Keyword Translation Accuracy and Cross-Lingual Question Answering in Chinese and Japanese

Teruko Mitamura
Carnegie Mellon University
Pittsburgh, PA USA
teruko@cs.cmu.edu

Mengqiu Wang
Carnegie Mellon University
Pittsburgh, PA USA
mengqiu@cs.cmu.edu

Hideki Shima
Carnegie Mellon University
Pittsburgh, PA USA
hideki@cs.cmu.edu

Frank Lin
Carnegie Mellon University
Pittsburgh, PA USA
frank+@cs.cmu.edu

Abstract

In this paper, we describe the extension of an existing monolingual QA system for English-to-Chinese and English-to-Japanese cross-lingual question answering (CLQA). We also attempt to characterize the influence of translation on CLQA performance through experimental evaluation and analysis. The paper also describes some language-specific issues for keyword translation in CLQA.

1 Introduction

The JAVELIN system is a modular, extensible architecture for building question-answering (QA) systems (Nyberg, et al., 2005). Since the JAVELIN architecture is language-independent, we extended the original English version of JAVELIN for cross-language question answering (CLQA) in Chinese and Japanese. The same overall architecture was used for both systems, allowing us to compare the performance of the two systems. In this paper, we describe how we extended the monolingual system for CLQA (see Section 3). Keyword translation is a crucial element of the system; we describe our translation module in Section 3.2. In Section 4, we evaluate the end-to-end CLQA systems using three different translation methods. Language-specific translation issues are discussed in Section 5.

2 Javelin Architecture

The JAVELIN system is composed of four main modules: the Question Analyzer (QA), 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, assigning the appropriate answer type to the question, and producing a set of keywords. The RS module is responsible for finding documents containing answers to the question, using keywords produced by the QA module. The IX module finds and extracts answers from the documents based on the answer type, and then produces a ranked list of answer candidates. The AG module normalizes and clusters the answer candidates to rerank and generate a final ranked list. The overall monolingual architecture is shown in Figure 1.

3 Extension for Cross-Lingual QA

Because of JAVELIN's modular design, significant changes to the monolingual architecture were not required. We customized the system in order to handle Unicode characters and "plug in" cross-lingual components and resources.

For the Question Analyzer, we created the Keyword Translator, a sub-module for translating keywords. The Retrieval Strategist was adapted to search in multilingual corpora. The Information Extractors use language-independent extraction algorithms. The Answer Generator uses language-specific sub-modules for normalization, and a language-independent algorithm for answer ranking. The overall cross-lingual architecture is shown in Figure 2. The rest of this section explains the details of each module.

3.1 Question Analyzer

The Question Analyzer (QA) is responsible for extracting information from the input question in order to formulate a representation of the

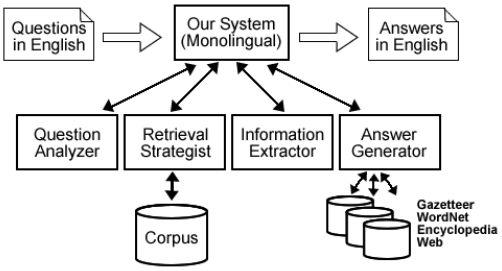

Figure1: Javelin Monolingual Architecture

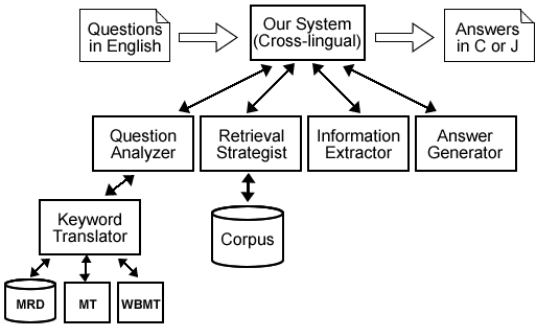

Figure2: Javelin Architecture with Cross-Lingual Extension

information required to answer the question. Input questions are processed using the RASP parser (Korhonen and Briscoe, 2004), 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 with correct answers. 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.

We extended the QA module with a keyword translation sub-module, so that translated keywords can be used to retrieve documents from multilingual corpora. This straightforward approach has been used by many other CLQA systems. An alternative approach is to first translate the whole question sentence from English to the target language, and then analyze the translated question. Our reasons for favoring keyword translation are two-fold. First, to translate the question to the target language and analyze it, we would have to replace the English NLP components in the Question Analyzer with their counterparts for the target language. In contrast, keyword translation decouples the question analysis from the translation, and requires no language specific resources during question analysis. The second reason is that machine translation is not perfect, and therefore the resulting translation(s) for the question may be incomplete or ungrammatical, thus adding to the complexity of the analysis task. One could argue that when translating the full sentence instead of just the keywords, we can better utilize state-of-art machine translation techniques because more context information is available. But for our application, an accurate translation of functional words (such as prepositions or conjunctions) is less important. We focus more on words that carry more content information, such as verbs and nouns. We will present more detail on the use of contextual information for disambiguation in the next section. In some recent work (Kwok, 2005, Mori and Kawagishi, 2005), researchers have combined these two approaches, but to date no studies have compared their effectiveness.

3.2 Translation Module

The Translation Module (TM) is used by the QA module to translate keywords into the language of the target corpus. Instead of combining multiple translation candidates with a disjunctive query operator (Isozaki et al., 2005), 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) (Nagata et al., 2001, Li et al., 2003). For translation from English to Japanese, we used two MRDs, eight MTs and one WBMT. If none of them return a translation, the word is transliterated into kana for Japanese (for details on transliteration, see Section 5.2). For translation from English to Chinese, we used one MRD, three MTs and one WBMT. After gathering all possible translations for every keyword, the TM uses a noisy channel model to select the best combination of translated keywords. The TM estimates model statistics using the World Wide Web. Details of the translation selection method are described in the rest of this subsection.

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 (Brown et al., 1990). We adopt this model for disambiguating 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:

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

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

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

We can now reduce the equation 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:

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

Estimating Probabilities using the World Wide Web: For estimating the probabilities of the translation model and the language model, we chose to gather statistics from the World Wide Web. There are three advantages in utilizing the web for gathering translation statistics: 1) it contains documents written in many different languages, 2) it has high coverage of virtually all types of words and phrases, and 3) it is constantly updated. However, we also note that the web contains a lot of noisy data, and building up web statistics is time-consuming unless one has direct access to a web search index.

Estimating Translation Model Probabilities: We make an assumption that terms that are translations of each other co-occur more often in mixed-language web pages than terms that are not translations of each other. This assumption is analogous to Turney's work on the co-occurrence of synonyms (Turney, 2001). We then define the translation probability of each keyword translation as:

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

Where s_i is the *i*-th term in the source language and $t_{i,j}$ is the *j*-th translation candidate for s_i. Let hits be a number of web pages retrieved from a certain search engine. $co(s_i, t_{i,j})$ is the hits given a query s_i and $t_{i,j}$., where *log* is applied to adjust the count so that translation probabilities can still be comparable at higher counts.

Estimating Language Model Probabilities: In estimating the language model, we simply obtain hits given a conjunction of all the candidate terms in the target language, 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)*.

Smoothing and Pruning: As with most statistical calculations in language technologies, there is a data sparseness problem when calculating the language model score. Also, because statistics are gathered real-time by accessing a remote search engine via internet, it can take a long time to process a single query when there is a large number of translation candidates. We describe methods for smoothing the language model and pruning the set of translation candidates below.

The data sparseness problem occurs when there are many terms in the query, and the terms are relatively rare keywords. When calculating the language model score, it is possible that none of the translation candidates appear on any web page. To address this issue, we propose a "moving-window smoothing" algorithm:

- When the target keyword co-occurrence count with *n* keywords is below a set threshold for all of the translation candidates, we use a moving window of size *n-1* that "moves" through the keywords in sequence, splitting the set of keywords into two sets, each with *n-1* keywords.

- If the co-occurrence count of all of these sets of keywords is above the threshold, return the product of the language model

score of these two sets as the language model score.

- If not, decrease the window and repeat until either all of the split sets are above the threshold or *n = 1*.

The moving window smoothing technique gradually relaxes the search constraint without losing the "connectivity" of keywords (there is always overlap in the split parts) before finally backing off to just the individual keywords. However, there are two issues worth noting with this approach:

1. "Moving-window smoothing" assumes that keywords that are next to each other are also more semantically related, which may not always be the case.

2. "Moving-window smoothing" tends to give the keywords near the middle of the question more weight, which may not be desirable.

A better smoothing technique may be used with trying all possible "splits" at each stage, but this would greatly increase the time cost. Therefore, we chose the moving-window smoothing as a trade-off between a more robust smoothing technique that tries all possible split combinations and no smoothing at all.

The set of possible translation candidates is produced by creating all possible combinations of the translations of individual keywords. For a question with *n* keywords and an average of *m* possible translations per keyword, the number of possible combinations is m^n. This quickly becomes intractable as we have to access a search engine at least m^n times just for the language model score. Therefore, pruning is needed to cut down the number of translation candidates. We prune possible translation candidates twice during each run, using early and late pruning:

1. Early Pruning: We prune possible translations of the individual keywords before combining them to make all possible translations of a query. We use a very simple pruning heuristic based on target word frequency using a word frequency list. Very rare translations produced by a resource are not considered.

2. Late Pruning: We prune possible translation candidates of the entire set of keywords after calculating translation probabilities. Since the calculation of the translation probabilities requires little access to the web, we can calculate only the language model score for the top *N* candidates with the highest translation score and prune the rest.

An Example of English to Chinese Keyword Translation Selection: Suppose we translate the following question from English to Chinese.

"What if Bush leaves Iraq?"

Three keywords are extracted: “Bush”, “leaves”, and “Iraq.” Using two MT systems and an MRD, we obtain the following translations:

	i=1	i=2	i=3
Source	Bush	leaves	Iraq
Target j=1	灌木	离去	伊拉克
Target j=2	布什	叶子	

Table 1. E-C Keyword Translation

"Bush" and *"leaves"* both have two translations because they are ambiguous keywords, while *"Iraq"* is unambiguous. Translation (1,1) means *bush* as in a *shrub*, and translation (1,2) refers to the person named *Bush*. Translation (2,1) is the verb *"to go away"*, and translation (2,2) is the noun for *leaf*. Note that we would like translation (1,2) and translation (2,1) because they match the sense of the word intended by the user. Now we can create all possible combinations of the keywords in the target language:

"灌木 离去 伊拉克"
"灌木 叶子 伊拉克"
"布什 离去 伊拉克"
"布什 叶子 伊拉克"

Query	"Bush" "灌木"	"Bush" "布什"	"leaves" "离去"	"leaves" "叶子"	"Iraq" "伊拉克"
hits	3790	41100	5780	7240	24500

Table 2. Translation Pair Page Counts

Candidate	Translation Score
"灌木 离去 伊拉克"	0.215615
"灌木 叶子 伊拉克"	0.221219
"布什 离去 伊拉克"	0.277970
"布什 叶子 伊拉克"	0.285195

Table 3. Translation Scores

By calculating hits, we obtain the statistics and the translation scores shown in Table 2 and 3. Now we can proceed to use the search engine to obtain language model statistics, which we use to obtain the language model. Then, together with the translation model score, we calculate the overall score[1].

Query	灌木	布什	离去	叶子	伊拉克
hits	428K	459K	1490K	1100K	9590K

Table 4. Individual Term Page Counts

Query	hits
"灌木 离去 伊拉克"	1200
"灌木 叶子 伊拉克"	455
"布什 离去 伊拉克"	17300
"布什 叶子 伊拉克"	2410

Table 5. Target Language Query Page Counts

Cand	Translation	Language	Overall
灌木 离去 伊拉克	2.1562E-1	1.0428E-4	2.2483E-5
灌木 叶子 伊拉克	2.2122E-1	4.0925E-5	9.0533E-6
布什 离去 伊拉克	2.7797E-1	1.4993E-3	4.1675E-4
布什 叶子 伊拉克	2.8520E-1	2.1616E-4	6.1649E-5

Table 6. Translation Score, Language Model Score, and Overall Score

As shown in Table 6, we select the most probable combination of translated keywords with the highest overall score (the third candidate), which is the correct translation of the English keywords.

3.3 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 (Ogilvie and Callan, 2001). Lemur supports structured queries using operators such as Boolean AND, Synonym, Ordered/Un-Ordered Window and NOT. An example of a structured query is shown below:

```
#BAND( #OD4(邪馬台国 王朝)
       女王
       #SYN(*organization *person) )
```

In formulating a structured query, the RS uses an incremental relaxation technique, starting from an initial query that is highly constrained; the 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.

3.4 Information Extraction

In the JAVELIN system, the Information Extractor (IX) is not a single module that uses one extraction algorithm; rather, it is an abstract interface which allows different information extractor implementations to be plugged into JAVELIN. These different extractors can be used to produce different results for comparison, or the results of running them all in parallel can be merged. Here we will describe just one of the extractors, the one which is currently the best algorithm in our CLQA experiment: the Light IX.

The Light IX module uses simple, distance-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 α + β = 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

[1] For simplicity, we don't apply smoothing and pruning.

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 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 E-C, we used character-based tokenization, whereas for the E-J, we use word-based tokenization. By heuristics obtained from training results, we used the linear *Dist* measure for E-C and logarithmic *Dist* measure for E-J in the evaluation.

This algorithm is a simple statistical approach which requires no language-specific external tools beyond word segmentation and a named-entity tagger. It is not as sophisticated as other approaches which perform deep linguistic analysis, but one advantage is faster adaptation to multiple languages. In our experiments, this simple algorithm performs at the same level as a FST-based approach (Nyberg, et al. 2005).

3.5 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, etc. are expressed in 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 its full potential in our E-C and E-J systems, since we lacked some language-specific resources required for multilingual answer merging.

4 Evaluation and Effect of Translation Accuracy

To evaluate the effect of translation accuracy on the overall performance of the CLQA system, we conducted several experiments using different translation methods. Three different runs were carried out for both the E-C and E-J systems, using the same 200-question test set and the document corpora provided by the NTCIR CLQA task. The first run was a fully automatic run using the original translation module in the CLQA system; the result is exactly same as the one we submitted to NTCIR5 CLQA. For the second run, we manually translated the keywords that were selected by the Question Analyzer module. This translation was done by looking at only the selected keywords, but not the original question. For both E-C and E-J tasks, the NTCIR organizers provided the translations for the English questions, which we assume are the gold-standard translations. Taking advantage of this resource, in the third run we simply looked up the corresponding term for each English keyword from the gold-standard translation of the question. The results for these runs are shown in Table 7 and 8 below.

	Translation Accuracy	Top1		Top1+U	
Run 1	69.3%	15	(7.5%)	23	(11.5%)
Run 2	85.5%	16	(8.0%)	31	(15.5%)
Run 3	100%	18	(9.0%)	38	(19.0%)

Table 7. Effect of Translation (E-C)

	Translation Accuracy	Top1		Top1+U	
Run 1	54.2%	20	(10.0%)	25	(12.5%)
Run 2	81.2%	19	(9.5%)	30	(15.0%)
Run 3	100%	18	(9.0%)	31	(15.5%)

Table 8. Effect of Translation (E-J)

We found that in the NTCIR task, the supported/correct document set was not complete. Some answers judged as unsupported were indeed well supported, but the supporting document did not appear in NTCIR's correct document set. Therefore, we think the Top1+U column is more informative for this evaluation. From Table 7 and 8, it is obvious that the overall performance increases as translation accuracy

increases. From Run1 to Run2, we eliminated all the overt translation errors produced by the system, and also corrected word-sense errors. Then from Run2 to Run3, we made different lexical choices among the seemingly all correct translations of a word. This type of inappropriateness cannot be classified as an error, but it makes a difference in QA systems, especially at the document retrieval stage. For example, the phrase "Kyoto Protocol" can have two valid translations: *京都協議* or *京都議定書*. Both translations would be understandable to a human, but the second translation will appear much more frequently than the first one in the document set. This type of lexical choice is hard to make, because we would need either subtle domain-specific knowledge, or knowledge about the target corpus; neither is easily obtainable.

Comparing Run 1 and 3 in Table 8, we see that improving keyword translation had less overall impact on the E-J system. Information extraction (including named entity identification) did not perform as well in E-J. We also compared the translation effect on cross-lingual document retrieval (Figure 3). As we can see, Run 3 retrieved supporting documents more frequently in rank 1 than in Run 1 or 2. From these preliminary investigations, it would seem that information extraction and/or answer generation must be improved for English-Japanese CLQA.

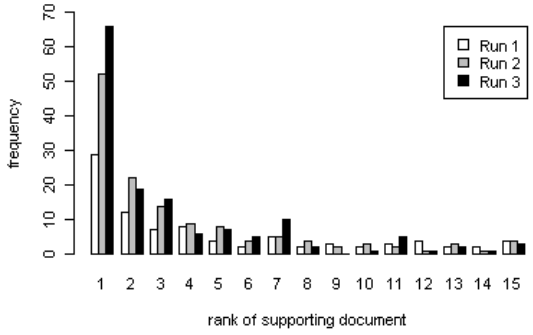

Figure3: Comparison of three runs: Cross-lingual document retrieval performance in E-J

5 Translation Issues

In this section, we discuss language specific keyword translation issues for Chinese and Japanese CLQA.

5.1 Chinese

One prominent problem in Chinese keyword translation is word sense disambiguation. In question answering systems, the translation results are used directly in information retrieval, which exhibits a high dependency on the lexical form of a word but not so much on the meaning. In other words, having a different lexical form from the corresponding term in corpora is the same as having a wrong translation. For example, to translate the word "bury" into Chinese, our system gives a translation of ***埋***, which means "bury" as the action of digging a hole, hiding some items in the hole and then covering it with earth. But the desired translation, as it appears in the document is ***葬***, which means "bury" too, but specifically for burial in funerals.

Even more challenging are regional language differences. In our system, for example, the corpora are newswire articles written in Traditional Chinese from Taiwan, and if we use an MT system that produces translations in Simplified Chinese followed by conversion to Traditional Chinese, we may run into problems. The MT system generates Simplified Chinese translations first, which may suggest that the translation resources it uses were written in Simplified Chinese and originate from mainland China. In mainland China and in Taiwan, people commonly use different words for describing the same thing, especially for proper nouns like foreign names. Table 9 lists some examples. Therefore if the MT system generates its output using text from mainland China, it may produce a different word than the one used in Taiwan, which may not appear in the corpora. This could lead to failure in document retrieval.

English	Mainland China	Taiwan
Band	樂隊	樂團
Computer Game	電腦遊戲	電玩
World Guinness Record	吉尼斯世界紀錄	金氏世界紀錄
The Catcher in the Rye	稻田裏的守望者	麥田捕手
Nelson	奈爾森	納爾遜
Salinger	塞林格	沙林傑
Creutzfeldt Jakob Disease	人類狂牛症	賈庫氏症
Luc Besson	呂克 貝松	盧貝松
Pavarotti	帕瓦洛蒂	帕華洛帝

Table 9. Different Translation in Chinese

5.2 Japanese

Representational Gaps: One of the advantages of using structured queries and automatic query formulation in the RS is that the system is able to handle slight representational gaps between a

translated query and corresponding target words in the corpus.

For example, *Werner Spies* appears as *ヴェルナー ・ シュピース* in our Japanese preprocessed corpus and therefore *ヴェルナー シュピース*, which is missing a dot between last and first name, is a wrong translation if our retrieval module only allows exact match. Lemur supports an *Ordered Distance Operator* where the terms within a *#ODN* operator must be found within *N* words of each other in the text in order to contribute to the document's belief value. This enables us to bridge the representational gaps; such as when *#OD1(ヴェルナー シュピース)* does not match any words in the corpus, *#OD2(ヴェルナー シュピース)* is formulated in the next step in order to capture *ヴェルナー ・ シュピース*.

Transliteration in WBMT: After detecting Japanese nouns written in romaji (e.g. *Funabashi*), we transliterated them into hiragana for a better result in WBMT. This is because we are assuming higher positive co-occurrence between kana and kanji (i.e. *ふなばし* and *船橋*) than between romaji and kanji (i.e. *funabashi* and *船橋*). When there are multiple transliteration candidates, we iterate through each candidate.

Document Retrieval in Kana: Suppose we are going to transliterate *Yusuke*. This romaji can be mapped to kana characters with relatively less ambiguity (i.e. ゆすけ, ゆうすけ), when compared to their subsequent transliteration to kanji (i.e. *雄介*, *祐介*, *佑介*, *勇介*, *雄輔* etc.). Therefore, indexing kana readings in the corpus and querying in kana is sometimes a useful technique for CLQA, given the difficulty in converting romaji to kana and romaji to kanji.

To implement this approach, the Japanese corpus was first preprocessed by annotating named entities and by chunking morphemes. Then, we annotated a kana reading for each named entity. At query time, if there is no translation found from other resources, the TM transliterates romaji to kana as a back-off strategy.

6 Conclusion

We described how we extended an existing monolingual (English) system for CLQA (English to Chinese and English to Japanese), including a translation disambiguation technique which uses a noisy channel model with probability estimations using web as corpora. We discussed the influence of translation accuracy on CLQA by presenting experimental results and analysis. We concluded by introducing some language-specific issues for keyword translation from English to Chinese and Japanese which we hope to address in ongoing research.

Acknowledgements

This work is supported by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligent (AQUAINT) Program.

References

Brown, P., J. Cocke, S.D. Pietra, V.D. Pietra, F. Jelinek., J. Lafferty, R. Mercer, and P. Roossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):38—45.

Isozaki, H., K. Sudoh and H. Tsukada. 2005. NTT's Japanese-English Cross-Language Question Answering System. In *Proceedings of the NTCIR Workshop 5 Meeting,* pages 186-193.

Korhonen, A. and E. Briscoe. 2004. Extended Lexical-Semantic Classification of English Verbs. *Proceedings of the HLT/NAACL '04 Workshop on Computational Lexical Semantics*, pages 38-45.

Kwok, K., P. Deng, N. Dinstl and S. Choi. 2005. NTCIR-5 English-Chinese Cross Language Question-Answering Experiments using PIRCS. In *Proceedings of the NTCIR Workshop 5 Meeting.*

Li, Hang, Yunbo Cao, and Cong Li. 2003. Using Bilingual Web Data To Mine and Rank Translations, *IEEE Intelligent Systems 18(4)*, pages 54-59.

Mori, T. and M. Kawagishi. 2005. A Method of Cross Language Question-Answering Based on Machine Translation and Transliteration. In *Proceedings of the NTCIR Workshop 5 Meeting.*

Nagata, N., T. Saito, and K. Suzuki. 2001. Using the Web as a Bilingual Dictionary, In *Proceedings of ACL 2001 Workshop Data-Driven Methods in Machine Translation,* pages 95-102

Nyberg, E., R. Frederking, T. Mitamura, J. M. Bilotti, K. Hannan, L. Hiyakumoto, J. Ko, F. Lin, L. Lita, V. Pedro, A. Schlaikjer. 2005. JAVELIN I and II in TREC2005. *In Proceedings of TREC 2005.*

Ogilvie, P. and J. Callan. 2001. Experiments Using the Lemur Toolkit. In *Proceedings of the 2001 Text REtrieval Conference (TREC 2001)*, pages 103-108.

Turney, P.D. 2001, Mining the Web for synonyms: PMI-IR versus LSA on TOEFL, *Proceedings of the Twelfth European Conference on Machine Learning*, pages 491-502.