query.

5. Retrieval Strategist

The Retrieval Strategist is responsible for retrieving relevant text from the corpus and passing it to the

Information Extractor. The corpus is indexed with Indri [5], a part of the open-source Lemur [1] toolkit. For Chinese, the locations of Named Entity types identified by the Chinese-language version of BBN Identifinder [6] are stored in the index. For Japanese, Named Entity types identified by Cabocha[2] are stored in the index, along with annotations corresponding to Japanese case markers and semantic role labels in the style of Propbank [7]. The unit of retrieval is an overlapping, three-sentence window that we call a *block*. For Chinese, block boundaries are calculated using sentence segmentation provided by S-MSRSeg [23] [3]; for Japanese, wrote regular expressions. For each language, block and sentence boundaries are stored in the index.

The Retrieval Strategist formulates Indri queries from the analysis provided by the Question Analyzer. Each query contains an outer clause scoring and ranking blocks based on the degree to which they match the keyterms identified by the Question Analyzer. A query contains one inner clause for each keyterm, weighted by its respective importance, which matches the original form of the keyterm, or an alternate form that receives a discounted match score according to the term weights provided by the Question Analyzer. The queries follow this template (the term weights pictured are notional):

#weight[**block**](
 $weight_1$ #wsyn(1.0 $term_1$ 0.85 alt_{1a} 0.60 alt_{1b} …)
 $weight_2$ #wsyn(1.0 $term_2$ 0.75 alt_{2a} …)
 …)

6. Answer Candidate Extraction

We are interested in extracting answer candidates given a set of potentially relevant documents. Our extraction approach for NTCIR-6 addressed some problems we noted with the surface pattern-based and proximity-based approaches that we implemented as FST IX and LIGHT IX, respectively, in NTCIR-5 CLQA-1 [8].

In the pattern-based approach, we have to spend a lot of human effort if we create patterns by hand. Even if we generate patterns automatically, the coverage of each pattern is usually limited due to the data sparseness of the training data, and thus the resulting system achieves high precision but low recall. And what is worse, the approach is not general and must be tailored for each new language.

The proximity-based approach utilizes surface distance between keyterms and the answer candidate, assuming that answer candidates occur close to keyterms. The simplicity of this algorithm makes it easy to implement; however, it often fails to capture linguistic relations among terms and is easily fooled by common linguistic phenomena (such as negation).

More advanced techniques (for example, the use of syntactic dependencies [9] [10] [11], or shallow semantics [12]) have been applied to complement the disadvantages of these simple approaches, but none of the single solutions that have been proposed work perfectly. Therefore, we decided to use machine learning techniques to take advantages of multiple features, which support a combination of algorithms which is tuned by training on real data, and can potentially outperform an ad-hoc combination of extraction algorithms.

We will present the language-specific implementations of the extractors in the following subsections.

6.1. Japanese

We first obtain Named Entities from the target documents using the CaboCha tool[4]. Then a pattern-based Named Entity Recognizer is used to extract more fine-grained categories of Named Entities from the corpus.

Ittycheriah et al. [13] used Maximum Entropy Models to automatically learn weights of features and predict the probability that answer candidates are correct. Other systems have followed the same approach and show a consistent improvement over ad-hoc scoring functions [14] [10].

Taking into account the previous successes of the Maximum Entropy models, we also model the distribution of correctness c being TRUE given a question q, document d and an answer candidate a as:

$$p(c = TRUE \mid q,d,a) = \frac{\exp(\vec{f}(c = TRUE,q,d,a)\cdot\vec{\theta})}{\sum_{c\in\{TRUE,FALSE\}}\exp(\vec{f}(c,q,d,a)\cdot\vec{\theta})}$$

where $\vec{f}$ is a vector of features and $\vec{\theta}$ is a vector of feature weights. Table 2 shows a list of the features we used.

6.2. Chinese

In our C-C system, we processed input questions using the same procedure used for Chinese corpus preparation (see Section 8), with an additional step that identifies the "question word" (e.g. *what*, *when*, *where*) based on hand-crafted patterns. In our E-C system, we created multiple full-sentence translations of the English question, and processed them as if they were Chinese questions. Then we select the best translation according to a word alignment score, which we will explain next.

To model the syntactic and semantic coherence between the question and the answer sentence, we want to first align the words in both sentences. Our matching scheme takes into account synonym matching, partial word matching and word-type matching. We then adopted a weighted maximum network flow algorithm ($O(n^3)$) [15] to find a bipartite matching with the highest alignment score. In the case of the E-C system, where we have multiple translations of the English question,

[1] See: http://www.lemurproject.org
[2] See: http://chasen.org/~taku/software/cabocha/
[3] See: http://research.microsoft.com/~jfgao/
[4] See: http://chasen.org/~taku/software/cabocha/

we pick the translated sentence that aligns the best when we evaluate an answer sentence.

After word alignment, we extracted all pairs of words that are aligned. We use $\{a_i, q_i\}$ to denote the set of pairs of aligned words, and we denote the answer candidate as *d*, and the "question word" as *w* (e.g. *what*, *when*, etc.). The next step is to find the relations between the keywords in the answer sentence ($\{a_i\}$) and the answer candidate (*d*), and compare these relations with the relations between the keywords in the question ($\{q_i\}$) and the "question word" (*w*). To do this, we extract the syntactic dependency path from a_i to *d* via their lowest common ancestor (LCA) in the answer sentence dependency tree, and pair it with the dependency path between q_i and *w* in the question tree to form a feature to our learning algorithm. The LCA finding algorithm we implemented is from [16], which has a remarkable constant ($O(1)$) complexity. Other features included word alignment score, named-entity and answer type matching features, term occurrence features, etc. The complete list of features is given in Table 3.

We experimented with two different learning algorithms for scoring and ranking answer candidates, using the same set of features. The first learner is the Maxent learner described in Japanese IX section. The second learner we experimented with is an in-house Ranking SVM algorithm that we have developed. Ranking SVM has also been applied to document retrieval and re-ranking of definitions. Due to length limitation, we will not give the algorithmic details about the Ranking SVM algorithm; please refer to [17] and [18] for full details.

Results in Table 1 show the evaluation results of IX outputs on E-C and C-C tasks using Maxent and Ranking SVM. There does not seem to be a salient difference between the two models. Ranking SVM seems to perform better on the E-C set, while Maxent gives better performance on the C-C set, but the relative difference is within 10%.

Table 1. Results of Chinese IX module output

Model	C-C	E-C
Maxent	0.280 (0.286)	0.160 (0.180)
Ranking SVM	0.260 (0.273)	0.173 (0.200)

Table 2. Features used for Japanese IX Module

Pattern Name	Description
RANK-OF-RS	Rank of the document in retrieval time
PROXIMITY-OF-KEYTERM	Proximity of keyterm and AC
COVERAGE-OF-KEYTERM	How many keyterms found in
POSITION-IN-SENTENCE	Distance of AC from the beginning of the sentence
TENSE-MATCHED	1 if the tense of the question and answer bearing sentence matched
PREDICATE-ARGUMENT	Structural similarity score of predicate argument structures
UNIT-MATCHED	1 if the unit from the question was attached to AC
ATYPE-NE	1 if each pair occurred (e.g. PERSON-PERSON)
PATTERN	1 if each pattern appeared (250 patterns used)

Table 3. Features used for Chinese IX Module

Pattern Name	Description
ALWAYS-ON	An always-on feature acting as a label prior
SENT-TERM-FREQ	The number of times question keyterms appeared in the AC sentence
BLOCK-TERM-FREQ	The number of times question keyterms appeared in the AC 3-sentence block
SENT-TERM-COVERAGE	The percentage of question keyterms that appeared in the AC sentence.
BLOCK-TERM-COVERAGE	The percentage of question keyterms that appeared in the AC 3-sentence block.
ALIGNMENT-SCORE	Question and AC sentence term alignment score
PREV-NEXT-WORD-MATCH	1 if AC contains the previous or next word to the question word (e.g. what, when) in the question.
UNIT-MATCHED	1 if the unit from the question was attached to AC
ATYPE-NE	1 if a pair occurred (e.g. PERSON-PERSON)
ANSWER-SUBTYPE-MATCH	1 if AC matches particular pattern defined by the subtype (e.g. CITY-市)
DEP-PATH-MATCHING	1 if a pair of dependency paths occurred (e.g. NMOD-SUB<->NMOD-OBJ), see Section 6.2

7. Answer Generator

The Answer Generator (AG) is responsible for selecting the correct answer from the answer candidates produced by the Information Extractor. In NTCIR-5, the AG used a simple answer clustering approach which merged similar or redundant answers and then calculated a new score for each answer cluster based on individual answer scores in the cluster. However, the answer selection task should consider the degree to which an answer candidate is relevant to the question and the amount of supporting evidence for the answer candidate. Formally, the first task is to estimate the probability $P(\text{correct}(A_i)|A_i,Q)$, where Q is a question and A_i is an answer candidate for the question. The second task is to estimate the probability $P(\text{correct}(A_i)|A_i, A_j)$, where A_j is similar to A_i. As both probabilities affect the decision about how much we boost the rank of an answer candidate, we should combine them in a unified framework and estimate the probability of an answer candidate as: $P(\text{correct}(A_i)|Q, A_1,..., A_n)$.

For the comprehensive answer selection task, we developed an answer ranking framework which estimates $P(\text{correct}(A_i)|Q, A_1,..., A_n)$ given multiple answer relevance features and answer similarity features. The evaluation results on English TREC questions show that the framework significantly improved answer selection performance in English QA [19]. In preparing for NTCIR6, we extended the framework to Chinese and Japanese QA by incorporating language-specific features. We used two sets of features: answer relevance features and answer similarity features. Answer relevance features utilized language-specific resources such as Chinese HowNet [20], Japanese Gengo

GoiTaikei dictionary[5], the Web and Wikipedia. These resources were used to generate an answer relevance score for each answer candidate. Answer similarity features calculated similarity between two answer candidates using string similarity metrics (e.g., Levenshtein, Jaro-Winkler, and Cosine similarity) and synonyms. Synonyms were acquired from Wikipedia, the Japanese EIJIRO dictionary and manual conversion rules which convert Chinese and Japanese temporal and numeric expressions into Arabic numbers. More details on the features are found in [21].

The answer ranking framework used logistic regression to estimate the probability of individual answer candidates using those features. Then, the answer candidates were re-ranked according to their estimated probability and the top one was chosen as the final answer to the question.

8. Corpus Preprocessing

To support information extraction processes in both Japanese and Chinese, the evaluation corpora were preprocessed with various NLP tools to produce annotations used during system training and runtime.

For Japanese language corpora the following preprocessing was performed: Morphology, dependency parses, and named entity tags were generated using CaboCha. Basic semantic predicate-argument structures were extracted from dependency parses in a three-stage process: First, those constituents containing either a verb or *sahen*-verb from the IPAL lexicon were selected as predicate targets; Then, constituents connected in the dependency parse to a target constituent were labeled as arguments of the target constituent. Finally, additional attributes were assigned to predicate arguments based on manually encoded rules, and predicate targets having no arguments were discarded. Additional patterns were applied to the text to improve named entity tagging of particular entity types such as numeric and date quantities.

For Chinese language corpora, additional work was required to handle Traditional Chinese text as all of our NLP tools for Chinese are trained on Simplified Chinese text. As a pre-processing step, we converted all documents in the corpus from Traditional Chinese to Simplified Chinese using an online converter[6]. The conversion is done based on simple character-mapping, and therefore cannot account for lexical gaps between Traditional and Simplified Chinese texts. For example, the original expression of the name of movie director Luc Besson in Traditional Chinese is 盧貝松, after converting to Simplified Chinese it becomes 卢贝松, but the correct form in Simplified Chinese is 吕克贝松. This problem of lexical semantic gap between the two forms of Chinese language is very hard to overcome as discussed in Mitamura et. al [22], and it is clearly the case that our downstream NLP tools (especially the Chinese word segmenter) suffer from cascaded error caused by this problem. Once the documents are converted into Simplified Chinese, word segmentation and named-entity tagging is done using S-MSRSeg [23]. We also performed NE tagging using BBN's Identifinder [6], and merged with NE tags produced by S-MSRSeg. Then we assigned POS tags to each word using a Chinese POS tagger described in [29]. Finally, using the POS and segmented words as input, we produced syntactic dependency analysis of the corpus sentences using MaltParser [24].

[5] http://www.kecl.ntt.co.jp/mtg/resources/GoiTaikei
[6] http://www.mandarintools.com/zhcode.html

9. Results

In this section we will first discuss our NTCIR-6 evaluation results, and then present a module-by-module analysis of the performance of the latest version of the system.

9.1 NTCIR-6 Evaluation

For NTCIR-6, our team participated in 4 CLQA subtasks. Table 4 shows the number of official and unofficial runs we submitted for formal evaluation, and Table 5 shows the score of the top scoring official runs as judged by NTCIR:

Table 4. Number of runs submitted by LTI to each subtask.

	Official	Unofficial
J-J	2	0
E-J	3	2
C-C	3	2
E-C	3	2

Table 5. Score of top official runs judged by NTCIR. R+U is the score when unsupported answers are included in calculation. The scores represent top 1 answer accuracy.

	R	R+U
Best J-J Run	0.335	0.360
Best E-J Run	0.095	0.115
Best E-J Run (2)	0.070	0.115
Best C-C Run	0.253	0.253
Best C-C Run (2)	0.240	0.260
Best E-C Run	0.147	0.200

For the J-J subtask, we achieved the highest score among all participants: 0.335(R)/0.360(R+U), with the next best system at 0.310(R)/0.335(R+U) [28]. On the other hand, the E-J system performed relatively poorly: 0.095(R)/0.115(R+U), which is only 28.3% of the accuracy of its monolingual counterpart. However, it is important to note that these two numbers cannot be directly compared against each other, since some of the degradation of performance is due to keyterm translation errors; due to time constraints, we could not train the IX for E-J with a complete training data. In the next section, we present E-J results using the same extractor we used for submitting the J-J run.

For the C-C subtask, we were at the median of the highest scoring official runs of 7 participants, both in R and R+U categories. For the E-C subtask, we were one above the median among the highest scoring official runs of 7 participants, both in R and R+U; the system achieved 58.1%(R) and 76.9%(R+U) of the accuracy of its best monolingual counterpart.

9.2 Post NTCIR-6 Evaluation and Analysis

After submitting our system output for the formal runs, we evaluated our system on our own with gold-standard data provided by NTCIR. Our goal was to obtain a more detailed module-by-module analysis of our NTCIR-6 system. In addition, we were able to evaluate runs that we could not submit for formal evaluation either due to time constraints or the limited number of runs we could submit. Tables below show detailed analysis.

QA Prec: Average # of questions where the answer type (including sub-type) matched the gold standard answer type.

RS Prec: Percentage of retrieved blocks containing the answer string, averaged over all the questions.

RS TopN: Average # of questions where the highest-ranked retrieved block containing the answer has a rank less than or equal to N.

RS Block MRR: Mean Reciprocal Rank of highest-ranked retrieved block containing the answer.

IX/AG R or R+U: R means extracted answer was correct and it came from supported document whereas R+U only look at surface string to judge.

IX/AG TopN: Average # of questions where the highest-ranked answer candidate that matches one of the gold standard answers has a rank less than or equal to N.

IX/AG MRR: MRR of the highest-ranked answer candidate that matches one of the gold standard answers (considering only the top 5 answer candidates).

Table 6. Detailed per-module analysis of J-J run

QA	**Prec**	0.810	
RS	**Prec**	0.203	
	Top1	0.385	
	Top5	0.530	
	Top10	0.650	
	Top20	0.720	
	MRR	0.456	
		R	**R+U**
IX	**Top1**	0.300	0.320
	Top5	0.435	0.480
	Top10	0.485	0.550
	Top20	0.540	0.640
	MRR	0.344	0.374
AG	**Top1**	0.330	0.370
	Top5	0.445	0.500
	Top10	0.495	0.580
	Top20	0.525	0.630
	MRR	0.368	0.412

Table 7. Detailed per-module analysis of E-J run

QA	**Prec**	0.890	
RS	**Prec**	0.119	
	Top1	0.220	
	Top5	0.365	
	Top10	0.455	
	Top20	0.490	
	MRR	0.286	
		R	**R+U**
IX	**Top1**	0.130	0.210
	Top5	0.180	0.290
	Top10	0.235	0.355
	Top20	0.255	0.425
	MRR	0.148	0.238
AG	**Top1**	0.135	0.220
	Top5	0.200	0.350
	Top10	0.250	0.420
	Top20	0.260	0.465
	MRR	0.159	0.267

Table 8. Detailed per-module analysis of C-C run using MaxEnt classifier

QA	**Prec**	0.680	
RS	**Prec**	0.160	
	Top1	0.253	
	Top5	0.340	
	Top10	0.407	
	Top20	0.440	
	MRR	0.296	
		R	**R+U**
IX	**Top1**	0.280	0.287
	Top5	0.440	0.493
	Top10	0.487	0.560
	Top20	0.533	0.620
	MRR	0.340	0.364
AG	**Top1**	0.327	0.393
	Top5	0.453	0.547
	Top10	0.507	0.613
	Top20	0.527	0.633
	MRR	0.380	0.458

Table 9. Detailed per-module analysis of E-C run using MaxEnt classifier

QA	**Prec**	0.860	
RS	**Prec**	0.104	
	Top1	0.113	
	Top5	0.267	
	Top10	0.307	
	Top20	0.353	
	MRR	0.176	
		R	**R+U**
IX	**Top1**	0.147	0.173
	Top5	0.233	0.307
	Top10	0.287	0.407
	Top20	0.320	0.520
	MRR	0.175	0.218
AG	**Top1**	0.147	0.193
	Top5	0.247	0.340
	Top10	0.280	0.440
	Top20	0.307	0.507
	MRR	0.179	0.240

As noted in previous section, the detailed per-module analysis of the E-J run shown above is not the same as the one submitted to NTCIR for formal evaluation.

In the Question Analyzer module, the English and Japanese question analysis achieved about the same performance; however, the performance of the Chinese question analysis is much lower. We attribute this to the fact that our training data (from NTCIR-5) had a significantly different distribution of question types.

In the Retrieval Strategist module, the performance drop is indicative of the performance of the Translation Module, since we maintain a similar document retrieval strategy across languages. For Japanese subtasks, we see E-J maintaining 68.1% and 62.7% of monolingual retrieval performance in Top20 and MRR categories, respectively. For Chinese subtasks, we see E-C maintaining 80.2% and 59.5% of monolingual retrieval performance in Top20 and MRR categories, respectively. The translation performance is about the same for E-J and E-C, and both leave room for much improvement.

The per-module analysis also shows that the Answer Generator (AG) was not effective in ranking the correct and supported answers to the top position. One reason was that the extractors sometimes produced answer candidates whose document frequency is high (e.g., "第一", "美国", "日本") and the AG gave them a high Web validation score. The Web validation routine sends a query consisting of an answer candidate and question keyterms to Google, and then calculates the word distance between keyterms and an answer candidate from the retrieved text snippets. Since answer candidates consisting of common terms had more retrieved snippets and therefore received a higher score, the AG often boosted such (incorrect) candidates to the top position. After the formal evaluation, we changed the Web validation by incorporating a tf-idf measure, which gives less weight for common terms. This boosted the score ("R") from 0.280 to 0.327 for the C-C run. Another reason is that AG was trained without considering whether a correct answer was extracted from supported documents or not. This is why the AG tended to improve the performance of (R+U), but not (R). How to identify correct answers extracted from supporting documents is one future research topic for answer generation.

10. Translation and Language-specific Issues

From the results section we see the difference between the performance of E-J and E-C systems compared to that of their monolingual counterparts, which leaves much room for improvement. An analysis of the translation errors points to three main problems: 1) vocabulary mismatch, 2) lack of translation for certain named entities, and 3) incorrect extraction of keyterms.

The vocabulary mismatch is a difficult problem, and one that not necessarily apply only to cross-lingual applications. Query-based document retrieval encounters the same problem when trying to match query terms from user input to terms found in the document collection. Also, based on our observation, vocabulary mismatch occurs mostly with common nouns and verbs, which are not as important for information retrieval or answer extraction as named entities. The lack of translation for certain named entities can only be remedied by incorporating more of certain translation resources; our incorporation of resources such as Wikipedia and web-mining translators seems to have improved our named entity translation over NTCIR-5 CLQA-1, but we may need more of these resources. Our unit of translation is a keyterm, which can be a word, a phrase, a book title, or even a short quote; the correct extraction of keyterms is extremely important for translation. As an example, from the question "When did the 1999 Hualien International Stone Sculptural Festival start?", we extracted "Hualien International Stone" as a keyterm, which resulted in erroneous translation. Also, there is the question of whether to translate "Hualien International Stone Sculptural Festival" as a phrase or "Hualien", "International", "Stone", "Sculptural", and "Festival" separately; maybe it is better to do both. Recent works on Cross-Lingual Information Retrieval show that finding a good unit of translation significantly improves CLIR performance [26].

Our post-NTCIR error analysis for the C-C and E-C runs showed that for a significant portion of the questions that our system failed to answer, the error came directly from the underlying Chinese NLP systems but not from our QA algorithms. Most severely, we observed many errors in the segmentation module (S-MSRSeg) and named-entity identification module (S-MSRSeg and Identifinder). The root of these problems is in the language difference between the NTCIR corpus (in Traditional Chinese) and the corpus which these NLP tools are trained on (in Simplified Chinese). Since we did not have access to Traditional Chinese tools or training resources, we converted the Traditional Chinese corpus into Simplified Chinese using a non-comprehensive character-mapping table and simple heuristics. The noise resulting from this conversion process, as well as the inherent differences between Simplified and Traditional Chinese [22], caused a cascade of errors in our Chinese pipeline, degrading the input quality to our retrieval, extraction and answer ranking modules.

On the other hand, we observed in E-J and J-J specific issues due to the properties of Japanese. Japanese text contains many instances of ellipsis when the topic or focus of a sentence is obvious to a human reader, making it difficult for a program to capture long term dependencies between keyterms and the answer. This form of ellipsis (known as zero-anaphora), seems to have had a negative impact on our block-level retrieval and answer extraction. According to a prior analysis of CLQA-1 formal data, about 20% of questions on average might be affected by this.

11. Future Work

Thanks to the many state-of-the-art techniques we implemented, our system performed better than the older version evaluated at NTCIR-5. However, we realize that the overall system becomes more and more complex as we implement new techniques.

In order to better understand errors in the system output, we have been analyzing system performance using different measures, including accuracy by answer type and module-by-module evaluation [25] [30]. Various Boolean properties can be assigned to each input question, and system performance can be analyzed according to the property values of each question. Each property represents one (usually challenging) aspect in answering the question. A sample of the properties we can make use of is listed below:

- A long-distance dependency (across multiple sentences) exists between the keyterm and the answer
- Question contains a Named Entity which is hard to translate
- Phrase level paraphrasing would help
- Co-reference resolution would help
- Question and answer-bearing sentence have the same predicate argument structure
- Answer-bearing sentence contains the answer in an extra-grammatical context

By comparing how well the system performs on questions with or without a certain property, we will be able to identify what problems can be solved in what degree, after introducing a specific algorithm, feature, resource or assumption.

Acknowledgment

This work was supported in part by the Disruptive Technologies Office (DTO)'s Advanced Question Answering for Intelligence (AQUAINT) Program. We thank Eric Riebling for his assistance in corpus preprocessing.

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 *Proc. of TREC'05*, 2005.

[2] S. Pradhan, W. Ward, K. Hacioglu, J. Martin and D. Jurafsky. Shallow semantic parsing using support vector machines. In *Proc. of the HLT/NAACL'04*, 2004.

[3] P. Kingsbury and M. Palmer. From Treebank to Propbank. In *Proc. of LREC'02*, 2002.

[4] E. Nyberg and T. Mitamura. The KANTOO Machine Translation Environment. In *Proc. of AMTA 2000*, 2000.

[5] T. Strohman, D. Metzler, H. Turtle and W. B. Croft. Indri: A language model-based search engine for complex queries. In *Proceedings of ICIA'05*, 2005.

[6] D. Bikel, R. Schwartz and R. Weischedel. An algorithm that learns what's in a name. *Machine Learning, Special Issue on Natural Language Learning, Volume 34, Issue 1-3, p211-231*, 1999.

[7] P. Kingsbury, M. Palmer and M. Marcus. Adding semantic annotation to the penn treebank. In *Proc. of HLT'02.* 2002.

[8] F. Lin, H. Shima, M. Wang, and T. Mitamura. CMU JAVELIN system for NTCIR CLQA1. In *Proc. of NTCIR-5*, 2005.

[9] T. Takahashi, K. Nawata, K. Inui and Y. Matsumoto. NAIST QA System for QAC2. In *Proc. of NTCIR-4*, 2004.

[10] D. Shen, G. M. Kruijff, and D. Klakow. Exploring syntactic relation patterns for question answering. In *Proc. Of IJCNL'05*, 2005.

[11] G. Kurata, N. Okazaki and M. Ishizuka. GDQA: Graph driven question answering system - NTCIR-4 QAC2 experiments. In *Proc. of NTCIR-4*. 2004.

[12] D. Kawahara, N. Kaji, and S. Kurohashi. Question and answering system based on predicate-argument matching, In *Proc. of NTCIR-3*, 2002.

[13] A. Ittycheriah, M. Franz, and S. Roukos. IBM's statistical question answering system – TREC-10. In *Proc. of TREC-10*, 2001.

[14] D. Shen, and D. Klakow. Exploring correlation of dependency relation paths for answer extraction. In *Proc. of COLING-ACL'06*, 2006.

[15] H. Gabow. Implementation of Algorithms for Maximum Matching on Nonbipartite Graphs. PhD thesis, Stanford University, 1974.

[16] B. Schieber and U. Vishkin. On finding lowest common ancestors: simplification and parallelization. SIAM Journal of Computing, Volumn 17, Issue 6, p1253-1262, 1998.

[17] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. of SIGKDD'02*, 2002.

[18] Y. Cao, J. Xu, T. Liu, H. Li, Y. Huang and H. Hon. adapting ranking SVM to document retrieval. In *Proc. of SIGIR'06*, 2006.

[19] J. Ko, L. Si, E. Nyberg, A probabilistic framework for answer selection in question answering, To appear in *NAACL-HLT'07*, 2007.

[20] Z. Dong. Hownet: http://www.keenage.com. 2000

[21] J. Ko, T. Mitamura, E. Nyberg, Language independent probabilistic answer ranking in multilingual question answering, To appear in *ACL'07*, 2007.

[22] T. Mitamura, M. Wang, H. Shima, and F. Lin. Keyword translation accuracy and cross-lingual question answering in Chinese and Japanese. In *Proc. of MLQA workshop, EACL 06'*, 2006.

[23] J. Gao, M. Li, A. Wu and C. Huang. Chinese word segmentation: a pragmatic approach. Microsoft Research Technical Report, MSR-TR-2004-123, 2004.

[24] J. Nivre and J. Hall. MaltParser: a language-independent system for data-driven dependency parsing. In *Proc. of the Fourth Workshop on Treebanks and Linguistic Theories*, 2005.

[25] H. Shima, M. Wang, F. Lin and T. Mitamura. Modular approach to error analysis and evaluation for multilingual question answering. In *Proc. of LREC'06*, 2006.

[26] J. Gao, and Nie, J. A study of statistical models for query translation: finding a good unit of translation. In *Proc. of SIGIR'06*, 2006.

[27] A. Berger, V. Della Pietra, and S. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics, 22(1):39-71*, 1996.

[28] Y. Sasaki, C. Lin, K. Chen and H. Chen. Overview of the NTCIR-6 cross-Lingual question answering (CLQA) task. In *Proc. of NTCIR-6,* 2007.

[29] Wang, M., K. Sagae and T. Mitamura. A fast, accurate deterministic parser for Chinese. In *Proc. of COLING-ACL'06,* 2006.

[30] Y. Sasaki, H. Isozaki, J. Suzuki, K. Kokuryou, T. Hirao, H. Kazawa, and E. Maeda. SAIQA-II: A Trainable Japanese QA System with SVM. *IPSJ Journal, Vol. 45, NO. 2, pp. 635-646*, 2004. (in Japanese)