Probabilistic Approaches for Answer Selection in Multilingual Question Answering

Jeongwoo Ko
Aug 27 2007

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This work was supported in part by ARDA/DTO Advanced Question Answering for Intelligence (AQUAINT) program award number NBCHC040164 and used the NTCIR 5-6 corpus.

Keywords: Answer ranking, answer selection, probabilistic framework, graphical model, multilingual question answering

To my family for love and support.

Abstract

Question answering (QA) aims at finding exact answers to a user's natural language question from a large collection of documents. Most QA systems combine information retrieval with extraction techniques to identify a set of likely candidates and then utilize some selection strategy to generate the final answers. This selection process can be very challenging, as it often entails ranking the relevant answers to the top positions. To address this challenge, many QA systems have incorporated semantic resources for answer ranking in a single language. However, there has been little research on a generalized probabilistic framework that models the correctness and correlation of answer candidates for multiple languages.

In this thesis, we propose two probabilistic models for answer ranking: independent prediction and joint prediction. The independent prediction model directly estimates the probability of an individual answer candidate given the degree of answer relevance and the amount of supporting evidence provided in a set of answer candidates. The joint prediction model uses an undirected graph to estimate the joint probability of all answer candidates, from which the probability of an individual candidate is inferred. The models consider both the relevance of individual answers as well as their correlation in order to rank the answer candidates.

As a general probabilistic framework, the models support answer selection (1) in monolingual QA as well as (2) cross-lingual QA (3) on answer candidates returned by multiple extraction techniques (4) provided by different question answering systems.

An extensive set of experiments was done for monolingual QA (English, Chinese and Japanese) as well as cross-lingual QA (English-to-Chinese and English-to-Japanese) using TREC and NTCIR questions. The empirical results demonstrate the effectiveness of the independent prediction model and the joint prediction model for answer selection in multilingual QA and the joint prediction model is useful to generate a set of unique and comprehensive answers.

Acknowledgments

One of my favorite movies is “Forrest Gump”. The first scene of the movie starts with Forrest sitting on a bench and saying to someone passing a chocolate, *“My momma always said, ’Life is like a box of chocolates. You never know what you’re gonna get.”’* It is a good metaphor for life. In my life, whenever I have chosen a path, I have been fortunate in having someone who believed me and could guide me to find out a good path. While studying at Carnegie Mellon University in particular, I have met many people who have provided me with priceless guidance and support.

First and foremost, I would like to thank my advisers, Eric Nyberg and Teruko Mitamura for their invaluable guidance, advice and encouragement. Eric gave me the opportunity to participate in this amazing research and inspired me to actively investigate many interesting and challenging problems in language and information technologies. Teruko guided me to look at the broader world by helping me to extend my research beyond one specific language and to explore difficult problems using multiple languages. In addition, when I had a hard time balancing research and family, Eric and Teruko always helped me to go through it and choose the best path. Their advice, support and encouragement helped me to finish this dissertation.

I would also like to thank my other thesis committee members, Jaime Carbonell and Luo Si. I was fortunate to have them as committee members. Jaime always gave me inspiring advice and wonderful suggestions. Especially, his guidance helped me to define the scope of this research. Luo always encouraged me and guided me to model probabilistic frameworks. His inspiring advice and research guidance made this thesis more thorough.

This work would not have been possible without the support and interest of many individuals at CMU. I would like to thank Jamie Callan and Eric Xing for their valuable discussion and suggestions. I would also like to thank the JAVELIN team members for their valuable comments and contribution: Shilpa Arora, Justin Betteridge, Matthew Bilotti, Kevyn Collins-Thompson, Krzysztof Czuba, Laurie Hiyakumoto, Yi-Fen Huang, Frank Lin, Lucian Lita, Vasco Pedro, James Rankin, Eric Riebling, Nico Schlaefer, Andrew Schlaikjer, Hideki Shima, Dave Svoboda and Mengqiu Wang. I also would like to thank Abhaya Agarwal, Bano Banerjee, Kethy Baker, Ulas Bardak, Kevin Chang, Betty Cheng, Shay Cohen, Ariadna Font-Llitjos, Kevin Gimpel, Fan Guo, Brooke Hyatt, Peter Jansen, Chun Jin, Chan Woo Kim, Jae Dong Kim, John Kominek, Anagha Kulkarni, Yan Liu, Thuy Linh Nguyen, Jean Oh, Radha Rao, Antoine Raux, Yanjun Qi, Wilson Tam, Stacey Young, Shinjae Yoo, Yi-Chia Wang, Bing Zhao and many others for their encouragement, freindship and support.

Finally, my most sincere gratitude to my parents who always had faith in me and gave me endless love and support, my sister Jungmin Ko who gave me insightful suggestions whenever I made a decision, and has been my role model in life, and my brother Junghoon Ko. Most importantly, throughout my Ph.D., my husband Chando Park encouraged me at all times with his love, patience, and understanding. My last thanks to God who blessed me with my lovely son Jonathan Park while I was at CMU.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Question Answering (QA) aims at finding exact answers to natural language questions from a large collection of documents. Typical QA systems [45, 80, 28, 34, 46, 15] combine document retrieval with question analysis and extraction techniques to identify a set of likely candidates, from which the final answer(s) are selected.

Questions can be classified into two categories: factoid questions and complex questions. Factoid questions ask for simple facts as answers (e.g. a person's name, a country name, an organization name). Complex questions require longer answers representing facts, relations or processes (e.g. *"What is a quasar?", "What is the relationship between Alan Greenspan and Robert Rubin?", "How did Egypt acquire nuclear weapons?"*). To answer many factoid questions, QA systems use several external resources such as ontologies and apply shallow parsing on questions and documents. In addition, many systems exploit external corpora (e.g. the Web and Wikipedia) as an additional source of answers [20, 51, 104, 2] and incorporate reasoning to verify the relationship between an answer candidate and the question [61].

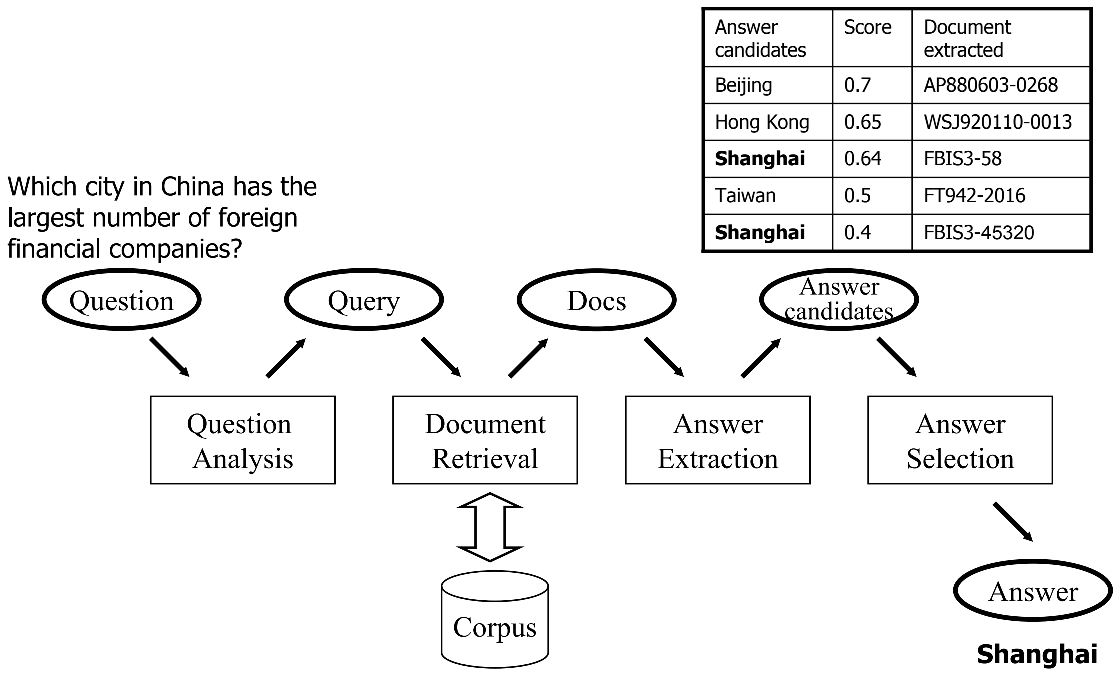
Statistical machine learning techniques are also widely used for QA subtasks such as answer type classification, document retrieval and answer extraction, etc. Recently, deeper semantic analysis has been incorporated into several QA systems to increase performance on factoid questions and answer complex questions [33, 13, 26, 74].

1.1 Motivation

Most QA systems [15, 34, 63, 80] 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 which analyzes a question and produces a list of keywords. Document retrieval is a step that searches for relevant documents or passages. Answer extraction extracts a list of answer candidates from the retrieved documents. Answer selection is a process which 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) amongst many incorrect ones.

Figure 1.1 shows a traditional QA architecture with an example question. 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. Due to imprecision in answer extraction, an incorrect answer ("Beijing") was ranked at the top position. The correct answer ("Shanghai") was extracted from two documents with different confidence scores and ranked at the third and the fifth positions. In order to rank "Shanghai" in the top position, we have to address two interesting challenges:

- *Answer Relevance.* How do we identify relevant answer(s) amongst irrele-



Answer candidates	Score	Document extracted
Beijing	0.7	AP880603-0268
Hong Kong	0.65	WSJ920110-0013
Shanghai	0.64	FBIS3-58
Taiwan	0.5	FT942-2016
Shanghai	0.4	FBIS3-45320

Figure 1.1: A traditional QA pipeline architecture

vant ones? This task may involve searching for evidence of a relationship between the 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)`).

- *Answer Similarity.* How do we exploit similarity among answer candidates? For example, when the candidate list contains redundant answers (e.g., "Shanghai" as above) or several answers which 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 question, several answer selection approaches have been developed that make use of external semantic resources. One of the most common approaches relies on precompiled lists or ontologies such as WordNet, CYC and gazetteers for answer validation or answer reranking. In this approach, answer candidates are either removed or discounted if they are not found within the portion of the resource's hierarchy corresponding to the expected answer type of the question (e.g. location, proper-name, or one of several predefined subtypes) [71, 103]. The Web also has been employed in answer reranking by exploiting search engine results produced by queries containing the answer candidate and question keywords [54]. This approach has also been found useful when using various languages for answer validation.

Even though each of these approaches uses one or more semantic resource to independently support an answer candidate, few have considered the potential benefits of combining resources together as evidence. As research that does combine resources, Schlobach et al. [86] combined geographical databases with WordNet in order to use more than one resource for answer type checking of location questions. However, in their experiments the combination actually hurt performance because of the increased semantic ambiguity that accompanies broader coverage of location names. This demonstrates that the method used to combine potential answers may matter as much as the choice of resources.

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 similarity in answer candidates and exploit similarity to boost answer confidence, especially for list questions that require a set of unique answers. However, this task raises the following issues [52]. When a partial overlap occurs between answer candidates, a decision must be made regarding whether or not to boost answer confidence (e.g. "April 1912", "14 Apr 1912", "14 April"). Granularity is another problem. For example, assume that there are two answer candidates "April" and "Spring" for a temporal question. Should we boost the confidence of one answer candidate ("Spring") by considering another ("April") as additional support?

One approach to address these challenges is to incorporate answer clustering [46, 72, 38]. For example, we might merge "April 1912" and "14 Apr 1912" into a cluster and then choose one answer as the cluster head. However, clustering raises new issues: how to choose the cluster label and how to calculate the scores of the clustered answers.

Exploiting redundancy is even more important in multi-strategy QA, in which multiple answering agents are used [14, 12, 21, 37, 2, 74]. The basic idea of this approach is that a combination of similar answers extracted from different sources using different strategies performs better than any individual answering strategy alone. As answer candidates come from different agents with different score distributions, exploiting answer redundancy plays an important role in answer ranking.

Although previous work has utilized evidence from similar answer candidates for a specific answer candidate, the algorithms 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.

1.2 Approach

In the previous section, 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 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 $P(correct(A_i)|Q, A_1, ..., A_n)$.

In this thesis, we propose two probabilistic models that conduct the two subtasks in a statistical framework: independent prediction and joint prediction. The independent prediction model directly estimates $P(correct(A_i)|Q, A_1, ..., A_n)$ from the probability that an answer candidate is correct given multiple answer similarity features and answer relevance features (Equation 1.1). This model is implemented with logistic regression, which is a statistical machine learning technique used to predict the probability of a binary variable from the input variables.

$$\begin{aligned} & P(correct(A_i)|Q, A_1, ..., A_n) \\ & \quad \approx P(correct(A_i)|rel_1(A_i), ..., rel_{K1}(A_i), sim_1(A_i), ..., sim_{K2}(A_i)) \end{aligned} \tag{1.1}$$

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 $P(correct(A_1)$,..., $correct(A_n)|Q$, $A_1, ..., A_n)$, where n is the number of answer candidates in consideration. From the joint probability, we can derive the marginal probability of $P(correct(A_i)|Q, A_1, ..., A_n)$ for each individual answer as well as the conditional probability $P(correct(A_i)|correct(A_j)$ $, Q, A_1, ..., A_n)$.

The joint prediction model uses an undirected graph to estimate the joint probability of all answer candidates. Each node S_i in the graph has a binary value to represent answer correctness: 1 represents a correct answer and 0 represents an incorrect answer. The weights on the edges represent the similarity between answer candidates. This model estimates the joint probability of all answer candidates, from which the probability of an answer candidate is inferred as shown in Equation 1.2.

$$P(correct(A_i)|Q, A_1, ..., A_n) \tag{1.2}$$
$$\approx \sum_{S_1} ... \sum_{S_{i-1}} \sum_{S_{i+1}} ... \sum_{S_n} P(S_i = 1, S_1, ..., S_{\ i-1}, S_{i+1}, ..., S_n)$$

1.3 Hypothesis

The hypothesis is that our probabilistic answer ranking models significantly improve performance of a QA system and provide a general framework that allows any relevance and similarity features to be easily incorporated.

Table 1.1: Hypothesis dimensions

Source language (for question)	Target language (for documents)	Extraction techniques	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

Specifically, the probabilistic answer ranking models provide a generalized probabilistic framework that supports answer selection (1) in monolingual QA as well as (2) cross-lingual QA (3) on answer candidates returned by multiple extraction techniques (4) provided by different question answering systems (Table 1.1), and perform better than state-of-the-art answer selection algorithms.

1.4 Contributions

In this research, we propose probabilistic answer ranking models for multilingual question answering. This work will contribute to research in question answering and answer selection by incorporating the following strategies:

- **Generalized probabilistic framework.** The models provide a general probabilistic framework which considers both the relevance of individual answers as well as their correlation; additionally, the models allow any answer relevance and similarity features to be easily incorporated in order to boost the rank of correct answers.

- **Support of multiple languages.** As the models are language independent, they can be easily extended to other languages with re-training for each individual language. New features and resources for other languages can be easily incorporated into the models.

- **Support of multi-strategy QA.** Architectures of recent QA systems typically employ multiple strategies and techniques to answer questions, thus requiring answers be merged to combine the results proposed by alternative approaches. The models described here can be easily extended to merge alternative approaches by using a score from each individual answering strategy as one feature and by learning their associated weights from the training data.

- **Combination of knowledge-driven and data-driven approaches.** Answer relevance scores and answer similarity scores can be calculated with knowledge-based approaches (e.g. using WordNet and gazetteers) as well as data-driven approaches (e.g exploiting the Web and Wikipedia). The models can easily learn how to combine them using different weights.

1.5 Outline

This thesis is organized as follows. Section 2 describes related work. Section 3 introduces the JAVELIN QA system and the EPHYRA QA system, both of which were used as testbeds to evaluate the answer ranking models. Section 4 describes the answer ranking models: the independent prediction model and the joint prediction model. Section 5 lists the features that generate similarity and relevance scores for factoid questions. In Section 6, we explain how the models were extended to support multiple languages. Section 7 describes the evaluation methodology used to measure the performance of the models. Section 8 summarizes the experimental setup and results. Finally, Section 9 concludes with potential future research.

Chapter 2

Related Work

This chapter describes previous research for answer selection in question answering.

2.1 Previous 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 approaches 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. For example, given the question, *"What continent is Egypt on?"*, the candidate list might include "Middle East", "Arab", "Islam", and "Africa". As the expected answer type is `LOCATION` (more specifically `CONTINENT`), if we know that "Middle East", "Arab" and "Islam" are not continent names, we can filter them out from the candidate list. This process is called type checking and used to filter out invalid answers. Ontologies such as WordNet and gazetteers are commonly-used resources to assess whether or not an answer candidate matches the expected answer type [10, 103, 86].

For numeric questions, filtering can be done using range checking [14]. Some resources such as CYC [48] and the CIA World Factbook[1] provide the latest geographic information about countries. For example, given the question *"What is the population of Iceland?"*, an answer candidate is "300". As the CIA World Factbook contains "299,388" as the population of Iceland, the answer candidate "300" is significantly different from the number in the resource and can be filtered out from the answer list.

Filtering is also important in QA systems that apply n-gram techniques to extract answer candidates [7, 2]. As the n-gram approach tends to produce many noisy answer candidates, the system requires filtering out invalid answer candidates using manually written or data-driven filters (e.g. a web hit counter filter to remove answer candidates without any hits from Google [2], and surface string filters to assess whether answer candidates are capitalized for proper name questions or whether numbers exist for numeric questions [7]).

Answer reranking is another popular approach for answer selection. This approach applies several different validation strategies in order to rerank answer candidates. Xu et al. [103] applied several type-specific constraints (e.g. constraint to check the sub type for location questions or constraint to check verb argument matching between an answer text and the question), and moved to the top position those answer candidates that best satisfied the constraints.

Moldovan et al. [61] converted question and answer candidates into logic representation, and used a logic prover to prove answer correctness using axioms obtained

[1]https://www.cia.gov/redirects/factbookredirect.html

from WordNet. This answer correctness assessment was then used to rerank answer candidates.

Prager et al. [81] extended CYC-based filtering to enable answer reranking. When CYC contained information about a numeric question, answer candidates for the question were categorized into five different groups: “answer is known to be correct”, “answer is known to be incorrect”, “answer is in range”, “answer is out of range”, and “unknown”. Then, this information was used for answer reranking by removing answer candidates that were classified into “answer is known to be incorrect” and increasing the confidence scores of the answer candidates categorized into correct or in-range answers.

Magnini et al. [54] proposed two answer reranking approaches: a statistical approach and a content-based approach. The statistical approach creates a query from question keywords and an answer candidate using the proximity operator (`NEAR`), and then sends it to AltaVista (http://www.altavista.com). Then, it counts the number of hits from the question keywords, the number of hits from the answer candidate and the number of hits from the combination of the question keywords and the answer candidate. These hits are used to compute answer relevance scores. Finally, answer candidates are reranked according to their relevance scores. On the other hand, the content-based approach analyzes the co-occurrence of question keywords and an answer candidate in text snippets returned from Google (http://www.google.com). It counts the word distance between an answer candidate and a question keyword in each text snippet, then calculates an answer relevance score using the distance. The answer whose score is highest is chosen as a final answer for factoid questions.

Ravichandran et al. [18] used maximum entropy to rerank answer candidates. They used four simple features such as a feature to count the frequency of an answer in the retrieved documents, a feature to check whether or not an answer matches the expected answer type, and a feature to calculate inverse term frequency of question keywords in the answer sentence. The scores from maximum entropy were used to rerank answer candidates. The LCC's Chaucer QA system [30] also used maximum entropy to merge answers returned from six answer extractors[2]. Chaucer answer selection consists of two steps. The first step was to rerank answer candidates using maximum entropy similar to Ravichandran et al. The second step used text entailment to select the final answer. It first took the top 25 answer candidates in the list provided by the first step, calculated a text entailment score between the question and each individual answer, and reranked answer candidates according to the entailment scores. In addition, Chaucer calculated a text entailment score between the original question and the predictive questions generated for the original question. If the original question entailed a predictive question, the answer of the predictive question was considered as the answer of the original question.

Buscaldi et al. [9] used Wikipedia for answer reranking by exploiting Wikipedia structured information such as category, definition and title. For example, for the question *"Which fruit contains vitamin C?"*, a list of fruits was obtained from the Wikipedia article[3], and then the list was compared with answer candidates.

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 reranking. Recently, Schlobach et al. [86] combined geographical databases with WordNet in a type checker for location questions. However, in their experiments the combination actually hurt perfor-

[2]Chaucer was the second best QA system for factoid questions in the TREC-2006 evaluation.
[3]http://en.wikipedia.org/wiki/Category:Fruit

mance, a result they attribute to the increased semantic ambiguity that accompanied broader coverage of location names.

Collecting evidence from similar answer candidates to boost the confidence of a specific answer candidate is also important for answer selection. 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 [15], summing the scores of all answers in the cluster [72, 46] or selecting the best score among the individual answer scores in the cluster [50]. Recently, Jijkoun et al. [38] used type checking scores when merging similar answers in Dutch monolingual QA. They multiplied each answer score with a probability calculated by their type checker. They also used a graph to consider non-transitiveness 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 [27, 43]. An exhaustive search to find all possible candidates was applied to find answers for list questions [106].

Recently, several QA systems have employed a multi-strategy architecture that allows multiple answering agents to answer a question [14, 12, 21, 37, 2, 74]. This architecture requires answer merging to combine the similar answers proposed by alternative approaches. This merging process is challenging, as it requires combining ranked answer lists with independent score distributions. To exploit this type of redundancy, simple confidence-based voting has been used to merge the top five answers returned by competing QA agents [14]. As a more advanced approach, a maximum-entropy model has been used to rerank the top 50 answer candidates from three different answering strategies [21]. This maximum-entropy reranking model contained more than 30 features generated from the internal QA components (e.g. the rank of answer candidates, the answer type, source document scores, and the count of each answer in the document corpus) and improved the performance of answer selection by combining complementary results provided by different answering strategies.

2.2 QA Evaluation

QA has been actively researched and evaluated 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 languages and European languages, respectively. We used TREC and NTCIR data to evaluate our answer ranking models for English, Chinese and Japanese QA. This section provides brief introduction to TREC and NTCIR.

2.2.1 TREC

TREC (http://trec.nist.gov) provides the infrastructure to evaluate various retrieval tasks such as document retrieval, filtering, novelty detection, question answering, etc. In 1992, NIST (National Institute of Standards and Technology) organized TREC under the support from Information Technology Laboratory's (ITL) Retrieval Group of the Information Access Division (IAD) and the Advanced Research and Development Activity (ARDA) of the U.S. Department of Defense.

```
<target id = "4" text = "James Dean">
  <qa>
     <q id = "4.1" type="FACTOID"> When was James Dean born?</q>
  </qa>
  <qa>
      <q id = "4.2" type="FACTOID">When did James Dean die? </q>
  </qa>
  <qa>
     <q id = "4.3" type="FACTOID">How did he die?</q>
  </qa>
  <qa>
      <q id = "4.4" type="LIST"> What movies did he appear in?</q>
  </qa>
  <qa>
      <q id = "4.5" type="FACTOID">Which was the first movie that he was in?</q>
  </qa>
  <qa>
      <q id = "4.6" type="OTHER"> Other</q>
  </qa>
</target>
```

Figure 2.1: Example question from TREC-2004

The TREC QA track [95, 96, 97, 98, 99, 100] has been offered every year since 1999 to evaluate the performance of English QA systems. The first two QA tracks focused on pinpointing small text snippets (either 50 or 250 bytes) from a newspaper document corpus to answer 200 factoid questions. MRR (Mean Reciprocal Rank) was used to evaluate system performance by calculating the average reciprocal rank of the top five answers.

The third QA track in TREC-2001 introduced “NIL” answer questions. When a system cannot find answers from the given document collection, the system should return ’NIL’. In addition, the answer should be a short text snippet whose length is less than 50 bytes. This track also incorporated list questions in order to evaluate the ability to extract a list of answers from multiple documents.

The TREC-2002 QA track added more constraints on answer: the participants should return exact answer strings (instead of text snippets) as well as provide information on the document where the answer was extracted. Even though an answer was correct for a question, the score of the answer would still be 0 if the answer was not extracted from the supporting documents.

In TREC-2003, the QA track introduced 30 definition questions that asked for information about person, organization and entity (e.g. feng shui, TB).

The latest TREC QA evaluations in TREC-2004, TREC-2005 and TREC-2006 have focused on answering series of questions for one topic. Each topic has several sub-questions, and the last sub-question requires the novel information which had not previously been mentioned in the earlier questions and answers for the topic (Figure 2.1) . In this task, natural language processing including coreference resolution is important to address anaphora resolution.

To evaluate performance on complex questions, TREC-2005 introduced the relationship track which included questions about eight different relations such financial, movement of goods, family ties, communication pathways, organizational ties, co-location, common interests, and temporal (e.g. "The analyst is concerned with arms tracking to Colombian insurgents. Specifically, the analyst would like to know of the different routes used for arms entering Colombia and the entities involved.")

2.2.2 NTCIR

More recently, QA systems have been extended to cover various Asian languages. NTCIR (http://research.nii.ac.jp/ntcir) focuses on evaluating information retrieval, question answering, text summarization and information extraction for Chinese, Japanese, Korean and English [41, 42].

In 2001, the NTCIR QA task started to search for answers from Japanese newspaper articles and has been extended to cover more complex questions (e.g. why and definition questions). Recent two NTCIR evaluations (NTCIR-5 and NTCIR-6) have incorporated cross-lingual question answering tasks for English-to-Chinese (E-C), English-to-Japanese (E-J), Chinese-to-English (C-E) and Japanese-to-English (J-E). Because NTCIR also provides translation of English questions into Chinese and Japanese, the translated questions can be used to measure the performance of Chinese-to-Chinese (C-C) and Japanese-to-Japanese (J-J). This allows a comparison to be made between monolingual QA (C-C and J-J) and cross-lingual QA (E-C and E-J) for the same question. For example, in the NTCIR-5 evaluation, the performance of C-C was three times better than that of E-C, and the performance of J-J was 5-10% better than E-J [83]. As they were evaluated using the same questions and the same document collection, the difference in the performance can be considered as the effect of question translation. When comparing Chinese and Japanese, less performance degradation occurred in Japanese because Chinese questions were generated by translating E-J questions and the translation occasionally caused vocabulary mismatch between the translated questions and Chinese documents.

Currently the cross-lingual QA tasks include only factoid questions and participants submit an exact answer string and document pairs for each question. Performance has been measured with the average top answer accuracy.

Chapter 3

System Overview

The primary testbed for our research is the JAVELIN QA system. This chapter describes the JAVELIN architecture, and its application for supporting multiple languages. The EPHYRA QA system is used as the secondary testbed to evaluate the extensibility of the answer ranking models.

3.1 The JAVELIN System

JAVELIN is an open-domain QA system to support a flexible and modular QA architecture [71, 72, 73]. To embody the QA pipeline architecture shown in Figure 1.1, JAVELIN defines abstract interfaces for each of the following modules:

- Question Analyzer: takes a question as an input and analyzes the question text to generate a *request object* containing a list of keywords, alternative terms for query expansion, the question type, and the expected answer type.
- Retrieval Strategist: takes the *request object* as an input and produces a list of

relevant documents retrieved from the corpus.

- Information Extractor: takes the *request object* and the retrieved documents as an input and produces a list of answer candidates extracted from the documents.

- Answer Generator: takes the *request object* and the extracted answer candidates as an input and generates a list of ranked answers as an output. Our answer ranking models are tested using this module.

The Execution Manager coordinates the processing of QA modules to imitate a QA pipeline architecture. It also stores process history to a centralized repository for information reuse [78].

In practice, the JAVELIN architectural design has been implemented in three different languages, each of which exploits the modularity of the architecture.

3.1.1 System for English QA

During ARDA's AQUAINT Phase I & II programs, JAVELIN was applied to English factoid questions and was evaluated in three TREC QA evaluations [71, 72, 73]. To handle English factoid questions, the four QA modules were implemented to incorporate multiple English resources and tools.

The Question Analyzer used the RASP parser [8], WordNet and a set of hand-coded rules for answer type classification. For factoid questions, each question was classified into one of eight answer types (`LOCATION`, `PROPER-NAME`, `PERSON-NAME`, `ORGANIZATION-NAME`, `TEMPORAL`, `NUMERIC-EXPRESSION`, `OBJECT`, `or LEXICON`). To extract keywords from a question, named entity detectors and syntactic information were utilized.

The Retrieval Strategist used the Lemur search engine[1] to execute queries and retrieve documents. For the TREC evaluation, the Retrieval Strategist supported answer justification to find supporting documents for answer candidates that were extracted from external corpora (e.g. the Web, Wikipedia). This process is called "answer projection" [57] and uses an answer candidate as a part of a query to retrieve additional documents that contain the answer candidate.

The Information Extractor was implemented with five different answer extraction techniques.

- EXPERT: an extractor that draws answers from a set of semantic resources including gazetteers and WordNet
- FST: an answer extractor based on finite state transducers that incorporate a set of extraction patterns (both manually created and generalized patterns)
- LIGHT: an extractor that selects answer candidates using a non-linear distance heuristic between the keywords and an answer candidate
- LIGHTv2: another extractor based on a different distance heuristic
- SVM: an extractor that uses Support Vector Machines [93, 94] to discriminate between correct and incorrect answers based on local semantic and syntactic context

As the extractors vary in accuracy, the types of questions they can answer, and the average number of answers returned for each question, the Answer Generator

[1]http://www.lemurproject.org/

should be robust and generalizable to handle different outputs returned from different extractors.

3.1.2 System for Multilingual QA

JAVELIN has been extended to support multiple languages: Japanese and Chinese monolingual QA as well as cross-lingual QA which searches for Chinese or Japanese answers to English questions. As JAVELIN supports a language-independent modular architecture, the system was customized to use multilingual resources and Unicode characters. Recently, JAVELIN was evaluated in the NTCIR-5 and NTCIR-6 [50, 59, 87, 58].

For cross-lingual QA, JAVELIN incorporates a new component called the Translation Module (TM). To find answers for cross-lingual QA, JAVELIN first extracts keyterms from the English question and then passes the extracted keyterms along with their associated properties (such as its named entity type or part-of-speech) on to the TM. For each English term, the TM returns a list of Chinese or Japanese translation candidates, in ranked order of their translation score.

The TM uses many different sources including Machine Readable Dictionaries (MRDs), Machine Translation systems (MTs) and web-mining-based translators (WBMTs) [67, 49]. Different resources and different types of resources have advantages and disadvantages. For example, MRDs are usually better for translating common nouns and verbs but have poor coverage of named entities, and web-mining-based translators are good for translating popular named entites but do a poor job of translating common nouns and verbs. Taking advantage of this, the TM uses different combinations of these resources based on the keyterm properties.

To rank the keyterm translation candidates, the TM assigns each translation candidate a score using co-occurrence statistics of the source keyterm (in this case English) and the target candidate translation (in this case Chinese or Japanese) on web pages. The co-occurrence information is obtained by counting the number of search results from a search engine, and the correlation statistics are calculated using chi-square. Then the set of translation candidates is ranked order from highest to the lowest score.

The translated keyterm candidates, along with their ranking, are used to retrieve Chinese or Japanese documents using Indri[2], a language model and inference network based search engine. The ranking assigned to each translation candidate is used to boost its confidence score when formulating the query.

Then the Chinese and Japanese answer extraction components extract answer candidates from the retrieved documents. Both the Chinese and Japanese answer extractors use maximum-entropy [25] to extract answer candidates based on multiple features such as answer type matching, dependency structure matching, and similarity score of predicate argument structures, etc. Finally answer candidates are reranked by our answer ranking models.

3.2 The EPHYRA System

EPHYRA is an open-domain question answering system which is based on a flexible pipeline architecture for combining multiple techniques [85]. It consists of three major components for query generation, search and answer selection.

[2]http://www.lemurproject.org/indri/

For factoid questions, EPHYRA first analyzes the question to generate a query and retrieves text snippets from Yahoo. The retrieved snippets are used then to extract answer candidates. EPHYRA supports two extraction techniques: one uses answer type to extract associated named entities and the other uses patterns which are automatically obtained from question-answer pairs in training data. As the recent TREC evaluations require that answers are extracted from documents in the given corpus, answer projection is required to find the documents in the given corpus. Finally, EPHYRA applies simple filters to select the final answer to the question (e.g., a Stopword Filter is used to remove malformed candidates, and a Duplicate Filter is used to merge similar answers by summing their scores).

Table 3.1: Performance of EPHYRA in TREC-2006

Score	Run1	Run2	Run3	Median of 59 runs
Factoid	0.196	0.196	0.196	0.186
List	0.092	0.096	0.097	0.087
Other	0.143	0.150	0.145	0.125
Average	0.139	0.143	0.141	0.134

EPHYRA was evaluated in TREC-2006 for the first time, and performed better than the median (Table 3.1). In this thesis, we apply our answer ranking models to answer candidates extracted by the two EPHYRA extractors. Application of the models to EPHYRA will show the extensibility of the answer ranking models to another QA system.

Chapter 4

Answer Ranking Models

In this chapter, we describe two probabilistic models for answer ranking: an independent prediction model and a joint prediction model.

4.1 Independent Prediction Model (IP)

The independent prediction model directly estimates the probability of an individual answer candidate using multiple answer relevance and similarity features. The model is implemented with logistic regression, which is a statistical machine learning technique used to estimate the probability of an output variable (Y) from input variables (X) [60]. Logistic regression is a discriminative method that directly models P(Y|X) by learning parameters from training data (Equation 4.1 and Equation 4.2).

$$P(Y=1|X) = \frac{exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + exp(w_0 + \sum_{i=1}^{n} w_i X_i)} \tag{4.1}$$

$$P(Y=0|X) = \frac{1}{1+exp(w_0+\sum_{i=1}^{n} w_i X_i)} \tag{4.2}$$

where X is a set of input variables (X=<X$_1$,...,X$_n$>) and X$_i$ is a single input variable.

Logistic regression has been successfully employed in many applications including multilingual document merging [84, 90]. We used logistic regression to predict the probability that an answer candidate is correct given the degree of answer correctness and the amount of supporting evidence provided in a set of answer candidates (Equation 4.3).

$$\begin{aligned} & P(correct(A_i)|Q, A_1, ..., A_n) \qquad (4.3)\\ & \approx P(correct(A_i)|rel_1(A_i), ..., rel_{K1}(A_i), sim_1(A_i), ..., sim_{K2}(A_i)) \\ & = \frac{exp(\alpha_0 + \sum\limits_{k=1}^{K1} \beta_k rel_k(A_i) + \sum\limits_{k=1}^{K2} \lambda_k sim_k(A_i))}{1+exp(\alpha_0 + \sum\limits_{k=1}^{K1} \beta_k rel_k(A_i) + \sum\limits_{k=1}^{K2} \lambda_k sim_k(A_i))} \\ & where, sim_k(A_i) = \sum_{j=1(j\neq i)}^{N} sim'_k(A_i, A_j). \end{aligned}$$

In Equation 4.3, K1 and K2 are the number of feature functions for answer relevance and answer similarity scores, respectively. N is the number of answer candidates. Each rel$_k$(A$_i$) is a feature function used to produce an answer relevance score for an individual answer candidate A$_i$. Each sim$'_k(A_i, A_j)$ is a scoring function used to calculate an answer similarity between A$_i$ and A$_j$.

Each sim$_k(A_i)$ represents one similarity feature for an answer candidate A$_i$ and is obtained by summing N-1 answer similarity scores to represent the similarity of one answer candidate to all other candidates. As some string similarity metrics (e.g. Levenshtein distance) produce a number between 0 and 1 (where a 1 means that two strings are identical, and a 0 means that they are different), similarity scores less than some threshold value can be ignored. In Chapter 5, we describe the answer relevance and similarity features in detail.

$$\vec{\alpha}, \vec{\beta}, \vec{\lambda} = \underset{\vec{\alpha}, \vec{\beta}, \vec{\lambda}}{\operatorname{argmax}} \sum_{j=1}^{R} \sum_{i=1}^{N_j} logP(correct(A_i)|rel_1(A_i), ..., rel_{K1}(A_i), sim_1(A_i), ..., sim_{K2}(A_i)) \tag{4.4}$$

The parameters $\vec{\alpha}, \vec{\beta}, \vec{\lambda}$ were estimated from training data by maximizing the log likelihood as shown in Equation 4.4, where R is the number of training questions and N_j is the number of answer candidates for each question Q_j. For parameter estimation, we used the Quasi-Newton algorithm [56].

After applying the independent prediction model, answer candidates are reranked according to their estimated probability. For factoid questions, the top answer is selected as the final answer to the question. As logistic regression can be used for a binary classification task with a default threshold of 0.5, we may use the model to identify incorrect answers: if the probability of an answer candidate is lower than 0.5, it may be considered wrong and is filtered out of the answer list. This is useful in deciding whether or not a valid answer exists in the corpus [97]. The estimated probability can also be used in conjunction with a cutoff threshold when selecting multiple answers to list questions.

4.2 Joint Prediction Model (JP)

The joint prediction model estimates the joint probability of all answer candidates, from which the probability of an individual candidate is inferred. This estimation is performed using a graphical model.

Graphical models have been used to represent and solve problems in many different domains such as artificial intelligence, computational biology, image processing, computer vision, information retrieval and natural language processing. A graphical model is either directed or undirected. Directed graphs can be used to model causal relationships between variables [76]. Undirected graphs can be used to model correlations between variables [16]. We used an undirected graphical model for joint prediction. In this section, we first introduce graphical models and explain how we use an undirected graph model for answer selection.

Graphical Models

Jordan [39] describes graphical models as follows:

> "Graphical models are a marriage between probability theory and graph theory. They provide a natural tool for dealing with two problems that occur throughout applied mathematics and engineering - uncertainty and complexity - and in particular they are playing an increasingly important role in the design and analysis of machine learning algorithms. Fundamental to the idea of a graphical model is the notion of modularity - a complex system is built by combining simpler parts. Probability theory provides the glue whereby the parts are combined, ensuring that the

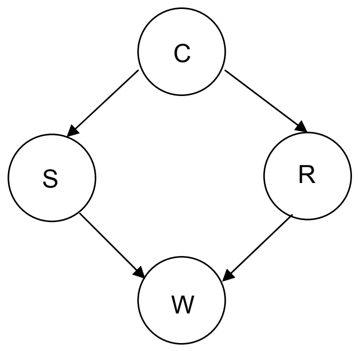


Figure 4.1: Example of directed graph (C:cloudy, S:sprinkler, R:rain, W:wet grass). The graph is extracted from (Russell and Norvig, 95) [82]

> system as a whole is consistent, and providing ways to interface models to data. The graph theoretic side of graphical models provides both an intuitively appealing interface by which humans can model highly-interacting sets of variables as well as a data structure that lends itself naturally to the design of efficient general-purpose algorithms."

Graphical models can be directed or undirected. Figure 4.1 shows an example of a directed graph extracted from (Russell and Norvig, 95) [82]. The graph has four nodes and four directed edges. Nodes C and R represent whether the weather is cloudy or rainy, respectively. Node S represents whether or not the sprinkler is on, and node W represents whether or not the grass is wet. The graph has two edges on the node W (one from S to W and another from R to W), but does not have any direct edge between C and W. This representation indicates that the grass can be wet only when it rains or the sprinkler is on, but it does not depend on the cloudy event. This illustrates conditional independence in directed graphical models: given its parents, a node does not depend on its non-descendants. Therefore, the joint

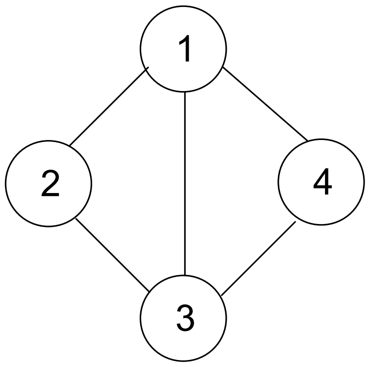


Figure 4.2: Example of undirected graph

distribution can be simplified by using conditional independence:

$$P(C, S, R, W) = P(C)P(S|C)P(R|C)P(W|S, R) \tag{4.5}$$

Figure 4.2 shows an example of an undirected graph. It has four random variables x_1, x_2, x_3, x_4 and each node in the graph represents a variable. The graph has two maximal cliques [1]: a clique with nodes 1,2,3 and a clique with nodes 1,3,4. The joint probability can be represented using a product over two cliques.

$$P(x_1, x_2, x_3, x_4) = \frac{1}{Z}\psi_{123}(x_{123}) \times \psi_{134}(x_{134}) \tag{4.6}$$

where Z=$\sum_{x_1,x_2,x_3,x_4} \psi_{123}(x_{123}) \times \psi_{134}(x_{134})$ and ψ_C is a potential function associated with a clique C.

A Boltzmann machine [32, 40] 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

[1] A clique is a complete subgraph whose nodes are fully connected.

probability of this graph is represented in Equation 4.7:

$$P(S) = \frac{1}{Z} exp(\sum_{i<j} \theta_{ij} S_i S_j + \sum_{i} \theta_{i0} S_i) \tag{4.7}$$

where Z is a normalization constant and θ_{ij}=0 if nodes S$_i$ and S$_j$ are not neighbors in the graph.

Joint Prediction Model

We adapted a Boltzmann machine for the answer selection 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.

$$S_i = \begin{cases} 1, & if\ A_i\ is\ correct \\ 0, & otherwise \end{cases} \tag{4.8}$$

The joint probability of the model can be calculated with Equation 4.9. 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, or less than some threshold, two nodes S$_i$ and S$_{N(i)}$ are not neighbors in the graph.

$$P(S_1, ..., S_n) = \frac{1}{Z} exp\left(\sum_{i=1}^{n}\left[(\sum_{k=1}^{K1} \beta_k rel_k(A_i))S_i + \sum_{N(i)}(\sum_{k=1}^{K2} \lambda_k sim_k(A_i, A_{N(i)}))S_i S_{N(i)})\right]\right) \tag{4.9}$$

The parameters $\vec{\beta}, \vec{\lambda}$ are estimated from training data by maximizing the joint probability (Equation 4.10). As the normalization constant Z is calculated by summing all configurations, $logZ$ does not decompose. In Chapter 8, we explain our implementation to address this issue by limiting the number of answers or applying approximate inference with the contrastive divergence learning method [31, 11].

$$\vec{\beta}, \vec{\lambda} = \underset{\vec{\beta},\vec{\lambda}}{\operatorname{argmax}} \sum_{j=1}^{R} log \frac{1}{Z} exp \left(\sum_{i=1}^{n} \left[(\sum_{k=1}^{K1} \beta_k rel_k(A_i)) S_i + \sum_{N(i)} (\sum_{k=1}^{K2} \lambda_k sim_k(A_i, A_{N(i)})) S_i S_{N(i)}) \right] \right) \tag{4.10}$$

As each node has a binary value (either 0 or 1), 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. Fig 4.3 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. In our algorithm, we have different weights (λ_1 and λ_2) for the marginal and conditional probability, respectively. The weights can be learned from the training data. In our initial experiments described in Chapter 8, we set λ_1 and λ_2 with the value of 1 to make the marginal and conditional probabilities equally important. Future work includes learning the weights from the training data

```
1. Create an empty answer pool.
2. Estimate the joint probability of all answer candidates: $P(S_1, ..., S_n)$
3. Calculate the marginal probability that an individual answer candidate is
   correct.
   $$P(correct(A_i)|Q, A_1, ..., A_n) \approx \sum_{S_1} ... \sum_{S_{i-1}} \sum_{S_{i+1}} ... \sum_{S_n} P(S_i = 1, S_1, ..., S_{i-1}, S_{i+1}, ..., S_n)$$
4. Choose the answer candidate whose marginal probability is highest, and
   move it to the answer pool.
5. For the remaining answer candidates, repeat the following steps.
   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 $P(correct(A_j)|correct(A_i), Q, A_1, ..., A_n)$.
   5.2. Calculate the score of each answer candidate ($A_j$) from the marginal
        and conditional probability.
        $$Score(A_j) = \lambda_1 P(correct(A_j)|Q, A_1, ..., A_n) - \max_i \lambda_2 P(correct(A_j)|correct(A_i), Q, A_1, ..., A_n)$$
   5.3. Choose the answer whose $Score(A_j)$ is maximum, and move it to the
        answer pool.
```

Figure 4.3: Algorithm to rank answers with the joint prediction model.

and evaluating their effect on answer selection performance.

Keeping consistent with the independent prediction model, answer candidates whose marginal probability is lower than 0.5 are removed from the answer list. If only one answer should be provided for any factoid question, the answer whose marginal probability is highest is selected as the final answer to the question.

We chose an undirected graphical model instead of a directed graphical model because it is difficult to learn the dependencies between answer candidates. However, we can measure answer dependency in an undirected graphical model with similarity scores. For example, assume that there are two answer candidates: "April 1912" and "14 Apr. 1912". If we have simple rules to convert temporal expressions into the ISO date format (YYYY:MM:DD), "April 1912" is converted into "1912:04:xx" and "14 Apr. 1912" is converted into "1912:04:14". Then, we can derive entailment information ("April 1912" entails "14 Apr. 1912"). This can be used as another similarity feature in the undirected graphical model [2].

4.3 Comparison of IP and JP

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. But the independent prediction model directly estimates the probability of an individual answer and the joint prediction model estimates the joint probability of all answers, from which the probability of correctness of an individual candidate is inferred. This section compares the two models and lists their advantages and disadvantages.

[2]Implementation of the entailment feature is left to future work.

Answer Ranking

The independent prediction model estimates the probability of correctness of each answer candidate. It considers two factors. The first factor is to identify relevant answers by estimating the probability $P(correct(A_i)|A_i, Q)$, where Q is a question and A_i is an answer candidate. The second factor is to exploit answer similarity by estimating the probability $P(correct(A_i)\ |A_i, A_j)$, where A_j is similar to A_i. By combining these two factors together, the independent prediction model estimates the probability of an answer as: $P(correct(A_i)|Q, A_1, ..., A_n)$, where n is the number of answer candidates in consideration.

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 $P(correct(A_1),..., correct(A_n)|\ Q, A_1, ..., A_n)$. The marginal probability of $P(correct(A_i)|Q, A_1, ..., A_n)$ for each individual answer as well as the conditional probability $P(correct(A_i)|correct(A_j), Q, A_1, ..., A_n)$ can be naturally derived from the joint probability to identify a set of distinct and comprehensive answers.

Advantages/disadvantages of Independent Prediction

The independent prediction model is simpler and more efficient than the joint prediction model. However, this model does not provide a formal framework to identify a list of distinct answers. In addition, it might have biased influence of incorrect similar answers. The next section describes how the joint prediction model address these issues.

Advantages/disadvantages of Joint Prediction

One advantage of the joint prediction model is that it provides formal framework to identify a distinct set of answers which is useful for list questions. For example, the question *"Who have been the U.S. 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 prediction model. For example, assume that we have three answer candidates for this question: "William J. Clinton", "Bill Clinton" and "George W. Bush". The probability of correctness of each answer has been calculated by marginalizing the joint probability of all answer candidates. Figure 4.4 shows the marginal probability of individual answers.

P(correct(William J. Clinton))= 0.758

P(correct(Bill Clinton)) = 0.755

P(correct(George W. Bush) = **0.617**

Figure 4.4: 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" are supporting 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".

P(correct(Bill Clinton)|correct(William J. Clinton)) = 0.803
P(correct(George W. Bush)|correct(William J. Clinton)) = **0.617**

Figure 4.5: Conditional probability given that "William J. Clinton" is correct

Fig 4.5 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) | correct(William J. Clinton)) is the same as P(correct(George W. Bush)) because the fact that "William J. Clinton" is correct does give any information on "George W. Bush". Next, we calculate a score for the remaining answers as shown in Fig 4.6. For simplicity, we set λ_1 and λ_2 with 1.

Score(Bill Clinton) = 0.755 - 0.803 = -0.048
Score(George W. Bush) = 0.617 - 0.617 = 0

Figure 4.6: Score calculation using marginal and conditional probability.

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 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. If we ignore similarity scores less than some threshold value similarly to the independent prediction model case, the graph is partially connected and we can use conditional independence to make calculation simpler. Even so, undirected graphical models need approximate approaches (e.g. Markov chain Monte Carlo sampling or variational inference) to estimate marginal and conditional probabilities when N is big (i.e. there are many answer candidates).

Chapter 5

Feature Representation

In the previous chapter, we introduced the notion of feature functions used to produce answer relevance scores and answer similarity scores. This chapter presents details of the feature functions and explains how answer relevance scores and answer similarity scores are generated for the answer ranking models. We will use a short word "features" instead of "feature functions".

5.1 Answer Relevance Features

Each answer relevance feature produces a relevance score to predict whether or not an answer candidate is correct given the question. This task can be done by exploiting internal and external QA resources. One important internal resource is the answer confidence score produced by an answer extractor. When extracting answer candidates from the retrieved documents, each answer extractor estimates a confidence score for an individual answer candidate based on the score of the retrieved document and its answering strategy. The rank of an individual answer candidate provided by an answer extractor can be used as another internal resource. As external resources, several semantic resources, such as the Web, databases, and ontologies have been used. For factoid questions, we used gazetteers and WordNet in a knowledge-based approach; we also used Wikipedia and Google in a data-driven approach. These two approaches are described below.

5.1.1 Knowledge-based Features

This section describes how knowledge-bases such as gazetteers and WordNet can be used to generate answer relevance scores.

a) Gazetteers

Electronic gazetteers provide geographic information, such as a country's population, languages, cities, continent and capital. As previously shown by Lita et al. [53], gazetteers such as the CIA World Factbook can answer specific types of TREC questions with high precision.

For answer ranking, we used three gazetteer resources: the Tipster Gazetteer, information about the US states provided by 50states.com [1] and the CIA World Factbook[2]. These resources were used to assign an answer relevance score between -1 and 1 to each candidate, following the algorithm in Figure 5.1. Effectively, a score of 0 means the gazetteers did not contribute to the answer ranking process for that candidate.

[1]http://www.50states.com
[2]https://www.cia.gov/library/publications/the-world-factbook/index.html

1) If the answer candidate directly matches the gazetteer answer for the question, its gazetteer score is 1.0 (e.g., given the question "What continent is Togo on?", the candidate "Africa" receives a score of 1.0).
2) If the answer candidate occurs in the gazetteer within the subcategory of the expected answer type, its score is 0.5 (e.g., given the question "Which city in China has the largest number of foreign financial companies?", the candidates "Shanghai" and "Boston" receive a score of 0.5 because they are both cities).
3) If the answer candidate is not the correct semantic type, its score is -1.0. (e.g., given the question "Which city in China has the largest number of foreign financial companies?", the candidate "Taiwan" receives a score of -1.0 because it is not a city).
4) Otherwise, the score is 0.0.

Figure 5.1: Algorithm to generate an answer relevance score from gazetteers.

For some numeric questions, range checking was added to validate numeric questions in a manner similar to the approach reported in Prager et al. [81]. For example, given the question *"How many people live in Chile?"*, if an answer candidate is within $\pm$ 10% of the population stated in the CIA World Factbook, it receives a score of 1.0. If it is in the range of 20%, its score is 0.5. If it significantly differs by more than 20%, it receives a score of -1.0. The threshold may vary based on when the document was written and when the census was taken[3].

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

1) If the answer candidate directly matches WordNet, its WordNet score is 1.0 (e.g., given the question "What is the capital of Uruguay?", the candidate "Montevideo" receives a score of 1.0).
2) If the answer candidate's hypernyms include a subcategory of the expected answer type, its score is 0.5 (e.g., given the question "Who wrote the book *Song of Solomon*?", the candidate "Mark Twain" receives a score of 0.5 because its hypernyms include *writer*).
3) If the answer candidate is not the correct semantic type, this candidate receives a score of -1.0 (e.g., given the question "What state is Niagara Falls located in?", the candidate "Toronto" gets a score of -1.0 because it is not a state).
4) Otherwise, the score is 0.0.

Figure 5.2: Algorithm to generate an answer relevance score from WordNet ontology.

b) Ontologies

Ontologies such as WordNet contain information about relationships between words and general meaning types (synsets, semantic categories, etc.). The WordNet lexical database includes English words organized in synonym sets, called *synsets* [23]. WordNet has been extensively used for different QA tasks, including construction of an answer type taxonomy [75] and as a source of axioms for reasoning about answer correctness [61].

We used WordNet in a manner analogous to gazetteers to produce an answer relevance score between -1 and 1. This score was computed for each candidate using the algorithm in Figure 5.2. As with the gazetteer score, a score of 0 means that WordNet did not contribute to the answer ranking process for a candidate.

5.1.2 Data-driven Features

Wikipedia and Google were used in a data-driven approach to generate answer relevance scores. Each resource is described below.

a) Wikipedia

Wikipedia (http://www.wikipedia.org) is a multilingual on-line encyclopedia. As it provides approximately 7.9 million articles in 253 languages (as of Aug 2007), Wikipedia has been used in many QA systems to answer definition questions [53, 3, 65]. For instance, to answer the question *"What is Friends of the Earth?"*, the Wikipedia article whose title is "Friends of the Earth" is retrieved and mined to extract answers. Wikipedia has also been used for answer validation by exploiting title, definition and category fields [9]. One issue with this approach is that it used only structured information, and could not address the vocabulary mismatch between answer candidates and the structured data. For example, if an answer string did not appear in the title or the definition fields, it was not considered to be a correct answer.

To address this issue, we used the Wikipedia documents in a data-driven approach using term frequency (TF) and inverse document frequency (IDF). TF and IDF have been widely used in information retrieval to compare the similarity between a query and a document as well as to measure the importance of a term in a document. We used TF and IDF to generate an answer relevance score from Wikipedia. Figure 5.3

```
For each answer candidate $A_i$,
    Initialize the Wikipedia score: ws($A_i$) = 0
    Search for a Wikipedia document whose title is $A_i$
        1. If a document is found,
            1.1. Calculate tf.idf score of $A_i$ in the retrieved Wikipedia document
                    ws($A_i$) += (1+log(tf)) $\times$ (1+log(idf))
        2. If not, for each question keyword $K_j$,
            2.1. Search for a Wikipedia document that includes $K_j$
            2.2. Calculate tf.idf score of $A_i$ in the retrieved Wikipedia document
                    ws($A_i$) += (1+log(tf)) $\times$ (1+log(idf))
```

Figure 5.3: Algorithm to generate an answer relevance score from Wikipedia (tf: term frequency, idf: inverse document frequency obtained from a large corpus)

shows the algorithm to generate an answer relevance score from Wikipedia. First, a query consisting of an answer candidate is sent to Wikipedia. If there is a document whose title matches the query, the document is analyzed to obtain TF and IDF of the answer candidate, from which a tf.idf score is calculated. When there is no matching document, each question keyword is sent to Wikipedia as a back-off strategy, and the answer relevance score is calculated by summing the tf.idf scores of the answer candidate. To obtain word frequency information, the TREC Web Corpus was used as a large background corpus[4].

[4]The TREC Web Corpus is a corpus of web pages crawled by the Internet archive. It can be found at http://ir.dcs.gla.ac.uk/test_collections/wt10g.html

```
For each snippet $s_i$:
  1. Initialize the snippet co-occurrence score: $\mathrm{cs}(s_i) = 1$
  2. For each question keyword $k_j$ in $s_i$:
    2.1. Compute distance d, the minimum number of words between $k_j$ and
         the answer candidate, excluding stopwords and other keywords
    2.2. Update the snippet co-occurrence score:
         $\mathrm{cs}(s_i) = \mathrm{cs}(s_i) \times 2^{(1+d)}$
  3. Add the snippet score to the web score
Normalize the web score by dividing by the constant C
```

Figure 5.4: Algorithm to generate an answer relevance score from Google.

b) Google

The Web has been used for many different QA tasks, including query expansion [105] and as a direct source of answers [15, 20, 51]. It also has been used to validate answer candidates [54] and to filter out answers [47].

Following Magnini et al [54], we used the Web to generate a numeric score for each answer 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 using the algorithm in Figure 5.4. In our experiments, we used 100 as the constant C in Figure 5.4.

5.2 Answer Similarity Features

As factoid questions require short text phrases as answer(s), the similarity between two answer candidates can be calculated with string distance metrics. We calculate the similarity between two answer candidates using multiple string distance metrics and a list of synonyms.

5.2.1 String Distance Metrics

There are several different string distance metrics which calculate the similarity of two strings. Each of them can be used as an individual similarity feature to calculate answer similarity scores.

- Levenshtein distance: this is a simple distance metric calculated by the minimum number of insertions, substitutions or deletions required to change one string into another.
- Cosine similarity: this is widely used in information retrieval to measure the distance between two documents or the distance between a document and a query. Each string is represented as a term vector where each term can be defined with term frequency and inverse document frequency (tf.idf) of a word. If s_i and s_j represent each term in two strings, cosine similarity is defined as $\frac{\sum s_i s_j}{\sqrt{\sum s_i^2}\sqrt{\sum s_j^2}}$.
- Jaccard similarity: this is calculated by dividing the size of the intersection between two strings by the size of the union of two strings. If S_1 and S_2 represent two strings, then the Jaccard similarity is $\frac{|S_1 \cap S_2|}{|S_1| + |S_2| - |S_1 \cap S_2|}$. The Jaccard similarity penalizes the case when there is low overlap between two strings.

- Jaro and Jaro-Winkler: The *NIST Dictionary of Algorithms and Data Structures* defines this as "Jaro is the weighted sum of the percentage of matched characters from each file and transposed characters. Winkler increased this measure for matching initial characters, then rescaled it by a piecewise function, whose intervals and weights depend on the type of string (first name, last name, street, etc.)". More details can be found in [35, 36, 102]

To measure string similarity mentioned above, we used the SimMetrics library[5], which provides many string similarity metrics.

5.2.2 Synonyms

Synonym information can be used as another metric to measure answer similarity. We defined a binary similarity score for synonyms as following:

$$sim(A_i, A_j) = \begin{cases} 1, & if\ A_i\ is\ a\ synonym\ of\ A_j \\ 0, & otherwise \end{cases} \tag{5.1}$$

To obtain 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".

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". As Wikipedia supports more than 200 language editions, this approach can be used for many languages.

[5]http://sourceforge.net/projects/simmetrics/

The CIA World Factbook is used to find synonyms for a country name. The Factbook includes five different names for a country: a conventional long form, a conventional short form, a local long form, a local short form and a 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 obtain synonyms for different types of answer candidates [72]:

- Date: 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). Unspecified fields are left as "xx" (e.g., "April 1914" is converted into "1914-04-xx").
- Time: Temporal expressions are converted into the HH:MM:SS format (e.g., "six thirty five p.m." and "6:35 pm" are converted into "18:35:xx" and are considered synonyms).
- Numeric expression: Numeric expressions are converted into numbers (e.g, "one million" and "1,000,000" are converted into "1e+06" and are considered synonyms).
- 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.").

Chapter 6

Model Extensions

The answer ranking models have been extended to merge results from multiple extractors and to support multiple languages. As our models are based on a language-independent probabilistic framework, they do not need to be changed to support other languages. We just incorporated language-specific resources and retrained the models for individual languages. For answer merging, we combined the confidence scores returned from individual extractors with the answer relevance and answer similarity features. This chapter describes the extension of the answer ranking models to multi-strategy QA and multilingual QA.

6.1 Extension to Multi-strategy QA

Many QA systems utilize multiple strategies to extract answer candidates, and then merge the candidates to find the most probable answer [14, 12, 21, 37, 2, 74]. This multi-strategy approach assumes that a combination of similar answers extracted from different sources with different strategies performs better than any individual answering strategy alone. As answer candidates come from different agents with different score distributions, it is important to consider how the results proposed by alternative approaches can be combined. This merging process is challenging, as it requires combining ranked answer lists with independent score distributions.

Our answer ranking models can be extended to support multi-strategy QA by combining the confidence scores returned from individual extractors with the answer relevance and answer similarity features. Equation 6.1 shows the extended independent prediction model for answer merging, where m is the number of extractors, n is the number of answers returned from one extractor, and $conf_k$ is the confidence score extracted from the k_{th} extractor whose answer is 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 JAVELIN LIGHT extractor returns two answers of "Bill Clinton" in the candidate list: one has a score of 0.7 and the other has 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 $sim_k(A_i)$ already considers this similarity information.

$$\begin{aligned} & P(correct(A_i)|Q, A_1, ..., A_{m*n}) \\ &= \frac{exp(\alpha_0 + \sum_{k=1}^{K1} \beta_k rel_k(A_i) + \sum_{k=1}^{K2} \lambda_k sim_k(A_i) + \sum_{\mathbf{k=1}}^{\mathbf{m}} \gamma_{\mathbf{k}} \mathbf{conf_k})}{1 + exp(\alpha_0 + \sum_{k=1}^{K1} \beta_k rel_k(A_i) + \sum_{k=1}^{K2} \lambda_k sim_k(A_i) + \sum_{\mathbf{k=1}}^{\mathbf{m}} \gamma_{\mathbf{k}} \mathbf{conf_k})} \end{aligned} \tag{6.1}$$

Similarly, the joint prediction model can be extended to incorporate multiple confidence scores, as shown in Equation 6.2.

$$P(S_1, ..., S_{m*n}) \tag{6.2}$$
$$= \frac{1}{Z} exp \left(\sum_{i=1}^{m*n} \left[(\sum_{k=1}^{K1} \beta_k rel_k(A_i) + \sum_{\mathbf{k=1}}^{\mathbf{m}} \gamma_{\mathbf{k}} \mathbf{conf_k}) S_i + \sum_{N(i)} (\sum_{k=1}^{K2} \lambda_k sim_k(A_i, A_{N(i)})) S_i S_{N(i)} \right] \right)$$

6.2 Extension to Different Monolingual QA

We extended the answer ranking models to Chinese and Japanese monolingual QA by incorporating language-specific features into the framework. These extensions are described below.

6.2.1 Answer Relevance Features

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) Knowledge-based Features

The knowledge-based features involve searching for facts in a knowledge base such as gazetteers and WordNet. We utilized comparable resources for Chinese and Japanese to generate an answer relevance score between -1 and 1.

a) Gazetteers

There are few available gazetteers for Chinese and Japanese. Therefore, we extracted location data from language-specific resources. For Japanese, we extracted Japanese location information from Yahoo[1], which contains many location names in Japan and the relationships among them. We also used Gengo GoiTaikei[2], a Japanese 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 (described in Section 3.1). As there is more than one translation per candidate, the top 3 translations were used. This gazetteer information was used to assign an answer relevance score between -1 and 1 using the algorithm described in Figure 5.1.

b) Ontologies

For Chinese, we used HowNet [19], 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 using the algorithm described in Figure 5.2.

2) Data-driven Features

This section describes how we used Wikipedia and Google to generate answer relevance scores.

[1]http://map.yahoo.co.jp/
[2]http://www.kecl.ntt.co.jp/mtg/resources/GoiTaikei

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

Table 6.1: Articles in Wikipedia for different languages

a) Wikipedia

As Wikipedia supports more than 200 language editions, the approach used in English can be used for different languages without any modification. Table 6.1 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 Chinese documents available, we used Baidu[3], which is similar to Wikipedia but contains more articles written in Chinese. We first search in Chinese Wikipedia documents. When there is no matching document in Wikipedia, we search in Baidu as a backoff strategy: each answer candidate is sent to Baidu and the retrieved document is analyzed in the same way to analyze Wikipedia documents.

The idf score was calculated using word statistics from the Japanese Yomiuri newspaper corpus and the NTCIR Chinese corpus [83].

[3]http://baike.baidu.com

b) Google

The algorithm described in Figure 5.4 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[4] was used. For Chinese segmentation, a maximum-entropy based parser was used [101].

6.2.2 Answer Similarity Features

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. It can be also calculated by using a list of synonyms.

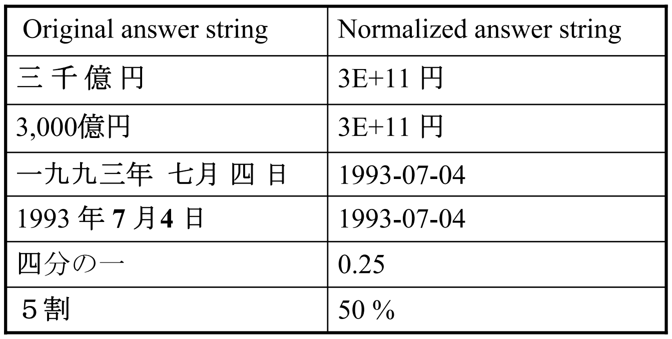
1) String Distance Metrics

The same string distance metrics used for English (see Section 5.2.1) were applied to calculate the similarity of Chinese/Japanese answer candidates.

2) Synonyms

To identify synonyms, Wikipedia was used for both Chinese and Japanese. The EIJIRO dictionary was also used to obtain Japanese synonyms. EIJIRO is a English-Japanese dictionary containing 1,576,138 words and provides synonyms for Japanese words.

[4]http://chasen.aist-nara.ac.jp/hiki/ChaSen



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 %

Figure 6.1: Example of normalized answer strings

For temporal and numeric expressions, new conversion rules were created; for example, a rule to convert Japanese Kanji characters to Arabic numbers is shown in Figure 6.1.

6.3 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 keyterms 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 [6]. This approach, however, requires documents for both the source language and the target language.

When a CLQA system uses translated questions or translated question keyterms 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 CLQA. To support English-to-Chinese and English-to-Japanese, we extend the features used for Chinese and Japanese monolingual answer ranking.

6.3.1 Answer Relevance Features

We reused knowledge-based features and extended data-driven features for CLQA as described below.

1) Knowledge-based Features

The knowledge-based features involve searching for facts in knowledge bases such as gazetteers and/or ontologies. We reused the features developed in Chinese and Japanese monolingual QA for English-to-Chinese and English-to-Japanese QA, respectively.

2) Data-driven Features

Data-driven features use Google and Wikipedia to generate answer relevance scores by counting word distance between a keyterm and an answer candidate or by calculating the tf.idf score of a keyterm or an answer candidates. As CLQA tends to have more than one translated keyterm candidate, the algorithms were extended to support multiple keyterm candidates.

a) Google

For monolingual QA, we generated a query consisting of an answer candidate and question keyterms. Since there are multiple translation candidates for each keyterm in CLQA, the algorithm was extended. As shown in Figure 6.2, the new algorithm uses each translated keyterm to create a query. As keyterm translation quality is typically poor for proper nouns, we added English proper noun keyterms to the query. For example, the question "In which city in Japan is the Ramen Museum located?" contains "Ramen Museum" as one keyterm. As the current translation module does not have a Chinese translation for this noun, it translates only Museum into a Chinese word. This partially translated phrase does not match any Web documents even though there are Chinese Web documents which matched the English term "Ramen Museum". Therefore, we used source proper noun keyterms as one alternation in the query.

However, some inaccurate translations may retrieve incorrect text snippets from the Web and give a high score to irrelevant answer candidates. Therefore, keyterms whose translation scores are less than some threshold value can be ignored, or only the top N keyterms can be used.

b) Wikipedia

We extended the algorithm to generate an answer relevance score using Wikipedia. First, a query consisting of an answer candidate is sent to Wikipedia. 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 keyterm is sent to Wikipedia as a back-off strategy. Similarly to the Google case, we search for English proper nouns appearing in the question and use keyterm thresholds. After retrieving relevant Wikipedia documents, each document is analyzed to obtain the tf.idf score of the answer candidate. The final answer relevance score is calculated by summing the tf.idf scores acquired from each keyterm.

6.3.2 Answer Similarity Features

In monolingual QA, answer similarity was calculated using string distance metrics and a list of synonyms (See Section 6.2.2). This approach was reused for answer ranking in CLQA.

```
For each answer candidate $A_i$:
  1. Initialize the Google score: $gs(A_i) = 0$
  2. Create a query from an answer candidate and the
     translated keyterms. For example, given the question
     "Which companies will merge with TECO Group?,"
     assume that we have three question  keyterms and
     their translation. Each translation has a  corresponding
     translation score.
       - company: 公司 (0.75)
       - merge: 合并 (0.78)
       - TECO Group: 东元集团, TECO 组, TECO 小组,
                     TECO 团体, 东芝集团

     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"
      or "TECO 团体" or "东芝集团")**

  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 keyterm translation $k$ in $s$:
          4.2.1. Compute distance $d$, the minimum number of
                 words between $k$ and the answer candidate,
                 excluding stopwords and other keywords
          4.2.2. Update the snippet co-occurrence score:
                 $cs(s) = cs(s) \times 2^{(1+d)^{-1}}$
     4.3 $gs(A_i) = gs(A_i) + cs(s)$
```

Figure 6.2: Algorithm to generate an answer relevance score from Google for cross-lingual QA.

Chapter 7

Evaluation Methodology

The answer ranking models were evaluated for three different languages: English, Japanese and Chinese. This chapter describes the data sets and evaluation metrics to measure performance of the models.

7.1 Data sets and Corpora

For English, the data set from the TREC 8-15 evaluation was used. For Chinese and Japanese, the data set from the NTCIR 5-6 evaluation was used.

7.1.1 English

A total of 2758 factoid questions from the TREC QA 8-15 evaluations [96, 97, 98, 99, 100, 17] [1] and their corresponding answers served as a dataset.

[1]http://trec.nist.gov/data/qamain.html

Two text corpora have been used in the TREC evaluations: TREC and AQUAINT. The TREC corpus consists of newspaper articles from the AP newswire, Wall Street Journal, San Jose Mercury News, Financial Times, Los Angeles Times and the Foreign Broadcast Information Service. The AQUAINT corpus contains AP newswire articles from 1998 to 2000, New York Times articles from 1998 to 2000, and Xinhua News Agency articles from 1996 to 2000. Each corpus contains approximately one million documents and three gigabytes of text.

7.1.2 Japanese

Questions from the NTCIR evaluation are used to evaluate answer selection for Japanese. The NTCIR5-6 CLQA task provides 700 Japanese questions and their corresponding answer patterns. The NTCIR CLQA Japanese corpus contains Yomiuri newspaper articles (2000-2001) and Mainichi newspaper (1998-2001). The corpus contains approximately 1,080,000 documents.

7.1.3 Chinese

As with Japanese, the data set from the NTCIR 5-6 CLQA evaluation is used to evaluate answer selection for Chinese. The data set contains 550 questions and their corresponding answer patterns. The NTCIR CLQA Chinese corpus contains United Daily News (1998-2001), United Express (2000-2001), Min Sheng Daily (1998-2001), Economic Daily News (1998-2001), United Evening News (1998-1999) and Start News (1998-1999). The corpus consists of approximately 900,000 documents and 3.6 gigabytes of text.

7.2 Evaluation Metrics

Answer selection performance can be evaluated using the following metrics:

- 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 5 answers to evaluate the performance of early QA systems. We will use MRR of the top 5 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 answers. 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.

As the performance of answer selection depends on the quality of answer extraction, we only considered questions where at least one correct answer exists in our evaluation.

Chapter 8

Evaluation

This chapter describes a series of experiments to evaluate the independent prediction model and the joint prediction model for answer ranking in multilingual question answering. The experiments were conducted with monolingual question answering systems (English, Chinese and Japanese), cross-lingual question answering systems (English-to-Chinese and English-to-Japanese) and multi-strategy QA systems to merge answer candidates produced by multiple English answer extractors.

8.1 Experimental Setup

As we had more than 2700 questions for the English QA system, 5-fold cross-validation was performed to evaluate our answer ranking models in English. For Chinese and Japanese, we had fewer available questions (550 questions for Chinese and 700 questions for Japanese), hence 3-fold cross-validation was performed to evaluate answer ranking in Chinese and Japanese.

Several baseline algorithms were used to compare the performance of our answer ranking models. These algorithms have been used for answer ranking in many QA systems.

- `IX`: 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. This approach was used in the LCC QA system [62] for the TREC-2002 evaluation. As it applied deep semantic analysis to find answers, the LCC system chose the answer which passed the logic prover without using any additional answer selection techniques.

- `Clustering`: This approach clusters identical or complementary answers and then assigns a new score to each cluster. The score of a cluster is calculated by counting the number of answers in the cluster [15], summing the scores of all answers in the cluster [46] or selecting the best score among the individual answer scores in the cluster [51]. In our experiments, we used the approach reported in (Nyberg et al.) [72] to assign a new score for each cluster. We clustered redundant answer candidates and calculated the score for an answer cluster given the assumption that all answers in the cluster are independent and equally weighted. For a cluster containing N answers whose extraction confidence scores are $S_1, S_2, ..., S_N$, the cluster confidence is computed with the following formula:

$$Score(Answercluster) = 1 - \prod_{i=1}^{N} (1 - S_i) \tag{8.1}$$

- `Filtering`: Many QA systems have applied filtering to remove improper answer candidates. This task involves comparing the expected answer type of the question with the type of answer candidates and then removing a candidate whose type does not match the expected answer type. Ontologies (such as WordNet) and gazetteers are one of the most popular resources to check whether or not an answer candidate matches the expected answer type [103, 14, 86]. In our experiments, we used both ontologies and gazetteers to filter out improper answer candidates. The algorithms described in Section 5.1.1 were used to identify improper answer candidates, and then these candidates were removed from the answer candidate list.

- `Web validation`: Magnini et al. [54, 55] validated answer candidates using a content-based approach that analyzes co-occurrence of question keywords and an answer candidate in Web text snippets. This approach 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 (called CombSum [24]) and then rerank answer candidates according to the sum, and (c) 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.

- `MaxEnt reranking`: In the latest TREC-2006 evaluation, the LCC's Chaucer QA system [30] was the second most accurate system for factoid questions. This system had an answer ranking component to merge answers returned from six answer extractors. This reranker was implemented using maximum entropy similar to (Ravichandran, Hovy, & Och [18]). We implemented a maximum entropy reranker as another baseline to be included with the features used in both LCC and Ravichandran et al. Altogether, six features were used: (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 of individual answer candidate provided by an extractor, and (6) the expected answer type. Answer candidates were reranked according to the scores from the maximum entropy reranker[1].

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

8.2 Monolingual QA

The models were evaluated for three different languages: English, Chinese and Japanese.

[1]The LCC Chaucer system had an additional answer selection step to find a final answer among the reranked top 25 answers using text entailment proposed in (Hickl et al. [29]). As it is beyond the scope of this thesis, we do not include this step in our reranking baseline system.

8.2.1 English

This section describes the experiments performed to evaluate our answer ranking models in English QA. The experiments were done with two QA systems: JAVELIN and EPHYRA.

(1) Experiments with JAVELIN

This section describes the experiments to evaluate our answer ranking models in the JAVELIN QA system.

Data Set

A total of 1760 questions from the TREC8-12 QA evaluations served as a data set. To better understand how the performance of our models can vary for different extraction techniques, we tested our answer ranking models with three JAVELIN answer extraction modules: FST, LIGHT and SVM. We did not include EXPERT and LIGHTv2 in the experiments because EXPERT covers only a small number of questions and does not provide sufficient training data to test our answer ranking models, and LIGHTv2 is very similar to LIGHT (both are based on distance heuristics).

Table 8.1 compares extractor performance on the test questions, and shows that extractors vary in the types of questions they can answer and in the average number of answers they return for each question. The third column in the table lists the number of answer sets returned by each extractor, and the fourth column shows the number of answer sets that included at least one correct answer. The fifth column shows the average size of the answer sets. The last two columns show the precision

Table 8.1: Performance characteristics of individual answer extractors: LEX (lexicon), LOC (location), OBJ (object), PER (person-name), ORG (organization-name), PROP (proper-name). 'Macro': precision at question-level. 'Micro': precision at answer-level.

Answer extractor	Types covered	#Answer sets	#Answer sets with correct answers	Avg size of answer sets	Precision	
					macro	micro
FST	DATE, LOC, ORG, OBJ, PER, PROP	837	301	4.19	0.171	0.237
LIGHT	All	1637	889	36.93	0.505	0.071
SVM	All	1553	871	38.70	0.495	0.077

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 number of total questions. Micro-level precision measures the precision of answer candidates. It was calculated by dividing the number of correct answers by the number of total answers. Generally, FST covers fewer questions than LIGHT and SVM, but it answers are more accurate than answers from the other extractors.

Even though the data set contains only factoid questions, approximately 36% of the questions have an average of five correct answers. This happens mostly for location, person name, numeric and temporal questions. 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 are “Gabriel Estavao Monjane”, “Robert Wadlow”, “Ali Nashnush”, “Barman”. In addition, TREC8-9 factoid questions include some list questions (e.g. “Name one of the major gods of Hinduism.”). Therefore, we evaluate the average precision of the top 5 answers in order to see how effectively the joint prediction model can identify unique answers when there is more than one correct answer.

For Wikipedia, we used data downloaded in November 2005, which contained 1,811,554 articles.

Results and Analysis

This section describes the experimental results for the independent prediction model and the joint prediction model.

1) Independent Prediction Model

Figure 8.1 shows the average top answer accuracy for the baseline systems and the independent prediction model. The result shows that baseline systems improved performance over `IX`. Among the baselines that uses a single feature (`Clustering, Filtering, Web validation`), `Web validation` produced the best performance for all three extractors. Among the combination of baseline systems, `C+F+W` achieved the best performance. This suggests that combining more resources was useful in answer selection for JAVELIN. On the other hand, `MaxEnt` worked well for FST, but did not work well for the LIGHT and SVM extractors. As `MaxEnt` depends only on internal resources such as frequency of answers and question word absent, it improved performance over `Clustering`, but did not gain benefit from the use of external resources such as Web, gazetteers and WordNet. As `Web validation` was more useful in LIGHT and SVM than in FST, `MaxEnt` worked well for FST, but did not work well for LIGHT and SVM.

When compared with the baseline systems, the independent prediction model obtained the best performance gain for all three extractors. The highest gain was achieved for the SVM extractor mostly because SVM produced more than one answer candidate with the same confidence score, but the independent prediction model could select the correct answer among many incorrect ones by exploiting answer similarity and relevance features.

Further analysis examined the degree to which the average top answer accuracy was affected by answer similarity features and answer relevance features. Figure 8.2 compares the average top answer accuracy using the answer similarity features, the answer relevance features and all feature combinations. As can be seen, similarity features significantly improved the performance, implying that exploiting redundancy improves answer selection. Relevance features also significantly improved the performance, and the gain was more significant than the gain from the similarity features.

When combining both types of features together, the answer selection performance increased for all three extractors: an average of 102% over `IX`, 31% over the similarity features alone and 1.82% over the relevance features alone. Adding the similarity features to the relevance features generated small but consistent improvement in all configurations. The biggest improvement was found with candidates produced by the SVM extractor: a 247% improvement over `IX`.

We also analyzed the average top answer accuracy when using individual features.

Figure 8.3 shows the effect of the individual answer relevance feature on different extraction outputs. The combination of all features significantly improved performance compared to answer selection using a single feature. Comparing data-driven features with knowledge-based features, we note that the data-driven features (such as Wikipedia and Google) increased performance more than the knowledge-based features (such as gazetteers and WordNet), mostly because the knowledge-based features covered fewer questions. The biggest improvement was found using Google, which provided a performance increase of an average of 74% over `IX`.

Table 8.2 shows the effect of individual similarity features on different extractors. As some string similarity features (e.g., Levenshtein distance) produce a number between 0 and 1 (where 1 means two strings are identical and 0 means they are different), similarity scores less than a threshold can be ignored. Table 8.2 compares the performance when using 0.3 and 0.5 as a threshold, respectively. When comparing five different string similarity features (Levenshtein, Jaro, Jaro-Winkler, Jaccard and Cosine similarity), Levenshtein and Jaccard tend to perform better than others. When comparing synonym with string similarity features, the synonym feature performed slightly better than the string similarity features.

As Levenshtein and Jaccard performed well among the five string similarity metrics, we also compared the combination of Levenshtein with synonyms and the combination of Jaccard with synonyms, and then chose Levenshtein and synonyms as the two best similarity features in the independent prediction model.

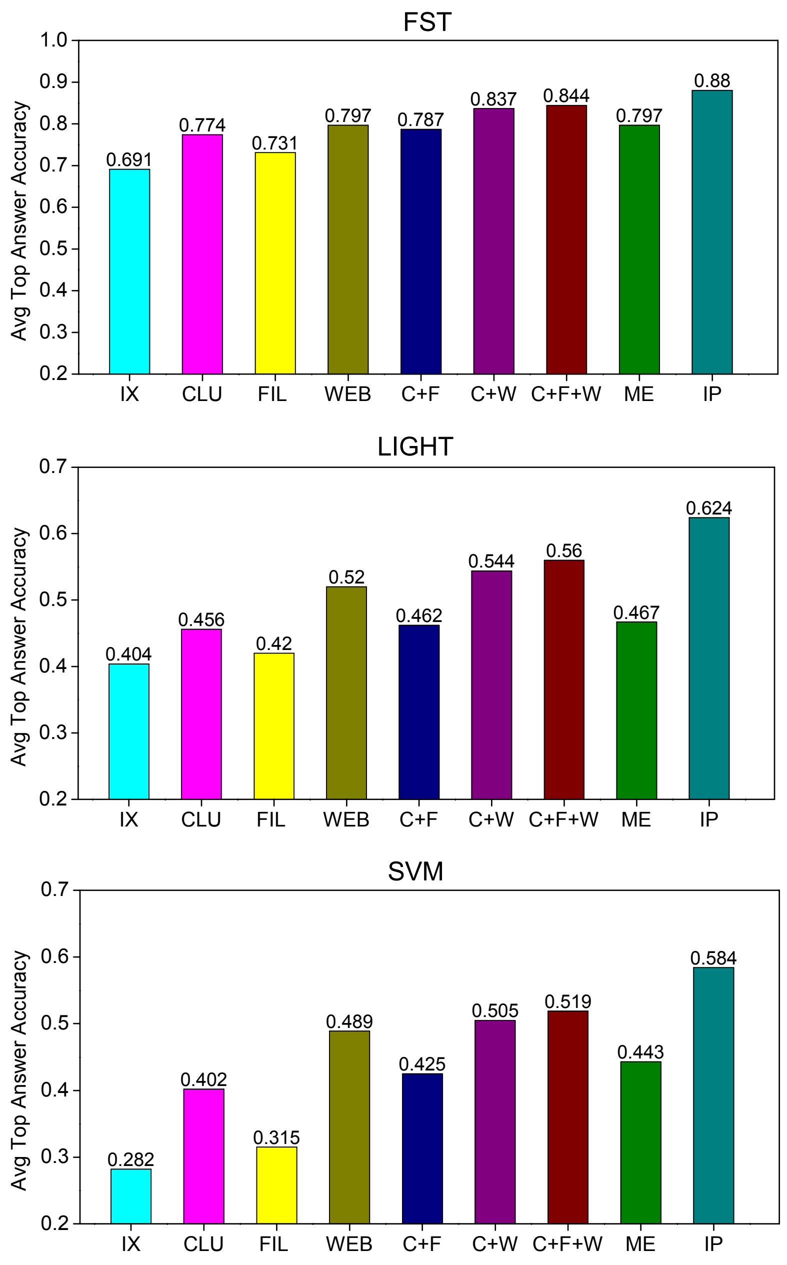


Figure 8.1: Performance of the baseline systems and the independent prediction model (C+F: combination of Clustering and Filtering, C+W: combination of Clustering and Web validation, C+F+W: combination of Clustering, Filtering and Web validation).

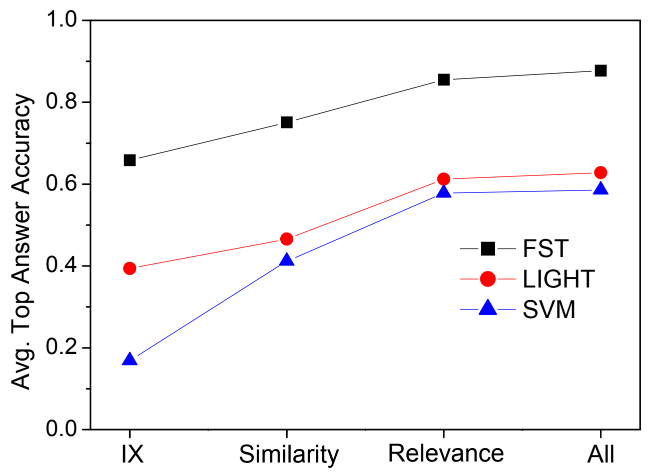


Figure 8.2: Average top answer accuracy of the independent prediction model (IX: performance of extractors, Similarity: merging similarity features, Relevance: merging relevance features, ALL: combination of all features)

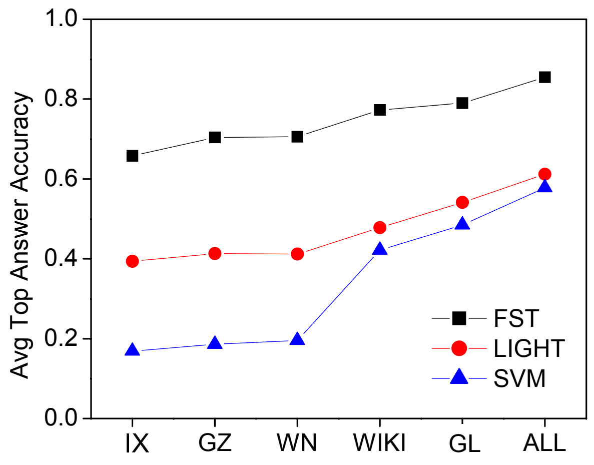


Figure 8.3: Average top answer accuracy of individual answer relevance features (GZ: gazetteers, WN: WordNet, WIKI: Wikipedia, GL: Google, ALL: combination of all relevance features).

Table 8.2: Average top answer accuracy of individual similarity features under different thresholds: 0.3 and 0.5.

Similarity feature	FST		LIGHT		SVM	
	0.3	0.5	0.3	0.5	0.3	0.5
Levenshtein	0.728	0.728	**0.471**	0.455	0.381	0.383
Jaro	0.708	0.705	0.422	0.440	0.274	0.282
Jaro-Winkler	0.701	0.705	0.426	0.442	0.277	0.275
Jaccard	0.738	0.738	0.438	0.448	0.382	0.390
Cosine	0.738	0.738	0.436	0.435	0.380	0.378
Synonyms	**0.745**	**0.745**	0.458	0.458	**0.412**	**0.412**
Lev+Syn	0.748	**0.751**	0.460	**0.466**	**0.420**	0.412
Jac+Syn	0.742	0.742	0.456	0.465	0.396	0.396

2) Joint Prediction Model

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, so that we could enumerate all possible configurations in a joint table, and then easily calculate marginal and conditional probabilities using the joint table. In the experiments, we used two different data sets: (1) the top 10 answer candidates provided by each individual answer extractor and (2) the top 10 answers returned from the independent prediction model. For approximate inference, we used Gibbs sampling.

a) 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. As it requires O(2^N) time and space where N is the size of the graph (i.e. number of answer candidates), we used only the top 10 answer candidates produced by each extractor. Figure 8.4 shows how to generate the joint table using the top 10 answers.

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.
2. For each row i, calculate a value using the relevance and similarity features and store it to JT(i,N+1).

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

2. Calculate the normalization constant Z: $Z = \sum_{i=1}^{2^N} JT(i, N+1)$
3. To make each value a probability, divide each value by Z.

$$JT(i, N+1) = JT(i, N+1)/Z$$

Figure 8.4: Algorithm 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.

$$P(correct(A_1)|Q, A_1, ..., A_n) = \sum_{i \in (JT(i,1)==1)} P(i, N+1)$$

The parameters for the model were estimated from the training data by maximizing the joint probability (Equation 8.2). This was done with the Quasi-Newton algorithm [56].

$$\vec{\beta}, \vec{\lambda} = \underset{\vec{\beta},\vec{\lambda}}{\text{argmax}} \sum_{j=1}^{R} log \frac{1}{Z} exp \left(\sum_{i=1}^{n} \left[(\sum_{k=1}^{K1} \beta_k rel_k(A_i)) S_i + \sum_{N(i)} (\sum_{k=1}^{K2} \lambda_k sim_k(A_i, A_{N(i)})) S_i S_{N(i)}) \right] \right) \tag{8.2}$$

Table 8.3 shows the performance of the joint prediction model, compared with the independent prediction model. As for TOP1, the joint prediction model performed as well as the independent prediction model in ranking the relevant answer at the top position. MRR5 shows the performance of the models 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 to what degree the joint prediction model could identify comprehensive results, we analyzed the average precision for the top 5 answers. Table 8.4 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

Table 8.3: Performance of IP and JP when using the top 10 answer candidates produced by each individual extractor.

	FST		LIGHT		SVM	
	IP	JP	IP	JP	IP	JP
TOP1	0.873	0.870	0.604	0.605	0.532	0.536
MRR5	0.936	0.952	0.699	0.729	0.618	0.652

Table 8.4: Average precision of IP and JP.

Average	FST		LIGHT		SVM	
Precision	IP	JP	IP	JP	IP	JP
at rank1	0.873	0.870	0.604	0.605	0.532	0.536
at rank2	0.420	0.463	0.359	0.383	0.311	0.339
at rank3	0.270	0.297	0.268	0.280	0.233	0.248
at rank4	0.175	0.195	0.222	0.222	0.193	0.199
at rank5	0.117	0.130	0.190	0.190	0.167	0.170

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 which is 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 answers exist. To address this issue, we performed exact inference using the answer candidates filtered by the independent

Table 8.5: Performance of IP and JP when using the top 10 answers produced by IP.

	FST		LIGHT		SVM	
	IP	JP	IP	JP	IP	JP
TOP1	0.880	0.874	0.624	0.637	0.584	0.583
MRR5	0.935	0.950	0.737	0.751	0.702	0.724

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 3 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 to what degree the joint prediction model could identify comprehensive results, we analyzed the average precision within the top 5 answers. Table 8.6 shows the average precision of the models. 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 described in the previous section (a) because the previous one improved the average precision at rank 2 by only 10% (FST), 6% (LIGHT) and 9% (SVM). This additional analysis on average precision clearly shows that the joint

Table 8.6: Average precision of IP and JP when using the top 10 answers produced by IP.

Average	FST		LIGHT		SVM	
Precision	IP	JP	IP	JP	IP	JP
at rank1	0.880	0.874	0.624	0.637	0.584	0.583
at rank2	0.414	0.548	0.377	0.541	0.350	0.498
at rank3	0.269	0.377	0.274	0.463	0.255	0.424
at rank4	0.178	0.259	0.220	0.399	0.203	0.366
at rank5	0.118	0.181	0.191	0.349	0.175	0.319

prediction model can generate more comprehensive results than the independent prediction model.

c) Approximate inference using Gibbs sampling

We tested the joint prediction model with only the top 10 answers either provided by each extractor or provided 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 ten 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. Gibbs sampling has been commonly used for undirected graphical models because it is simple and requires only conditional probability $P(S_i|S_{-i})$, where S_{-i} represents all nodes except S_i (Equation 8.4).

$$\begin{aligned} &P(S_i = 1|S_{-i}) \qquad (8.3)\\ &= \frac{P(S_i = 1, S_{-i})}{P(S_i = 1, S_{-i}) + P(S_i = 0, S_{-i})}\\ &= \frac{1}{1 + \frac{P(S_i=0,S_{-i})}{P(S_i=1,S_{-i})}} \end{aligned}$$

Using this conditional probability, Gibbs sampling generates a set of samples: $S^{(0)}, S^{(1)}, S^{(2)}, ..., S^{(T)}$. Equation 8.4 shows how Gibbs sampling generates one sample $S^{(t+1)}$ from the previous sample $S^{(t)}$. In each sequence, each component $S_i^{(t+1)}$ is generated from the distribution conditional on the other components. This result $S_i^{(t+1)}$ is then used for sampling of the next component.

$$\begin{aligned} &1.\ S_1^{(t+1)} \sim p(S_1|S_2^{(t)}, ..., Sn^{(t)}) \qquad (8.4)\\ &2.\ S_2^{(t+1)} \sim p(S_2|S_1^{(t+1)}, x_3^{(t)} ..., S_n^{(t)})\\ &3.\ S_i^{(t+1)} \sim p(S_i|S_1^{(t+1)}, ..., S_{i-1}^{(t+1)}, S_{i+1}^{(t)}, ..., S_n^{(t)})\\ &4.\ S_n^{(t+1)} \sim p(S_n|S_1^{(t+1)}, ..., S_{n-1}^{(t+1)}) \end{aligned}$$

As it takes time for Gibbs sampling to converge, we ignored the first 2000 samples. This process is called burn-in. In addition, all samples are not independent and 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) [44] as shown in Equation 8.5. This learning method has been popularly used with Gibbs sampling because it quickly converges after a few steps. More details about contrastive divergence can be found at [31, 11].

$$CD_n = KL(p_0||p_\infty) - KL(p_n||p_\infty) \tag{8.5}$$

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

Table 8.7 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 Table 8.1). 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.

Table 8.7: Performance of JP when using Gibbs sampling

	FST		LIGHT		SVM	
	IP	JP	IP	JP	IP	JP
TOP1	0.880	**0.870**	0.624	**0.537**	0.584	**0.480**
MRR	0.935	0.930	0.737	0.657	0.702	0.638

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.

The experimental results show that exact inference using the outputs from the independent prediction model produced the best performance. Therefore, in the rest of the experiments, we will use this approach to implement the joint prediction model and compare its performance with the independent prediction in other systems.

To address the unbalanced data problem, resampling including over-sampling and under-sampling [1, 107] 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) [108] 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 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.

(2) Experiments with 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 answer ranking models for list questions.

Data Set

The data set we used in the previous section contained many questions which had more than one correct answers. To evaluate the answer ranking models for list questions, we extracted 482 questions whose number of correct answers is more than three from the TREC8-12 questions; an average number of the correct answers was 5.7.

Table 8.8 shows the characteristics of the LIGHT and SVM extractors. As the FST extractor returns the average of 4.19 answers, we only used the LIGHT and SVM extractors for this experiment. It can be seen that precision is higher than the factoid case (shown in Table 8.1) because the questions had more correct answers.

Table 8.8: Performance characteristics of the LIGHT and SVM extractors

	#Answer	#Answer sets with	Average size	Precision	
Extractor	sets	correct answers	of answer sets	macro	micro
LIGHT	299	203	36.4	0.679	0.110
SVM	284	196	35.3	0.690	0.125

Table 8.9: Average precision of the answer ranking models when there are more than three correct answers per question ("Diff": difference between IP and JP at each rank).

	LIGHT			SVM		
	IP	JP	Diff	IP	JP	Diff
at rank 1	0.532	0.547	0.015	0.473	0.493	0.020
at rank 2	0.355	0.461	0.106	0.318	0.411	0.094
at rank 3	0.250	0.386	0.136	0.236	0.343	0.107
at rank 4	0.217	0.346	0.129	0.195	0.308	0.113
at rank 5	0.188	0.315	0.127	0.174	0.286	0.111
at rank 6	0.164	0.284	0.120	0.159	0.260	0.101
at rank 7	0.144	0.251	0.107	0.143	0.242	0.099
at rank 8	0.127	0.228	0.101	0.132	0.233	0.102
at rank 9	0.113	0.207	0.094	0.120	0.214	0.094
at rank 10	0.104	0.193	0.089	0.112	0.200	0.088

Results and Analysis

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 8.9 shows the average precision of the models. 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.

However, the improvement of JP over IP tends to decrease as there are more answers. “Diff” in Table 8.9 shows the performance difference between IP and JP at each rank. For the LIGHT case, the difference started to decrease at rank 4 and after rank 6, it significantly decreased. In the SVM case, the difference started to decrease at rank 5. When considering that the average number of the correct answers was 5.7, we can tell that JP tends to improve the average precision while there are correct answers, but this improvement tends to decrease when adding more answers to the input (i.e. the input contains more incorrect answers).

(3) Experiments with EPHYRA

We also evaluated the answer ranking models with the answer candidates provided by the EPHYRA QA system. This section describes experiments done with the EPHYRA QA system [85].

Data Set

A total of 998 factoid questions from the TREC13-15 QA evaluations served as a data set. For the Wikipedia feature, we used a version of Wikipedia downloaded in Feb 2007.

EPHYRA has two extractors: Extractor1 and Extractor2. Extractor1 exploits answer types to extract associated named entities, and Extractor2 uses patterns which were automatically obtained from question-answer pairs in the training data.

Table 8.10 shows the characteristics of the EPHYRA extractors. It can be seen that micro-level precision was lower here than in the JAVELIN case mostly because the EPHYRA extractors returned much more answers than the JAVELIN extractors. In addition, the correct answer occurs only once in the EPHYRA candidate list because the input to the answer ranking models was already merged. On the other hand, the JAVELIN extractors tend to contain redundant correct answer candidates extracted from different documents. The redundancy of correct answers is another reason why micro-level precision was higher in JAVELIN.

Table 8.10: Performance characteristics of EPHYRA extractors

	#Answer	#Answer sets with	Average size	Precision	
Extractor	sets	correct answers	of answer sets	macro	micro
Extractor 1	813	464	27	0.465	0.026
Extractor 2	535	305	104	0.306	0.008

Results and Analysis

Figure 8.5 shows the performance of baseline systems, IP and JP on the EPHYRA answer candidates. `Clustering` did not affect performance even though it was useful in the JAVELIN case. As EPHYRA already combined the answer candidates whose surface strings were the same, there was no answer to be clustered. On the other hand, JAVELIN extractors return multiple redundant answer candidates from different documents. Therefore, exploiting answer redundancy was important in JAVELIN, but not in EPHYRA.

`Filtering` did not significantly affect performance because the number of EPHYRA answer types is smaller than that in JAVELIN. In addition, JAVELIN has two-tier answer types: one for the named entity information (e.g. location, person, organization) and the other for more specific information such as city, river, writer, etc. As `Filtering` heavily depends on the second answer type and EPHYRA does not produce this information, the gain from `Filtering` was not significant.

Among the baseline systems, `Web validation` produced the best performance for both Extractor1 and Extractor2. When compared with `Web validation`, the answer ranking models did not significantly improve performance. To explain this result, further analysis was done with the answer relevance and similarity features. Figure 8.6 (a) shows the utility of the relevance features on the independent prediction model. It can be seen that the knowledge-based features (gazetteers and WordNet) did not significantly boost performance; this is similar to the `Filtering` case. The biggest improvement was found with Google. Adding Wikipedia slightly improved results only in Extractor2. Figure 8.6 (b) shows the effect of the similarity features on the independent prediction model. As there was much less redundancy in the answer candidates, the similarity features also had little impact on performance either. Therefore, Google was the only feature that affected the answer selection performance in EPHYRA. This explains why `Web validation` achieved performance gain comparable to that achieved by our answer ranking models.

One interesting result is that `C+F+W` produced lower performance than `Web validation` because each method (`Clustering, Filtering, Web validation`, respectively) sometimes made a conflict decision. This demonstrates that combination of multiple approaches is hard. However, our answer ranking models made a small but significant improvement over one single baseline even though they merged multiple approaches.

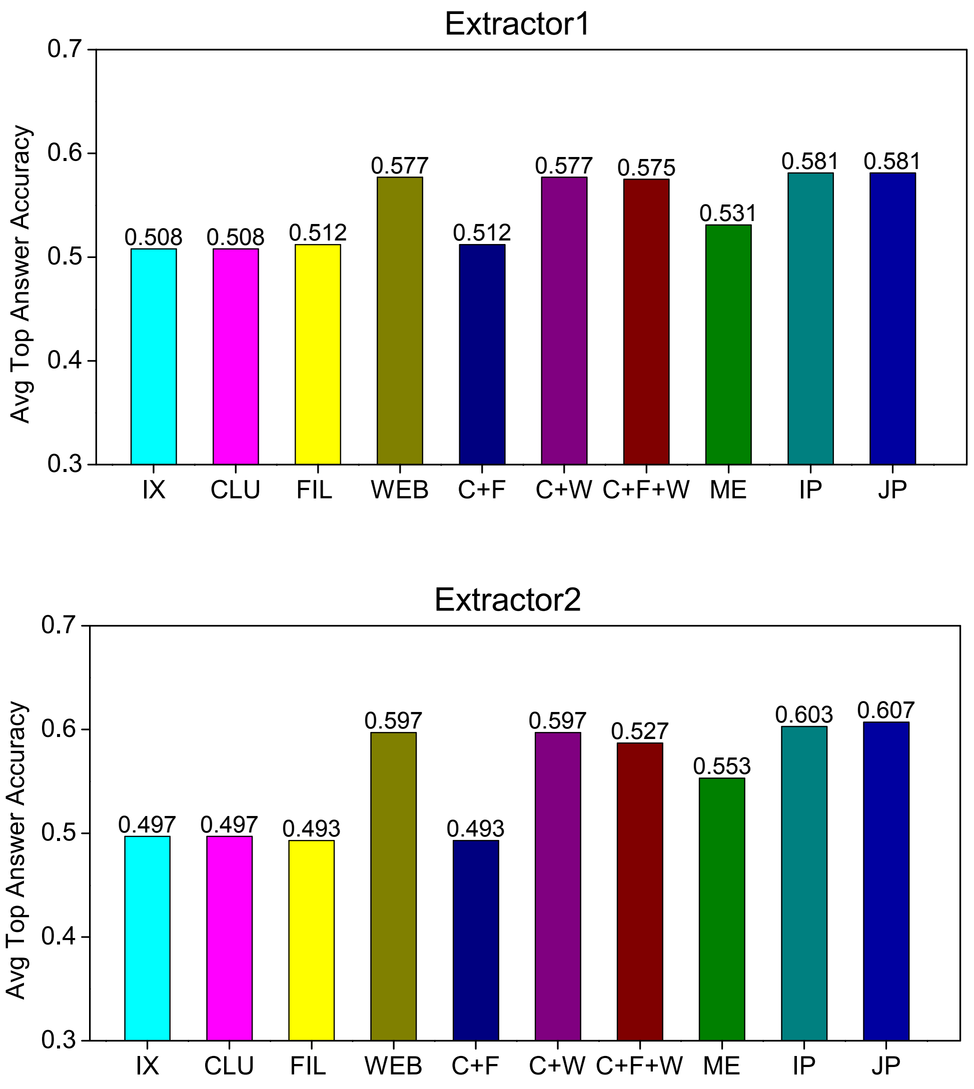
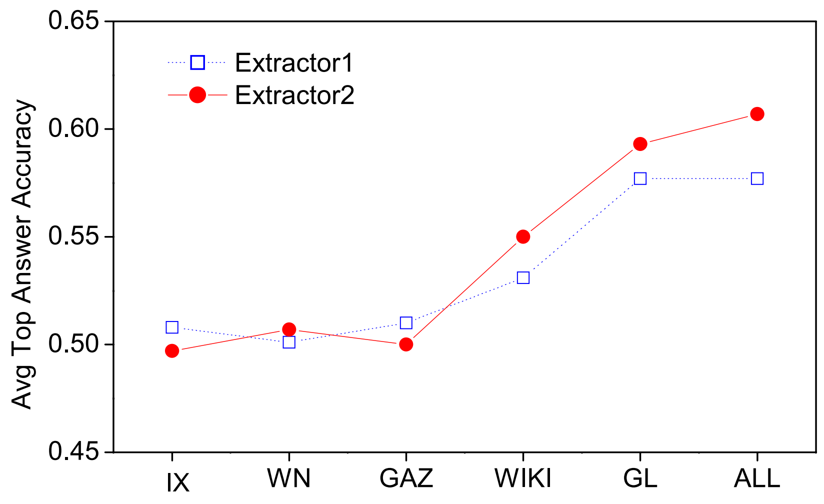


Figure 8.5: Performance of the answer ranking models for the EPHYRA QA system.



(a) Relevance features

Similarity feature	Extractor1		Extractor2	
	0.3	0.5	0.3	0.5
Cosine	0.499	0.497	0.547	0.557
Jaccard	0.495	0.492	0.553	0.543
Jaro	0.497	0.501	0.530	0.520
JaroWinkler	0.492	0.499	0.527	0.520
Levenshtein	0.505	0.501	0.523	0.527
Synonyms	**0.514**	**0.514**	0.523	0.523
JAR+Syn	**0.514**	0.503	**0.560**	0.540
Lev+Syn	0.510	0.512	0.537	0.543

(b) Similarity features

Figure 8.6: Average top answer accuracy when using individual relevance features and similarity features

8.2.2 Chinese

This section describes the experiments we used to evaluate the answer ranking models for Chinese QA. The JAVELIN QA system [58] was used as a testbed for the evaluation.

Data Set

550 Chinese questions provided by the NTCIR 5-6 QA evaluations served as the data set. Among them, 200 questions were used to train the Chinese answer extractor and 350 questions were used to evaluate our answer ranking models. Table 8.11 shows the characteristics of the Chinese extractor. As the Chinese extractor returned many answer candidates (the average number of answer candidates was 565.8), micro-level precision was very low. Therefore, we preprocessed the data to remove answer candidates having rank lower than 100.

Table 8.11: Performance characteristics of the Chinese extractor

	#Answer sets	#Answer sets with correct answers	Average size of answer sets	Precision	
				macro	micro
Chinese extractor	350	272	565.8	0.777	0.010

For the Wikipedia feature, we used a version of Wikipedia downloaded in Feb 2007.

Results and Analysis

Figure 8.7 compares the average top answer accuracy when using the baseline systems, the independent prediction model and the joint prediction model. Among the

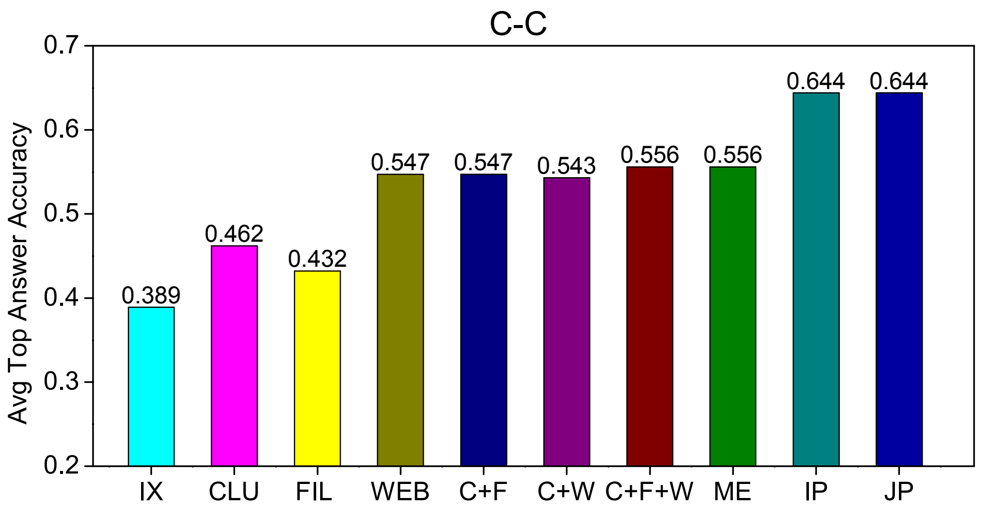


Figure 8.7: Performance of the answer ranking models for Chinese answer selection

Table 8.12: Average precision of the answer ranking models for Chinese answer selection

	at rank1	at rank2	at rank3	at rank4	at rank5
IP	0.644	0.356	0.247	0.200	0.167
JP	0.644	0.401	0.290	0.226	0.186

baseline systems which used a single feature, `Web validation` produced the best performance ; observe that this is similar to results for the English case. The best baseline systems were (`C+F+W` and `MaxEnt reranking`). When comparing the baseline systems with the answer ranking models, we note that the answer ranking models obtained beter performance gain than the baseline systems. Both of the models improved performance by 15.8% over the best baseline systems (`C+F+W` and `MaxEnt reranking`).

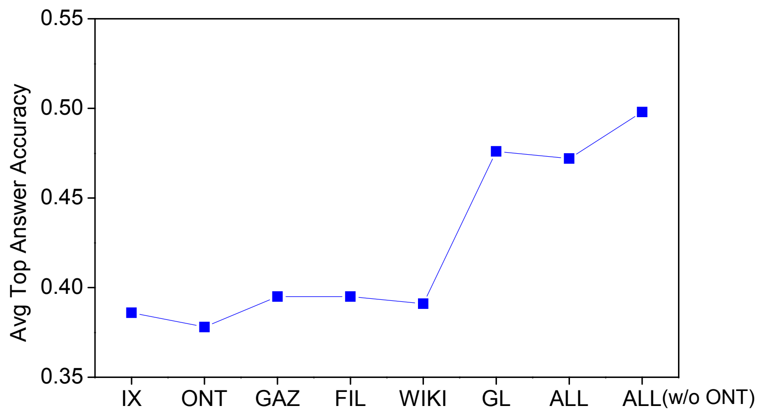
As there was no difference between independent prediction and joint prediction in selecting the top answer, we further investigated to what degree the joint prediction model could identify comprehensive results. Table 8.12 compares the average precision of IP and JP and shows that JP performed better than IP when selecting the top 5 answers. This was because joint prediction could identify unique correct answers by estimating conditional probability.

We further analyzed the utility of individual relevance features in the independent prediction model (Figure 8.8 (a)). The manual filtering and gazetteers features were somewhat useful in ruling out wrong answers, but the ontology did not improve performance; we assume that this is because the Chinese ontology (HowNet) contains much less information overall than the English ontology (WordNet). As for Wikipedia, there were fewer Chinese Wikipedia documents available. Even though we used Baidu as a supplemental resource for Chinese, this did not improve answer selection performance. Among the relevance features, Google produced the best performance.

Given that the limited coverage of the ontology feature decreased performance, we also compared the performance when combining all relevance features with the performance when combining all features except ontology ("ALL" v.s. "ALL (except ontology)" in Figure 8.8 (a)). The latter improved performance by 5.5% over the former.

Figure 8.8 (b) shows the effect of individual similarity features on Chinese answer selection when using two thresholds (0.3 and 0.5). In our experiments, the combination of Levenshtein and Synonyms worked best.

In our previous experiments on English QA, the relevance features improved performance by an average of 67.5% over `IX` and the similarity features improved performance by an average of 33.4% over `IX`. In Chinese, both the similarity and the relevance features significantly improved answer selection performance compared to the baseline: 63.5% improvement when using similarity features and 29% improvement when using relevance features. However, answer relevance features played less important roles than in English, because of fewer Chinese resources to identify answer relevance. For example, as of Feb 2007, Chinese Wikipedia contained around 263,000 documents, and English Wikipedia contained 3,583,699 documents. Due to the smaller coverage in Chinese, the answer relevance features had less impact on answer ranking in Chinese QA.



(a) Relevance features

Similarity feature	0.3	0.5
Cosine	0.614	0.614
Jaccard	0.614	0.614
Jaro	0.579	0.554
JaroWinkler	0.584	0.558
Levenshtein	0.588	0.579
Synonyms	0.609	0.609
Cos+Syn	0.605	0.605
Lev+Syn	**0.631**	0.618

(b) Similarity features

Figure 8.8: Average top answer accuracy in Chinese QA when using (a) individual relevance features and (b) similarity features.

8.2.3 Japanese

This section describes the experiments we conducted to evaluate the answer ranking models for Japanese QA. The JAVELIN QA system [58] was used as a testbed for the evaluation.

Data Set

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 models. Among 400 test questions, the Japanese extractor did not produce any answers for 4 questions and 251 questions contained at least one correct answer (Table 8.13).

Table 8.13: Performance characteristics of the Japanese extractor

	#Answer sets	#Answer sets with correct answers	Average size of answer sets	Precision	
				macro	micro
Japanese extractor	396	251	58.5	0.628	0.077

For the Wikipedia feature, we used a version of Wikipedia downloaded in Feb 2007.

Results and Analysis

Figure 8.9 compares the average accuracy when using baselines, the independent prediction model and the joint prediction model. Among the baseline systems, `MaxEnt reranking` produced the best performance. On the other hand, `Web validation`

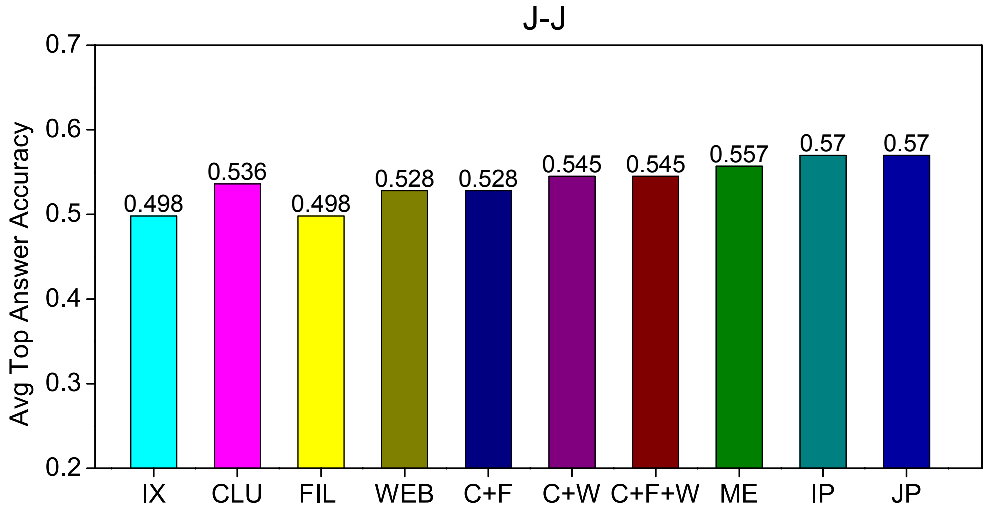


Figure 8.9: Performance of the answer ranking models for Japanese QA

Table 8.14: Average precision of the answer ranking models for Japanese QA.

	at rank1	at rank2	at rank3	at rank4	at rank5
IP	0.570	0.315	0.237	0.185	0.156
JP	0.570	0.379	0.271	0.209	0.171

was not as useful for Japanese as in the Chinese and English cases. This can be explained by analyzing the difference in the data set. Figure 8.10 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%).

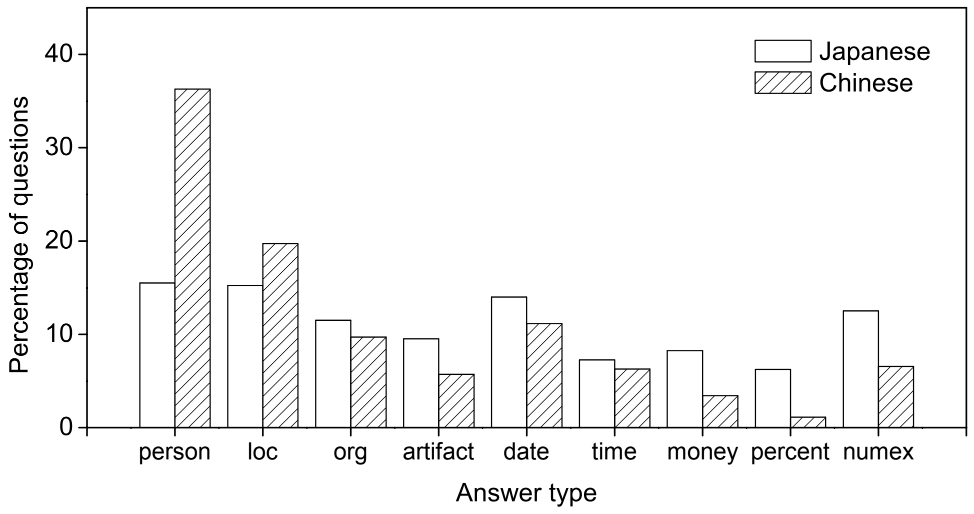


Figure 8.10: Answer type distribution in Chinese and Japanese data set

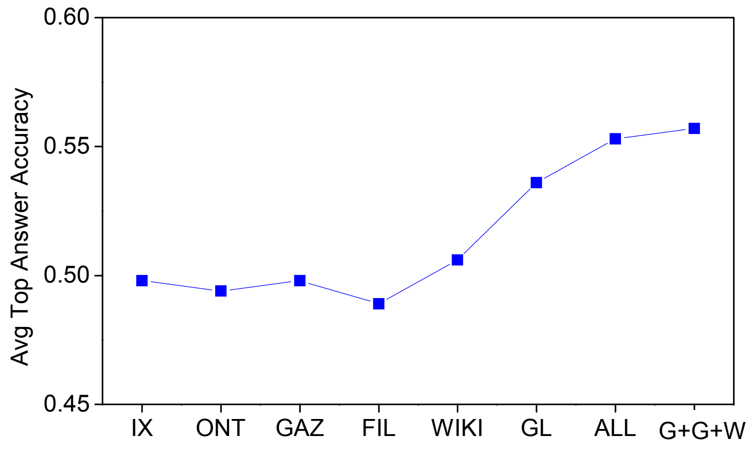
`Web validation` is less useful in validating numeric and temporal questions because correct answers to numeric and temporal questions may vary over even short periods of time. In addition, some answers are too specific and hard to find within Web documents (e.g. "At what hour did a truck driven by Takahashi rear-end a truck driven by Hokubo?" or "How many cardboard boxes were there on the truck driven by Hiroshi Hokubo?"). As Japanese question set contained much more numeric and temporal questions, `Web validation` was not as useful as in the Chinese case.

The smaller performance gain from `Web validation` also limited the improvement of both IP and JP relative to `MaxEnt`. Even though both IP and JP slightly improved the average top answer accuracy (an increase of 2.25% over `MaxEnt reranking`), the performance gain was not statistically significant.

More analysis was performed by examining the utility of relevance features. Figure 8.11 (a) shows that the data-driven features were more useful than the knowledge-based features. This is similar to the Chinese case, but the gain from data-driven features was much less than in the Chinese case. For example, the Google feature feature led to improvements of 7.6% in Japanese and 23.3% in Chinese. In addition, the gain from the similarity features was less than in the Chinese case (6% gain in Japanese and 63.5% gain in Chinese). Therefore, there was less opportunity to boost correct answers to the top position when using our answer ranking models.

Figure 8.11 (b) shows the effect of individual similarity features on Japanese answer selection. In our experiments, Levenshtein tends to perform better than the other features with threshold “0.3”. However, the performance gain from the similarity features was much less than that in the Chinese case: 58.9% gain in Chinese and 6% gain in Japanese. In addition, the combination of two similarity features (“Lev+Syn” and “Cos+Syn”) did not improve performance in Japanese. Therefore, we only used Levenshtein feature in our answer ranking models for Japanese.

We also analyzed the difference between IP and JP. Table 8.14 shows the average precision at rank N. Similar to the English and Chinese cases, the joint prediction model could identify unique correct Japanese answers by estimating conditional probability.



(a) Relevance features

Similarity feature	0.3	0.5
Cosine	0.511	0.511
Jaccard	0.511	0.511
Jaro	0.502	0.506
JaroWinkler	0.502	0.506
Levenshtein	**0.528**	0.506
Synonyms	0.515	0.515
Cos+Syn	0.494	0.506
Lev+Syn	0.506	0.494

(b) Similarity features

Figure 8.11: Average top answer accuracy in Japanese QA when using (a) individual relevance features ("G+G+W": combination of Google, Gazetteers and Wikipedia) and (b) similarity features.

8.2.4 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 Google, 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.

	IX	Data-driven features	All features
English (FST)	0.691	0.840	0.880
English (LIGHT)	0.404	0.617	0.624
English (SVM)	0.282	0.556	0.584
Chinese	0.386	0.635	0.644
Japanese	0.478	0.553	0.570

Table 8.15: Average top answer accuracy when using data-driven features v.s. when using all features.

Table 8.15 shows the performance when using data-driven features v.s. all features in the independent prediction model. For all three languages, data-driven features alone achieved significant improvement over `IX`. This indicates that the model can easily be extended to any language where appropriate data resources are available, even if knowledge-based features and resources for the language are still under development.

8.3 Cross-lingual QA

This section describes our experiments which evaluate the answer ranking models for cross-lingual QA. Empirical results for NTCIR cross-lingual questions (English-to-Chinese and English-to-Japanese) show the effectiveness and robustness of the models in cross-lingual answer ranking.

8.3.1 English-to-Chinese

This section describes the experiments that evaluate the answer ranking models for English-to-Chinese QA. The JAVELIN QA system [58] was used as a testbed for the evaluation.

Data Set

550 questions provided by NTCIR 5-6 were used as a data set. Among them, 200 questions were used to train the Chinese answer extractor and the other 350 questions were used to evaluate the answer ranking models. Among the 350 test questions, the Chinese extractor could not find any answers for one question; 190 questions contained at least one correct answer in the candidate list (Table 8.16).

Table 8.16: Performance characteristics of the extractor

#Answer sets	#Answer sets with correct answers	Average size of answer sets	Precision	
			macro	micro
349	190	76.6	0.543	0.029

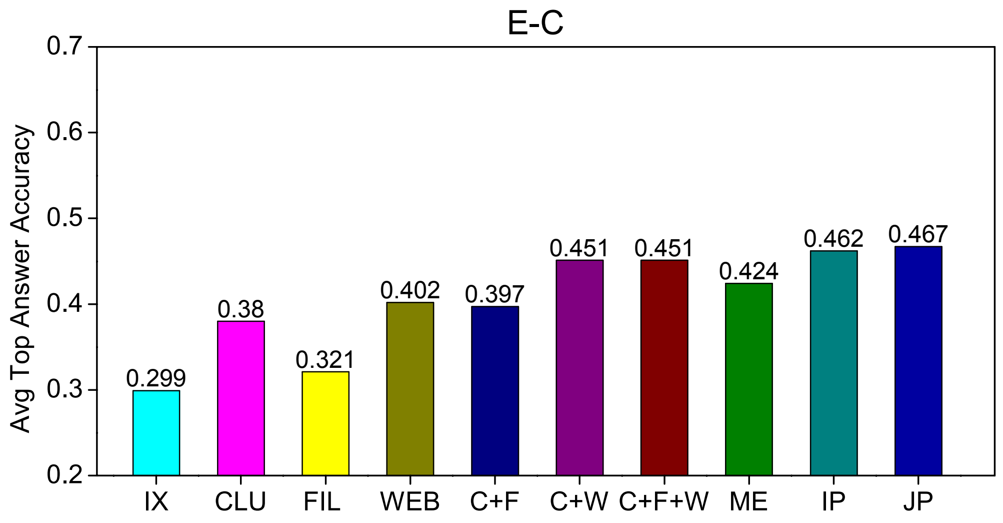


Figure 8.12: Performance of the answer ranking models in English-to-Chinese QA.

Table 8.17: Average precision of the answer ranking models in English-to-Chinese QA.

	at rank1	at rank2	at rank3	at rank4	at rank5
IP	0.462	0.255	0.190	0.155	0.135
JP	0.467	0.293	0.246	0.196	0.164

Results and Analysis

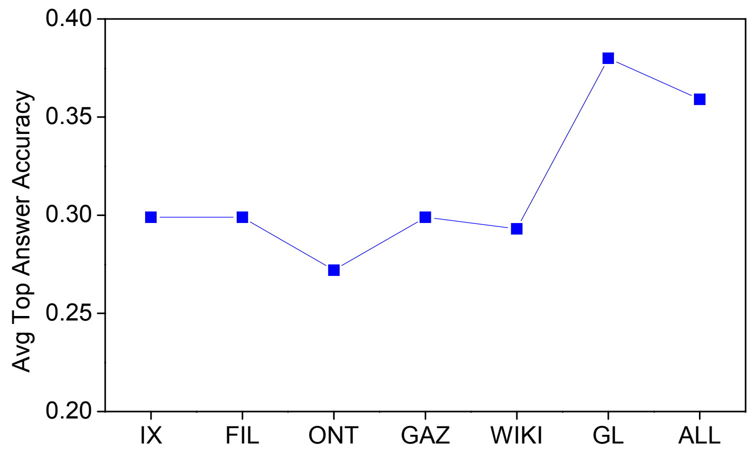
Figure 8.12 compares the average accuracy when using the baseline systems, IP and JP. Among the baseline systems,"C+W" and "C+W+F" produced the best performance. Compared to the best baseline systems, IP and JP slightly improved performance, but the difference was not statistically significant. This differs from the Chinese monolingual QA where IP and JP achieved significant performance gain over "C+W" and "C+W+F".

Table 8.17 compares the average precision of IP and JP. Similar to the Chinese monolingual case, JP performed better than IP when selecting the top 5 answers by identifying unique answers for English-to-Chinese QA.

We also analyzed to what extent the average top answer accuracy was affected by the use of individual features. Figure 8.13 (a) shows the performance of individual relevance features. Google produced the best performance, but the other relevance features did not provide performance gains because they covered too few questions. Therefore, combining all relevance resources did not improve the performance over Google. This result is unlike that for the monolingual case, where a combination of relevance features obtained a performance gain.

Figure 8.13 (b) shows the performance of similarity features under different similarity thresholds. Among the different string similarity features we tested, Cosine and Jaccard features performed better than other features; further, the combination of the Cosine and Synonym features produced the best performance for both threshold 0.3 and 0.5.

Similar to the Chinese monolingual QA case, we achieved greater performance gain from the similarity features than the relevance features; this is because we had fewer available resources to estimate answer relevance in Chinese.



(a) Relevance features

Similarity feature	0.3	0.5
Cosine	0.451	0.451
Jaccard	0.451	0.451
Jaro	0.37	0.38
JaroWinkler	0.37	0.386
Levenshtein	0.413	0.424
Synonyms	0.418	0.418
Cos+Syn	**0.457**	**0.457**
Lev+Syn	0.429	0.424

(b) Similarity features

Figure 8.13: Average top answer accuracy in English-to-Chinese QA when using (a) individual relevance features and (b) similarity features.

8.3.2 English-to-Japanese

This section describes the experiments we conducted to evaluate the answer ranking models for English-to-Japanese QA. The JAVELIN QA system [58] was used as a testbed for the evaluation.

Data Set

700 questions from the NTCIR 5-6 evaluation were used as a data set. Among them, 300 questions were used to train the Japanese answer extractor and 400 questions were used to evaluate the models. Among 400 test questions, the Japanese extractor found no answers for two questions, and only 166 questions contained at least one correct answer (Table 8.18).

Table 8.18: Performance characteristics of the extractor

#Answer sets	#Answer sets with correct answers	Average size of answer sets	Precision	
			macro	micro
398	166	53.3	0.415	0.043

Results and Analysis

Figure 8.14 compares the average accuracy of the baseline algorithms and the answer ranking models. The results show that there was less performance gain in English-to-Japanese answer selection than the English-to-Chinese case. The best baseline system was `Web validation`, but the combination of `Web validation` with `Filtering` and/or `Clustering` did not improve performance over `Web validation`; this demonstrates that in this case combining multiple strategies is hard. However,

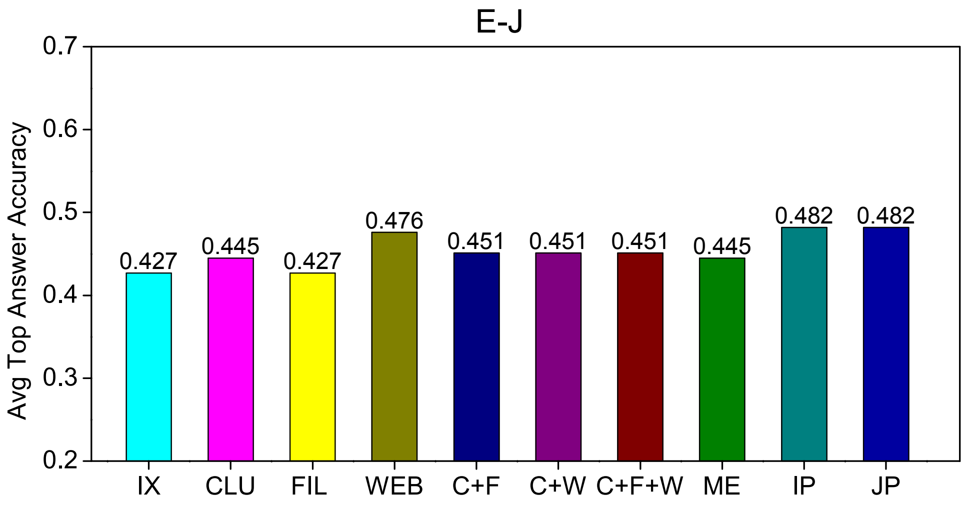


Figure 8.14: Performance of baseline, IP and JP for English-to-Japanese QA

Table 8.19: Average precision of IP and JP for English-to-Japanese QA.

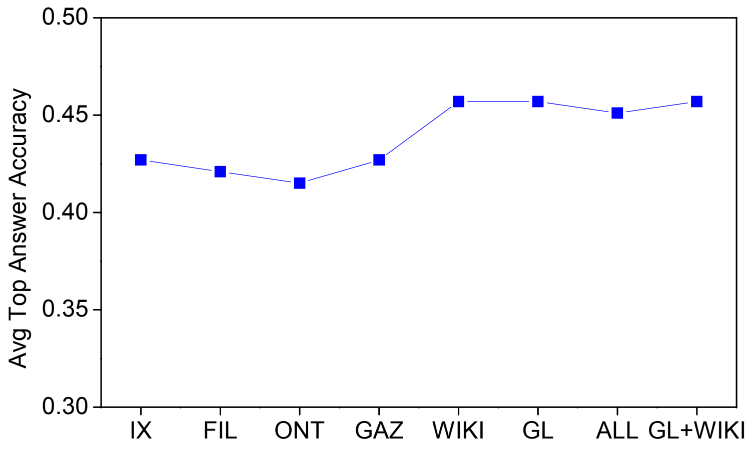
	at rank1	at rank2	at rank3	at rank4	at rank5
IP	0.482	0.277	0.209	0.165	0.140
JP	0.482	0.308	0.226	0.181	0.150

IP and JP were robust and still produced the best performance even though they combined multiple features.

Further analysis was done by comparing the average precision. Table 8.19 compares the average precision of IP and JP when ranking all answer candidates. In English-to-Japanese QA, JP performed better than IP when selecting the top 5 answers.

Figure 8.15 (a) shows the performance of individual relevance features. For English-to-Japanese, the Google and Wikipedia features produced the best performance. Compared with English-to-Chinese, the Wikipedia feature was more useful in answer ranking. However, the combination of all relevance features did not improve performance because the ontology and gazetteer features covered few questions.

Figure 8.15 (b) shows the performance of similarity features under different similarity thresholds. Among the different string similarity features we tested, Levenshtein performed better than other features. Similar to the Japanese monolingual case, we had greater performance gain from the relevance features than from the similarity features.



(a) Relevance features

Similarity feature	0.3	0.5
Cosine	0.421	0.421
Jaccard	0.421	0.421
Jaro	0.427	0.409
JaroWinkler	0.427	0.415
Levenshtein	0.415	**0.445**
Synonyms	0.439	0.439
Cos+Syn	0.39	0.439
Lev+Syn	0.439	**0.445**

(b) Similarity features

Figure 8.15: Average top answer accuracy in English-to-Japanese QA when using (a) individual relevance features and (b) similarity features.

8.4 Multi-strategy QA

This section describes the experiments that we conducted to examine answer merging for multi-strategy QA. The experiments were done with two English QA systems, JAVELIN and EPHYRA.

8.4.1 JAVELIN

In Section 8.2.1, we evaluated the answer ranking models with the answer list produced by each individual JAVELIN extractor: FST, LIGHT and SVM. As different extractors cover different types of questions with varying precision and recall, we merged their answers with the extended answer ranking models (described in Section 6.1). These models use the confidence scores provided by individual extractors as additional features to measure answer relevance.

Experimental Setup

The same data set used to evaluate each individual JAVELIN extractor was used to evaluate the answer ranking models for multi-strategy QA. As different extractors returned different numbers of answer candidates, we only considered the top 50 answer candidates produced by each individual extractor. Among 1760 questions in the data set, there were 978 questions for which at least one extractor identified a correct answer.

The baseline was the result from the independent prediction model, which was run on the answer candidates provided by each individual extractor. As each extractor covers different questions, the performance was measured using answer coverage, calculated by the percentage of correct top answers among the 978 questions for which at least one correct answer exists.

Results and Analysis

Figure 8.16 shows the experimental results. Figure 8.16 (a) is the coverage of the independent prediction model for answer candidates produced by the FST extractor. As can be seen, the model selected the correct top answers only for 27.1% of the questions. Figure 8.16 (b) and (c) shows the coverage of the independent prediction model for answer candidates produced by the LIGHT and SVM extractors, respectively. The independent prediction model identified the correct top answers for 56.1% of the questions in LIGHT; the model selected the correct top answers for 52% of the questions provided by SVM. Comparing these three extractor results, the coverage of FST is small because it can answer only specific types of questions.

Figure 8.16 (d) and (e) show the coverage of answer merging. Answer merging with the independent prediction model improved the coverage by 7.5% over LIGHT alone. Answer merging with the joint prediction model produced the best performance, having a 9.8% improvement over LIGHT alone. The results show that the models are effective in merging answers produced by different extraction strategies.

8.4.2 EPHYRA

In Section 8.2.1, we evaluated the answer ranking models with the answer list produced by each individual EPHYRA extractor: Extractor1 and Extractor2. This section describes the experiments on answer merging in EPHYRA.

Experimental Setup

The same 998 questions used to evaluate each individual EPHYRA extractor were used to evaluate answer merging. Similar to the JAVELIN case, we only considered the top 50 answer candidates produced by each individual extractor. Among the total of 998 questions, only 55.4% of questions (553 questions) had at least one correct answer.

The baseline was the independent prediction model which was run on the answer candidates provided by each individual extractor. As each extractor covers different questions, the performance was measured with the answer coverage.

Results and Analysis

Figure 8.17 shows the experimental results. Figure 8.17 (a) is the coverage of the independent prediction model on the answer candidates produced by Extractor1. It selected the correct top answers for 49.7% of the questions. Extractor2 identified the correct answers for 34% of the questions.

Figure 8.17 (c) and (d) show the coverage of answer merging. Answer merging with the independent prediction model and the joint prediction model improved the coverage by 7.2% over Extractor1 alone. Again, the models were able to improve answer selection performance when merging the two EPHYRA extractor results.

When comparing JAVELIN and EPHYRA, we had greater performance gain when merging JAVELIN extractors because JAVELIN had one more extractor than EPHYRA and the coverage of the LIGHT and SVM extractors was higher than for the EPHYRA extractors.

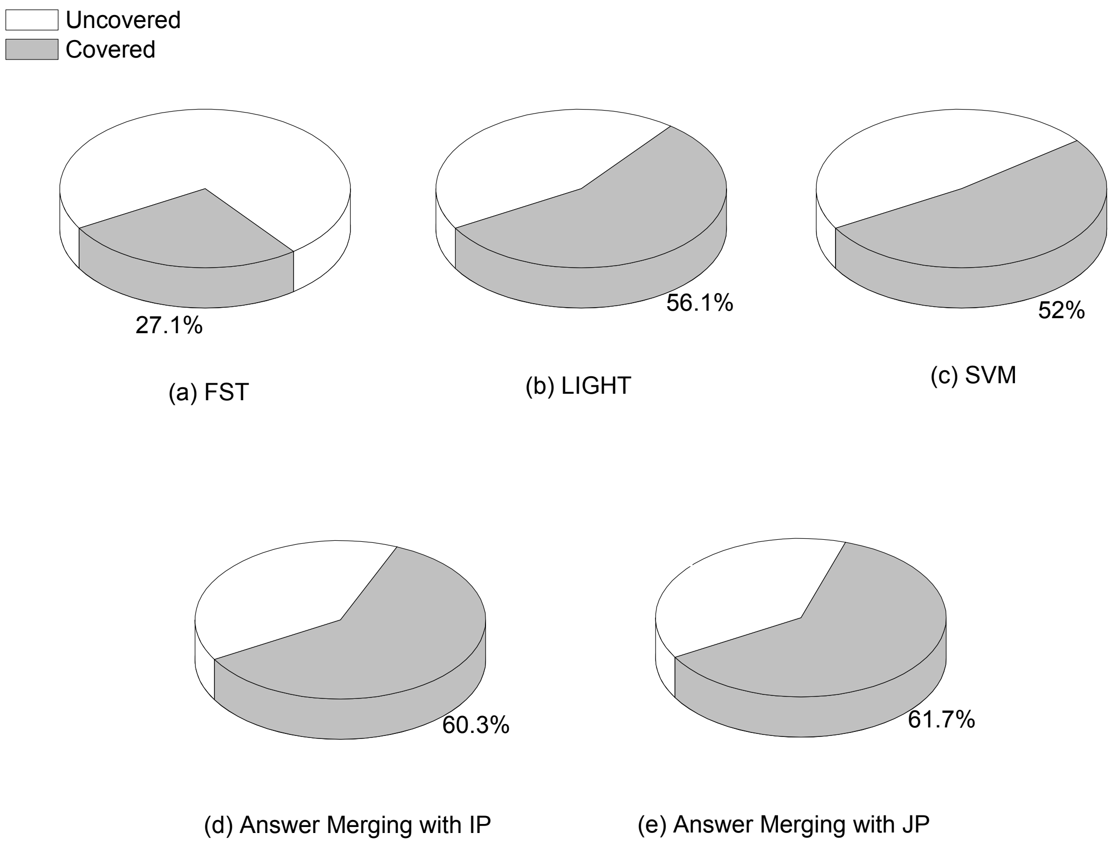


Figure 8.16: Answer merging in JAVELIN. (a),(b) and (c) show the coverage of IP on FST, LIGHT and SVM, respectively. (d) shows the coverage of IP when merging answers from the three extractors. (e) shows the coverage of JP when merging answers from the three extractors.

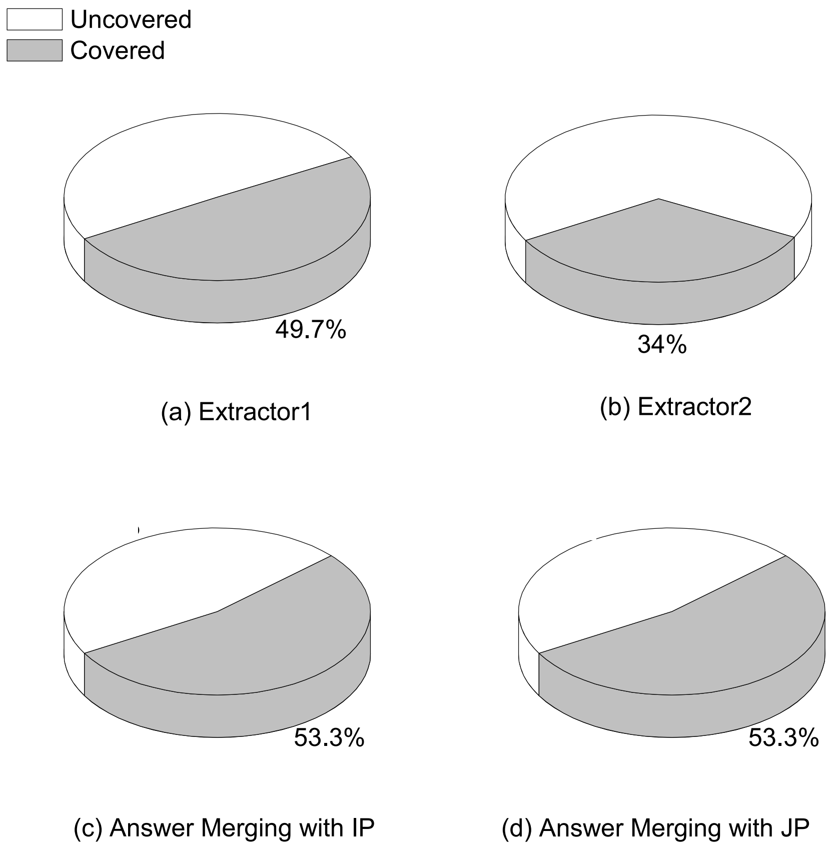


Figure 8.17: Answer merging in EPHYRA. (a) and (b) show the coverage of IP on Extractor1 and Extractor2, respectively. (c) shows the coverage of IP when merging answers from the two extractors. (d) shows the coverage of JP when merging answers from the two extractors.

8.5 Comparison with State-of-the-art Systems

In the previous sections, we evaluated the answer ranking models with cross-validation in order to see how much the models improved the average answer accuracy in one QA system. In this section, we compare the QA systems which incorporate our answer ranking models with state-of-the-art QA systems.

Experimental Setup

Questions from the latest TREC and NTCIR evaluations served as a test set: the latest TREC-2006 evaluation contains 403 English factoid questions, and the latest NTCIR-6 evaluation contains 150 Chinese questions and 200 Japanese 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, 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.

Results and Analysis

Table 8.20 shows the performance of EPHYRA and JAVELIN without and with the independent prediction model for answer selection. It can be seen that JAVELIN and EPHYRA with IP worked much better than the TREC and NTCIR median runs for all languages. As for Japanese (both Japanese-to-Japanese and English-to-Japanese),

Table 8.20: Performance comparison with the latest 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

JAVELIN with IP performed better than the best QA system in NTCIR-6.

8.6 Summary

We conducted a series of experiments to evaluate the performance of our answer ranking models and analyzed the effect of individual relevance and similarity features. We also tested the answer ranking models further by applying them to multi-strategy QA and multilingual QA. Multi-strategy QA tends to have greater answer redundancy, and identifying answer similarity was more important. Multilingual QA includes two tasks: Chinese/Japanese monolingual QA and cross-lingual QA. The former provides a testbed to evaluate to what degree our models are robust in different languages. The latter entails question translation from English to another language and tends to have poor quality data. Applying the models to cross-lingual QA shows the degree to which the models are noise-resistant in supporting data of poor quality.

Table 8.21 summarizes performance gain of the independent prediction model over the baseline systems. We only compared the performance of the independent prediction model because there was no significant difference between IP and JP in selecting the top answer. The performance of the answer ranking model varied according to the characteristics of input quality (e.g. score distribution, degree of answer redundancy, availability of external resources, question distribution, etc).

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.

Table 8.21: Performance gain of IP over baselines and characteristics of testbed systems. (* means the difference is statistically significant (p=0.05)).

System	Improvement over IX	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`)	IX 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`)	

Chapter 9

Conclusions and Future Research

In the previous chapters, we proposed probabilistic answer ranking models and described a series of experiments to evaluate their performance on answer selection in monolingual, cross-lingual & multi-strategy question answering. This chapter summarizes the research results and discusses future research.

9.1 Conclusions

In this thesis, we described our probabilistic answer ranking models for answer selection in question answering. Our hypothesis is that the answer ranking models provide a generalized probabilistic framework that supports answer selection (1) in monolingual QA as well as (2) cross-lingual QA (3) on answer candidates produced by multiple extraction techniques (4) in different question answering systems, and perform better than state-of-the-art answer selection algorithms.

Probabilistic Answer Ranking Framework

As a probabilistic answer ranking framework to estimate the probability of an individual answer candidate, we proposed two answer ranking models: the independent prediction model and the joint prediction model. Both models consider the relevance of individual answers as well as their correlation in order to rank the answer candidates.

The independent prediction model directly estimates the probability of correctness for each answer candidate. It considers two factors: answer relevance and answer similarity. The relevance of an answer to a question 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. Answer similarity is is based on an estimate of the probability $P(correct(A_i)|A_i, A_j)$, where A$_j$ is similar to A$_i$. Since both probabilities influence answer selection performance, it is important to combine them in a unified framework and to estimate the probability of an answer candidate as: $P(correct(A_i)|Q, A_1, ..., A_n)$, where n is the number of answer candidates under consideration. The independent prediction model uses logistic regression to directly estimate this probability.

The joint prediction model estimates the joint probability of correctness of available answer candidates. In particular, the joint prediction model estimates the probability of $P(correct(A_1)$,..., $correct(A_n)|$ $Q, A_1, ..., A_n)$. The marginal probability of $P(correct(A_i)|Q, A_1, ..., A_n)$ for each individual answer as well as the conditional probability $P(correct(A_i)|correct(A_j)$ $, Q, A_1, ..., A_n)$ can be naturally derived from the joint probability. This enables better answer ranking results for a more accurate and comprehensive answer list. An extensive set of empirical results on TREC and NTCIR questions shows that the joint prediction model significantly improved answer ranking performance and was better at finding a unique set of correct answers (e.g. for a list-type question).

As the models provide a unified framework, they can be easily extended to support answer selection in other QA systems. This is different from most answer selection approaches (e.g. IBM's PIQUANT QA system [13] has separate modules to validate answer candidates and to merge answers provided by multiple answering agents, each of which requires a separate training). As our answer ranking models have one unified framework to run answer validation and answer merging, they can be easily integrated into another QA system with only one trainable step using the training data provided by the QA system. To our best understanding, this is the first research work that proposes a formal unified probabilistic framework to model the correctness and correlation of answer candidates in question answering. In addition, this is a novel approach to design a flexible and extensible system architecture for answer selection.

Support of Monolingual QA

As the models are based on a statistical framework, they can be easily extended to support multiple languages by incorporating language-specific features and retraining them for individual languages. We evaluated the models for answer selection in three monolingual QA systems for English, Chinese and Japanese. As English differs markedly from Chinese and Japanese, extending the models to Chinese and Japanese shows that the models are language-independent.

An extensive set of experiments on TREC English questions and NTCIR Chinese and Japanese questions shows that the models significantly improved answer selection performance over the baseline (`IX`) for all three languages. Further experiments on data-driven features show that data-driven features alone achieved significant improvements. This indicates that the models can easily be extended to any language where appropriate data resources are available, even if knowledge-based features and resources for the language are still under development.

As the models were evaluated in three different languages, the utility of individual features can be compared. In English QA (JAVELIN), the combination of relevance and similarity features improved performance by an average of 102% over the baseline `IX`. The relevance features improved performance by an average of 67.5% over the baseline, and the similarity features improved performance by an average of 33.4% over the baseline. On the other hand, in Chinese QA, the relevance features improved performance by 29% and the similarity features improved performance by 63.5% over the baseline. In Chinese, answer relevance features played less important roles than in English because there are fewer Chinese resources suitable for identifying answer relevance. Due to the smaller coverage in available Chinese resources, the answer relevance features had less impact on answer ranking in the Chinese case. As there are more available resources in Japanese, the answer relevance features produced a more performance gain than in Chinese, but the gain was smaller than in the English case.

On the other hand, similarity features heavily depend on redundancy of answer candidates provided by answer extractors. In JAVELIN, similarity features improved performance because the JAVELIN answer extractors produced many redundant answers extracted from different documents. But in EPHYRA similarity features had much less impact on answer ranking because answer candidates provided by EPHYRA tend to have much less redundancy.

Support of Cross-lingual QA

The models were extended to support cross-lingual QA (CLQA). As CLQA involves question and/or keyterm translation to find answers from the target corpus, the answer candidates tend to contain a lot of noisy data. This can be explained by the following; (1) while there are numerous incorrect answer candidates, there are few correct answer candidates, and (2) correct answers have very low rankings. This causes answer ranking to be more challenging in CLQA, as it requires an additional degree of robustness in answer selection. Applying the models to cross-lingual QA shows the degree to which the models are noise-resistant in supporting low quality data.

To support CLQA, we extended the data-driven features used by monolingual answer ranking, while reusing the knowledge-based features. This extension was to incorporate multiple translated keyterms into answer relevance features. As more than one translation candidate might exist for each keyterm, multiple translation candidates were used to search for Web and Wikipedia documents. As some inaccurate translations might retrieve incorrect information, only the top 3 keyterms were used to search for data-driven features. In addition, we used English proper noun keyterms to retrieve Web and Wikipedia documents because the translation quality was not high for proper nouns.

The models were evaluated for answer selection in two cross-lingual QA systems: English-to-Chinese and English-to Japanese. The experimental results on NTCIR questions showed that the models improved answer selection performance in both English-to-Chinese and English-to-Japanese.

Support of Multiple Extraction Techniques

The models were evaluated with multiple extraction techniques: (1) an extractor based on finite state transducers that incorporate a set of extraction patterns (both manually created and generalized patterns), (2) an extractor that selects answer candidates using a non-linear distance heuristic between the keywords and an answer candidate, (3) an extractor that uses Support Vector Machines to discriminate between correct and incorrect answers (4) an extractor that uses patterns automatically obtained from question-answer pairs in training data, (5) an extractor that uses answer types to extract associated named entities, and (6) an extractor that uses maximum entropy models with multiple features such as answer type matching, dependency structure matching, and similarity score of predicate argument structures.

A series of experiments showed that the answer ranking models improved answer selection performance on answer candidates provided by different extraction techniques. This is evidence that the models are robust and generalizable for different extraction techniques.

Support of Multiple QA Systems

The models were evaluated using two different question answering systems: JAVELIN and EPHYRA. Even though both are open-domain question answering systems, implementation details differ. One major difference is how a list of answer candidates is produced. JAVELIN directly searches the given corpus (the TREC/AQUAINT corpora for English and the NTCIR corpora for Chinese and Japanese) to extract documents, so that the list of answer candidates contains several redundant or similar answers retrieved from different documents. On the other hand, EPHYRA extracts answer candidates from Web snippets and combines the answer candidates whose surface strings are the same. Then it conducts answer projection to find supporting documents from the TREC/AQUAINT corpus. Therefore, the input to the answer ranking models does not include redundant answers. In addition, they have different answer type hierarchy and different answer candidate score distributions. Nevertheless, a series of experiments show that the models improved answer selection performance for both JAVELIN and EPHYRA.

Support of Answer Merging

Many recent QA systems incorporate multiple answering agents that extract answer candidates from multiple sources with different strategies. The basic idea of this approach is that a combination of similar answers extracted from different sources with different strategies performs better than any individual answering strategy alone. As answer candidates come from different agents with different score distributions, exploiting answer redundancy plays a more important role in answer ranking.

The models were extended to merge answer candidates provided by multiple extraction strategies. Experiments were done to merge three JAVELIN English extractors and merge two EPHYRA extractors. The experimental results show that the models were effective in merging answer candidates in both systems and produced better performance than the case in which used an individual extractor output was used alone.

Combination of Knowledge-based and Data-driven Features

The models use multiple answer relevance and answer similarity features that are appropriate for the language in question. The features generate scores using knowledge-based approaches (e.g. using ontologies and gazetteers) as well as data-driven approaches (e.g exploiting the Web and Wikipedia). The models can easily learn how to combine them using different weights. In addition, the models provide a framework to evaluate the utility of individual features.

9.2 Future Research

This section describes possible directions for future research.

Extension to complex questions

We conducted a series of experiments on factoid questions whose answers are short noun phrases or named entities. The experimental results show that our answer ranking models improve performance on three languages (English, Chinese and Japanese). As a generalized framework, the models should be able to support different types of questions. As recent TREC data set includes complex questions as well as factoid questions, it would be very important to support complex questions in our answer ranking models. Complex questions require longer answers representing facts or relations (e.g., *"What is the relationship between Alan Greenspan and Robert Rubin?"*, *"Who is Bill Clinton?"*). Hence, we cannot reuse the answer relevance and similarity features used for factoid questions, and need to develop different features for answer selection in complex questions.

One simple relevance feature is to check whether or not an answer candidate contains the required question keywords. For example, given the question *"What is the relationship between Egypt and Syria?"*, "Egypt" and "Syria" are chosen as the required keywords by the Question Analyzer. If an answer candidate does not contain the required keywords, it will probably not be a good answer to the question. Therefore, a relevance score for each answer candidate can be calculated by dividing the number of required keywords in each answer candidate by the number of required keywords in the question.

Another feature is predicate structure match [73]. 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 which 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 [92] can be applied. The Novelty track has experimented with finding relevant and novel sentences from a document collection. We can reuse some approaches evaluated in Allan et al. [4] (e.g., the number of new named entities, the number of new words, or language models) as a feature to measure the similarity of two answer candidates. 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.

Inference for joint prediction

The joint prediction model was implemented with exact inference and approximate inference. For exact inference, we used enumeration on the top 10 answers produced by either an extractor or 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 ten correct answers. As a more general approach, we applied Gibbs sampling. However, our experiments show that Gibbs sampling 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 [1, 107] 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) [108] proposed bootstrap-based over-sampling to reduce issues in over-sampling. Applying resampling to the data from the LIGHT and SVM extractors is likely future research. In addition, more experiments should be done to support different exact and approximate inferences such as variable elimination, loopy belief propagation, mean field, etc.

Feature extension

Experimental results show that answer relevance features had a greater impact for English QA than for Chinese and Japanese QA because the quality and coverage of the data resources available for English answer validation are much higher than the quality and coverage of existing resources for Japanese and Chinese. Acquiring more data resources for answer validation in Chinese and Japanese can be done to improve the performance of answer selection. Similarly, adding more data resources to estimate answer relevance and answer similarity is another future work.

Recently, there has been research on supporting a unified ontology search. Federated Ontology Search (FOS) is one system which provides a unified interface to search for several knowledge-bases [77]. It incorporates multiple ontologies such as

CYC [48], SUMO [68], Omega [79], Scone [22], ThoughtTreasure [66] and WordNet [23], and it supports functionality to identify relationships between two English words (e.g., IS-A, PART-OF). FOS can be used as an additional answer relevance feature. In our experiments, we only used gazetteers and WordNet as a knowledge-based approach, but their utility was significantly less than that of data-driven features. Adding FOS as an additional feature can be one future work to improve the utility of knowledge-based features.

Answer Clustering for Interactive QA

In this thesis, we focused on the degree to which the answer ranking models could identify the correct answers in a batch test. But recent TREC focuses on evaluating the ability to support question answering in an interactive mode so that the user can ask a series of questions. In the interactive QA, it is important to display the answers as well as identify the correct answers. To support a better user interface for interactive QA, the answer ranking models can be extended to support hierarchical answer clustering. For example, given the question, ”What missiles does Egypt have?”, we have multiple answers extracted from different document corpora (Figure 9.1). If we can create feature structure for each answer using more sophisticated natural language processing (NLP) for individual languages (e.g. KANTOO [69, 70]), we can generate normalized answers with a feature structure as shown in Figure 9.2. Now we can cluster answers using a weighted feature unification and ontologies. Figure 9.3 shows a hierarchical cluster, whose top cluster is the Missile cluster which represents the semantic type of the question.

Answers from English documents	1. Egypt has a plant to produce improved Scud-B missiles with North Korea. 2. Egypt is now believed to be able, with North Korean assistance, to manufacture Scuds. 3. Egypt purchased 50 Scud-B missiles. 4. The shipments included steel sheets and support equipment, and gave Egypt everything it would need to produce 310-mile-range Scud-C missiles. 5. Egypt and North Korea have reportedly had a licensing agreement to produce Scud-C missiles since the 1980s.
Answers from Japanese documents	1. この時、エジプト軍はイスラエル軍施設にスカッド B を発射した。 2. 北朝鮮はエジプトから ソ連製のスカッド B ミサイルを提供された。 3. 朝鮮民主主義人民共和国はエジプトから 2 基のスカッドミサイルを購入した。 4. 北朝鮮は 1976 年にエジプトから旧ソ連製のスカッド-B ミサイルを導入。 5. エジプトはソ連からスカッド C を提供されていた。
Answers from Chinese documents	1. 朝鲜 1976 年从埃及 引进了飞毛腿 B 型 导弹 2. 刚果共和国和埃及除飞毛腿-B 之外，还购买了飞毛腿-C. 3. 埃及擁有"飛毛腿" 導彈 4. 埃及和 叙利亚向以色列的腹地 发射了 约 28 枚 苏制“飞毛腿 C” 导弹

Figure 9.1: Answer candidates extracted from multiple document corpora for the question “What missiles does Egypt have?’

```
((quantifier 50)
 (name Scud)
 (subtype B)
 (head missile)
 (num plural)
 (string "50 Scud-B
missiles”))
```

```
((quantifier 2)
 (name Scud)
 (head missile)
 (num plural)
 (string "2 基のスカッド
ミサイル”))
```

```
((quantifier 28)
 (name Scud)
 (subtype C)
 (head missile)
 (num plural)
 (made-by Russia)
 (string "约 28 枚 苏制飞
毛腿 C 导弹”))
```

Figure 9.2: Normalized answers for English (left), Japanese (middle) and Chinese (right).

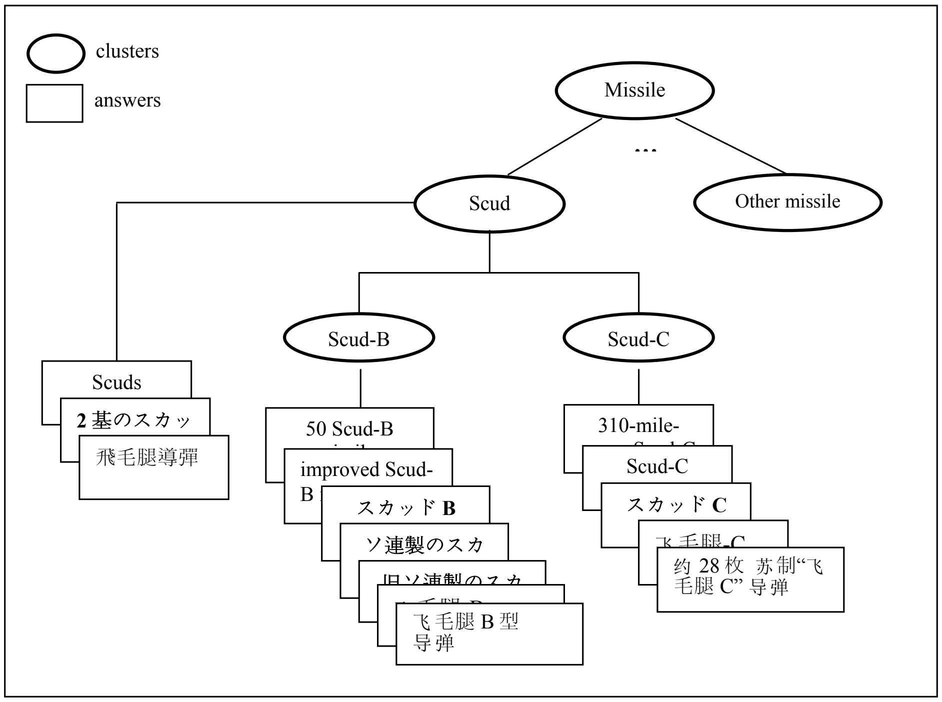


Figure 9.3: Answer clustering for interactive QA.

To support this functionality, we need research on more advanced NLP techniques to create feature structure in individual languages and to support feature unification. In addition, creating ontologies is important to hierarchically cluster answers.

Application to other QA systems and other languages

We evaluated the models with QA systems: JAVELIN and EPHYRA. As the models are based on a probabilistic framework, they can be extended to other QA systems with re-training for each individual system. More specifically, the re-training process should consider the following steps:

- Tuning of relevance features: Our experimental results show that the effect of the relevance features highly depends on the characteristics of the extractor outputs. In addition, some relevance features require manual effort to provide access to language-specific resources and tools (e.g. segmentation for Chinese and Japanese). Therefore, tuning relevance features for a specific system and/or a language is one important task when applying the models to another QA system.

- Tuning of similarity features: As each QA system requires different sets of similarity features and thresholds, it is important to learn the best similarity features and cutoff thresholds for each system.

- Training data acquisition: To retrain the models for another QA system, we need to obtain training data provided by the system. The training data should contain question keywords, answer type information, answer strings and their associated scores.

- Extractor score normalization: Each extractor has a different score distribution. As the extractor score is one of the important features, it is necessary to normalize the extractor scores for consistency.

- Mapping of answer type classification: Each QA system has a different answer type hierarchy. As our answer ranking models highly depend on answer types (e.g., the web feature was less useful in validating temporal and numeric types of questions), we need to convert the answer types of each individual QA system into the answer types which the models use.

In addition, we can extend the model 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 questions and merge their results. Merging has been extensively studied in the context of the information retrieval problems known as metasearch and federated search. In metasearch, multiple search engines are queried simultaneously and the resulting ranked document lists are merged to optimize their combined precision and recall. Many different metasearch merging algorithms have been implemented and evaluated in (Aslam and Montague, 2001; Manmatha and Sever, 2002) [5, 64]. Federated search [91] is useful when the participants work in uncooperative environments. In this environment, it is hard to obtain training data and a semi-supervised merging approach [88, 89] has been proposed to efficiently merge documents by using a small number of documents. 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 models. This is one interesting area of future research.

Bibliography

[1] T. Jo A. Estabrooks and N. Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20:18–36(19), February 2004.

[2] D. Ahn, V. Jijkoun, G. Misha, K. Mller, M. de Rijke, and S. Schlobach. Using Wikipedia at the TREC QA Track. In *Proceedings of the Text REtrieval Conference*, 2004.

[3] K. Ahn, J. Bos, J. R. Curran, D. Kor, M. Nissim, and B. Webber. Question Answering with QED at TREC-2005. In *Proceedings of the Text REtrieval Conference*, 2005.

[4] J. Allan, C. Wade, and A. Bolivar. Retrieval and novelty detection at the sentence level. In *Proceedings of ACM SIGIR Conference on Research and Development on Information Retrieval*, 2003.

[5] J. Aslam and M. Montague. Models for meta-search. In *Proceedings of ACM SIGIR Conference on Research and Development on Information Retrieval*, 2001.

[6] J. Bos and M. Nissim. Cross-Lingual Question Answering by Answer Translation. In *Working Notes for the CLEF 2006*, 2006.

[7] E. Brill, S. Dumais, and M. Banko. An analysis of the askmsr question-answering system. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[8] E. Briscoe and J. Carroll. Robust accurate statistical annotation of general text. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2002.

[9] D. Buscaldi and P. Rosso. Mining Knowledge from Wikipedia for the Question Answering task. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2006.

[10] C. Cardie, D. Pierce V. Ng, and C. Buckley. Examining the Role of Statistical and Linguistic Knowledge Sources in a General-Knowledge Question-Answering System. In *Proceedings of 6th Applied Natural Language Processing Conference and 1st Meeting of the North American Chapter of the Association for Computational Lingusitics*, 2000.

[11] Miguel Á. Carreira-Perpiñ and Geoffrey Hinton. On contrastive divergence learning. In *Proceedings of Artificial Intelligence and Statistics*, pages 33–40, 2005.

[12] J. Chu-Carroll, K. Czuba, J. Prager, and A. Ittycheriah. In question answering, two heads are better than one. In *Proceedings of HLT/NAACL*, 2003.

[13] J. Chu-Carroll, K. Czuba, J. Prager, A. Ittycheriah, and S. Blair-Goldensohn.

IBM's PIQUANT II in TREC2004. In *Proceedings of the Text REtrieval Conference*, 2004.

[14] J. Chu-Carroll, J. Prager, C. Welty, K. Czuba, and D. Ferrucci. A Multi-Strategy and Multi-Source Approach to Question Answering. In *Proceedings of the Text REtrieval Conference*, 2002.

[15] C. Clarke, G. Cormack, and T. Lynam. Exploiting redundancy in question answering. In *Proceedings of ACM SIGIR Conference on Research and Development on Information Retrieval*, 2001.

[16] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Springer, 1999.

[17] H. T. Dang, J. Lin, and D. Kelly. Overview of the TREC 2006 Question Answering Track. In *Proceedings of the Text REtrieval Conference*, 2006.

[18] Eduard Hovy Deepak Ravichandran and Franz Josef Och. Statistical qa - classifier vs. re-ranker: What's the difference? In *in Proceedings of the ACL Workshop on Multilingual Summarization and Question Answering*, 2003.

[19] Zhendong Dong. Hownet: http://www.keenage.com. 2000.

[20] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: Is more always better? In *Proceedings of ACM SIGIR Conference on Research and Development on Information Retrieval*, 2002.

[21] A. Echihabi, U. Hermjakob, E. Hovy, D. Marcu, E. Melz, and D. Ravichandran. Multiple-Engine Question Answering in TextMap. In *Proceedings of the Text REtrieval Conference*, 2003.

[22] S. E. Fahlman. Scone user's manual. http://www.cs.cmu.edu/ sef/scone/, 2005.

[23] C. Fellbaum. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

[24] E. Fox and J. Shaw. Combination of multiple searches. In *Proceedings of the Text REtrieval Conference*, 1994.

[25] S. Guiasu and A. Shenitzer. The principle of maximum entropy. *The Mathematical Intelligencer*, 7(1):43–38, 1985.

[26] S. Harabagiu and F. Lacatusu. Strategies for advanced question answering. In *HLT-NAACL 2004: Workshop on Pragmatics of Question Answering*. Association for Computational Linguistics, 2004.

[27] S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, J. Williams, and J. Bensley. Answer mining by combining extraction techniques with abductive reasoning. In *Proceedings of the Text REtrieval Conference*, 2003.

[28] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunsecu, R. Girju, V. Rus, and P. Morarescu. Falcon: Boosting knowledge for answer engines. In *Proceedings of the Text REtrieval Conference*, 2000.

[29] A. Hickl, J. Williams, J. Bensley, K. Roberts, B. Rink, and Y. Shi. Recognizing Textual Entailment with LCC. In *Proceedings of the Second PASCAL Challenges Workshop*, 2006.

[30] A. Hickl, J. Williams, J. Bensley, K. Roberts, Y. Shi, and B. Rink. Question Answering with LCC's CHAUCER at TREC 2006. In *Proceedings of the Text REtrieval Conference*, 2007.

[31] G. Hinton. Training products of experts by minimizing contrastive divergence, 2000.

[32] G.E. Hinton and T.J. Sejnowski. Learning and relearning in Boltzmann machines. In Rumelhart, editor, *Parallel Distributed Processing*, pages pp. 282–317. MIT Press, 1986.

[33] E. Hovy, L. Gerber, U. Hermjakob, C. Lin, and D. Ravichandran. Toward Semantics-Based Answer Pinpointing. In *Proceedings of the Human Language Technology Conference (HLT). San Diego*, 2001.

[34] E. H. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C. Lin. Question Answering in Webclopedia. In *Proceedings of the Text REtrieval Conference*, 2000.

[35] M. A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[36] M. A. Jaro. Probabilistic linkage of large public health data files. *Statistics in Medicine*, 14(5–7):491–498, 1995.

[37] V. Jijkoun, J. Kamps, G. Mishne, C. Monz, M. de Rijke, S. Schlobach, and O. Tsur. The University of Amsterdam at TREC 2003. In *Proceedings of the Text REtrieval Conference*, 2003.

[38] V. Jijkoun, J. van Rantwijk, D. Ahn, E. Tjong Kim Sang, and M. de Rijke. The University of Amsterdam at CLEF@QA 2006. In *Working Notes CLEF*, 2006.

[39] Michael Jordan, editor. *Learning in Graphical Models.* MIT Press, 1999.

[40] Michael Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence Saul. An introduction to variational methods for graphical models. In Michael Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, Boston, 1998.

[41] Noriko Kando. Overview of the Second NTCIR Workshop. In *Proceedings of the Second NTCIR Workshop Meeting on Evaluation of Chinese and Japanese Text Retrieval and Text Summarization*, 2001.

[42] Noriko Kando. Overview of the Third NTCIR Workshop. In *Working notes of the Third NTCIR Workshop*, 2002.

[43] Boris Katz, Jimmy J. Lin, Daniel Loreto, Wesley Hildebrandt, Matthew Bilotti, Sue Felshin, Aaron Fernandes, Gregory Marton, and Federico Mora. Integrating web-based and corpus-based techniques for question answering. In *Text REtrieval Conference*, pages 426–435, 2003.

[44] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:76–86, 1951.

[45] Julian Kupiec. MURAX: A Robust Linguistic Approach For Question-Answering Using An On-Line Encyclopedia. In *Proceedings of ACM SIGIR Conference on Research and Development on Information Retrieval*, 1993.

[46] C. Kwok, O. Etzioni, and D. Weld. Scaling Question Answering to the Web. In *Proceedings of the Text REtrieval Conference*, 2001.

[47] J. Leidner, J. Bos, T. Dalmas, J. Curran, S. Clark, C. Bannard, B. Webber, and M. Steedman. QED: The Edinburgh TREC-2003 Question Answering System. In *Proceedings of the Twelfth Text Retrieval Conference (TREC 2003)*, 2003.

[48] D.B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):32–38, 1995.

[49] Hang Li, Yunbo Cao, and Conf Li. Using Bilingual Web Data To Mine and Rank Translations. In *Proceedings of IEEE Intelligent Systems 18(4)*, 2003.

[50] F. Lin, H. Shima, M. Wang, and T. Mitamura. CMU JAVELIN System for NTCIR5 CLQA1. In *Proceedings of the 5th NTCIR Workshop*, 2005.

[51] J. Lin and B. Katz. Question answering from the web using knowledge annotation and knowledge mining techniques. In *Proceedings of the Text REtrieval Conference*, 2003.

[52] L.V. Lita. Instance-based Question Answering. Thesis proposal, March 2005.

[53] L.V. Lita, W. Hunt, and E. Nyberg. Resource Analysis for Question Answering. In *Proceedings of ACL*, 2004.

[54] B. Magnini, M. Negri, R. Pervete, and H. Tanev. Comparing statistical and content-based techniques for answer validation on the web. In *Proceedings of the VIII Convegno AI*IA*, 2002.

[55] B. Magnini, M. Negri, R. Pervete, and H. Tanev. Is It the Right Answer? Exploiting Web Redundancy for Answer Validation. In *Proceedings of ACL*, 2002.

[56] T. Minka. A Comparison of Numerical Optimizers for Logistic Regression. Unpublished draft, 2003.

[57] G. Mishne and M. de Rijke. Query formulation for answer projection. In *Proceedings of the European Conference on Information Retrieval*, 2005.

[58] T. Mitamura, F. Lin, H. Shima, M. Wang, J. Ko, J. Betteridge, M. Bilotti, A. Schlaikjer, and E. Nyberg. JAVELIN III: Cross-Lingual Question Answering from Japanese and Chinese Documents. In *Proceedings of the 6th NTCIR Workshop*, 2007.

[59] T. Mitamura, M. Wang, H. Shima, and F. Lin. Keyword Translation Accuracy and Cross-Lingual Question Answering in Chinese and Japanese. In *Proceedings of the EACL Workshop on Multilingual Question Answering*, 2006.

[60] Tom Mitchell. *Machine Learning*, chapter 1 (draft for second edition). McGraw Hill, 1997.

[61] D. Moldovan, D. Clark, S. Harabagiu, and S. Maiorano. Cogex: A logic prover for question answering. In *Proceedings of HLT-NAACL*, 2003.

[62] D. Moldovan, S. Harabagiu, R. Girju, P. Morarescu, F. Lacatusu, A. Novischi, A. Badulescu, and O. Bolohan. LCC tools for Question Answering. In *Proceedings of the 11th Text REtrieval Conference (TREC-2002)*, 2002.

[63] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus. The structure and performance of an open-domain question answering system. In *Proceedings of ACL*, 2000.

[64] M. Montague and J. Aslam. Relevance score normalization for metasearch. In *Proceedings of the ACM Tenth International Conference on Information and Knowledge Management (CIKM)*, 2001.

[65] F. Mora, J. Louis-Rosenberg, G. Marton, and B. Katz. Using Structured, Knowledge-Rich Corpora in Question Answering. In *CSAIL Student Workshop*, 2005.

[66] Erik Mueller. *Natural language processing with ThoughtTreasure*. New York: Signiform, 1998.

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

[68] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS)*, pages 2–9. ACM Press, 2001.

[69] E. Nyberg and T. Mitamura. The KANT System: Fast, Accurate, High-Quality Translation in Practical Domains. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 1992.

[70] E. Nyberg and T. Mitamura. The Kantoo machine translation environment. In *Proceedings of the Association for Machine Translation in the Americas Conference (AMTA)*, 2000.

[71] E. Nyberg, T. Mitamura, J. Callan, J. Carbonell, R. Frederking, K. Collins-Thompson, L. Hiyakumoto, Y. Huang, C. Huttenhower, S. Judy, J. Ko, A. Kupse, L. Lita, V. Pedro, D. Svoboda, and B. Van Durme. A multi-strategy approach with dynamic planning. In *Proceedings of the Text REtrieval Conference*, 2003.

[72] E. Nyberg, T. Mitamura, J. Carbonell, J. Callan, K. Collins-Thompson, K. Czuba, M. Duggan, L. Hiyakumoto, N. Hu, Y. Huang, J. Ko, L. Lita, S. Murtagh, V. Pedro, and D. Svoboda. The JAVELIN Question-Answering System at TREC 2002. In *Proceedings of the Text REtrieval Conference*, 2002.

[73] E. Nyberg, T. Mitamura, R. Frederking, M. Bilotti, K. Hannan, L. Hiyakumoto, J. Ko, F. Lin, V. Pedro, and A. Schlaikjer. JAVELIN I and II Systems at TREC 2005. In *Proceedings of the Text REtrieval Conference*, 2005.

[74] E. Nyberg, T. Mitamura, R. Frederking, V. Pedro, M. Bilotti, A. Schlaikjer, and K. Hannan. Extending the javelin qa system with domain semantics. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, 2005.

[75] M. Pasca and S. Harabagiu. High-Performance Question Answering. In *Proceedings of ACM SIGIR Conference on Research and Development on Information Retrieval*, 2001.

[76] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[77] Vasco Pedro. Federated Ontology Search. Thesis proposal, October 2006.

[78] Vasco Pedro, Jeongwoo Ko, and Eric Nyberg. An Information Repository Model For Advanced Question Answering Systems. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, 2004.

[79] A. Philpot, M. Fleischman, and E. H. Hovy. Semi-Automatic Construction of a General Purpose Ontology. In *Proceedings of the International Lisp Conference*, 2003.

[80] J. Prager, E. Brown, A. Coden, and D. Radev. Question answering by predictive annotation. In *Proceedings of ACM SIGIR Conference on Research and Development on Information Retrieval*, 2000.

[81] J. Prager, J. Chu-Carroll, K. Czuba, C. Welty, A. Ittycheriah, and R. Mahindru. IBM's PIQUANT in TREC2003. In *Proceedings of the Text REtrieval Conference*, 2003.

[82] S.J. Russell and P. Norvig. *Artificial Intelligence: a modern approach*. Prentice-Hall, 1995.

[83] Y. Sasaki, H. Chen, K. Chen, and C. Lin. Overview of the NTCIR-5 Cross-Lingual Question Answering Task (CLQA1). In *Proceedings of NTCIR-5 Workshop*, 2005.

[84] J. Savoy and P-Y Berger. Selection and Merging Strategies for Multilingual Information Retrieval. In *Proceedings of Cross-Language Evaluation Forum*, 2004.

[85] N. Schlaefer, P. Gieselman, and G. Sautter. The Ephyra QA System at TREC 2006. In *Proceedings of the Text REtrieval Conference*, 2006.

[86] S. Schlobach, M. Olsthoorn, and M. de Rijke. Type checking in open-domain question answering. In *Proceedings of European Conference on Artificial Intelligence*, 2004.

[87] H. Shima, M. Wang, F. Lin, and T. Mitamura. Modular Approach to Error Analysis and Evaluation for Multilingual Question Answering. In *Proceedings of International Conference on Language Resources and Evaluation*, 2006.

[88] L. Si and J. Callan. Using sampled data and regression to merge search engine results. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[89] L. Si and J. Callan. A semi-supervised learning method to merge search engine results. *ACM Transactions on Information Systems*, 21(4):457–491, 2003.

[90] L. Si and J. Callan. Clef2005: Multilingual retrieval by combining multiple multilingual ranked lists. In *Proceedings of Cross-Language Evaluation Forum*, 2005.

[91] Luo Si. Federated Search of Text Search Engines in Uncooperative Environments. Thesis, 2006.

[92] I. Soboroff. Overview of the TREC 2004 Novelty Track. In *Proceedings of the Text REtrieval Conference*, 2004.

[93] V Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[94] Vladimir N. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, 1998.

[95] E. Voorhees. The TREC-8 question answering track report. In *Proceedings of the Text REtrieval Conference*, 1999.

[96] E. Voorhees. Overview of the TREC-9 question answering track. In *Proceedings of the Text REtrieval Conference*, 2001.

[97] E. Voorhees. Overview of the TREC 2002 question answering track. In *Proceedings of the Text REtrieval Conference*, 2002.

[98] E. Voorhees. Overview of the TREC 2003 question answering track. In *Proceedings of the Text REtrieval Conference*, 2003.

[99] E. Voorhees. Overview of the TREC 2004 question answering track. In *Proceedings of the Text REtrieval Conference*, 2004.

[100] E. Voorhees. Overview of the TREC 2005 question answering track. In *Proceedings of the Text REtrieval Conference*, 2005.

[101] M. Wang, K. Sagae, and T. Mitamura. A Fast, Accurate Deterministic Parser for Chinese. In *Proceedings of COLING/ACL*, 2006.

[102] W. E. Winkler. The state of record ligrenkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 1999.

[103] J. Xu, A. Licuanan, J. May, S. Miller, and R. Weischedel. TREC 2002 QA at BBN: Answer Selection and Confidence Estimation. In *Proceedings of the Text REtrieval Conference*, 2003.

[104] J. Xu, A. Licuanan, and R. Weischedel. TREC 2003 QA at BBN: Answering Definitional Questions. In *Proceedings of the Text REtrieval Conference*, 2004.

[105] H. Yang, T.S. Chua, S. Wang, and C.K. Koh. Structured use of external knowledge for event-based open domain question answering. In *Proceedings of ACM SIGIR Conference on Research and Development on Information Retrieval*, 2003.

[106] Hui Yang, Hang Cui, Mstislav Maslennikov, Long Qiu, Min-Yen Kan, and Tat-Seng Chua. QUALIFIER In TREC-12 QA Main Task. In *Text REtrieval Conference*, pages 480–488, 2003.

[107] Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Trans. Knowl. Data Eng.*, 18(1):63–77, 2006.

[108] Jingbo Zhu and Eduard Hovy. Active Learning for Word Sense Disambiguation with Methods for Addressing the Class Imbalance Problem. In *Proceedings of ACL*, 2007.