CMU JAVELIN System for NTCIR5 CLQA1

Frank Lin   Hideki Shima   Mengqiu Wang   Teruko Mitamura
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{frank+, hideki, mengqiu, teruko}@cs.cmu.edu

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 NTCIR5 CLQA1 evaluation, our system achieved 7.5% and 10.0% 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: *Multi-lingual Question Answering, Information Retrieval, Named Entity Translation.*

1 Introduction

The JAVELIN system is a modular, extensible architecture for building question-answering systems [7]. The JAVELIN architecture is language-independent, and we have been working to extend the original English version of JAVELIN for cross-language question answering in Chinese and Japanese. Our submissions to the NTCIR5 CLQA1 evaluation represent the first evaluation of the JAVELIN system on a CLQA task. Out of 13 groups participating in the CLQA1 task, we are the only group to submit formal runs for both the English-to-Chinese (EC) and the English-to-Japanese (EC) subtasks. The same overall architecture was used for both systems, allowing us to compare the performance of the two systems on a per-module basis. We did not use any external resources for query expansion or information extraction. After analyzing the observed performance of each module on the evaluation data, we created gold standard data (perfect input) for each module in order to determine upper bounds on module performance.

2 JAVELIN 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.

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 which might contain answers to the question, using translated keywords produced by the TM. 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.

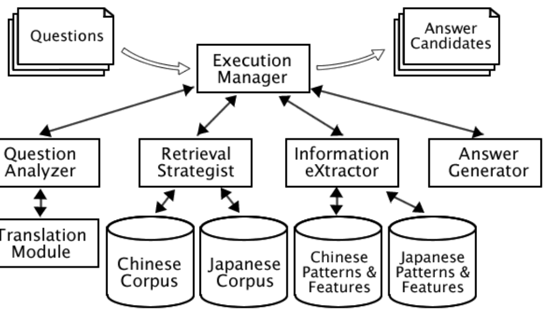

Figure 1. System Architecture

2.1 Question Analyzer

The Question Analyzer (QA) is responsible for extracting information from the input question in order to formulate a representation of the information required to answer the question. Input questions are processed using the RASP parser [6], and the module output contains three main components: a) selected keywords; b) the answer type (e.g. numeric-expression, person-name, location); and c) the answer subtype (e.g. author, river, city). The selected keywords are words or phrases which are expected to appear in documents which contain a correct answer. In order to reduce noise in the document retrieval phase, we use stop-word lists to eliminate high-frequency terms; for example, the term “old” is not included as a keyword for “how-old” questions.

For the EJ and EC subtasks, the QA module translates keywords by calling the Translation Module (see Section 2.2), so that translated keywords can be used to retrieve documents from multilingual corpora.

2.2 Translation Module

The Translation Module (TM) is used by the QA module to translate keywords into the language of the target corpus. In our approach, the TM selects the best combination of translated keywords from several sources: Machine Readable Dictionaries (MRDs), Machine Translation systems (MTs) and Web-mining-Based Keyword Translators (WBMTs). For the EJ subtask, we used two MRDs, eight MTs and one WBMT. If none of them return translation, the word is tranliterated into kana. See the details of transliteration in Section . For the EC subtask, we used one MRD, three MTs and one WBMT. The TM uses a noisy channel model for keyword translation, and estimates model statistics using the World Wide Web as a resource. Details of the model are described below.

2.2.1 The Noisy Channel Model

In the noisy channel model, an undistorted signal passes through a noisy channel and becomes distorted. Given the distorted signal, we are to find the original, undistorted signal. IBM applied the noisy channel model idea to translation of sentences from aligned parallel corpora, where the source language sentence is the distorted signal, and the target language sentence is the original signal [4]. We adopt this model for keyword translation, with the source language keyword terms as the distorted signal and the target language terms as the original signal. The TM's job is to find the target language terms given the source language terms, by finding the probability of the target language terms given the source language terms $P(T|S)$.

Using Bayes' Rule, we can break the equation down to several components:

$$P(T|S) = \frac{P(T) \cdot P(S|T)}{P(S)}$$

Because we are comparing probabilities of different translations of the same source keyword terms, we can simplify the problem to be:

$$P(T|S) = P(T) \cdot P(S|T)$$

We can now reduce the equation down to two components. $P(T)$ is the language model and $P(S|T)$ is the translation model. If we assume independence among the translations of individual terms, we can represent the translation probability of a keyword by the product of the probabilities of the individual term translations:

$$P(S|T) = \prod_i P(s_i|t_i)$$

2.2.2 Estimating Translation Model Probabilities

We make the assumption that terms that are translations of each other co-occur more often in mixed-language webpages than terms that are not translations of each other. We then define the translation probability of each keyword translation as:

$$P(s_i|t_{t,j}) = \frac{log(co(s_i, t_{i,j}))}{\sum_j log(co(s_i, t_{i,j}))}$$

Where s_i is the i-th term in the source language, $t_{i,j}$ is the j-th translation candidate for s_i, and $co(s_i, t_{i,j})$ is the number of web pages that contain both s_i and $t_{i,j}$, according to the search engine. The log function is used to adjust the count so that translation probabilities can still be comparable at higher counts.

2.2.3 Estimating Language Model Probabilities

In estimating the language model, we simply count the number of pages in which all the terms in a set of target language term candidates co-occur, and divide that count by the sum of the occurrences of the individual terms:

$$P(T) = \frac{co(t_1, t_2, ..., t_n)}{\sum_i o(t_i)}$$

The final score of a translation candidate for a query is the product of the translation model score $P(S|T)$ and the language model score $P(T)$.

2.3 Retrieval Strategies

The Retrieval Strategist (RS) module retrieves documents from a corpus in response to a query. For document retrieval, the RS uses the Lemur 3.0 toolkit [8]. Lemur supports structured queries using operators such as Boolean AND, Synonym, Un-ordered and NOT. An example of a structured query generated by the RS is shown below:

#BAND (キャプテン 市立船橋 チーム サッカー
#SYN (*organization *person))

In deciding how to formulate a structured query, the RS uses an incremental query relaxation technique, starting from an initial query that is highly constrained; the query formulation algorithm searches for all the keywords and data types in close proximity to each other. The priority is based on a function of the likely answer type, keyword type (word, proper name, or phrase) and the inverse document frequency of each keyword. The query is gradually relaxed until the desired number of relevant documents is retrieved [7]. Language-specific corpus indexing is described in Section 3.1 and Section 3.2.

2.4 Information Extraction

For the CLQA1 EC and EJ subtasks, we submitted three formal runs using three different information extractors (IXes): FST IX, Light IX and Combo IX.

FST IX is a trained pattern-based extractor. Light IX uses simple distance-based algorithms on named-entity-tagged corpora. Combo IX uses a combination of the FST IX and Light IX modules. All of the IX modules assume that the answer has been tagged with the appropriate named entity tags in the corpus.

2.4.1 FST IX

The FST IX implements a set of finite state transducers which are modeled for each question type and corresponding answer sets drawn from training data [7]. Note that the pattern descriptions contains labels that are expanded to synonyms which are created manually: e.g. “設立” can expanded to the regular expression “(設立|創立|創設|創業|起業)”.

For each passage retrieved by the RS module, the FST IX module applies all patterns in sequential order, so that patterns with higher precision can be tried earlier; the corresponding matched passages are therefore assigned higher confidence scores.

Table 1 shows an example of the trained FST model (for “Who founded” questions), along with the number of times the pattern extracted the correct answer, the number of times the pattern was matched, and the corresponding precision score.

Table 1. Excerpt of Trained FST Model for Who-Founded Questions

Pattern Description	Correct	Matched	Precision
ANSWERによって設立された	66	336	0.196
KEYWORDのANSWER会長	6	29	0.207
KEYWORDＣＥＯのANSWER	7	26	0.270
KEYWORDはANSWERが創業	44	188	0.234
ANSWERが設立	2	5	0.400

One disadvantage of the FST IX approach is that it requires a significant amount of training data; performance is reduced when there is only a small amount of training data available (for this task, we had 200 and 300 training examples for the EC and EJ subtasks, respectively). For example, if a pattern was matched and the correct answer was returned in just one case, the pattern precision score on the training data will be 1, which may result in overfitting the model to the training data. For this reason, FST IX performance is likely to improve as more training data is made available.

2.4.2 Light IX

The Light IX module uses simple, distanced-based algorithms to find a named entity that matches the expected answer type and is “closest” to all the keywords according to some distance measure. The algorithm considers as answer candidates only those terms that are tagged as named entities which match the desired answer type. The score for an answer candidate a is calculated as follows:

$$Score(a) = \alpha \cdot OccScore(a) + \beta \cdot DistScore(a)$$

Where $\alpha + \beta = 1$, $OccScore$ is the occurrence score and $DistScore$ is the distance score. Both $OccScore$ and $DistScore$ return a number between zero and one, and likewise $Score$ returns a number between zero and one. Usually, α is much smaller than β. The occurrence score formula is:

$$OccScore(a) = \frac{\sum_{i=1}^{n} Exist(k_i)}{n}$$

Where a is the answer candidate and k_i is the i-th keyword, and n is the number of keywords. $Exist$ returns 1 if the i-th keyword exists in the document, and 0 otherwise. The distance score for each answer candidate is calculated according to the following formula:

$$DistScore(a) = \frac{\sum_{i=1}^{n} \frac{1}{Dist(a,k_i)}}{n}$$

This formula produces a score between zero and one. If the i-th keyword does not exist in a document, the equation inside the summation will return zero. If the i-th keyword appears more than once in the document, the one closest to the answer candidate is considered. An additional restriction is that the answer candidate cannot be one of the keywords. The Dist function is the distance measure, which has two definitions:

1. $Dist(a, b) = TokensApart(a, b)$
2. $Dist(a, b) = log(TokensApart(a, b))$

The first definition simply counts the number of tokens between two terms. The second definition is a logarithmic measure. The $TokensApart$ function returns the number of tokens from a to b; if a and b are adjacent, the count is 1; if a and b are separated by one token, the count is 2, and so on. A token can either be a character or a word; for the EC subtask, we used character-based tokenization, whereas for the EJ subtask, we use word-based tokenization. By heuristics obtained from training results, we used the linear $Dist$ measure for EC and logarithmic $Dist$ measure for EJ in the formal run.

2.4.3 Combo IX

The Combo IX module implements a combination of the FST IX and Light IX modules. We first applied the FST IX trained with tight[1] patterns, and subsequently, applied the Light IX for questions where no answer was found using the FST IX. Our underlying assumptions were that a) the first IX module does not produce false-positive answers (i.e., the combination is a form of back-off), and b) the first IX to run should produce fewer answer candidates, with higher precision.

[1]“Tightness” is the term we use for describing how generic a pattern is. For example, pattern “KEYWORD *is* ANSWER” is tighter than pattern “KEYWORD*(,.{,30},)? is* ANSWER”.

2.5 Answer Generator

The task of the Answer Generator (AG) module is to produce a ranked list of answer candidates from the IX output. The AG is designed to normalize answer candidates by resolving representational differences (e.g. in how numbers, dates, names etc. are expressed in surface text). This canonicalization makes it possible to combine answer candidates that differ only in surface form.

Even though the AG module plays an important role in JAVELIN, we did not use the full potential of the AG in our EC and EJ systems due to the lack of certain language-specific resources, more details are discussed in Section 5.3.

3 Corpus Preprocessing

In this section, we discuss some of the language specific issues that are related to named entity tagging in corpus.

3.1 Chinese Specific

We first tagged the Chinese corpus using the Chinese Identifinder [3] from BBN. We used the toolkit directly off-the-shelf, without additional training because of the lack of corpus-specific training data. Chinese Identifinder was used to tag the following named entities: ORGANIZATION, PERSON, LOCATION, ARTIFACT, TIME and PERCENT. For the DATE and MONEY entities, we used our original rule-based tagger. We automatically removed any single-character named entities (for ORGANIZATION, PERSON, and LOCATION), since these are very unlikely in Chinese text and were assumed to be erroneous.

In the end, we analyzed a random document sample to assess the performance of tagging. In general, we find our named entity tagging performed best in LOCATION, DATE, PERCENT, and MONEY; not as well in PERSON; worst in ORGANIZATION and ARTIFACT.

3.2 Japanese Specific

Named entities convey important information for question answering but are difficult to recognize because entity names are less likely to appear in existing MRDs. For example, questions like “When did Sumie Tanaka turn 88?” will be more difficult to answer than questions like “Who is the 26th U.S. president?”, because they contain person names written in romanized form (romaji), which must be mapped to the native representation (e.g., kanji for Japanese names) that is likely to appear in the target language corpus. Foreign person names (e.g. “Bill Clinton”) are mapped to a different representation (katakana).

Japanese corpus was preprocessed first by annotating named entities with bar [2] and chunking morphemes with ChaSen [1]. In addition, for each named entity, we also used ChaSen to tag character readings in kana.

Indexing kana readings in the corpus and querying in kana is a useful strategy for CLQA. Given a question sentence with named entities in romaji, we are able to retrieve related documents by searching for kana terms which are transliterated from romaji. This approach allows us to avoid the much harder task of mapping romaji to kanji characters[2].

On the other hand, transliteration can generate noise when: a) misclassifying an English term as a Japanese romaji term (e.g. home), or b) the mapping is ambiguous (e.g. Okuma Kodo in romaji $\rightarrow$ { *Ookuma Koudou, Oukuma Koudou, Okuma Koudou, ...* in kana }).

4 Result and Analysis

Our overall submission to the NTCIR CLQA1 task included three submissions for the English-Chinese (EC) subtask, and three submissions for the English-Japanese (EJ) subtask. The three runs for each subtask were carried out using the three different IX strategies described in Section 2.4. 200 input questions were provided for each of the subtasks. For each question, only the top answer candidate that was returned by the system was judged. Correct answers that were not properly supported by the returned document were judged to be unsupported answers.

4.1 Formal Run Results

Formal run results are shown in Table 2. The highest accuracies for the EC and EJ subtasks are 7.5% and 10.0%, respectively, and both were achieved using the Light IX.

Table 2. Formal Run Performance

	EC		EJ	
	Corr[a]	Unsup[b]	Corr[a]	Unsup[b]
FST	14 (7.0%)	19 (9.5%)	17 (9.0%)	20 (10.0%)
LIGHT	15 (7.5%)	19 (9.5%)	20 (10.0%)	25 (13.0%)
COMBO	10 (5.0%)	12 (6.0%)	17 (9.0%)	20 (10.0%)

[a]Corr – Answer which are correct and supported
[b]Unsup – Unsupported correct answer

4.2 Overall Analysis

We analyzed the number of successfully returned answers and answer accuracy with respect to answer type. Nine categories of answer type are given in Table 3, together with the corresponding performance of the system using the three different IX strategies.

In the EC subtask, about 70%(139 out of 200) of the questions are about PERSON, LOCATION and ORGANIZATION. In the EJ subtask, there were many

[2]Romaji to kana transliteration is much less ambiguous than romaji to kanji.

fewer questions for these three types. Instead, about half(98 out of 200) of the questions were temporal or numerical questions.

For the EC subtask, all three IX strategies achieved relatively higher accuracy on PERSON and LOCATION questions, but lower accuracy on temporal and numerical questions. For the EJ subtask, the system had much higher accuracy on PERSON, ARTIFACT, LOCATION and ORGANIZATION questions in comparison to EC (Light IX achieved 25.0% accuracy), but because of the higher frequency of temporal and numerical questions in the EJ subtask, the overall accuracy of the EJ system is only slightly higher than the EC system.

To our surprise, both Light IX and FST IX outperformed the Combo IX. There was only one case(PERSON) in the EC subtask where Combo IX out-performed the other two approaches. In the EJ subtask, we observed more cases in which the Combo IX out-performed or matched the highest accuracy of the other two approaches, but there were more cases in which it failed to do so. The possible reasons for the unexpectedly low performance of the Combo IX in the formal run[3] are discussed in Section 5.4 .

4.3 Module-by-Module Analysis

In order to gain different perspectives on the tasks and our systems' performance, a module-by-module analysis was performed. This analysis was based on gold-standard answer data, which also provides information about the documents that contain the correct answer for each question. We judged the QA module by the accuracy of its answer type classification, and the TM module by the accuracy of its keyword translation. For the RS and IX modules, if a correct document or answer is returned, regardless of its ranking, we consider the module to be successful. To separate the effects of errors introduced by earlier modules, we created gold-standard data by manually correcting answer-type and keyword translation errors. We also create "perfect" IX input using the gold-standard document set.

The results are shown in Table 4. Note that because Light IX performed best in the formal run, for both EC and EJ, we will focus our discussion on Light IX in this section. More discussion about Combo IX and FST IX can be found in Section 5.4

4.3.1 QA Performance

The QA module performed well in identifying the answer type in both subtasks. As we can see from the QA_{ATYPE} column in Table 4, the QA achieved 86.5% for the EC subtask and 93.5% for the EJ subtask. An additional analysis of accuracy by answer type is shown in Table 5. Compared to row $TM{+}QA_{ATYPE}$ in Table 4, we can see that further improvement of the answer type accuracy via manual correction did not make a significant difference.

[3]On the training data set, the FST IX and Combo IX performed better than the Light IX, partly because of overfitting in FST IX.

4.3.2 TM Performance

The average precision of translation was 69.3% for the EC subtask and 72.6% for the EJ subtask. By taking advantage of translation by web-mining, we could successfully translate some named entities. Table 7 shows some sample input and output pairs.

4.3.3 RS Performance

The RS module achieved an accuracy of 30.5% in the EC subtask and 44.5% in the EJ subtask, as shown in column *RS* in Table 4. After manual correction of keyword translation errors, we immediately gained over 20.0% accuracy in the RS module performance for both the EC and EJ subtasks, as shown in row *TM* in Table 4. This shows that translation errors have a significant negative impact on keyword-based document retrieval. To further illustrate the difference before and after manual translation of keywords, a CLIR-style analysis of the RS module is provided in Table 6.

For all the questions that showed an improved MRR score after manual correction of keyword translation errors, the TM failed to translate 43 and 88 keywords in the EJ and EC subtasks, respectively. Among these keywords, 65.0% for the EJ subtask and 43.0% for the EC subtask were classified as proper nouns and phrases by the QA module. Most of the proper nouns are person, location and organization names. Referring back to the observation at the beginning of Section 4.2 that the majority of the questions are drawn from these three types, this helps to explain the 20.0% accuracy gain achieved from corrected key term translation. Accurate translation of these types of keywords without adequate context and background knowledge is a challenging problem, even for humans. We did an experiment in which the human translator for the EC subtask was not allowed to reference the official translation of the questions. The translation accuracy was 85.5% when compared to the gold standard translation. This is an on-going research topic which leaves a lot of room for future improvement, and our analysis shows that improvements in this area are likely to have a significant position impact on overall CLQA performance.

4.3.4 IX Performance

In the formal run data (row *None* in Table 4), we observed big accuracy drops at the RS module and after the IX module for both the EC and EJ subtasks, and bigger accuracy drops at the IX module for the EJ subtask. The drop in RS accuracy is expected, but the

Table 3. Accuracy by Answer Type

	EC							EJ						
Answer type	#[a]	F[b]		L[c]		C[d]		#[a]	F[b]		L[c]		C[d]	
PERSON	79	6	7.6%	6	7.6%	7	8.9%	34	5	14.7%	5	14.7%	2	5.9%
LOCATION	45	4	8.9%	4	8.9%	3	6.7%	34	4	11.8%	4	11.8%	6	17.6%
ORGANIZATION	15	1	6.7%	1	6.7%	0	0.0%	13	1	7.7%	3	23.1%	2	15.4%
ARTIFACT	26	1	3.9%	2	7.7%	0	0.0%	21	1	4.8%	3	14.3%	1	4.8%
DATE	19	2	10.5%	2	10.5%	0	0.0%	25	1	4.0%	4	16.0%	1	4.0%
TIME	1	0	0.0%	0	0.0%	0	0.0%	14	0	0.0%	0	0.0%	0	0.0%
MONEY	5	0	0.0%	0	0.0%	0	0.0%	20	3	15.0%	1	5.0%	3	15.0%
NUMEX	10	0	0.0%	0	0.0%	0	0.0%	31	1	3.2%	0	0.0%	1	3.2%
PERCENT	0	0	-	0	-	0	-	8	1	12.5%	0	0.0%	1	12.5%
Overall	200	14	7.0%	15	7.5%	10	5.0%	200	17	8.5%	20	10.0%	17	8.5%

[a]# stands for the number of questions for each answer type
[b]F stands for FST IX
[c]L stands for Light IX
[d]C stands for Combo IX

Table 4. Performance from Partially Gold Standard Input

	Gold Standard	QAATYPE[a]	TM[b]	RS[c]	IX[d] (MRR[e])	Accu[f] (Unsup[g])
EC	None	86.5%	69.3%	30.5%	30.0% (0.130)	7.5% (9.5%)
	TM	86.5%	—	57.5%	50.0% (0.254)	9.5% (20.0%)
	TM+QAATYPE	—	—	57.5%	50.5% (0.260)	9.5% (20.5%)
	TM+QAATYPE+RS	—	—	—	63.0% (0.489)	41.0% (43.0%)
EJ	None	93.5%	72.6%	44.5%	31.5% (0.116)	10.0% (12.5%)
	TM	93.5%	—	67.0%	41.5% (0.154)	9.5% (15.0%)
	TM+QAATYPE	—	—	68.0%	45.0% (0.164)	10.0% (15.5%)
	TM+QAATYPE+RS	—	—	—	51.5% (0.381)	32.0% (32.5%)

[a]Average presicion of answer-type detection
[b]Average presicion of keyword translation over 200 formal run questions
[c]Average precision of document retrieval. Counted if correct document was ranked between 1st–15th
[d]Average precision of answer extraction. Counted if correct answer was ranked between 1st–100th
[e]The MRR measure of IX performance, calculated by averaging the sum of the reciprocal of each answer's rank
[f]Overall accuracy of the system
[g]Accuracy including unsupported answers

	EC			EJ		
A-type	# of Q	correct	%	# of Q	correct	%
PER	79	64	81%	34	34	100%
LOC	45	44	98%	34	33	97%
ORG	15	12	80%	13	8	62%
ARTI	27	23	85%	21	19	90%
DATE	18	18	100%	25	25	100%
TIME	1	1	100%	14	13	93%
MONEY	5	4	80%	20	18	90%
NUMEX	10	7	70%	31	29	94%
PCNT	0	0	-	8	8	100%
Sum	200	173	86.5%	200	187	93.5%

Table 5. QA Performance by Answer Type

	No man-trans		With man-trans	
Rank	EC	EJ	EC	EJ
1	11	29	44	52
2-5	30	31	38	53
6-9	14	12	20	15
10-15	6	17	13	14
no match	139	111	85	66
Sum	200	200	200	200
MRR	0.12	0.22	0.31	0.37
Success Rate	30.5%	44.5%	57.5%	67.0%

Table 6. RS Evaluation: Number of correct documents by retrieved rank

Table 7. Examples of Named Entity Translation

English	Chinese	English	Japanese
Milosevic	米洛舍維奇	Ichiritsu Funabashi	市立船橋
WTO	世界貿易組織	Suzakumon	朱雀門
NMD	國家飛彈防禦	Sumie Tanaka	田中澄江
NHK	日本廣播協會	Werner Spies	ヴェルナー・シュピース
Pakistan Liberation Organization	巴基斯坦解放組織	Leisure Development Center	余暇開発センター

difference between Light IX performance in the EC and EJ subtasks is surprising. After eliminating errors carried over from earlier modules, the IX in the EC and the EJ subtasks show a performance difference of 11.5%(63.0%-51.5%); see row $TM+QA_{ATYPE}+RS$.

The Light IX used the same algorithm in the EC and EJ subtasks, but with different distance measure functions and different α and β parameter settings. The IX in the EC subtask achieved a higher MRR score in all cases, and better accuracy in most cases (except in the formal run). But the EC system had worse overall accuracy than the EJ system, except in the $TM+QA_{ATYPE}+RS$ case. We cannot conclude at this point which Light IX setting is more effective, because other factors such as corpus tagging precision differences are involved. In general however, we found the Light IX in the EC system to be more accurate and produced more answer candidates.

Because the answer validation function was not yet implemented in the AG module to filter out noise, the overall accuracy of the EC and EJ systems is much lower than the accuracy of the IX module in both cases. We can see the degradation caused by the noise in IX output by examining the $TM+QA_{ATYPE}$ row and $TM+QA_{ATYPE}+RS$ row in the EC part of Table 4. The accuracy of the IX differs only by 12.5%(63.0%-50.5%), but this measure does not take into account noise in other answer candidates. The effect of the noise is delayed until the output of the AG module, where a 31.5%(41.0%-9.5%) difference in overall answer accuracies and a 22.5%(43.0%-20.5%) difference including unsupported answers are seen.

As the performance of the RS increased after manual correction of keyword translation errors, the IX module showed a similar increase in performance of 20.0%(50.0%-30.0%) in the EC subtask and 10.0%(41.5%-31.5%) in the EJ subtask. But as we increase the accuracy of RS from 57.5% in EC and 67.0% in EJ to 100.0%, by manually creating "perfect" RS output, the performance of the IX module did not increase as much. The upper bound on IX performance was 63.0% for the EC subtask and 51.5% for the EJ subtask.

One observation regarding the differences between the EC and EJ corpora drew our attention. We noticed that although both subtasks had the same number of questions(200), the EC corpus had more documents (641) that contained an answer for some question(s) when compared to the EJ corpus (314 documents). In order to determine if this difference in document-to-question density contributed to the performance difference in the IX modules, we did some additional analysis. Although the document-to-question density differs in the corpora for the EC and EJ subtasks, we found that the numbers of retrieved correct documents were very close – 133 documents in the EC task versus 134 documents in the EJ subtask. Therefore we can conclude that the difference we observed in the corpora did not affect the performance of the IX module in our evaluation.

5 Issues and Proposed Solutions

In this section, we discuss some issues that we observed through our analysis, and propose possible ways to improve the system.

5.1 Multiple Representations and Canonicalization

We noticed that multiple representations of the same meaning sometimes occur in the corpus, and if we do not treat them identically, it will affect retrieval and extraction accuracy. A few examples are shown below. Example (1) shows how the year is often written in different ways in Japanese documents. Example (2) shows how the dot symbol "Nakaguro" sometimes separates syllables in words written in katakana. Example (3) is an example of orthographic variation in the use of katakana to represent a proper name ("Pinatubo").

One way to tackle this issue would be to use Lemur's "SYN"(synonym) operator. To handle multiple representations of the year, we are able to generate and index different representations based on predefined rules. For the "Nakaguro" problem, we may be able to insert a Nakaguro between syllables (if we can detect syllable boundaries using a lexicon). The last problem is rather difficult to handle, because we do not know whether automatically-generated representational variants would have the same meaning as original; any solution to this problem must take care not to increase noise in document retrieval.

(1) (a) １９９８年	(2) (a) テルアビブ	(3) (a) ピナツボ
(b) ９８年	(b) テル・アビブ	(b) ピナトゥボ
(c) 九八年		

5.2 Using answer subtype information in Light IX

One type of information that the QA module produces is the answer subtype information. Table 8 shows some examples of questions and the subtype produced by the QA module.

Table 8. Subtype examples

Question	A-type	Subtype
Which year did Nelson defeat the French navy?	temporal	year
What is the date of the World Asthma Day?	temporal	date
What is the height in meters of the Eiffel Tower?	numerical	length
How much in damages did Steven Herman claim?	numerical	money

This information is not being used in the current Light IX module. To estimate the reliability of the subtype information, we collected all the subtype information produced by QA in both the training and formal question set, for both EC and EJ subtasks, and judged them by hand. There are in total 605 questions in which subtype is given, and 586 of them(97%) were judged correct.

Unlike FST IX, Light IX does not do any intelligent semantic processing, and merely relies on corpus tags and distance-based matching. Some of the subtypes are too specific, e.g. “shrine” as a subtype of ARTIFACT and “baseball fielder” as a subtype of PERSON. Our tagger cannot tag these subtype informations in the corpus, and therefore are not very useful for the Light IX. But we found that for numerical and temporal questions, subtypes such as ‘year’ and ‘percentage’ are very informative and could be easily tagged in the corpus. We plan to re-tag the corpus with numerical and temporal subtypes and fine tune the parameters used in Light IX as a prelude to additional experimentation.

5.3 Validation in the AG Module

In the JAVELIN system for English, the AG module has a very important function – answer validation. Answer validation increases answer precision by eliminating irrelevant answers produced by earlier modules. It also boosts confidence scores for the correct answer candidates, which raises their ranking in the candidate answer list. Quite often the correct answer is extracted, but not ranked as the top answer in our system. Since only the topmost answer candidate is judged in CLQA1, the answer validation step becomes crucial. However, the validation process in JAVELIN utilizes a set of knowledge-based and web-based approaches, developed initially for English, which are not readily applied to the CLQA task. To provide answer validation for CLQA is one of the key tasks in our future work.

5.4 Integrating the JAVELIN Planner

Table 2 showed that the Combo IX module did not work as effectively as we expected. One of the reasons is that our initial assumption that the FST IX would not produce many false-positive answers did not hold in the formal run.

It is difficult to decide how much recall should be sacrificed for accuracy when one IX module is used in combination with others. In our experiment, simply combining the FST IX and the Light IX without adjusting the “tightness” did not provide any improvement. But for some question types, we observed increases in performance when using the Combo IX. This suggests that a selecting the best combination of IX modules per question type might achieve better overall performance. The JAVELIN system for English incorporates a Planner module which can select among the set of available IX modules at run-time [5]. One of our future tasks involves adapting the Planner for use in CLQA.

6 Conclusion

Our analysis of per-module performance from gold-standard input shows that the QA module and the RS module are already performing fairly well, but there is still room in the IX module and the AG module for future improvement. Also, we found that keyword translation accuracy greatly affects overall performance on the CLQA task. We also discussed and presented solutions to issues such as the canonicalization problem in Japanese, the use of answer subtypes, and answer validation.

Acknowledgements

This work was supported in part by the Advanced Research and Development Activity (ARDA)’s Advanced Question Answering for Intelligence (AQUAINT) Program. We thank Matt Bilotti, Kerry Hannan, Dave Svoboda, Jeongwoo Ko for their assistance in building the CLQA JAVELIN system. We also thank Eric Nyberg for his help in the final preparation of this paper.

References

[1] M. Asahara and Y. Matsumoto. Extended models and tools for high-performance part-of-speech tagger. *In Proceedings of COLING 2000*, July 2000.

[2] M. Asahara and Y. Matsumoto. Japanese named entity extraction with redundant morphological analysis. *HLT-NAACL*, pages 8–15, 2003.

[3] D. Bikel, S. Miller, R. Schwartz, , and R. Weischedel. Nymble: a high-performance learning name-finder. *In Fifth Conference on Applied Natural Language Processing*, pages 194–201, 1997.

[4] P. Brown, J. Cocke, S. D. Pietra, V. D. Pietra, F. Jelinek., J. Lafferty, R. Mercer, and P. Roossin. A statistical approach to machine translation. *In Computational Linguistics*, 16(2):38–45, 1990.

[5] L. S. Hiyakumoto. Planning in the javelin qa system. *Carnegie Mellon Computer Science Technical Report CMU-CS-04-132*, 2004.

[6] A. Korhonen and E. Briscoe. Extended lexical-semantic classification of english verbs. *Proceedings of the HLT/NAACL0̂4 Workshop on Computational Lexical Semantics*, pages 38–45, 2004.

[7] E. Nyberg, T. Mitamura, J. Callan, J. Carbonell, R. Frederking, K. Collins-Thompson, L. Hiyakumoto, Y. Huang, C. Huttenhower, S. Judy, J. Ko, A. Kupsc, L. V. Lita, V. Pedro, D. Svoboda, and B. V. Durme. The javelin question-answering system at trec 2003: A multi-strategy approach with dynamic planning. *In Proceedings of TREC 12*, November 2003.

[8] P. Ogilvie and J. Callan. Experiments using the lemur toolkit. *In Proceedings of the 2001 Text REtrieval Conference (TREC 2001)*, pages 103–108, 2001.