Modular Approach to Error Analysis and Evaluation for Multilingual Question Answering

Hideki Shima, Mengqiu Wang, Frank Lin, Teruko Mitamura

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213 USA
{hideki, mengqiu, frank+, teruko}@cs.cmu.edu

Abstract

Multilingual Question Answering systems are generally very complex, integrating several sub-modules to achieve their result. Global metrics (such as average precision and recall) are insufficient when evaluating the performance of individual sub-modules and their influence on each other. In this paper, we present a modular approach to error analysis and evaluation; we use manually-constructed, gold-standard input for each module to obtain an upper-bound for the (local) performance of that module. This approach enables us to identify existing problem areas quickly, and to target improvements accordingly.

1. Introduction

In this paper, we present a new approach for the evaluation of Multilingual Question Answering (MLQA) systems. Our focus is the JAVELIN MLQA system for factoid questions, which integrates multiple modules in a sequential pipeline with no backtracking or dynamic planning (Lin et al., 2005). The system requires complex integration of several modules. In order to evaluate our system, we analyzed the performance of each module on the evaluation data from the NTCIR CLQA1 task[1]. We created gold-standard data (perfect input) for each module, in order to establish performance upper bounds for each module. Our analysis allows us not only to identify several research issues, but also to compare the performance of our system across different languages (English-Chinese and English-Japanese) on a per-module basis.

For evaluating the performance of the same system handling different languages, modular analysis is also useful in identifying language-specific issues in individual modules.

Since our evaluation focuses on the performance of the system, it is a form of *information-based* evaluation rather than *utility-based* or *architectural* evaluation (Nyberg & Mitamura, 2002). We adopted a fully-automatic, information-based approach to support regular batch evaluation during development and maintenance of the system.

2. Javelin Architecture

Our JAVELIN MLQA system consists of five modules: Question Analyzer (QA), Translation Module (TM), Retrieval Strategist (RS), Information eXtractor (IX) and Answer Generator (AG). Input question sentences in English are processed by these modules in the order listed above. The answer candidates are returned in one of the two languages (Japanese and Chinese) as final outputs. The QA module is responsible for parsing the input question, choosing the expected answer type, and producing a set of keywords. The QA module calls the TM module, which translates the keywords into the language(s) required by the task.

We use a combination of machine translation (MT) approaches for translating keywords: web-based-MT, dictionary-based-MT and text-mining-based-MT. The system selects the combination of translated keywords which are most likely to co-occur. Subsequently, translated keywords are passed to the RS module in order to retrieve a ranked list of relevant documents. Given these documents, the IX module extracts answer candidates and assigns confidence scores to each candidate. Finally, the AG module normalizes and clusters the answers, and attempts to boost the ranks of the most probable answer candidates. The overall architecture is shown in Figure 1.

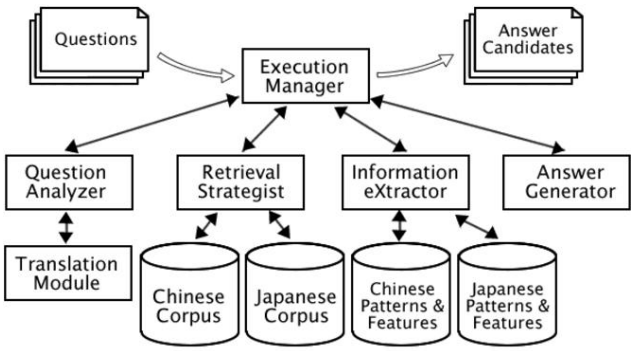

Figure 1: System Architecture

3. Result and Analysis

In addition to evaluating the overall performance of our system (e.g., by measuring average answer precision), we performed evaluations on a per-module basis in order to identify and analyze specific failure points. We used the formal run dataset from NTCIR task CLQA1, which includes English-Chinese (EC) and English-Japanese (EJ) subtasks. 200 input questions were provided for each of the subtasks.

[1] http://www.slt.atr.jp/CLQA/

	Gold Standard Input	QA$_{ATYPE}$ Accuracy	TM Accuracy	RS Top15	IX Top100	MRR	Overall Top1 R	Top1 R+U
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+QA$_{ATYPE}$	-	-	57.5%	50.5%	0.260	9.5%	20.5%
	TM+QA$_{ATYPE}$+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+QA$_{ATYPE}$	-	-	68.0%	45.0%	0.164	10.0%	15.5%
	TM+QA$_{ATYPE}$+RS	-	-	-	51.5%	0.381	32.0%	32.5%

Table 1: Modular basis performance given gold standard input

To evaluate the system's output, we used the gold standard data from NTCIR, which includes correct answers and the documents where they came from. We distinguish "correct and well-supported" answers from "correct but unsupported" answers, in the following way. We define documents in the gold standard dataset as supporting documents.

Correct answers that came from supporting documents are deemed *correct and supported* (denoted by R), whereas correct answers that did not come from supporting documents are called *unsupported* answers (denoted by U). Let "top *n* frequency" be the frequency of the event where at least one correct answer was included in the top *n* answer candidates returned. And let "average top *n* accuracy" be an average of the top *n* frequency over the questions. Note that this metric does not evaluate the number of correct answers returned for each question, whereas "average precision at *n*", commonly used in the field of information retrieval, does for the number of correct documents (Buckley & Voorhees, 2000). Following the evaluation method in the CLQA1 formal run, we will use *top 1 average accuracy* as the metric for evaluating overall performance.

In Table 1, the overall performance (top 1 average accuracy) is shown in the last two columns of the top rows for EC and EJ. If we examine only such global measures, we will not be able to understand the performance of individual modules in a complex system. The next section provides further discussion of the modular analysis results shown in Table 1.

3.1. Module-by-Module Analysis

In order to gain different perspectives from the test runs and to compare the system's capabilities across two different target languages, we conducted a module-by-module analysis. This analysis was based on gold-standard answer data, composed of answers and their supporting documents. In the third column of Table 1, we evaluated the QA module by the average accuracy of its answer type classification, and the TM module (fourth column) by the average accuracy of its keyword translation. For the RS module (fifth column) and IX module (sixth column), if a correct document or answer is returned, regardless of its ranking, we consider the module to be successful, because all top *n* documents (e.g. 15 in our case) are used in the extraction following the RS, and then top *m* (e.g. 100 in our case) answer candidates are used for the re-ranking process in the AG. Mean Reciprocal Rank or MRR is calculated for the IX in order to evaluate correct answers and their ranks. More precisely, MRR is calculated by averaging the reciprocal of the extracted correct answer's rank over questions. If there are multiple correct answers in the ranked list given one question, we use the highest rank for the calculation.

To separate the effects of errors introduced by earlier modules, we created additional gold standard data for answer type and keyword translation by manually correcting answer type and keyword translation errors. We also created "perfect" IX input using the gold-standard document set. The rows in Table 1 show what perfect input was fed to the system; the row "None" indicates the original performance without any manually-created perfect input.

3.1.1. Question Analyzer (QA) Performance

The QA module performed well in identifying the answer type in both runs. As we can see from the QA$_{ATYPE}$ column in Table 1, the QA achieved 86.5% for the EC run and 93.5% for the EJ run. An additional analysis of accuracy by answer type is shown in Table 2. Compared to the row TM+QA$_{ATYPE}$ in Table 1, we can see that further improvement of the answer type accuracy via manual correction did not make a significant difference in EJ.

	EC			EJ		
A type	# of Q	# of correct A type	%	# of Q	# of correct A type	%
PER	79	64	81.0%	34	34	100.0%
LOC	45	44	97.8%	34	33	97.1%
ORG	15	12	80.0%	13	8	61.5%
ARTIFACT	27	23	85.2%	21	19	90.5%
DATE	18	18	100.0%	25	25	100.0%
TIME	1	1	100.0%	14	13	92.9%
MONEY	5	4	80.0%	20	18	90.0%
NUMBER	10	7	70.0%	31	29	93.5%
PERCENT	0	0	-	8	8	-
total	200	173	86.5%	200	187	93.5%

Table 2: Question Analyzer performance by answer type

3.1.2. Translation Module (TM) Performance

The average accuracy of translation was 69.3% for the EC run and 72.6% for the EJ run. By taking advantage of translation by web mining, we could successfully translate some named entities. 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 runs, as shown in row TM in Table 1. This shows that translation errors have a significant negative impact on keyword-based document retrieval.

3.1.3. Retrieval Strategist (RS) Performance

The RS module achieved a top 15 accuracy of 30.5% in the EC run and 44.5% in the EJ run, as shown in column "RS Top 15" in Table 1. 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 runs, respectively. Among these keywords, 65.0% for the EJ run and 43.0% for the EC run were classified as proper nouns and phrases by the QA module. Most of the proper nouns are person, location and organization names. We also observed that in the corpus, the majority of the questions were drawn from these three types. This helps to explain the 20.0% accuracy gain achieved from corrected key term translation.

To illustrate the performance of the RS module, a CLIR-style analysis is also provided. In Figure 2, the cumulative frequency of correctly retrieved documents is plotted for each rank. From the heuristics in the past training run, we decided to use 15 as the cut-off value for the number of documents to retrieve. However, the figure shows that for EC, documents that are ranked 8th or higher contribute very little to the set of correct documents. Although it is hard to predict the number of documents to retrieve in order to reach the best balance between recall and precision, we should have set a lower cut-off value for EC.

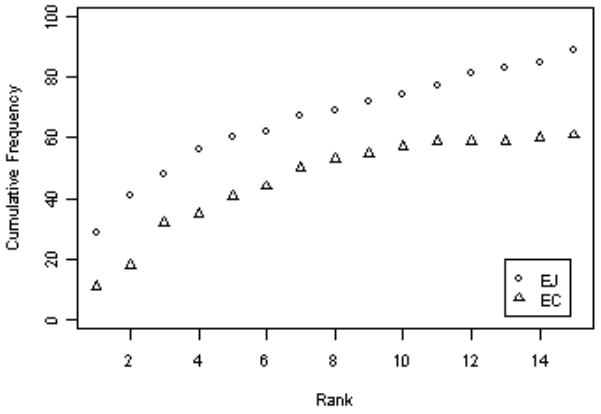

Figure 2: RS evaluation on cumulative frequency and rank of all the retrieved documents judged with correct supporting document

3.1.4. Information Extractor (IX) Performance

In the formal run data (row None in Table 1), we observed big accuracy drops at the RS module and after the IX module for both EC and EJ, and even bigger accuracy drops at the IX module for the EJ. The drop in RS accuracy is expected, but the difference between IX performance in the EC and EJ run is surprising. After eliminating errors carried over from earlier modules (see row TM+QAATYPE+RS), the IX in the EC and the EJ runs show a performance difference of 11.5% (63.0%-51.5%).

As the performance of the RS module 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 run and 10.0% (41.5%-31.5%) in the EJ run. 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 run and 51.5% for the EJ run.

The IX in the EC run 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+QAATYPE+RS case.

Because the answer validation function was not yet implemented in the AG module to filter out noise such as answers different from expected answer type, 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+QAATYPE row and TM+QAATYPE+RS row in the EC part of Table 1. 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.

We also performed an evaluation of the overall accuracy based on the answer type. Figure 3 shows the overall result classified by answer type, and indicates poor extraction performance on numeric (money, number, percent) and temporal (date, time) questions. Following this analysis, we began experimenting with answer type-specific approaches in order to address these weak points.

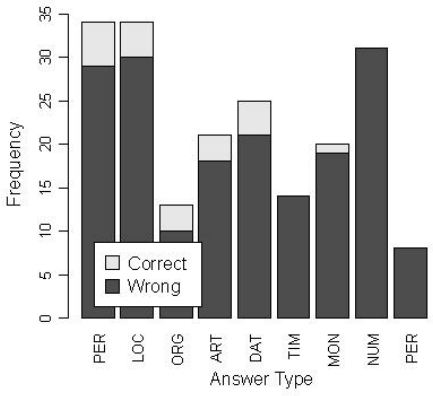

Figure 3: Number of top 1 correct answer for EJ run, classified by answer type

3.1.5. Answer Generator (AG) Performance

In order to evaluate the effectiveness of the AG module, we compared the ranks of answers before the AG (i.e. IX output) and after the AG module. To evaluate the AG module on a larger sample, we included gold-standard answers with or without supporting document.

We can categorize the results of the AG module's processing into three groups or answers: answers whose ranks were successfully raised or stayed at the top, answers whose rank did not change (except when remaining at the 1st rank), and answers whose ranks were lowered. Formally, let $Rank_{BeforeAG}$ be the rank of the answer which before input to the AG and let $Rank_{AfterAG}$ be the rank of the answer after processing by the AG.

$$\delta_+ = freq(Rank_{BeforeAG} > Rank_{AfterAG}) + freq(Rank_{BeforeAG} = 1 \wedge Rank_{AfterAG} = 1)$$

$$\delta_0 = freq(Rank_{BeforeAG} = Rank_{AfterAG}) - freq(Rank_{BeforeAG} = 1 \wedge Rank_{AfterAG} = 1)$$

$$\delta_- = freq(Rank_{BeforeAG} < Rank_{AfterAG})$$

As we consider the successful cases to be ones in which answer ranks were raised or stayed at the top, the accuracy is calculated in the formula below:

$$Accuracy = \frac{\delta_+}{\delta_+ + \delta_0 + \delta_-}$$

	δ_+	δ_0	δ_-	Accuracy
EC	34	12	14	56.7%
EJ	47	5	11	74.6%

Table 3: # of correct answers whose rank were affected by AG

Table 3 shows the summary of AG accuracy and how ranks of correct answers were affected by the AG. Even though the accuracy reflects the ratio of successfully boosted answers, it does not reflect to what degree ranks were improved. To visualize this, we generate a graphical view of the AG's performance such as the one shown in Figure 4. The Y-axis corresponds to the answer rank and $Rank_{BeforeAG}$ and $Rank_{AfterAG}$ are connected by a solid line in cases where $Rank_{BeforeAG} > Rank_{AfterAG}$, and by a dotted line otherwise. A histogram of answer ranks is also displayed on the side.

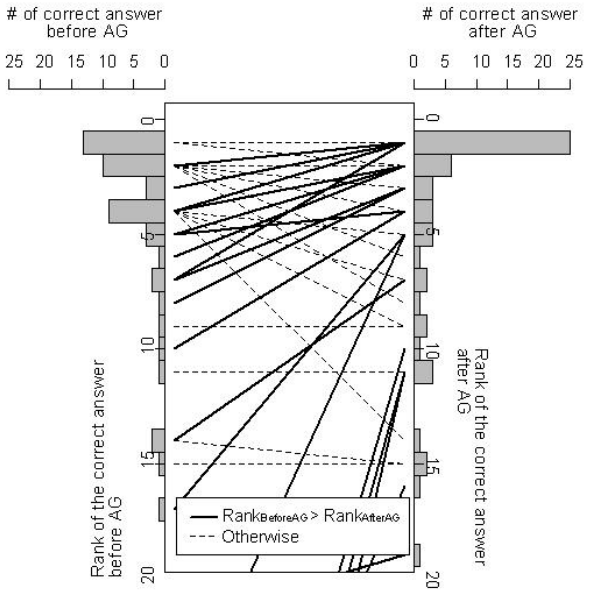

Figure 4: Visualization of AG performance (EJ)

4. Issues and Proposed Solutions

From the modular analysis, we observed low performance of the IX module on numerical and temporal questions. We also note that for these types of questions, subtype information (e.g., 'year', 'percentage') is very informative and could be used to improve extraction performance. The AG module is also partially responsible for the low performance on those types of questions, since the answer re-ranking algorithm should filter out answers with incorrect types.

It is difficult to decide how much recall should be sacrificed for accuracy when the IX module is used in combination with others. The JAVELIN system for English incorporates a Planner module which can select among the set of available IX modules at run-time (Hiyakumoto, 2004; Nyberg et al, 2005). It is left to future work to adapt the Planner for use in MLQA.

5. Conclusion

We presented a modular method for evaluating a complex multilingual Question Answering system. Our analysis using different levels of 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.

6. Acknowledgement

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.

7. References

Buckley C., E. M. Voorhees (2000). Evaluating Evaluation Measure Stability. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 33-40.

Hiyakumoto, L.S. (2004). Planning in the JAVELIN QA System. *Carnegie Mellon Computer Science Technical Report CMU-CS-04-132*.

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

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 Systems at TREC 2005. In *Proceedings of The Fourteenth Text REtrieval Conference*.

Nyberg, E., T. Mitamura. (2002). Evaluating QA Systems on Multiple Dimensions. In *Proceedings of LREC 2002 Workshop on QA Strategy and Resources*.