Probabilistic Models for Answer-Ranking in Multilingual Question-Answering

JEONGWOO KO
Google Inc
LUO SI
Purdue University
and
ERIC NYBERG and TERUKO MITAMURA
Carnegie Mellon University

This article presents two probabilistic models for answering ranking in the multilingual question-answering (QA) task, which finds exact answers to a natural language question written in different languages. Although some probabilistic methods have been utilized in traditional monolingual answer-ranking, limited prior research has been conducted for answer-ranking in multilingual question-answering with formal methods. This article first describes a probabilistic model that predicts the probabilities of correctness for individual answers in an independent way. It then proposes a novel probabilistic method to jointly predict the correctness of answers by considering both the correctness of individual answers as well as their correlations. As far as we know, this is the first probabilistic framework that proposes to model the correctness and correlation of answer candidates in multilingual question-answering and provide a novel approach to design a flexible and extensible system architecture for answer selection in multilingual QA. An extensive set of experiments were conducted to show the effectiveness of the proposed probabilistic methods in English-to-Chinese and English-to-Japanese cross-lingual QA, as well as English, Chinese, and Japanese monolingual QA using TREC and NTCIR questions.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process*

General Terms: Algorithm, Design, Experimentation

Additional Key Words and Phrases: Question-answering, answer-ranking, answer selection, answer-merging, probabilistic graphical model

ACM Reference Format:

Ko, J., Si, L., Nyberg, E., and Mitamura, T. 2010. Probabilistic models for answer-ranking in multilingual question-answering. ACM Trans. Inform. Syst. 28, 3, Article 16 (June 2010), 37 pages.
DOI = 10.1145/1777432.1777439 http://doi.acm.org/10.1145/1777432.1777439

This research was supported in part by ARDA/DTO Advanced Question Answering for Intelligence (AQUAINT) program, award number NBCHC040164.
Contact author's address: J. Ko, Google, Inc.; email: jko@google.com.

DOI 10.1145/1777432.1777439 http://doi.acm.org/10.1145/1777432.1777439

1. INTRODUCTION

For the past several years, open-domain question-answering (QA) has been actively studied to find exact answers to a natural language question in many different languages. TREC (Text REtrieval Conference) provides the infrastructure to evaluate QA systems for English. NTCIR (NII Test Collection for IR Systems) and CLEF (Cross-Language Evaluation Forum) focus on evaluating QA systems for Asian and European languages, respectively.

Even though there are many variations in the QA architecture for specific languages, traditional QA systems [Prager et al. 2000; Clarke et al. 2001; Harabagiu et al. 2001] adopt a pipeline architecture that incorporates four major processes: (1) question analysis; (2) document retrieval; (3) answer extraction; and (4) answer selection. Question analysis is a process that analyzes a question and produces a list of keywords and a question-processing strategy. Document retrieval is a step that searches for relevant documents or passages from given corpora. Answer extraction extracts a list of answer candidates from the retrieved documents. Answer selection is a process that pinpoints correct answer(s) from the extracted candidate answers. Since the first three processes in the QA pipeline may produce erroneous outputs, the final answer-selection process often entails identifying correct answer(s) among many incorrect ones.

For example, given the question "*Which city in China has the largest number of foreign financial companies?*", the answer-extraction component produced a list of five answer candidates: Beijing (AP880603-0268),[1] Hong Kong (WSJ920110-0013), Shanghai (FBIS3-58), Taiwan (FT942-2016), and Shanghai (FBIS3-45320). Due to imprecision in answer extraction, an incorrect answer ("Beijing") was ranked in the first position. The correct answer ("Shanghai") was extracted from two documents and was ranked in the third and the fifth positions. In order to select "Shanghai"" as the final answer, we have to address two interesting challenges.

—*Estimating answer relevance*. How do we estimate answer relevancy to identify relevant answer(s) among irrelevant ones? This task may involve searching for evidence of a relationship between t e answer and the answer type or a question keyword. For example, we might wish to query a knowledge base to determine if Shanghai is a city (*IS-A(Shanghai, city)*), or to determine if Shanghai is in China (*IS-IN(Shanghai, China)*) to answer the question.

—*Exploiting answer redundancy*. How do we exploit answer redundancy among answer candidates? For example, when the candidate list contains redundant answers (e.g., "Shanghai" as above) or several answers that represent a single instance (e.g., "U.S.A." and "the United States"), to what extent should we boost the rank of the redundant answers? Note also that effective handling of redundancy is particularly important when identifying a set of novel answers for list or definition questions.

To address the answer relevance, several answer-selection approaches have been developed that make use of external semantic resources for answer

[1]Answer candidates are shown with the identifier of the TREC document where they were found.

validation. However, few have considered the potential benefits of combining resources as evidence.

The second challenge is to exploit redundancy in the set of answer candidates. As answer candidates are extracted from different documents, they may contain identical, similar, or complementary text snippets. For example, the United States may be represented by the strings "U.S.," "United States," or "USA" in different documents. It is important to detect redundancy in answer candidates and exploit this redundancy to boost answer confidence, especially for list questions that require a set of unique answers. Some previous work [Kwok et al. 2001; Nyberg et al. 2003; Jijkoun et al. 2006] used heuristic methods like manually compiled rules to cluster evidence from similar answer candidates. However, previous work only modeled each answer candidate separately and did not consider both answer relevance and answer correlation to prevent the biased influence of incorrect similar answers. As far as we know, no previous work has jointly modeled the correctness of available answer candidates in a formal probabilistic framework, which is very important for generating an accurate and comprehensive answer list.

Extensibility is another important consideration in answer selection: how easy is it to extend answer selection to multilingual QA? As most answer-selection processes are language-dependent and require language-specific external resources, it is not easy to extend answer-ranking to multilingual QA. Although many QA systems have incorporated individual features and/or resources for answer selection in a single language, little previous research has examined a generalized probabilistic framework that supports answer selection in multiple languages using answer relevance and answer similarity features appropriate for the language in question. A generalized probabilistic framework will help QA systems to easily add new resources and easily support different languages.

In our previous work [Ko et al. 2009, 2007a], we proposed a probabilistic answer-ranking model to address these challenges. The model used logistic regression to estimate the probability that an individual answer candidate is correct given the relevance of the answer and the amount of supporting evidence provided by a set of similar answer candidates. The experimental results show that the model significantly improved answer-ranking performance in English, Chinese, and Japanese monolingual QA. However, the model considered each answer candidate separately, and did not consider the correlation of the correctness of answer candidates, which is problematic for generating accurate and comprehensive answers. For example, several similar answers may be ranked high in the final answer list, but a less redundant answer may not be ranked high enough to reach the user's attention.

In this article, we complement this logistic regression model by extending it to cross-lingual QA (English-to-Japanese and English-to-Chinese) and propose a new probabilistic answer-ranking model which considers both the correctness of individual answers and their correlation, to generate an accurate and comprehensive answer list. The proposed model uses an undirected graphical model to estimate the joint probability of the correctness of all answer candidates, from which the probability of the correctness of an individual candidate

Table I. Hypothesis Dimensions

Source Language (Question)	Target Language (Document)	Extraction Technique	QA System
English	English	FST, SVM, Heuristics	JAVELIN
English	English	Answer type-matching, Pattern-matching	EPHYRA
Chinese	Chinese	MaxEnt	JAVELIN
Japanese	Japanese	MaxEnt	JAVELIN
English	Chinese	MaxEnt	JAVELIN
English	Japanese	MaxEnt	JAVELIN

can be inferred. In comparing the previous logistic regression model (which considers answers independently) to the new graphical model (which jointly considers answers), we refer to the former as the *independent prediction model* and the latter as the *joint prediction model*. In our previous preliminary work [Ko et al. 2007b], we implemented a limited version of the joint prediction model which takes only the top 10 answer candidates for answer-reanking. This article extends this preliminary research. In particular, we have done more extensive experiments using approximate inference to avoid the limitation of exact inference which used only the top 10 answers. In addition, we demonstrate the effectiveness of the model for answer-ranking in cross-lingual QA (English-to-Chinese and English-to-Japanese) as well as multiple monolingual QA for English, Chinese, and Japanese.

Our hypothesis is that our models significantly improve performance of a multilingual QA system and provide a general framework that allows any relevance and similarity features for multiple languages to be easily incorporated. Specifically, these probabilistic answer-ranking models provide a generalized probabilistic framework that supports answer selection in cross-lingual QA as well as monolingual QA on answer candidates returned by multiple extraction techniques provided by different question-answering systems (Table I).

The rest of the article is arranged as follows: Section 2 reviews prior research on answer selection in question-answering. Section 3 describes the independent prediction model and proposes the novel joint prediction model. Section 4 lists the features that generate similarity and relevance scores for English factoid questions. In Section 5, we explain how the models are extended to support multistrategy QA and multilingual QA. Sections 6 and 7 describe the experimental results on English, Chinese, and Japanese answer-ranking. Section 8 compares the QA systems that incorporate our answer-ranking approach with other QA systems, participants in a recent TREC and NTCIR. Section 9 summarizes our findings and Section 10 concludes with potential future research.

2. PRIOR RESEARCH

To select the most probable answer(s) from the answer candidate list, QA systems have applied several different answer-selection approaches. One of the common is filtering. Due to errors or imprecision in earlier modules of the QA pipeline, extracted answer candidates sometimes contain irrelevant answers, leading to answer candidates that do not match the question. Manual rules and ontologies such as WordNet and gazetteers are commonly-used to delete

answer candidates that do not match the expected answer type [Cardie et al. 2000; Xu et al. 2003; Schlobach et al. 2004].

Answer-reranking is another popular approach for answer selection, which applies several different validation strategies in order to rerank answer candidates. One of the most common approaches relies on WordNet, CYC and gazetteers for answer validation or answer reranking, where answer candidates are pruned or discounted if they are not found within a resource's hierarchy corresponding to the expected answer type [Xu et al. 2003; Moldovan et al. 2003; Prager et al. 2004]. In addition, the Web has been used for answer-reranking by exploiting search engine results produced by queries containing the answer candidate and question keywords [Magnini et al. 2002]. Wikipedia's structured information has also been used for answer type-checking in Spanish monoligual QA [Buscaldi and Rosso 2006].

Even though each of these approaches uses one or more semantic resources to independently support an answer, few have considered the potential benefits of combining resources and using the result as evidence for answer-ranking. For example, Schlobach et al. combined geographical databases with WordNet in a type checker for location questions [Schlobach et al. 2004]. However, in their experiments the combination actually hurt performance—a result they attribute to the increased semantic ambiguity that accompanied broader coverage of location names.

Collecting evidence from similar answer candidates to boost confidence for a specific answer candidate is also important. As answer candidates are extracted from different documents, they may contain identical, similar, or complementary text snippets. One of the most popular approaches is to cluster identical or complementary answers. The score of each cluster is calculated by counting the number of answers in the cluster [Clarke et al. 2001]; summing the scores of all answers in the cluster [Nyberg et al. 2003; Kwok et al. 2001]; or selecting the best score among the individual answer scores in the cluster [Lin et al. 2005]. Recently, Jijkoun et al. used type-checking scores when merging similar answers in Dutch monolingual QA [Jijkoun et al. 2006]. They multiplied each answer score with a probability calculated by their type checker; they also used a graph to consider nontransitiveness in similarity.

Similarity detection is more important in list questions that require a set of unique answers (e.g., "Which countries produce coffee?"). In many systems, a cutoff threshold was used to select the most probable top N answers [Harabagiu et al. 2003; Katz et al. 2003]. An exhaustive search to find all possible candidates was applied to find answers for list questions [Yang et al. 2003].

Recently, several QA systems have employed a multistrategy architecture that allows multiple answering agents to answer a question [Chu et al. 2003; Echihabi et al. 2004; Jijkoun et al. 2003; Ahn et al. 2004; Nyberg et al. 2005]. This architecture requires answer-merging to combine the similar answers proposed by alternative approaches. As a simple approach, confidence-based voting has been used to merge the top five answers returned by competing QA agents [Chu et al. 2003]. As a more advanced approach, a maximum-entropy model has been used to rerank the top 50 answer candidates from three different answering strategies [Echihabi et al. 2004]. The LCC's Chaucer QA system

[Hickl et al. 2007] also used maximum entropy to merge answers returned from six answer extractors.

3. METHOD

Most of the answer-ranking solutions described in Section 2 model each answer candidate separately and do not consider both answer relevance and answer correlation in order to prevent the biased influence of incorrect similar answers. In this section, we first define the answer-ranking task from a probabilistic point of view. We then summarize the independent prediction model and propose the joint prediction model. Finally, we compare the advantages and disadvantages of the two models.

3.1 Answer-Ranking Task

In Section 1, we raised two challenges for answer selection: how to identify relevant answers and how to exploit answer redundancy to boost the rank of relevant answers. In order to address the two issues, the answer selection process should be able to conduct two subtasks. One task is to estimate the probability that an answer is relevant to the question. This task can be estimated by the probability $P(correct(A_i)|A_i, Q)$, where Q is a question and A_i is an answer candidate. The other task is to exploit answer redundancy in the set of answer candidates. This task can be done by estimating the probability $P(correct(A_i)|A_i, \{A_j\})$, where $\{A_j\}$ is a set of answers similar to A_i. Since both tasks influence answer-selection performance, it is important to combine the two tasks in a unified framework and estimate the probability of an answer candidate, $P(correct(A_i)|Q, A_1, \ldots, A_n)$.

The independent prediction model directly estimates this probability by combining answer relevance and similarity features on logistic regression. Instead of addressing each answer candidate separately, the joint prediction model estimates the joint probability of available answer candidates. In particular, the joint model estimates the probability of $P(correct(A_1), \ldots, correct(A_n)|Q, A_1, \ldots, A_n)$. The marginal probability of $P(correct(A_i)|Q, A_1, \ldots, A_n)$ for each individual answer as well as the conditional probability $P(correct(A_i)|correct(A_j), Q, A_1, \ldots, A_n)$ can be derived naturally from the joint probability to identify a set of distinct and comprehensive answers.

In both models, the answer whose probability is highest is selected as the final answer to the TREC factoid questions. As the models provide answer probability, we can also use them to classify incorrect answers. For example, if the probability of an answer candidate is lower than 0.5, it can be considered a wrong answer, and is filtered out of the answer list. This is useful in deciding whether or not a valid answer exists in the corpus, which is an important aspect of the TREC QA evaluation [Voorhees, 2003]. In addition, it can be used to find the answers for list questions.

3.2 Independent Prediction Model

The independent prediction model directly estimates a probability that an answer is correct given multiple answer relevance features and answer similarity

features. The framework was implemented with logistic regression (Eq. (1)).

$$\begin{aligned} & P(correct(A_i)|Q, A_1, \ldots, A_n) \\ & \approx P(correct(A_i)|rel_1(A_i), \ldots, rel_{K1}(A_i), sim_1(A_i), \ldots, sim_{K2}(A_i)) \\ & = \frac{\exp\left(\alpha_0 + \sum_{k=1}^{K1} \beta_k rel_k(A_i) + \sum_{k=1}^{K2} \lambda_k sim_k(A_i)\right)}{1 + \exp\left(\alpha_0 + \sum_{k=1}^{K1} \beta_k rel_k(A_i) + \sum_{k=1}^{K2} \lambda_k sim_k(A_i)\right)} \end{aligned} \tag{1}$$

$$\text{where, } sim_k(A_i) = \sum_{j=1(i \neq j)}^{N} sim'_k(A_i, A_j)$$

In Eq. (1), each $rel_k(A_i)$ is a feature function used to produce an answer relevance score for an answer candidate A_i. Each $sim'_k(A_i, A_j)$ is a similarity function used to calculate an answer similarity between A_i and A_j. *K1* and *K2* are the number of answer relevance and answer similarity features, respectively. N is the number of answer candidates. To incorporate multiple similarity features, each $sim_k(A_i, A_j)$ is obtained from an individual similarity metric. For example, if Levenshtein distance is used as one similarity metric, $sim_k(A_i, A_j)$ is calculated by summing the N-1 Levenshtein distances between one answer candidate and all the other candidates. The parameters α, β, λ were estimated from training data by maximizing the log likelihood. For parameter estimation, we used the Quasi-Newton algorithm [Minka 2003]. The estimated probability from the model is used to rank answer candidates and select final answers from the ranked list.

3.3 Joint Prediction Model

The joint prediction model uses an undirected graphical model to estimate the joint probability of all answer candidates from which the probability of an individual candidate is inferred. We used a probabilistic graphical model called a Boltzmann machine [Hinton and Sejnowski 1986], which is a special undirected graphical model whose nodes have a binary value. Each node S_i can be either $\{0, 1\}$ or $\{-1, 1\}$. The joint probability of this graph is represented in Eq. (2).

$$P(S_1, S_2, \ldots, S_n) = \frac{1}{Z} \exp\left(\sum_{i<j} \theta_{ij} S_i S_j + \sum_i \theta_{io} S_i\right) \tag{2}$$

where Z is a normalization constant and $\theta_{ij} = 0$ if nodes S_i and S_j are not neighbors in the graph. θ_{io} is the threshold of node i.

We adapted a Boltzmann machine for the answer-ranking process. Each node S_i in the graph represents an answer candidate A_i and its binary value represents answer correctness. The weights on the edges represent answer similarity between two nodes. If two answers are not similar, the weight between them is 0. The joint probability of the model can be calculated with Eq. (3). Each $rel_k(A_i)$ is a feature function used to produce an answer relevance score for an individual answer candidate, and each $sim_k(A_i, A_{N(i)})$ is a feature function used to calculate the similarity between an answer candidate A_i and its neighbor

answer $A_{N(i)}$. If $sim_k(A_i, A_{N(i)})$ is zero, two nodes S_i and $S_{N(i)}$ are not neighbors in the graph.

$$P(S_1, S_2, \ldots, S_n) = \frac{1}{Z} \exp\left(\sum_{i=1}^{n}\left[\left(\sum_{k=1}^{K1}\beta_k rel_k(A_i)\right)S_i + \sum_{N(i)}\left(\sum_{k=1}^{K2}\lambda_k sim_k(A_i, A_{N(i)})\right)S_i S_{N(i)}\right]\right) \quad (3)$$

The parameters β and λ are estimated from training data by maximizing the joint probability, as shown in Eq. (4). R is the number of training data and Z is the normalization constant calculated by summing all configurations. As $logZ$ does not decompose, we explain in Section 6 our implementation details to address this issue by limiting the number of answer candidates or applying approximate inference with the contrastive divergence learning method [Hinton 2000].

$$\vec{\beta}, \vec{\lambda} = \arg\max_{\vec{\beta},\vec{\lambda}} \sum_{r=1}^{R} \log \frac{1}{Z} \exp\left(\sum_{i=1}^{n}\left[\left(\sum_{k=1}^{K1}\beta_k rel_k(A_i)\right)S_i + \sum_{N(i)}\left(\sum_{k=1}^{K2}\lambda_k sim_k(A_i, A_{N(i)})\right)S_i S_{N(i)}\right]\right) \quad (4)$$

As each node has a binary value, this model uses the answer relevance scores only when an answer candidate is correct ($S_i = 1$), and uses the answer similarity scores only when two answer candidates are correct ($S_i = 1$ and $S_{N(i)} = 1$). If $S_i = 0$, then the relevance and similarity scores are ignored. If $S_{N(i)} = 0$, the answer similarity scores are ignored. This prevents the biased influence of incorrect similar answers.

As the joint prediction model is based on a probabilistic graphical model, it can support probabilistic inference to identify a set of accurate and comprehensive answers. Figure 1 shows the algorithm for selecting answers using the joint prediction model. After estimating the marginal and conditional probabilities, we calculate the score of each answer candidate A_j by subtracting the conditional probability from the marginal probability.

3.4 Comparing JP with IP

Both the independent prediction model and the joint prediction model provide a general probabilistic framework to estimate the probability of an individual answer candidate from answer relevance and similarity features.

One advantage of the joint prediction model is that it provides a formal framework to identify a distinct set of answers, which is useful for list questions. For example, the question "*Who have been the US presidents since 1993?*" requires a list of person names as the answer. As person names can be represented in several different ways (e.g., "Bill Clinton", "William J. Clinton," "Clinton, Bill"), it is important to find unique names as the final answers. This task can be done by using the conditional probability inferred from the joint

1. Create an empty answer pool.
2. Estimate the joint probability of all answer candidates: $P(S_1, S_2, ..., S_n)$
3. Calculate the marginal probability that an individual answer candidate is correct.

$$P(correct(A_i) \mid Q, A_1, ..., A_n) \approx \sum_{S_1,...,S_{i-1},S_{i+1}...,S_n} P(S_i = 1, S_1, ..., S_{i-1}, S_{i+1}, ..., S_n)$$

4. Choose the answer whose marginal probability is highest, and move it to the answer pool.
5. For the remaining answer candidates, repeat 5.1 ~ 5.3
   5.1. Calculate the conditional probability of the remaining answers (A_j) given an answer in the answer pool. For example, if A_i is an answer in the answer pool, calculate
   $P(correct(A_j) \mid correct(A_i), Q, A_1, ..., A_n)$
   5.2. Calculate the score of each answer A_j from the marginal and conditional probability.

$$Score(A_j) = P(correct(A_j) \mid Q, A_1, ..., A_n) - \max_i P(correct(A_j) \mid correct(A_i), Q, A_1, ..., A_n)$$

   5.3. Choose the answer whose $Score(A_j)$ is maximum, and move it to the answer pool.

Fig. 1. Algorithm to rank answers with the joint prediction model.

P(correct("William J. Clinton") | Q, A_1,…,A_n)= 0.758

P(correct("Bill Clinton") |Q, A_1,…,A_n) = 0.755

P(correct("George W. Bush") |Q, A_1,…,A_n) = 0.617

Fig. 2. Marginal probability of individual answers.

prediction model. For example, assume that we have three answer candidates for this question: "William J. Clinton," "Bill Clinton," and "George W. Bush". As shown in Figure 1, the probability of correctness of each answer was calculated by marginalizing the joint probability of all answer candidates. Figure 2 shows the marginal probability of individual answers.

In this example, the marginal probability $P(correct(Bill\ Clinton))$ and $P(correct(William\ J.\ Clinton))$ are high because "Bill Clinton" and "William J. Clinton" support each other. Based on the marginal probabilities, we first choose the answer candidate A_i whose marginal probability is the highest. In this example, "William J. Clinton" is chosen and added to the answer pool. Then we calculate the conditional probability of the remaining answer candidates given the chosen answer "William J. Clinton".

Figure 3 shows the conditional probability given "William J. Clinton". The conditional probability of "Bill Clinton" is higher than the marginal probability of "Bill Clinton," which indicates that "Bill Clinton" depends on "William J. Clinton". On the other hand, $P(correct(George\ W.\ Bush) \mid correct(William\ J.\ Clinton))$ is the same as $P(correct(George\ W.\ Bush))$ because the fact that "William J. Clinton" is correct does not give any information for "George W. Bush". Next, we calculate a score for the remaining answers using the marginal and conditional probabilities (Figure 4).

As the score of "George W. Bush" is higher than the score of "Bill Clinton," "George W. Bush" is chosen as the second answer even though its marginal

P(correct("Bill Clinton")|correct("William J. Clinton"), Q, A_1,…,A_n) = 0.803

P(correct("George W. Bush")|correct("William J. Clinton"), Q, A_1,…,A_n) = 0.617

Fig. 3. Conditional probability given that "William J. Clinton" is correct.

Score("Bill Clinton") = 0.755 – 0.803 = -0.048

Score("George W. Bush") = 0.617 – 0.617 = 0

Fig. 4. Score calculation using marginal and conditional probability.

probability is lower than "Bill Clinton". In this way we can select the best unique answers from a list of answer candidates.

However, the joint prediction model is less efficient than the independent prediction model. For example, when the graph is fully connected, the joint prediction model requires $O(2^N)$ time to calculate the joint probability, where N is the size of the graph. This is the worst case in terms of algorithmic efficiency, but which is still manageable for a small N. To make the model provide answers in interactive time, efficiency is more important, and we need to implement it with faster approximate approaches such as Gibbs sampling, Markov chain Monte Carlo sampling, and variational inference. In Section 6, we describe how we used Gibbs sampling to implement the joint prediction model. On the other hand, the independent prediction model needs complex optimization for parameter estimation (e.g., the Quasi-Newton algorithm [Minka 2003]), but after training, it needs $O(N)$ time for answer-reranking.

4. ANSWER RELEVANCE AND SIMILARITY MEASUREMENT

The models need two sets of scores: answer relevance and answer similarity scores. This section describes how we used several semantic resources to generate answer relevance and similarity scores for factoid and list questions. More details on features can be found in Ko et al. [2009].

4.1 Measuring Answer Relevance

Each answer relevance score predicts whether or not an answer candidate is correct for the question. For factoid and list questions, we used gazetteers and ontologies in a knowledge-based approach to get answer relevance scores; we also used Wikipedia and the Web in a data-driven approach.

4.1.1 *Utilizing External Knowledge-Based Resources.* (a) *Gazetteers*: We used gazetteers to generate an answer relevance score. For example, given the question "*What continent is Togo on*?" the answer candidate "*Africa*" receives a score of 1.0 because gazetteers can answer this question. The candidate "*Asia*" receives a score of 0.5 because it is a continent name in gazetteers and matches the expected answer type of the question. But the candidate "*Ghana*" receives a score of −1.0 because it is not a continent in gazetteers. A score of 0 means the gazetteers did not contribute to the answer-selection process for that candidate. For English QA, three gazetteers were used: the Tipster

Gazetteer, information about the states in the US provided by 50states.com (http://www.50states.com) and the CIA World Factbook. Because the CIA World Factbook can answer questions about languages and populations (e.g., "*What is the primary language of the Philippines?*" "*How many people live in Chile?*"), we used it for range checking. For population, if an answer candidate is in the range of 10% stated in the CIA Factbook, it is considered a correct answer and receives a score of 1.0. If it in the range of 20%, it receives a score of 0.5. If it differs significantly by more than 20%, it receives a score of −1. The threshold may vary based on when the document was written and when the census was taken.[2]

(b) *Ontologies*: Ontologies such as WordNet [Fellbaum, 1998] contain information about relationships between words and general meaning types (synsets, semantic categories, etc.). We used WordNet to produce an answer relevance score in a manner analogous to gazetteers. For example, given the question "*What is the capital of Uruguay*?" the candidate "*Montevideo*" receives a score of 1.0 because Montevideo is described as the "capital of Uruguay" in WordNet. For the question "*Who wrote the book 'Song of Solomon'?*" the candidate "*Mark Twain*" receives a score of 0.5 because its hypernyms include the expected answer type "writer".

4.1.2 *Utilizing External Resources in a Data-Driven Approach.* (a) *Wikipedia*: Wikipedia (http://www.wikipedia.org) is a multilingual free online encyclopedia. To generate an answer relevance score, an answer candidate is compared with the Wikipedia document titles. If there is a document whose title matches the answer candidate, the document is analyzed to obtain the term frequency (tf) and the inverse term frequency (idf) of the answer candidate. The answer relevance score is calculated from the tf.idf score of this answer candidate. When there is no matched document, each question keyword is used as a back-off strategy to find the matched Wikipedia documents, and then the answer relevance score is calculated by summing the tf.idf scores for all question keywords.

(b) *Web*: Following Magnini et al. [2002], we used the Web to generate a numeric score for each candidate. A query consisting of an answer candidate and question keywords was sent to the Google search engine. To calculate a score, the top 10 text snippets returned by Google were then analyzed to generate an answer relevance score by computing the word distance between a keyword and the answer candidate.

4.2 Measuring Answer Similarity

As factoid and list questions require short text phrases as an answer, the similarity between two answer candidates is measured using string distance metrics and synonyms.

4.2.1 *Utilizing String Distance Metrics.* There are several string distance metrics to calculate the similarity of short strings (e.g., Levenshtein, Cosine

[2]The ranges used here were found to work effectively, but were not explicitly validated or tuned.

similarity, Jaccard, Jaro, and Jaro and Winkler [Jaro 1995; Winkler 1999]). In our experiments, we used Levenshtein as a string distance metric, and when a Levenshtein score is less than 0.5, it is ignored.

4.2.2 *Utilizing Synonyms.* Synonyms can be used as another metric to calculate answer similarity. We define a binary similarity score for synonyms as

$$\text{sim}(A_i, A_j) = \begin{cases} 1, \text{if } A_i \text{ is a synonym of } A_j \\ 0, \text{otherwise} \end{cases}$$

For English, to get a list of synonyms, we used three knowledge bases: WordNet, Wikipedia, and the CIA World Factbook. WordNet includes synonyms for English words. For example, "U.S." has a synonym set containing "United States," "United States of America," "America," "US," "USA," and "U.S.A". All the terms in the synonym set were used to find similar answer candidates.

For Wikipedia, redirection is used to obtain another set of synonyms. For example, "Calif." is redirected to "California" in English Wikipedia. "Clinton, Bill" and "William Jefferson Clinton" are redirected to "Bill Clinton". The CIA World Factbook is used to find synonyms for a country name. It includes five different names for a country: the conventional long form, conventional short form, local long form, local short form, and former name. For example, the conventional long form of Egypt is "Arab Republic of Egypt," the conventional short form is "Egypt," the local short form is "Misr," the local long form is "Jumhuriyat Misr al-Arabiyah," and the former name is "United Arab Republic (with Syria)". All are considered to be synonyms of "Egypt".

In addition, manually generated rules are used to canonicalize answer candidates which represent the same entity. Dates are converted into the ISO 8601 format (YYYY-MM-DD) (e.g., "April 12 1914" and "12th Apr. 1914" are converted into "1914-04-12" and are considered synonyms). Temporal expressions are converted into the HH:MM:SS format and numeric expressions are converted into numbers. For location, a representative entity is associated with a specific entity when the expected answer type is COUNTRY (e.g., "the Egyptian government" is considered "Egypt" and "Clinton administration" is considered "U.S."). This representative entity rule was only applied to the Unite States. As there will be have new U.S. presidents, this rule should be updated every four years to add a new entity.

5. EXTENSION TO MULTISTRATEGY QA AND MULTILINGUAL QA

This section describes the extension of the models to multistrategy QA and multilingual QA.

5.1 Extension to Multistrategy QA

Many QA systems utilize multiple strategies to extract answer candidates, and then merge the candidates to find the most probable answer [Chu et al. 2003; Echihabi et al. 2004; Jijkoun et al. 2003; Ahn et al. 2004; Nyberg et al. 2005]. The joint prediction model can be extended to support multistrategy QA

by combining the confidence scores returned from individual extractors with the answer relevance and answer similarity features. Equation 5 shows the extended joint prediction model for answer-merging, where m is the number of extractors, n is the number of answer candidates returned from one extractor, and $conf_k$ is the confidence score extracted from the k_{th} extractor whose answer is the same as A_i. When an extractor extracts more than one answer from different documents with different confidence scores, the maximum confidence score is used as $conf_k$. For example, the LIGHT extractor in the JAVELIN QA system [Nyberg et al. 2004] returns two answers for "Bill Clinton" in the candidate list: one has a score of 0.7 and the other a score of 0.5. In this case, we ignore 0.5 and use 0.7 as $conf_k$. This is to prevent double counting of redundant answers because $sim_k(A_i, A_{N(i)})$ already considers this similarity information.

$$P(S_1, S_2, \ldots, S_{m*n}) = \frac{1}{Z} \exp\left(\sum_{i=1}^{m*n}\left[\left(\sum_{k=1}^{K1} \beta_k\, rel_k(A_i) + \sum_{k=1}^{m} \gamma_k\, conf_k\right) S_i + \sum_{N(i)}\left(\sum_{k=1}^{K2} \lambda_k\, sim_k(A_i, A_{N(i)})\right) S_i S_{N(i)}\right]\right) \tag{5}$$

Equation 6 shows the extended independent prediction model for answer-merging (reported in Ko et al. [2009]).

$$\begin{aligned} &P(correct(A_i)|Q, A_1, \ldots, A_n) \\ &\quad \approx P(correct(A_i)|rel_1(A_i), \ldots, rel_{K1}(A_i), sim_1(A_i), \ldots, sim_{K2}(A_i)) \\ &\quad = \frac{\exp\left(\alpha_0 + \sum_{k=1}^{K1} \beta_k\, rel_k(A_i) + \sum_{k=1}^{K2} \lambda_k\, sim_k(A_i) + \sum_{k=1}^{m} \gamma_k\, conf_k\right)}{1 + \exp\left(\alpha_0 + \sum_{k=1}^{K1} \beta_k\, rel_k(A_i) + \sum_{k=1}^{K2} \lambda_k\, sim_k(A_i) + \sum_{k=1}^{m} \gamma_k\, conf_k\right)} \end{aligned} \tag{6}$$

5.2 Extension to Different Monolingual QA

We extended the models to Chinese and Japanese monolingual QA by incorporating language-specific features into the models. As the models are based on a probabilistic framework, they do not need to be changed to support other languages. We only retrained the models for individual languages. To support Chinese and Japanese QA, we incorporated new features for individual languages. This section summarizes the relevance and similarity scores for Chinese and Japanese.

5.2.1 *Measuring Answer Relevance.* We replaced the English gazetteers and WordNet with language-specific resources for Japanese and Chinese. As Wikipedia and the Web support multiple languages, the same algorithm was used in searching language-specific corpora for the two languages.

(1) *Utilizing external knowledge-based resources.* (a) *Gazetteers*: There are few available gazetteers for Chinese and Japanese, so, we extracted location data from language-specific resources. For Japanese, we extracted Japanese location information from Yahoo (http://map.yahoo.co.jp), which contains many location names in Japan and the relationships among them. We also used *Gengo GoiTaikei* (http://www.kecl.ntt.co.jp/mtg/resources/GoiTaikei), a Japanese

Table II. Articles in Wikipedia for Different Languages

	# of Articles	
	Nov. 2005	Aug. 2006
English	1,811,554	3,583,699
Japanese	201,703	446,122
Chinese	69,936	197,447

lexicon containing 300,000 Japanese words with their associated 3,000 semantic classes. We utilized the *GoiTaikei* semantic hierarchy for type-checking of location questions. For Chinese, we extracted location names from the Web. In addition, we translated country names provided by the CIA World Factbook and the Tipster gazetteers into Chinese and Japanese using the JAVELIN Translation Module [Mitamura et al. 2007]. As there is more than one translation per candidate, the top three translations were used. This gazetteer information was used to assign an answer relevance score between -1 and 1 using the algorithm described in Section 4.1.1.

(b) *Ontologies*: For Chinese, we used HowNet [Dong 2000], which is a Chinese version of WordNet. It contains 65,000 Chinese concepts and 75,000 corresponding English equivalents. For Japanese, we used semantic classes provided by *Gengo GoiTaikei*. The semantic information provided by HowNet and *Gengo GoiTaikei* was used to assign an answer relevance score between -1 and 1.

(2) *Utilizing external resources in a data-driven approach.* (a) *Web:* The algorithm used for English was applied to analyze Japanese and Chinese snippets returned from Google by restricting the language to Chinese or Japanese so that Google returned only Chinese or Japanese documents. To calculate the word distance between an answer candidate and the question keywords, segmentation was done with linguistic tools. For Japanese, Chasen (http://chasen.aist-nara.ac.jp/hiki/ChaSen) was used. For Chinese segmentation, a maximum-entropy based parser was used [Wang et al. 2006].

(b) *Wikipedia*: As Wikipedia supports more than 200 language editions, the approach used in English can be used for different languages without any modification. Table II shows the number of text articles in the three languages. Wikipedia's coverage in Japanese and Chinese does not match its coverage in English, but coverage in these languages continues to improve.

To supplement the small corpus of available Chinese documents, we used Baidu (http://baike.baidu.com), which is similar to Wikipedia but contains more articles in Chinese. We first search in Chinese Wikipedia documents, and when there is no matching document in Wikipedia, we search in Baidu as a back-off strategy. Each answer candidate is sent to Baidu and the retrieved document is analyzed in the same way to analyze Wikipedia documents.

5.2.2 *Measuring Answer Similarity.* As Chinese and Japanese factoid questions require short text phrases as answers, the similarity between two answer candidates can be calculated with string distance metrics and a list of synonyms [Japanese on WordNet; Chen et al. 2000; Mei et al. 1982].

Original answer string	Normalized answer string
三千億円	3E+11 円
3,000億円	3E+11 円
一九九三年 七月 四 日	1993-07-04
1993 年 7 月4 日	1993-07-04
四分の一	0.25
5割	50 %

Fig. 5. Example of normalized answer strings.

(1) *Utilizing string distance metrics.* The same string distance metrics used for English were applied to calculate the similarity of Chinese/Japanese answer candidates.

(2) *Utilizing synonyms.* To identify synonyms, Wikipedia was used for both Chinese and Japanese. The EIJIRO dictionary was also used to obtain Japanese synonyms. EIJIRO is an English-Japanese dictionary containing 1,576,138 words, and provides synonyms for Japanese words. For temporal and numeric expressions, new conversion rules were created; for example, a rule to convert Japanese *Kanji* characters to Arabic numbers (Figure 5).

5.3 Extension to Cross-Lingual QA

Recently, QA systems have been extended to cover cross-lingual question-answering (CLQA), which accepts questions in one language (source language) and searches for answers from documents written in another language (target language). CLQA has been evaluated for various language pairs in CLEF and NTCIR.

For CLQA, most systems translate a question or question keywords into the target language and then apply a monolingual QA approach to find answers in the corpus. One recently reported system used a monolingual QA system to find answers in the source language, and then translated the answers into the target language [Bos and Nissim 2006]. This approach, however, requires documents for both the source language and the target language. Translating questions or keywords into the target language has been the more common approach to date. In this target, we only focus on supporting the models for the question or keyword translation case.

When a CLQA system uses translated questions or translated question keywords to find answers from the target corpus, it tends to produce lower-quality answer candidates: (1) there are numerous incorrect answer candidates and few correct answer candidates; and (2) correct answers are ranked very low. This phenomenon makes answer-ranking more challenging in CLQA, as it requires an additional degree of robustness in answer-ranking.

This section describes how we extend the models for CLQA answer-ranking, especially for English-to-Chinese and English-to-Japanese. For this,

we extended the features used for Chinese and Japanese monolingual answer-ranking.

5.3.1 *Measuring Answer Relevance.* We reused knowledge-based features and extended data-driven features for CLQA, as described below.

(1) *Utilizing external knowledge-based resources.* The knowledge-based features involve searching for facts in knowledge bases such as gazetteers and/or ontologies. As the question is already translated into Chinese or Japanese, the question and answer candidates were written in the same language. Therefore, we can reuse the features developed in Chinese and Japanese monolingual QA for English-to-Chinese and English-to-Japanese QA, respectively. However, CLQA tends to have more than one translated candidate for the expected answer type, and the algorithms were extended to use multiple translation candidates for the answer type. The final relevance score is the sum of relevance scores for all answer type translation candidates.

When using translation candidates, some inaccurate translations may retrieve incorrect relations form the knowledge bases and give a high score to irrelevant answer candidates. Therefore, keywords whose translation scores are less than some threshold value can be ignored, or only the top N keywords can be used. In our experiments, we used the top three translation candidates provided by the Translation Module (TM) in the Javelin QA system. The TM uses a noisy channel translation model and produces translated keywords with the associated scores. More details on TM can be found Lin et al. [2005].

(2) *Utilizing external resources in a data-driven approach.* (a) *Web*: For monolingual QA, we generated a query consisting of an answer candidate and question keywords. Since there are multiple translation candidates for each keyword in CLQA, the algorithm was extended to use multiple translated keywords to create the query. As keyword translation quality is typically poor for proper nouns and these English terms tend to appear in many documents, we added English proper noun keywords to the query. For example, the question "*In which city in Japan is the Ramen Museum located*?" contains "*Ramen Museum*" as one keyword. As the JAVELIN current translation module does not have a Chinese translation for this noun, it only translates *Museum* into a Chinese word. This partially translated phrase does not match any Web documents, even though there are Chinese Web documents that matched the English term "Ramen Museum". Hence, we used source proper noun keywords as one alternation in the query (Figure 6). However, this algorithm may change as translation quality improves. Similar to the knowledge-based features, we use the top three translation candidates to generate the Web relevance score.

(b) *Wikipedia*: We extended the algorithm to generate an answer relevance score using Wikipedia. First, a query consisting of an answer candidate is compared with the Wikipedia titles. When there is a matching document, the document is analyzed to obtain the tf.idf score of the answer candidate. When there is no matching document, each translated question keyword is sent to Wikipedia as a back-off strategy. Similarly to the Web case, we also search for English proper nouns in the question. After retrieving matched Wikipedia documents, each document is analyzed to obtain the tf.idf score. The final

For each answer candidate A_i:
1. Initialize the Web relevance score: $ws(A_i) = 0$
2. Create a query from an answer candidate and the translated keywords. For example, given the question "*Which companies will merge with TECO Group?*," assume that we have three question keywords and their translations.
   - company: 公司
   - merge: 合并
   - TECO Group: 东元集团, TECO 组, TECO 小组

   For an answer candidate "聲寶公司 ", a query is created:
   "聲寶公司" "公司" "合并" ("TECO Group" or "东元集团" or "TECO 组" or "TECO 小组")
3. Send the query to the Google search engine and choose the top 10 snippets returned from Google.
4. For each snippet *s*:
   4.1. Initialize the snippet co-occurrence score: $cs(s) = 1$
   4.2. For each question keyword translation *k* in *s*:
      4.2.1. Compute distance *d*, the minimum number of words between *k* and the answer candidate, excluding stopwords and the remaing keywords in the question.
      4.2.2. Update the snippet co-occurrence score: $cs(s) = cs(s) \times 2^{(1+d)^{-1}}$
   4.3 $ws(A_i) = ws(A_i) + cs(s)$

Fig. 6. Algorithm to generate an answer relevance score from the Web for cross-lingual QA.

answer relevance score is calculated by summing the tf.idf scores acquired from each keyword translation. As in the Web case, we used the top three translation candidates for this feature.

5.3.2 *Measuring Answer Similarity.* In monolingual QA, answer similarity was calculated using string distance metrics and a list of synonyms. As CLQA answer candidates are written in one language, the features for the monolingual QA in Section 5.2.2 was reused for CLQA.

6. EXPERIMENTAL RESULTS: ENGLISH QA

This section describes the experimental results done with two English QA systems. In our earlier work [Ko et al. 2009], we already demonstrated that the independent prediction model significantly improved answer-ranking performance over several popular answer-ranking approaches in English QA. Therefore, in this section we focus on evaluating the joint prediction model with three different implementations.

6.1 Experimental Setup

As the performance of answer selection depends on the quality of answer extraction, in our evaluation we calculated the performance by only using questions where at least one correct answer exists in the candidate list provided *by* an extractor. More specifically, three metrics were used for evaluation.

—Average top answer accuracy (TOP1). This is calculated by dividing the number of correct top answers by the number of questions where at least one correct answer exists in the candidate list provided by an extractor.

—Mean Reciprocal Rank (MRR5). This is the average reciprocal rank of the top N answers. For example, the answer ranked in the first position receives a score of 1, the answer ranked in the second position receives a score of 1/2, etc. The TREC evaluation used MRR of the top five answers to evaluate the performance of early QA systems. We will use MRR of the top five answers as another metric to evaluate the performance of our models.

—Average Precision at rank N. The average precision is calculated by counting the number of *unique* correct answers among the top N answers. Redundant answers are not considered as correct. For example, when the first two answers are "William J. Clinton" and "George Bush," and the third answer is "Clinton, Bill," the precision at rank 3 is 2/3. This will be useful to evaluate list questions.

The baseline was calculated with the answer candidate scores provided by each individual extractor; the answer with the best extractor score was chosen, and no validation or similarity processing was performed.

As a testbed to evaluate the joint prediction model, we used two QA systems: JAVELIN [Nyberg et al. 2004] and EPHYRA [Schlaefer et al. 2006]. Even though they are open-domain question-answering systems, they apply different extraction techniques. JAVELIN extracts answer candidates from the given TREC corpus to answer TREC questions. On the other hand, EPHYRA extracts answer candidates from the text snippets returned by Yahoo, merges the same answers, and then conducts answer projection to find the documents in the given TREC corpus.[3] We first describe the experiments done with JAVELIN and then report the experimental results on EPHYRA.

6.2 Experiments with JAVELIN

A total of 1,818 factoid questions from the TREC QA 8-12 evaluations and their corresponding answers served as a dataset, and 5-fold cross-validation was performed to evaluate our joint prediction model. Cross validation is a very popular approach to thoroughly evaluate the performance of a learning algorithm by splitting and alternating the roles of training and test data, to avoid overfitting to a specific data set when applying machine-learning techniques. In our experiment, we split the questions into five groups, and used the first group for testing and the others for training. Then we used the second group for testing and the others for training. This procedure was repeated five times by switching training and test data. Finally, the score is calculated by the average of scores produced from each test-training set pair. In each fold, we learned parameters directly from the training set and applied them to the test set.

For a string similarity metric, 0.5 was used as a threshold so that when a Levenshtein score is less than 0.5, it is ignored.

To better understand how the performance of our model can vary for different extraction techniques, we tested the joint prediction model with three

[3]As TREC requires submitting the supporting document as well as the answer to a question, the answer projection is a very important task in EPHYRA.

Table III. Performance Characteristics of Individual JAVELIN Answer Extractors ("macro" precision at question-level; "micro" precision at answer-level)

Extractor	# Questions with Correct Answers	Avg. Num of Answers per Question	Precision	
			Macro	Micro
FST	301	4.19	0.166	0.237
LIGHT	889	36.93	0.489	0.071
SVM	871	38.70	0.479	0.077

JAVELIN answer-extraction modules: FST, LIGHT, and SVM. FST is an answer extractor based on finite state transducers that incorporate a set of extraction patterns (both manually created and generalized patterns), and are trained for each answer type. LIGHT is an extractor that selects answer candidates using a nonlinear distance heuristic between the keywords and an answer candidate. SVM uses support vector machines to discriminate between correct and incorrect answers based on local semantic and syntactic context.

Table III compares JAVELIN extractor performance on the test questions and shows that extractors vary in the average number of answers they return for each question. The last two columns show the precision of individual extractors. Precision was calculated at both the macro-level and the micro-level. Macro-level precision measures the *precision of the questions*; the number of questions where at least one correct answer exists was divided by the total number of questions. Micro-level precision measures the *precision of answer candidates for one question*. It was calculated by dividing the number of correct answers by the number of total answers for one question. Generally, the macro precision of FST is lower than the one of LIGHT and SVM, which states that FST covers fewer questions than LIGHT and SVM, but its micro precision is higher than the others (This says that its answers are more accurate than answers from the other extractors). Micro and macro notions are popular in the text categorization task [Yang and Lui 1999], and we used this notion to explain how this affects the performance of the joint prediction model.

6.2.1 *Experimental Results.* As the joint prediction model is based on a graphical model, it requires $O(2^N)$ time complexity where N is the size of the graph (i.e., number of answer candidates). Therefore, we implemented the joint prediction model using two different inference methods: exact inference and approximate inference. For exact inference, we limited the number of answers to the top 10. If extractors provide good quality answers in its top 10 candidate list, this approach would work well without complex approximate approaches. In our preliminary work [Ko et al., 2007b], we implemented the joint prediction model using the exact inference approach and evaluated its performance on English factoid questions using the JAVELIN system. In this article, we extend this work by applying approximate inference using Gibbs sampling to implement the joint prediction model. This section describes three variations of the joint prediction model.

(1) *Exact inference with the top 10 answers produced by extractors.* This approach enumerates all possible configurations in a joint table and then calculates the marginal and conditional probabilities from the joint table so that it

Table IV(a). Performance of JP Using Top 5, 10, 11, and 12 Answer Candidates Produced by Each Individual Extractor

	5 candidates		10 candidates		11 candidates		12 candidates	
	TOP1	MRR5	TOP1	MRR5	TOP1	MRR5	TOP1	MRR5
FST	0.827	0.923	0.870	0.952	0.869	0.933	0.845	0.970
LIGHT	0.570	0.667	0.605	0.729	0.609	0.770	0.609	0.774
SVM	0.468	0.569	0.536	0.652	0.538	0.693	0.545	0.704

Table IV(b). Performance of IP and JP Using the Top 10 Answer Candidates Produced by Each Individual Extractor (BL: baseline, IP: independent prediction, JP: joint prediction)

	FST			LIGHT			SVM		
	BL	IP	JP	BL	IP	JP	BL	IP	JP
TOP1	0.691	0.873*	0.870*	0.404	0.604*	0.605*	0.282	0.532*	0.536*
MRR5	0.868	0.936*	0.952*	0.592	0.699*	0.729*	0.482	0.618*	0.652*

Table IV(c). Average Precision of IP and JP at a Different Rank Using the Top 10 Answer Candidates Produced by Each Individual Extractor

Average Precision	FST			LIGHT			SVM		
	BL	IP	JP	BL	IP	JP	BL	IP	JP
at rank1	0.691	0.873*	0.870*	0.404	0.604*	0.605*	0.282	0.532*	0.536*
at rank2	0.381	0.420*	0.463*	0.292	0.359*	0.383*	0.221	0.311*	0.339*
at rank3	0.260	0.270	0.297*	0.236	0.268*	0.268*	0.188	0.293*	0.248*
at rank4	0.174	0.195	0.195	0.201	0.222	0.222	0.167	0.193	0.199
at rank5	0.117	0.117	0.130	0.177	0.190	0.190	0.150	0.167	0.170

(*means the difference over the baseline is statistically significant ($p < 0.05$, t-test)).

requires $O(2^N)$ time and space where N is the size of the graph (i.e., number of answer candidates). Table IV(a) shows the answer-ranking performance when using the top 5, 10, 11, and 12 answer candidates, respectively. As can be seen, using more candidates tends to produce better performance. FST is exceptional because it tends to return a small number of candidates (i.e., the average number of answer candidates from FST is 4.19) and low-ranked answers are less reliable. As for LIGHT and SVM, accuracy tends to improve when using more answer candidates, but the improvement is small, even though the computation cost with 11 or 12 answer candidates will be two or four times more than the cost of using 10 candidates. Therefore, in this section, we use 10 candidates for the experiment, since we think this number is a reasonable trade-off between effectiveness and efficiency. Figure 7 shows how to generate the joint table using the top 10 answers. Given the joint table, we calculate the conditional and marginal probabilities. For example, the marginal probability of A_1 is calculated by summing the rows where the value of the 1st column is 1 (Eq. 7).

$$P(correct(A_1)|Q, A_1, \ldots, A_n) \approx \sum_{j \in \{JT(j,1)=1)_n} P(j, N+1) \tag{7}$$

The parameters for the model were estimated from the training data by maximizing the joint probability (Eq. (4)). This was done with the Quasi-Newton algorithm [Minka 2003].

1. Create a joint table (JT) whose size is $2^N \times (N+1)$, where N=10. Fill the column 1 to column N with all combinations of binary values and initialize the column N+1 with 0.
2. For each row i, calculate a value using the relevance and similarity features and store it to JT(i,N+1).

$$\mathrm{JT(i,N+1)} = \exp\Big(\sum_{i=1}^{n}\Big[(\sum_{k=1}^{K1}\beta_k rel_k(A_i))S_i + \sum_{N(i)}(\sum_{k=1}^{K2}\lambda_k sim_k(A_i, A_{N(i)}))S_i S_{Nn(i)}\Big]\Big)$$

3. To make each value a probability, divide each value by Z: $Z = \exp(\sum_{i=1}^{2^N} JT(i, N+1))$
4. To make each value a probability, divide each value by Z: $\mathrm{JT(i,N+1)} = \mathrm{JT(i,N+1)}/Z$

Fig. 7. Algorithm to generate the joint table using the top 10 answers.

Table IV(b) shows the performance of the joint prediction model, compared with the independent prediction model. As for TOP1, the joint prediction model significantly improved over baseline and performed as well as the independent prediction model in ranking the relevant answer at the top position. MRR5 shows the performance of the joint prediction model when they return multiple answers for each question. It can be seen that the joint prediction model performed better than the independent prediction model because it could identify unique correct answers by estimating conditional probability.

To further investigate the degree to which the joint prediction model could identify comprehensive results, we analyzed the average precision for the top five answers. Table IV(c) shows the average precision of the three models. It can be seen that the joint prediction model produced the answer list whose average precision is higher than the independent prediction model. This is additional evidence that the joint prediction model can produce a more comprehensive answer list.

However, this approach only uses the top 10 answer candidates, and hence misses the opportunity to boost the correct answer candidate which was ranked lower than 10. In the next section, we describe another approach to address this issue.

(b) *Exact inference with the top 10 answers produced by IP.* In the previous experiment, we limited the number of answers to the top 10. However, this is not extensible when more than 10 answer candidates were extracted from an extractor. To address this issue, we performed exact inference using the answer candidates filtered by the independent prediction model. We first applied the independent prediction model with all the candidates provided by each answer extractor. Then we chose the top 10 answer candidates returned from the independent prediction model as the input to the joint prediction model. Finally we did exact inference using enumeration.

Table V(a) compares the performance of the joint prediction model with the independent prediction model. It shows that the joint prediction model performed as well as the independent prediction model when selecting the top relevant answer for all extractors. When comparing MRR5, the joint prediction model performed better than the independent prediction model because it could identify unique correct answers by estimating conditional probability.

To further investigate the degree to the joint prediction model could identify comprehensive results, we analyzed the average precision within the top five

answers. Table V(b) shows the average precision of the model. It can be seen that the joint prediction model performed much better than the independent prediction model. For example, the average precision at rank 2 increased by 33% (FST), 43% (LIGHT), and 42% (SVM) over independent prediction. This is a significant improvement over the joint prediction model implemented in the previous section; as reported in Table IV(b), the previous implementation improved the average precision at rank 2 by only 10% (FST), 6% (LIGHT), and 9% (SVM).

(c) *Approximate inference using Gibbs sampling.* We tested the joint prediction model with only the top 10 answers provided either by each extractor or by the independent prediction model. Even though this worked well for factoid questions, limiting the number of answers may not be useful for list and complex questions because they may have more than 10 correct answers.

To address this issue, approximate inference can be used (e.g., Markov chain Monte Carlo sampling, Gibbs sampling, or variational inference). We used Gibbs sampling in our experiments, which has commonly been used for the undirected graphical model because it is simple and requires only conditional probability $P(S_i|S_{-i})$, where S_{-i} represents all nodes except S_i (Eq. (8)).

$$\begin{aligned} P(S_i|S_{-i}) &= \frac{P(S_i=1,S_{-i})}{P(S_i=1,S_{-i})+P(S_i=0,S_{-i})} \\ &= \frac{1}{1+\frac{P(S_i=0,S_{-i})}{P(S_i=1,S_{-i})}} \end{aligned} \tag{8}$$

Using this conditional probability, Gibbs sampling generates a set of samples: $S^{(0)}$, $S^{(1)}$, $S^{(2)}, \ldots, S^{(T)}$. Equation 9 shows how Gibbs sampling generates one sample $S^{(t+1)}$ from the previous sample $S^{(t)}$. In each sequence, each component $S^{(t+1)}$ is generated from the distribution conditional on the other components. This result $S^{(t+1)}$ is then used for sampling the next component.

$$\begin{aligned} &1.\ S_1^{(t+1)} \sim P\left(S_1|S_2^{(t)},\ldots,S_n^{(t)}\right) \\ &2.\ S_2^{(t+1)} \sim P\left(S_2|S_1^{(t+1)},S_3^{(t)},\ldots,S_n^{(t)}\right) \\ &3.\ S_i^{(t+1)} \sim P\left(S_i|S_1^{(t+1)},\ldots,S_{i-1}^{(t+1)},S_{i+1}^{(t)}S,\ldots,S_n^{(t)}\right) \\ &4.\ S_n^{(t+1)} \sim P\left(S_n|S_1^{(t+1)},\ldots,S_{n-1}^{(t+1)}\right) \end{aligned} \tag{9}$$

As it takes time for Gibbs sampling to converge, the first N samples are not reliable. Therefore, we ignored the first 2000 samples (this process is called burn-in). In addition, as all samples are not independent, we only used every 10th sample generated by Gibbs sampling (this process is called thinning).

The model parameters were estimated from training data using contrastive divergence learning, which estimates model parameters by approximately minimizing contrastive divergence. Contrastive divergence (CD) is defined using Kullback-Leibler divergence (KL), as shown in Eq. (10). This learning method has been used extensively with Gibbs sampling because it quickly converges after a few steps. More details about contrastive divergence can be found Hinton [2000].

$$CD_n = KL(p_0\|p_\infty) - KL(p_n\|p_\infty) \tag{10}$$

Table V(a). Performance of IP and JP Using the Top 10 Answers Produced by IP

	FST			LIGHT			SVM		
	BL	IP	JP	BL	IP	JP	BL	IP	JP
TOP1	0.691	0.880*	0.874*	0.404	0.624*	0.637*	0.282	0.584*	0.583*
MRR5	0.868	0.935*	0.950*	0.592	0.737*	0.751*	0.482	0.702*	0.724*

Table V(b). Average Precision of IP and JP at a Different Rank Using the Top 10 Answers Produced by IP

Average	FST			LIGHT			SVM		
Precision	BL	IP	JP	BL	IP	JP	BL	IP	JP
at rank1	0.691	0.880*	0.874*	0.404	0.624*	0.637*	0.282	0.584*	0.583*
at rank2	0.381	0.414*	0.548*	0.292	0.377*	0.541*	0.221	0.350*	0.498*
at rank3	0.260	0.269	0.377*	0.236	0.274*	0.463*	0.188	0.255*	0.424*
at rank4	0.174	0.178	0.259*	0.201	0.220	0.399*	0.167	0.203*	0.366*
at rank5	0.117	0.118	0.181*	0.177	0.191	0.349*	0.150	0.175*	0.319*

(*means the difference over the baseline is statistically significant ($p<0.05$, t-test)).

Table VI. Performance of JP Using Gibbs Sampling

	FST			LIGHT			SVM		
	BL	IP	JP	BL	IP	JP	BL	IP	JP
TOP1	0.691	0.880*	0.870*	0.404	0.624*	0.537*	0.282	0.584*	0.480*
MRR5	0.868	0.935*	0.930*	0.592	0.737*	0.657*	0.482	0.702*	0.638*

where p_0 is the data distribution, p_n is the empirical distribution at the n^{th} step and p_∞ is the model distribution.

Table VI shows the performance of Gibbs sampling. For the FST data set, Gibbs sampling worked as well as the independent prediction model. However, it did not work well for the LIGHT and SVM extractors, mostly because the answer list produced by LIGHT and SVM contained a lot of incorrect answers. The FST extractor contains 23.7% of correct answers in the answer candidate list. But LIGHT contains only 7.1% of correct answers in the candidate list and SVM contains only 7.7% of correct answers (see micro-level precision in Table III). Due to a significant imbalance between correct and incorrect answer candidates, Gibbs sampling and contrastive divergence learning did not work well for the LIGHT and SVM extractors.

(d) *Summary.* We implemented the joint prediction model in three different ways using exact inference and approximate inference. For exact inference, we applied enumeration by using the top 10 answers provided by either each individual answer extractor or the independent prediction model. For approximate inference, we used Gibbs sampling. While Gibbs sampling does not limit the number of answers, it still did not work well for the extractors, which produced significantly unbalanced data.

To address the unbalanced data problem, resampling, including over-sampling and under-sampling [Zhou 2006], can be applied. Over-sampling generates training data for the minority class, and under-sampling randomly removes training data from the majority class. Recently, Zhu and Hovy [2007]

proposed bootstrap-based over-sampling to reduce issues in over-sampling. Applying resampling to the data from the LIGHT and SVM extractors is one extension of this work that we intend to perform in the future. On the other hand, we can sample questions from the existing test data set so that we have a balanced training data set between negative and positive examples. This will make the training set smaller, but it may allow us to learn proper weights for the joint prediction model. This is another extension we intend to perform in the future. In addition, implementing the joint prediction model with different approaches (e.g., variable elimination, loopy belief propagation) is another expected extension.

In the rest of the experiments, we will use the second approach (exact inference using the outputs from the independent prediction model) to demonstrate the extensibility of the joint prediction model to other questions for other QA systems for other languages.

6.2.3 *List Questions.* The previous section shows that JP is better than IP when selecting multiple answers. In this section, we report another experiment to evaluate the performance of the joint prediction model for list questions.

The data set we used in the previous section contained many questions that had more than one correct answer. Especially, location, a person's name, numeric and temporal questions tend to have more than one correct answer. For example, given the question "Where is the tallest roller coaster located?", there are three answers in the TREC corpus: Cedar Point, Sandusky, Ohio. All of them are correct, although they represent geographical areas of increasing generality. Some questions require more than one correct answer. For example, for the question "Who is the tallest man in the world?", the correct answers in the TREC corpus are "Monjane, Gabriel Estavao, Robert Wadlow, AliNashnush, Barman". In addition, there are some list questions (e.g., "Name one of the major gods of Hinduism.").

To evaluate these list questions, we extracted 482 questions whose number of correct answers is more than three from the TREC8-12 questions; an average number of correct answers is 5.7. As the FST extractor returns the average of 4.19 answer candidates (shown in Table III), we only used the LIGHT and SVM extractors for this experiment. Table VII(a) shows the characteristics of the LIGHT and SVM extractors for the list questions.

To investigate the degree to which the joint prediction model could identify comprehensive results, we analyzed the average precision within the top 10 answers. Table VII(b) shows the average precision of the model. It can be seen that the joint prediction model performed much better than the independent prediction model. For example, the average precision at rank 10 increased by 85% (LIGHT) and 78% (SVM) over independent prediction. This demonstrates that the joint prediction model is better at finding a unique set of correct answers for list-type questions.

6.3 Experiments with Ephyra

To test the joint prediction model in the EPHYRA QA system, we used a total of 998 factoid questions from the TREC13-15 QA evaluations. This data

Table VII(a). Performance Characteristics of the LIGHT and SVM Extractors on List Questions

Extractor	# Questions with Correct Answers	Avg. Num of Answers per Question	Precision	
			Macro	Micro
LIGHT	203	36.4	0.679	0.110
SVM	196	35.3	0.690	0.125

Table VII(b). Average Precision on List Questions

	LIGHT		SVM	
	IP	JP	IP	JP
at rank 1	0.532	0.547	0.473	0.493
at rank 2	0.355	0.461*	0.318	0.411*
at rank 3	0.250	0.386*	0.236	0.343*
at rank 4	0.217	0.346*	0.195	0.308*
at rank 5	0.188	0.315*	0.174	0.286*
at rank 6	0.164	0.284*	0.159	0.260*
at rank 7	0.144	0.251*	0.143	0.242*
at rank 8	0.127	0.228*	0.132	0.233*
at rank 9	0.113	0.207*	0.120	0.214*
at rank 10	0.104	0.193*	0.112	0.200*

Table VIII. Performance Characteristics of EPHYRA Extractors

Extractor	# Questions with Correct Answers	Avg. Num of Answers per Question	Precision	
			Macro	Micro
Extractor1	464	27	0.465	0.026
Extractor2	305	104	0.306	0.008

Table IX. Performance of IP and JP in EPHYRA

	Extractor 1			Extractor 2		
	BL	IP	JP	BL	IP	JP
TOP1	0.508	0.581*	0.581*	0.497	0.603*	0.607*
MRR5	0.706	0.755*	0.758*	0.664	0.749*	0.745*

set is quite different from the previous TREC8-12 data set which was used to evaluate JAVELIN; the questions from TREC8-12 include many list questions, but questions from the TREC13-15 factoid task tend to have only one correct answer. As there is a separate task for list questions in the recent TREC QA tasks, most factoid questions require only one correct answer. EPHYRA has two extractors: Extractor1 and Extractor2. Extractor1 exploits answer types to extract associated named entities, and Extractor2 uses patterns that were obtained automatically from question-answer pairs in the training data. Table VIII shows the characteristics of the EPHYRA extractors. It can be seen that microlevel precision was lower here than in the JAVELIN case, which means there are many more incorrect answer candidates.

Table IX shows the performance of the joint prediction model in the EPHYRA system. TOP1 shows that the joint prediction model improved performance significantly over the baseline, and performed as well as the independent prediction model in ranking the relevant answer at the top position for the EPHYRA case. When comparing MRR5, there is no significant difference between IP and

JP, due to the fact that the questions from TREC 13-15 factoid questions tend to have only one correct answer. However, this experiment demonstrates that the joint prediction model still performs well for another QA system whose microprecision is quite low (i.e., there are a lot of incorrect answer candidates).

6.4 Answer-Merging

In the previous sections, we evaluated the joint prediction model with the answer list produced by each individual extractor. As different extractors cover different types of questions with varying precision and recall, we merged their answers with the extended models (described in Section 5.1).

We used the same data set as used to evaluate each individual JAVELIN and EPHYRA extractor. As different extractors returned different numbers of answer candidates, we only considered the top 50 answer candidates produced by each individual extractor. In case of JAVELIN, among 1760 questions in the data set, there were 978 questions for which at least one extractor identified a correct answer. In case of EPHYRA, among the total of 998 questions, only 553 questions had at least one correct answer. As each extractor covers different questions, the performance was measured using answer coverage, calculated by the percentage of correct top answers among the questions for which at least one correct answer exists (998 questions for JAVELIN and 553 questions for EPHYRA).

Two baselines were used: MaxScore and CombSum. MaxScore picks the highest score among the scores provided by the extractors. CombSum sums the scores from the extractors and then reranks the answers according to the sum.

Tables X(a) and X(b) show the experimental results on answer-merging when using four different answer-merging models. As can be seen, answer-merging with IP and JP significantly improved the coverage over CombSum and MaxScore. This shows that both IP and JP are very effective in merging answers produced by different extraction strategies. To analyze the performance of JP in answer-merging, we also compared the coverage of JP when using the results from each individual extractor v.s. multiple extractors (X(c) and X(d)). "LIGHT with JP" is the coverage of the joint prediction model for answer candidates produced by the LIGHT extractor. As can be seen, JP selected the correct top answers for 57.9% of the questions when taking the answer candidates provided by the LIGHT extractor as an input, but merging its results with the results from FST and SVM improved the performance. For the EPHYRA case, answer-merging with JP also improved the performance over the individual result. As we add more answer extractor results, we expect to have more performance gain. Analyzing how much JP performs better while adding more extractors is one of our future work.

7. EXPERIMENTAL RESULTS: CHINESE AND JAPANESE QA

This section describes the experiments we used to evaluate the models for Chinese and Japanese monolingual QA and English-to-Chinese and English-to-Japanese cross-lingual QA. The JAVELIN QA system [Mitamura et al. 2007] was used as a testbed for the evaluation.

Table X(a). Answer Merging in JAVELIN Using Different Models

CombSum	MaxScore	Merging with IP	Merging with JP
29%	40.4%	60.3%	61.7%

Table X(b). Answer Merging in EPHYRA Using Different Models

CombSum	MaxScore	Merging with IP	Merging with JP
48.1%	50.6%	53.3%	53.3%

Table X(c). Coverage of JP for Individual Extractor vs. Answer Merging in JAVELIN

FST with JP	LIGHT with JP	SVM with JP	Merging with JP
26.9%	57.9%	51.9%	61.7%

Table X(d). Coverage of JP for Individual Extractor vs. Answer-Merging in EPHYRA

Extractor1 with JP	Extractor2 with JP	Merging with JP
49.7%	34.5%	53.3%

Table XI. Performance Characteristics of Chinese and Japanese Extractors for Monolingual and Cross-Lingual QA

Extractor	# Questions with Correct Answers	Avg. Num of Answers per Question	Precision	
			Macro	Micro
C-C (Chinese-to-Chinese)	272	565.8	0.777	0.010
E-C (English-to-Chinese)	190	76.6	0.543	0.029
J-J (Japanese-to-Japanese)	251	58.5	0.628	0.077
E-J(English-to-Japanese)	166	53.3	0.415	0.043

7.1 Data Set

The 550 Chinese questions provided by the NTCIR 5-6 QA evaluations served as the data set. As we have a small number of questions compared to the English case, we split the question for extraction and answer-ranking to avoid overfitting. Among them, 200 questions were used to train the Chinese answer extractor and 350 questions were used to evaluate our answer-ranking model. For Japanese, we used 700 Japanese questions provided by the NTCIR 5-6 QA evaluations as the data set. Among them, 300 questions were used to train the Japanese answer extractor, and 400 questions were used to evaluate our model.

Table XI shows the characteristics of the extractors. As for Chinese-to-Chinese, the extractor returned many answer candidates (the average number of answer candidates was 565.8) and micro-level precision was very low. Therefore, we preprocessed the data to remove answer candidates having rank lower than 100. When comparing macro-precision between monolingual and cross-lingual QA, the macro-precision is much lower in the cross-lingual case, which shows the difficulty in cross-lingual QA.

7.2 Baselines

As there has been little research that compares answer selection performance of different answer-ranking approaches for Chinese and Japanese, we report

the performance of other baseline algorithms as well as that of our answer-ranking models. These baseline algorithms have been used extensively for answer-ranking in many QA systems.

(1) *Extractor*: Answer extractors apply different techniques to extract answer candidates from the retrieved documents or passages, and assign a confidence score for each individual answer. As a simple baseline, we reranked the answer candidates according to the confidence scores provided by answer extractors.

(2) *Clustering*: This approach clusters identical or complementary answers and then assigns a new score to each cluster. In our experiments, we used the approach reported in Nyberg et al. [2003]. For a cluster containing N answers whose extraction confidence scores are $S_1, S_2, \ldots, S_n$, the cluster confidence is computed with the following formula:

$$Score(Answer\ Cluster) = 1 - \prod_{i=1}^{n}(1 - S_i) \tag{11}$$

(3) *Filtering*: We used both ontologies and gazetteers to filter out improper answer candidates. The algorithms described in Section 4.1.1 were used to identify improper answer candidates, and then these candidates were removed from the answer candidate list.

(4) *Web validation*: The approach proposed by Magnini et al. [2002] for QA was used as another baseline. We implemented three variants to rerank answer candidates using Web validation scores: (1) rerank answer candidates according to the Web validation scores; (2) add the Web score to the extractor score and then rerank answer candidates according to the sum; and (3) use a linear regression to learn the weight for both extractor scores and Web scores and then rerank candidates according to the results from the linear regression. In our experiments, the linear regression method was more accurate than the other methods. Therefore, we used linear regression to combine the Web validation scores with the extractor scores.

(5) *Combination*: We also combined three baseline systems (*Clustering, Filtering, and Web validation*) using linear regression in order to see the extent to which the combined approaches could improve answer-ranking performance. Three combinations were tested: Clutering+Filtering(C+F), Clustering+Web (C+W), and Clustering+Filtering+Web ($C + F + W$).

(6) *MaxEnt* with internal resources: Maximum entropy has been used for the answer-ranking task in several QA systems [Hickl et al. 2007; Ravichandran et al. 2003] using internal resources such as answer type, frequency of answers in the candidate list, and so on. We implemented a maximum entropy reranker as another baseline with extensively used internal features: (1) frequency of answers in the candidate list; (2) answer type matching; (3) question word absent in the answer sentence; (4) inverse term frequency of question keywords in the answer sentence; (5) confidence in individual answer candidate provided by an extractor; and (6) the expected answer type. As this baseline is not using any external resources, comparing it with our models can show the degree to which the external resources are useful in answer-ranking.

Table XII. Average Top Answer Accuracy in Chinese and Japanese QA (C+F: combination of clustering and filtering, C+W: combination of clustering and web validation, C+F+W: combination of clustering, filtering and web validation)

	EXT	CLU	FIL	WEB	C+F	C+W	C+F+W	ME	IP	JP
C-C	0.389	0.462	0.432	0.547	0.547	0.543	0.556	0.556	**0.644**	**0.645**
E-C	0.299	0.380	0.321	0.402	0.397	0.451	0.451	0.424	**0.462**	**0.467**
J-J	0.498	0.536	0.498	0.528	0.528	0.545	0.545	0.557	**0.570**	**0.572**
E-J	0.427	0.445	0.427	0.476	0.451	0.451	0.451	0.445	**0.482**	**0.482**

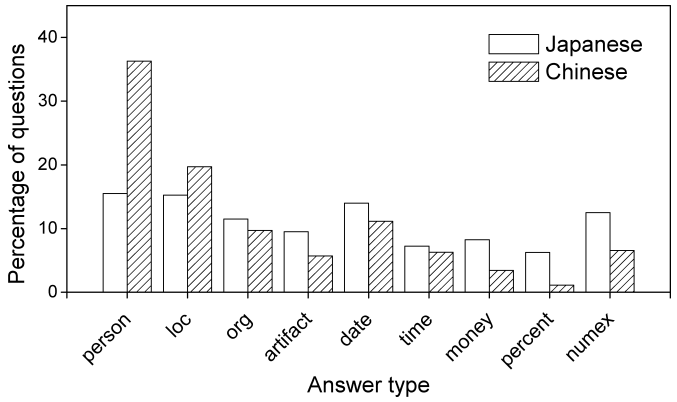

Fig. 8. Answer type distribution in a Chinese and Japanese data set.

7.3 Results and Analysis

Table XII compares the average top answer accuracy when using the baseline systems, the independent prediction model, and the joint prediction model. Among the baseline systems which used a single feature, *Web validation* produced the best performance in Chinese (both C-C and E-C). However, *Web validation* was less useful in Japanese. This can be explained by analyzing the difference in the data set. Figure 8 compares answer type distribution in Chinese and Japanese. In the Chinese data set, 66% of questions look for names (person name, organization name, and location name), 11% for numbers, and 17% for temporal expressions. But in the Japanese data set, far fewer questions look for names (42%) while more questions search for numbers (27%) and temporal expressions (21%). *Web validation* is less useful in validating numeric and temporal questions because correct answers to numeric and temporal questions may vary over even for short periods of time. In addition, some answers are too specific and hard to find within Web documents (e.g., "At what hour did a truck driven by Takahashi rear-end a truck driven by Hokubo?"). As the Japanese question set contains many more numeric and temporal questions, *Web validation* was not as useful as in the Chinese case.

When comparing the combination of baseline systems, $C + F$ worked better than individual clustering and filtering, which suggests that combining more resources was useful in answer selection. However, $C + F + W$ and $C + W$ did not perform well all the time. For the English-to-Japanese case, $C + F + W$ hurt the answer selection performance compared to *Web validation*. For the

Table XIII. Average Precision of IP and JP at a Different Rank

Average Precision	C-C		J-J		E-C		E-J	
	IP	JP	IP	JP	IP	JP	IP	JP
at rank1	0.644	0.645	0.570	0.572	0.462	0.467	0.482	0.482
at rank2	0.356	0.401*	0.315	0.379*	0.255	0.293*	0.277	0.308
at rank3	0.247	0.290*	0.237	0.271*	0.190	0.246*	0.209	0.226
at rank4	0.200	0.226*	0.185	0.209*	0.155	0.196*	0.165	0.181
at rank5	0.167	0.186*	0.156	0.171*	0.135	0.164*	0.140	0.150

Chinese-to-Chinese case, $C + W$ produced lower scores than *Web validation*. This again demonstrates that in this case combining multiple strategies is hard.

However, when comparing the baseline systems with the independent prediction model, we note that it always obtained a better performance gain than the baseline systems, and the joint prediction model worked as well as the independent prediction model. As for Chinese-to-Chinese, both models improved performance by 15.8% over the best baseline systems ($C + F + W$ and *MaxEnt reranking*). In Japanese-to-Japanese, both models slightly improved the average top answer accuracy (an increase of 2.25% over *MaxEnt reranking*). As for the cross-lingual case, there was less performance gain than the monolingual case, which is the expected result when considering the difficulty of cross-lingual QA.

As there is no significant difference between independent prediction and joint prediction in selecting the top answer, we further investigated the degree to which the joint prediction model could identify comprehensive results. Table XIII compares the average precision of IP and JP at rank N and shows that JP performed better than IP when selecting the top five answers for all the cases. This shows that joint prediction could successfully identify unique correct answers by estimating conditional probability in other languages.

7.4 Utility of Data-Driven Features

In our experiments, we used data-driven features as well as knowledge-based features. As knowledge-based features require manual effort to provide an access to language-specific resources for each language, we conducted an additional experiment with data-driven features, in order to see how much performance gain is available without the manual work. As Web, Wikipedia and string similarity metrics can be used without any additional manual effort when extended to other languages; we used these three features and compared performance in JAVELIN.

Table XIV shows the performance when using data-driven features vs all features in the independent prediction model. For all three languages, data-driven features alone achieved significant improvement over *Extractor*. This indicates that our approach can be easily extended to any language where appropriate data resources are available, even if knowledge-based features and resources for the language are still under development.

Table XIV. Average Top Answer Accuracy Using Data-Driven Features vs Using All Features

	Extractor	Data-driven Features	All Features
E-E (FST)	0.691	0.840*	0.880*
E-E (LIGHT)	0.404	0.617*	0.624*
E-E (SVM)	0.282	0.556*	0.584*
C-C	0.386	0.635*	0.644*
E-C	0.299	0.424*	0.462*
J-J	0.478	0.553*	0.570*
E-J	0.427	0.457*	0.482*

(* means the difference over Extractor is statistically significant ($p < 0.05$, t-test)).

8. COMPARISON WITH OTHER QA SYSTEMS

In the previous sections, we evaluated the models with cross-validation in order to see how much they improved the average answer accuracy in one QA system. In this section, we compare the QA systems that incorporate our approach with other QA systems that participated in the recent TREC and NTCIR QA task.

8.1 Experimental Setup

Questions from the recent TREC and NTCIR evaluations served as a test set: the TREC-2006 evaluation contains 403 English factoid questions, and the NTCIR-6 evaluation contains 150 Chinese factoid questions and 200 Japanese factoid questions. All other questions from the previous TREC and NTCIR evaluations were used as a training set.

As both TREC and NTCIR use the top answer accuracy as an evaluation metric to evaluate factoid questions, we used the top answer accuracy to compare the performance. As the experiments in the previous sections showed that there was no significant difference in selecting the top answer between the independent prediction model and the joint prediction model, we only used the independent prediction model for this experiment.

8.2 Results and Analysis

Table XV shows the performance of EPHYRA and JAVELIN with and without the independent prediction model for answer selection. It can be seen that JAVELIN and EPHYRA with the model worked much better than the TREC and NTCIR median runs for all languages. As for Japanese (both Japanese-to-Japanese and English-to-Japanese), JAVELIN with IP performed better than the best QA system in NTCIR-6.

9. SUMMARY

We conducted a series of experiments to evaluate the performance of our models in multilingual QA. Multilingual QA includes two tasks: English/Chinese/Japanese monolingual QA and English-to-Chinese/English-to-Japanese cross-lingual QA. The former provides a testbed to evaluate the degree to which our models are extensible to different languages. The latter entails question translation from English to another language and tends to have poor quality data.

Table XV. Performance Comparison with TREC-2006 (English) and NTCIR-6 (Chinese and Japanese) Systems

	Testbed	Testbed Score w/o IP	Testbed Score with IP	TREC/NTCIR Best Score	TREC/NTCIR Median Score
E-E	Ephyra	0.196	0.238	0.578	0.134
C-C	Javelin	0.287	0.393	0.547	0.260
E-C	Javelin	0.167	0.233	0.340	0.107
J-J	Javelin	0.320	0.370	0.360	0.295
E-J	Javelin	0.215	0.235	0.195	0.140

Table XVI. Performance Gain of IP over Baselines and Characteristics of Testbed Systems

System	Improvement Over Extractor	Improvement Over CLU, FIL, WEB, ME	Characteristics
E-E (FST)	27.35%*	10.41% (WEB)*	Redundant answers exist in the candidate list, and exploiting redundancy is important. Fine-granulated answer type and subtypes (useful for filtering)
E-E (LIGHT)	54.46%*	20.00%(WEB)*	
E-E (SVM)	107.09%*	19.43%(WEB)*	
E-E (Ephyra1)	14.37%*	0.69%(WEB)	The extractor already merged redundant answers (no gain from similarity features). High variance in extractor scores. Not enough subtype information.
E-E (Ephyra2)	21.33%*	1.01%(WEB)	
C-C	65.55%*	15.83%(ME)*	Data set has many name questions (web validation is useful for them).
E-C	54.52%*	8.96%(ME)*	
J-J	14.46%*	2.33% (ME)	Extractor output is more accurate than Chinese (higher baseline than Chinese). Data set has more numeric questions and fewer name questions (numeric questions are hard to validate: corpus-specific).
E-J	12.88%*	1.26% (WEB)	

(* means the difference is statistically significant ($p < 0.05$, t-test)).

Applying the models to cross-lingual QA shows the degree to which the models are noise-resistant in supporting data of poor quality.

Table XVI summarizes the performance gain of the independent prediction model over the baseline systems; the performance results of baseline systems for English QA come from our earlier work [Ko et al. 2009], which showed the effectiveness of the independent prediction model on English QA. As can be seen in Table XVI, the performance of the model varies according to the characteristics of the input quality (e.g., score distribution, degree of answer redundancy, availability of external resources, question distribution, etc), but in all cases the model improved answer selection performance over baseline systems.

However, answer-ranking performance is inherently system-dependent. Although we may be able to characterize contexts in which different approaches are likely to perform well, many of the details (e.g., cutoff threshold decisions, feature selection) must be learned for specific QA systems (corpora, languages,

etc.). Additionally, as translation quality improves, the best approach to answer-ranking may change.

When comparing the performance of the models in different question types, the models worked well for lexical types of questions (such as location, person name, organization name, etc.), but did not improve much on numeric and temporal questions because it was hard to validate them with the resources we used (this is also shown in Lita et al. [2004]). To address this issue, more general canonicalization can be used to identify candidates that represent the same entity. For example, the application of unit conversion will make "1kg" equal to "1000g". Or we can consider containment and use it to support another answer candidate (e.g., "April 15 1912" supports "April 1912" for the question "When did the Titanic sink?"). Incorporating more encyclopedias is another way to increase the coverage of range-checking on numeric questions (described in Section 4.1.1).

10. CONCLUSIONS

This article described two probabilistic answer-ranking models for multilingual question-answering. Both models consider the relevance of individual answers as well as their correlation in order to rank the answer candidates. However, they estimate the answer probability in different ways. The independent prediction model directly estimates the probability of each individual answer candidate using logistic regression. The joint prediction model estimates the joint probability of all available answer candidates using a probabilistic graphical model and then drives the marginal and conditional probability for each individual answer to generate a more accurate and comprehensive answer list. An extensive set of empirical results on TREC and NTCIR questions shows that the models significantly improved answer-ranking performance and the joint prediction model was better at finding a unique set of correct answers (e.g., for a list-type question).

This article extends the prior research in three important directions. First, it addresses the criticism that the joint prediction model is not thorough enough because it was implemented by using the top 10 answer candidates and was tested using English factoid questions in one QA system. In this article, we conducted thorough experiments on the joint prediction model by applying approximate inference to rerank more than 10 answer candidates, evaluating its performance on list questions, and using it to merge the answer candidates provided by multiple extraction techniques in multiple QA systems. As far as we know, no previous work has jointly modeled the correctness of available answer candidates in a formal probabilistic framework for question-answering, which is very important for generating an accurate and comprehensive answer list.

Second, we demonstrated that both models are effective on cross-lingual QA (English-to-Chinese and English-to-Japanese) as well as monolingual QA. Chinese and Japanese were chosen to show the extensibility of the models to multilingual answer-ranking because they are very different from English, and translation accuracy (from English to Chinese and English to Japanese) tends

to be lower than that in European languages. This causes answer-ranking to be more challenging in these languages, as it requires an additional degree of robustness to handle low-quality answer candidates. Hence, showing the effectiveness of the models in these languages can clearly demonstrate that the models provide a general probabilistic framework to support multilingual QA. In addition, we showed that our approach can be easily extended to any language using data-driven relevance and similarity features, even if knowledge-based features and resources for the language are still under development.

Third, the prior research only evaluated the models with cross-validation in order to see how much they improved the average answer accuracy in one QA system. In this article, we compared our approach with other QA systems that participated in recent TREC and NTCIR evaluations, and showed that our approach worked much better than the TREC and NTCIR median runs for all three languages. As for Japanese (both Japanese-to-Japanese and English-to-Japanese), our system performed better than the best QA system in NTCIR-6.

To our best understanding, this is the first work that proposes a formal unified probabilistic framework to model the correctness and correlation of answer candidates in multilingual question-answering. In addition, this is a novel approach to design a flexible and extensible answer-ranking architecture which can easily incorporate multiple relevance and similarity features with one training step for the language in question.

We plan to extend the models for other types of questions and languages, and here are possible directions for future research.

(1) *Extension to complex questions*: We conducted a series of experiments on factoid and list questions whose answers are short noun phrases or named entities. As a generalized framework, the models should be able to support complex questions that require longer answers representing facts or relations (e.g., "What is the relationship between Alan Greenspan and Robert Rubin?", "Who is Bill Clinton?"). For these questions, we need to develop different features for answer selection. One relevance feature is to use a predicate structure match [Nyberg et al. 2006]. For example, given the question "Did Egypt sell Scud missiles to Syria?", the key predicate from the question is Sell(Egypt, Syria, Scud missile). If there is a sentence that contains the predicate structure Buy(Syria, Scud missile, Egypt), we can calculate the predicate structure distance and use it as a relevance feature. For answer similarity, some approaches used in the TREC Novelty track can be applied [Soboroff 2005]. The Novelty track has experimented with finding relevant and novel sentences from a document collection, so we can reuse some approaches in this task. For example, the number of new named entities, the number of new words, or the language model as a feature to measure the similarity of two answer candidates [Allan et al. 2003]. Semantic match can be applied to measure the semantic similarity (e.g., "Sell (Egypt, Syria, Scud missile)" vs "Buy (Syria, Scud missile, Egypt)") between answer candidates.

(2) *Inference for joint prediction*: The joint prediction model was implemented with exact inference and approximate inference. However, the experimental results show that our approximate inference approach did not work well when the data set is significantly unbalanced. To address the unbalanced data problem,

resampling such as over-sampling and under-sampling was applied to the data from the LIGHT and SVM extractors. In addition, more experiments should be done to support different exact and approximate inferences such as variable elimination, loopy belief propagation, mean field, and so on.

(3) *Merging the results from multiple QA systems*: We can extend the models to unify answers provided by several QA systems in order to have higher quality answers. As each QA system has different score distribution and different coverage on specific types of questions for different languages, it is important to select the best QA systems for each type of question and merge the results. Merging has been extensively studied in the context of information retrieval [Aslam and Montague 2001; Si 2006]. A natural question is whether algorithms proven successful for this task can be applied to answer-merging in the QA domain using our answer-ranking model. This is an interesting area for future research.

(4) *Extending features*: In this article, we used dictionary-based semantic similarity and surface-level string similarity (such as Levenshtein) to estimate the similarity among answer candidates. Future work is extending it to utilize contextual similarity by considering contexts from which the answers are extracted, and adding more data resources to estimate answer relevance. Recently, there has been research on supporting a unified ontology search; the Federated Ontology Search (FOS) is one system that provides a unified interface to search for several knowledge-bases [Pedro 2006]. Adding more resources like FOS can also be future work.

ACKNOWLEDGMENTS

We would like to thank NTCIR for providing the Japanese and Chinese corpora and data set. We would also like to thank Jamie Callan for his valuable discussion and suggestions.

REFERENCES

AHN, D., JIJKOUN, V., MISHA, G., MLLER, K., DE RIJKE, M., AND SCHLOBACH, S. 2004. Using Wikipedia at the TREC QA Track. In *Proceedings of the Text Retrieval Conference*.

ALLAN, J., WADE, C., AND BOLIVAR, A. 2003. Retrieval and novelty detection at the sentence level. In *Proceedings of the ACM SIGIR Conference on Research and Development on Information Retrieval*. ACM, New York.

ASLAM, J. AND MONTAGUE, M. 2001. Models for meta-search. In *Proceedings of the ACM SIGIR Conference on Research and Development on Information Retrieval*. ACM, New York.

BOS, J. AND NISSIM, M. 2006. Cross-lingual question answering by answer translation. In *Working Notes of the Cross Language Evaluation Forum*.

BRILL, E., DUMAIS, S., AND BANKO, M. 2002. An analysis of the AskMSR question answering system. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

BUSCALDI, D. AND ROSSO, P. 2006. Mining knowledge from Wikipedia for the question answering task. In *Proceedings of the International Conference on Language Resources and Evaluation*.

CARDIE, C., PIERCE, D., NG, V., AND BUCKLEY, C. 2000. Examining the role of statistical and linguistic knowledge sources in a general-knowledge question-answering system. In *Proceedings of the 6th Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

CHEN, H. H., LIN, C. C., AND LIN, W. C. 2000. Construction of a Chinese-English WordNet and its application to CLIR. In *Proceedings of the 5th International Workshop on Information Retrieval with Asian Languages*.

CHU-CARROLL, J., CZUBA, K., PRAGER, J., AND ITTYCHERIAH, A. 2003. In question answering, two heads are better than one. In *Proceedings of the Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

CHU-CARROLL, J., CZUBA, K., PRAGER, J., ITTYCHERIAH, A., AND BLAIR-GOLDENSOHN, S. 2005. IBM's PIQUANT II in TREC2004. In *Proceedings of the Text Retrieval Conference*.

CLARKE, C., CORMACK, G., AND LYNAM, T. 2001. Exploiting redundancy in question answering. In *Proceedings of* the *ACM SIGIR Conference on Research and Development on Information Retrieval*.

DONG, Z. 2000. Hownet. http://www.keenage.com.

ECHIHABI, A., HERMJAKOB, U., HOVY, E., MARCU, D., MELZ, E., AND RAVICHANDRAN, D. 2004. How to select an answer string? In *Advances in Textual Question Answering*, Kluwer, Amsterdam.

FELLBAUM, C. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

HARABAGIU, S., MOLDOVAN, D., CLARK, C., BOWDEN, M., WILLIAMS, J., AND BENSLEY, J. 2003. Answer mining by combining extraction techniques with abductive reasoning. In *Proceedings of the Text Retrieval Conference*.

HARABAGIU, S., MOLDOVAN, D., PASCA, M., MIHALCEA, R., SURDEANU, M., BUNSECU, R., GIRJU, R., RUS, V., AND MORARESCU, P. 2001. Falcon: Boosting knowledge for answer engines. In *Proceedings of the Text Retrieval Conference*.

HICKL, A., WILLIAMS, J., BENSLEY, J., ROBERTS, K., SHI, Y., AND RINK, B. 2007. Question answering with LCC's CHAUCER at TREC 2006. In *Proceedings of the Text Retrieval Conference*.

HINTON, G. AND SEJNOWSKI, T. J. 1986. Learning and relearning in Boltzmann machines. In *Parallel Distributed Processing*, 282–317.

HINTON, G. 2000. Training products of experts by minimizing contrastive divergence. Japanese on WordNet. http://nlpwww.nict.go.jp/wn-ja/index.en.html.

JARO, M. A. 1995. Probabilistic linkage of large public health data files. *Stat. Medicine 14*, 5–7, 491–498.

JIJKOUN, V., VAN RANTWIJK, J., AHN, D., SANG, E.T. K., AND DE RIJKE, M. 2006. The University of Amsterdam at CLEF@QA 2006. In *Working Notes of* the *Cross Language Evaluation Forum*.

JIJKOUN, V., KAMPS, J., MISHNE, G., MONZ, C., DE RIJKE, M., SCHLOBACH, S., AND TSUR, O. 2003. The University of Amsterdam at TREC 2003. In *Proceedings of the Text Retrieval Conference*.

KATZ, B., LIN, J. J., LORETO, D., HILDEBRANDT, W., BILOTTI, M., FELSHIN, S., FERNANDES, A., MARTON, G., AND MORA, F. 2003. Integrating web-based and corpus-based techniques for question answering. In *Proceedings of the Text Retrieval Conference*.

KO, J., SI, L., AND NYBERG, E. 2010. Combining evidence with a probabilistic framework for answer ranking and answer merging in question answering. *Inform. Process, Manage.* To appear.

KO, J., MITAMURA, T., AND NYBERG, E. 2007a. Language-independent probabilistic answer ranking for question. In *Proceedings of the Association for Computational Linguistics*.

KO, J., SI, L., AND NYBERG, E. 2007b. A probabilistic graphical model for joint answer ranking in question answering. In *Proceedings of the ACM SIGIR Conference on Research and Development on Information Retrieval*.

KWOK, C., ETZIONI, O., AND WELD, D. 2001. Scaling question answering to the Web. In *Proceedings of the Text Retrieval Conference*.

LIN, F., SHIMA, H., WANG, M., AND MITAMURA, T. 2005. CMU JAVELIN system for NTCIR5 CLQA1. In *Proceedings of the 5th NTCIR Workshop* (NII Test Collection for IR Systems).

LITA LV., HUNT W., AND NYBERG E. 2004. Resource analysis for question answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

MAGNINI, B., NEGRI, M., PERVETE, R., AND TANEV, H. 2002. Comparing statistical and content-based techniques for answer validation on the web. In *Proceedings of the VIII Convegno AI*IA*.

MEI, J., ZHU, Y., GAO, Y., AND YIN, H. 1982. *Tongyici Cilin*. Shanghai Dictionary Press.

MINKA, T. 2003. A comparison of numerical optimizers for logistic regression. Unpublished draft.

MITAMURA, T., LIN, F., SHIMA, H., WANG, M., KO, J., BETTERIDGE, J., BILOTTI, M., SCHLAIKJER, A., AND NYBERG, E. 2007. JAVELIN III: Cross-lingual question answering from Japanese and Chinese documents. In *Proceedings of the 6th NII Test Collection for IR Systems Workshop*.

MOLDOVAN, D., CLARK, D., HARABAGIU, S., AND MAIORANO, S. 2003. Cogex: A logic prover for question answering. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

NYBERG, E., MITAMURA, T., FREDERKING, R., BILOTTI, M., HANNAN, K., HIYAKUMOTO, L., KO, J., LIN, F., PEDRO, V., AND SCHLAIKJER, A. 2006. JAVELIN I and II systems at TREC 2005. In *Proceedings of Text Retrieval Conference*.

NYBERG, E., MITAMURA, T., FREDERKING, R., PEDRO, V., BILOTTI, M., SCHLAIKJER, A., AND HANNAN, K. 2005. Extending the JAVELIN QA system with domain semantics. In *Proceedings of the 20th National Conference on Artificial Intelligence*.

NYBERG, E., MITAMURA, T., CARBONELL, J., CALLAN, J., FREDERKING, R., COLLINS-THOMPSON, K., HIYAKUMOTO, L., HUANG, Y., HUTTENHOWER, C., JUDY, S., KO, J., KUPSE, A., LITA, L., PEDRO, V., SVOBODA, D., AND VAN DURME, B. 2004. The JAVELIN question-answering system at TREC 2003: A multi-strategy approach with dynamic planning. In *Proceedings of the Text Retrieval Conference.*

NYBERG, E., MITAMURA, T., CARBONELL, J., CALLAN, J., COLLINS-THOMPSON, K., CZUBA, K., DUGGAN, M., HIYAKUMOTO, L., HU, N., HUANG, Y., KO, J., LITA, L., MURTAGH, S., PEDRO, V., AND SVOBODA, D. 2003. The JAVELIN question-answering system at TREC 2002. In *Proceedings of the Text Retrieval Conference*.

PEDRO, V. 2006. Federated ontology search. Thesis proposal.

PRAGER, J., BROWN, E., CODEN, A., AND RADEV, D. 2000. Question answering by predictive annotation. In *Proceedings of the ACM SIGIR Conference on Research and Development on Information Retrieval*. ACM, New York.

PRAGER, J., CHU-CARROLL, J., CZUBA, K., WELTY, C., ITTYCHERIAH, A., AND MAHINDRU, R. 2004. IBM's PIQUANT in TREC2003. In *Proceedings of the Text Retrieval Conference*.

RAVICHANDRAN, D., HOVY, E., AND OCH, F. J. 2003. Statistical QA—Classifier vs. Re-ranker: What's the difference? In *Proceedings of the ACL Workshop on Multilingual Summarization and Question Answering*.

SCHLOBACH, S., OLSTHOORN, M., AND DE RIJKE, M. 2004. Type checking in open-domain question answering. In *Proceedings of the European Conference on Artificial Intelligence*.

SI, L. 2006. Federated search of text search engines in uncooperative environments. Thesis.

SOBOROFF, I. 2005. Overview of the TREC 2004 novelty track. In *Proceedings of the Text Retrieval Conference*.

VOORHEES, E. 2003. Overview of the TREC 2003 question answering track. In *Proceedings of the Text Retrieval Conference.*

VOORHEES, E. 2002. Overview of the TREC 2002 question answering track. In *Proceedings of the Text Retrieval Conference*.

WANG, M., SAGAE, K., AND MITAMURA, T. 2006. A fast, accurate deterministic parser for Chinese. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*.

WINKLER, W. E. 1999. The state of record linkage and current research problems. Tech. rep., Statistical Research Division, U.S. Census Bureau, Washington, D.C.

XU, J., LICUANAN, A., MAY, J., MILLER, S., AND WEISCHEDEL, R. 2003. TREC 2002 QA at BBN: Answer selection and confidence estimation. In *Proceedings of the Text Retrieval Conference*.

YANG, H., CUI, H., MASLENNIKOV, M., QIU, L., KAN, M.-Y., AND CHUA, T.-S. 2003. QUALIFIER In TREC-12 QA main task. In *Proceedings of the Text Retrieval Conference*.

YANG, Y. AND LIU, X. 1999. A re-examination of text categorization methods. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York.

ZHOU, Z. AND LIU, X. 2006. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Trans. Knowl. Data Engin. 18*, 1, 63–77.

ZHU, H. AND HOVY, E. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proceedings of the Association for Computational Linguistics*.

Received March 2009; revised September 2009; accepted December 2009