

Combining Evidence with a Probabilistic Framework for Answer Ranking and Answer Merging in Question Answering

Jeongwoo Ko ^{a,*}, Luo Si ^b, Eric Nyberg ^c

^a*Google Inc, Mountain View, CA 94043, USA*

^b*Department of Computer Science, Purdue University,
Lafayette, IN 47907, USA*

^c*School of Computer Science, Carnegie Mellon University,
Pittsburgh, PA 15213, USA*

Abstract

Question answering (QA) aims at finding exact answers to a user's 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 ranking strategy to generate the final answers. This ranking process can be challenging, as it often entails identifying the relevant answers amongst many irrelevant ones. This is more challenging in multi-strategy QA, in which multiple answering agents are used to extract answer candidates. As answer candidates come from different agents with different score distributions, how to merge answer candidates plays an important role in answer ranking. In this paper, we propose a unified probabilistic framework which combines multiple evidence to address challenges in answer ranking and answer merging. The hypotheses of the paper are that (1) the framework effectively combines multiple evidence for identifying answer relevance and their correlation in answer ranking, (2) the framework supports answer merging on answer candidates returned by multiple extraction techniques, (3) the framework can support list questions as well as factoid questions, (4) the framework can be easily applied to a different QA system, and (5) the framework significantly improves performance of a QA system. An extensive set of experiments was done to support our hypotheses and demonstrate the effectiveness of the framework. All of the work substantially extends the preliminary research in (Ko et al., 2007a).

Key words: Answer Ranking, Answer Merging, Question Answering

* Corresponding author. Tel.: +1 650 506 6923; fax: +1 650 506 7224.
Email address: jeongwooko@gmail.com (Jeongwoo Ko).

1 Introduction

Question answering (QA) aims at finding exact answers to a user’s natural language question. For simple questions, direct database lookup has been applied to find answers from large knowledge-bases. In this approach, the question is transformed into a database query and the answer is extracted from the knowledge-bases using the query. However, this approach is not sufficient for many questions because of the limited information in knowledge-bases, and was applied for specific domains. For the past several years, TREC (Text Retrieval Conference) facilitated the development of open-domain question answering to answer more general and arbitrary questions from a large collection of documents. Most QA systems [29,6,13,18] in TREC 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 step to produce a strategy for answering the question, which includes a list of question keywords, alternative forms of the query terms, and the answer type of the question. In recent QA systems which use predicate structure match, question analysis also produces the predicate structure from the question using semantic tools [28,2]. This strategy produced by the question analysis step is used in the document retrieval step to search for relevant documents or passages from a large collection of documents. In the answer extraction step, a list of answer candidates is extracted from the retrieved documents or passages. Finally, answer selection 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 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 irrelevant 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 ($IS-A(\text{Shanghai}, \text{city})$), or to determine if Shanghai is in China ($IS-IN(\text{Shanghai}, \text{China})$).

- *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.

Although many QA systems address these issues separately, there has been little research on generating a probabilistic framework that considers both answer relevance and similarity. In our recent work [17], we proposed a probabilistic framework which uses an undirected graphical model to consider answer relevance and similarity. This framework estimates the joint probability of all answer candidates and then infers the probability of an individual candidate. Even though this joint approach provides a formal framework for answer ranking, it requires $O(2^N)$ time complexity where N is the size of the graph (i.e. number of answer candidates). This paper focuses on a simpler but effective framework which uses logistic regression to estimate 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.

The hypotheses of the paper are that (1) the framework effectively combines multiple evidence for identifying answer relevance and their correlation in answer ranking, (2) the framework supports answer merging on answer candidates returned by multiple extraction techniques, (3) the framework can support list questions as well as factoid questions, (4) the framework can be easily applied to a different QA system, and (5) the framework significantly improves performance of a QA system.

The rest of the paper is arranged as follows: Section 2 describes related work. Section 3 describes our answer ranking framework and Section 4 lists the features used for the framework. In Section 5, we describe the experimental setup. Section 6 describes experimental results on TREC factoid question and Section 7 describes experimental results on list questions. Section 8 discusses the significance of the work. Finally, Section 9 concludes with suggestions for future research.

2 Related Work

To select the most probable answer(s) from the answer candidate list, QA systems have applied several different answer ranking approaches. One of the common approaches is filtering. Filtering relies on precompiled lists or ontologies such as WordNet, CYC and gazetteers to compare the expected answer

type of the question with the type of answer candidates and then remove a candidate whose type does not match the expected answer type [39,26,30,35].

Answer validation is another popular approach for answer ranking. (Xu et al., 2003) applied several type-specific constraints (e.g. constraint to check the subtype for location questions), and moved to the top position those answer candidates that best satisfied the constraints. (Moldovan et al., 2003) converted question and answer candidates into logic representation, and used a logic prover to prove answer correctness using axioms obtained from WordNet. (Magnini et al., 2002) reranked answers according to the validation scores calculated by co-occurrence of question keywords and an answer candidate in the Web text snippets.

Even though each of these approaches uses one or more resources 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., 2004) 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.

Collecting evidence from similar answer candidates to boost the confidence of a specific answer candidate is also important for answer ranking. As answer candidates are extracted from different documents, they may contain identical, similar or complementary text snippets. One approach to exploit redundancy is to incorporate answer clustering [19,25,15,31]. 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. Some simple approaches calculated the cluster scores by counting the number of answers in the cluster [6], summing the scores of all answers in the cluster [19] or selecting the best score among the individual answer scores in the cluster [20].

Exploiting redundancy is even more important in multi-strategy QA, in which multiple answering agents are used [4,9,14,28]. As answer candidates come from different agents with different score distributions, exploiting answer redundancy plays an important role in answer ranking. To exploit this type of redundancy, simple confidence-based voting has been used to merge the top five answers returned by multiple QA agents [4]. As a more advanced approach, a maximum-entropy model has been used to rerank the top 50 answer candidates returned from three different answering strategies [9]. This model

contained more than 30 features generated from the internal QA components (e.g. the rank of answer candidates, 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. 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.

3 Answer Ranking Framework

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 ranking 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 $P(\text{correct}(A_i) | A_i, Q)$, 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 $P(\text{correct}(A_i) | A_i, A_j)$, where A_j is similar to A_i . Since both tasks influence answer selection performance, we believe it is important to combine the two tasks in a unified framework and estimate the probability of an answer candidate $P(\text{correct}(A_i) | Q, A_1, \dots, A_n)$.

Our answer ranking framework estimates the probability $P(\text{correct}(A_i) | Q, A_1, \dots, A_n)$ using multiple answer relevance and similarity features. The framework 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). Logistic regression is a discriminative method that directly models $P(Y|X)$ by learning parameters from training data and has been successfully employed in many applications including multilingual document merging [32,36]. 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 as shown in Equation 1. More details on this equation can be found in our preliminary research work in (Ko et al., 2007a).

$$\begin{aligned}
& P(\text{correct}(A_i)|Q, A_1, \dots, A_n) & (1) \\
& \approx P(\text{correct}(A_i)|\text{rel}_1(A_i), \dots, \text{rel}_{K1}(A_i), \text{sim}_1(A_i), \dots, \text{sim}_{K2}(A_i)) \\
& = \frac{\exp(\alpha_0 + \sum_{k=1}^{K1} \beta_k \text{rel}_k(A_i) + \sum_{k=1}^{K2} \lambda_k \text{sim}_k(A_i))}{1 + \exp(\alpha_0 + \sum_{k=1}^{K1} \beta_k \text{rel}_k(A_i) + \sum_{k=1}^{K2} \lambda_k \text{sim}_k(A_i))} \\
& \text{where, } \text{sim}_k(A_i) = \sum_{j=1(j \neq i)}^N \text{sim}'_k(A_i, A_j).
\end{aligned}$$

In Equation 1, 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 $\text{rel}_k(A_i)$ is a feature function used to produce an answer relevance score for an individual answer candidate A_i . Each $\text{sim}'_k(A_i, A_j)$ is a scoring function used to calculate an answer similarity between A_i and A_j .

Each $\text{sim}_k(A_i)$ 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. In the next Section, we describe the answer relevance and similarity features in detail.

For simplicity, we sum the N-1 answer similarity scores to exploit answer redundancy. But other approaches can be used. For example, if an extractor provides reliable confidence scores for answer candidates, the similarity scores can be calculated by multiplying extractor scores with the string similarity scores.

The parameters $\vec{\alpha}, \vec{\beta}, \vec{\lambda}$ were estimated from training data by maximizing the log likelihood as shown in Equation 2, 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 [23].

$$\vec{\alpha}, \vec{\beta}, \vec{\lambda} = \underset{\vec{\alpha}, \vec{\beta}, \vec{\lambda}}{\text{argmax}} \sum_{j=1}^R \sum_{i=1}^{N_j} \log P(\text{correct}(A_i)|Q, A_1, \dots, A_n) \quad (2)$$

The estimated answer probability is used to rank answer candidates. For factoid questions, the top answer is selected as a final answer to the question. In addition, we can use the estimated probability to classify incorrect answers. For example, if the probability of an answer candidate is lower than 0.5, it is considered to be a wrong answer and is filtered out of the answer list. This can be learned from training data and is useful in deciding whether or not a valid answer to a question exists in a given corpus [37]. The estimated probability can also be used in conjunction with a cutoff threshold when selecting multiple answers to list questions.

3.1 Extension to support answer merging for multi-strategy QA

Many QA systems utilize multiple strategies to extract answer candidates, and then merge the candidates to find the most probable answer [4,9,14,28]. 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.

Our framework 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 3 shows the extended answer ranking framework to merge answer candidates provided by multiple extraction techniques, 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, in Figure 1, the answer extractor returns two instances of “Shanghai” in the answer candidate list: one has a score of 0.64 and the other has score of 0.4. In this case, we ignore 0.4 and use 0.64 as $conf_k$. This is to prevent double counting of redundant answers because $sim_k(A_i)$ already considers this similarity information.

$$\begin{aligned}
 &P(\text{correct}(A_i)|Q, A_1, \dots, A_{m*n}) \tag{3} \\
 &= \frac{\exp(\alpha_0 + \sum_{k=1}^{K1} \beta_k rel_k(A_i) + \sum_{k=1}^{K2} \lambda_k sim_k(A_i) + \sum_{\mathbf{k}=1}^{\mathbf{m}} \gamma_{\mathbf{k}} \mathbf{conf}_{\mathbf{k}})}{1 + \exp(\alpha_0 + \sum_{k=1}^{K1} \beta_k rel_k(A_i) + \sum_{k=1}^{K2} \lambda_k sim_k(A_i) + \sum_{\mathbf{k}=1}^{\mathbf{m}} \gamma_{\mathbf{k}} \mathbf{conf}_{\mathbf{k}})}
 \end{aligned}$$

When extractors do not provide confidence scores, we can use answer ranks instead of scores. If the variance of scores is high, we can apply simple score normalization approaches such as CombSum, CombMNZ[22] as a preprocessing step.

4 Feature Representation

This section presents details of the feature functions and explains how answer relevance scores and answer similarity scores are generated for the answer ranking framework.

4.1 Answer Relevance Features

Each answer relevance feature produces a relevance score to predict whether or not an answer candidate is correct given the question. For factoid and list questions, we used gazetteers and WordNet in a knowledge-based approach; we also used Wikipedia and Google in a data-driven approach. For more complex questions (such as how, why and opinions, etc.), these features are not effective. Extension of relevance features to complex questions is one of future work.

The approaches described in this section use information produced by the question analysis process (e.g., question keywords and the expected answer type). The question analysis process uses semantic resources such as WordNet to produce that information. For example, given the question “Who wrote the book *Song of Solomon?*”, “writer” is the expected answer type by considering normalization of the question focus verb “wrote” [25]. These two approaches are described below.

4.1.1 Knowledge-based Features

a) Gazetteers: For answer ranking, we used three gazetteer resources: the Tipster Gazetteer, information about the US states provided by 50states.com [40] and the CIA World Factbook [41]. These resources were used to assign an answer relevance score between -1 and 1 to each candidate (Figure 2). A score of 1 means the answer is correct given information provided by gazetteers. A score of 0.5 means the answer matches the expected type of the question. Effectively, a score of 0 means the gazetteers did not contribute to the answer ranking process for that candidate. A score of -1 means the answer should be filtered out because it does not match the expected answer type of the question.

For some numeric questions, range checking was added to validate numeric questions in a manner similar to the approach reported in (Prager et al., 2003). 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 taken¹.

b) WordNet: The WordNet lexical database includes English words organized in synonym sets, called *synsets* [11]. We used WordNet in a manner analogous

¹ The ranges used here were found to work effectively, but were not explicitly validated or tuned.

to gazetteers to produce an answer relevance score between -1 and 1. This score was computed for each candidate using the algorithm in Figure 3. As with the gazetteer score, a score of 0 means that WordNet did not contribute to the answer ranking process for a candidate.

4.1.2 Data-driven Features

a) Wikipedia: We used the Wikipedia documents in a data-driven approach using term frequency (TF) and inverse document frequency (IDF). Figure 4 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 each keyword, from which a Wikipedia score is calculated. For example, assume an answer candidate ‘Africa’ for the question “Which continent is Togo on?”. We first find a Wikipedia document about Africa and then check how many times each question keyword (‘continent’ and ‘Togo’) appears in the retrieved document in order to calculate a wikipedia score.

For many numeric and temporal questions, answer candidates tend to be numeric expressions and Wikipedia does not have information related with the numeric expressions. For example, for the question “How many stories are in the Sears Tower?”, we have an answer candidate ‘110’. In Wikipedia, there is no document whose title is ‘110’. In this case, each question keyword is sent to Wikipedia as a back-off strategy, and the answer relevance score is calculated by using tf and idf of the answer candidate. For example, to validate the answer candidate ‘110’, we search for documents for question keywords (‘Sears Tower’ and ‘story’). As the document about ‘Sears Tower’ contains the answer candidate ‘101’, we calculate the Wikipedia score from this document.

To obtain word frequency information, the TREC Web Corpus was used as a large background corpus [42].

b) Web: Following (Magnini et al., 2002), 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. Then the top 10 text snippets returned by Google were analyzed to generate an answer relevance score by computing the minimum number of words between a keyword and the answer candidate (Figure 5).

4.2 Answer Similarity Features

As factoid and list 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.

4.2.1 String Distance Metrics

There are several different string distance metrics which calculate the similarity of two strings. We used Levenshtein distance, Cosine similarity, Jaccard similarity, Jaro and Jaro-Winkler [7]. Each of them can be used as an individual similarity feature to calculate answer similarity scores.

4.2.2 Synonyms

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

$$sim(A_i, A_j) = \begin{cases} 1, & \text{if } A_i \text{ is a synonym of } A_j \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

To obtain a list of synonyms, we used three knowledge-bases: WordNet, Wikipedia and the CIA World Factbook. WordNet includes synonyms for English words. We used the most common senses (i.e., sense 1 and sense 2) to get synonyms from WordNet. Wikipedia redirection is used to obtain another set of synonyms. For example, “Calif.” is redirected to “California” in Wikipedia. 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. These names are considered to be synonyms.

In addition, manually generated rules are used to obtain synonyms for different types of answer candidates [25]. Temporal expressions are converted into the ISO 8601 format (YYYY-MM-DD HH:MM:SS) (e.g., “April 12 1914” and “12th Apr. 1914” are converted into “1914-04-12” and are considered synonyms). Numeric expressions are converted into numbers (e.g, “one million” and “1,000,000” are converted into “1e+06” and are considered synonyms). In addition, we used manually created rules for a representative entity so that it is associated with a specific country when the expected answer type is COUNTRY (e.g., “the Egyptian government” is considered “Egypt” and “Clinton administration” is considered “U.S.”).

5 Experiments: Answer Ranking

This section describes the experiments performed to evaluate our answer ranking framework. The experiments were done with TREC factoid and list questions using two QA systems: JAVELIN [28] and EPHYRA [33]. JAVELIN is the primary test bed for evaluation and EPHYRA is used as the secondary testbed to evaluate the degree to which the framework is applicable to another QA system.

Even though both JAVELIN and EPHYRA 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) 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 clusters the answer candidates whose surface strings are the same. The clustered answer score is calculated by summing the scores in the cluster. Then it conducts answer projection to find supporting documents from the TREC/AQUAINT corpus². Therefore, the input to the answer ranking framework does not include redundant answers and the score is not normalized between 0 and 1. In addition, the two systems have a different answer type hierarchy.

In this section, we first describe the baseline systems and then report the effectiveness of the answer ranking framework in both systems.

5.1 Baseline systems

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

- **Extractor:** Answer extractors apply different techniques to extract answer candidates from the retrieved documents or passages, and assign a confidence score for each individual answer. As a simple baseline, we reranked the answer candidates according to the confidence scores provided by answer extractors.
- **Clustering:** This approach clusters identical or complementary answers and then assigns a new score to each cluster. In our experiments, we used the approach reported in (Nyberg et al., 2003). For a cluster containing

² This is necessary because TREC requires all answers to be extracted from the given corpus

N answers whose extraction confidence scores are S_1, S_2, \dots, S_N , the cluster confidence is computed with the following formula:

$$Score(Answercluster) = 1 - \prod_{i=1}^N (1 - S_i) \quad (5)$$

- **Filtering:** We used both ontologies and gazetteers to filter out improper answer candidates. The algorithms described in Section 4.1.1 were used to identify improper answer candidates, and then these candidates were removed from the answer candidate list.
- **Web validation:** The approach proposed by (Magnini et al., 2002) 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 [12]) and then rerank answer candidates according to the sum, and (3) use a linear regression to learn the weight for both extractor scores and Web scores and then rerank candidates according to the results from the linear regression. In our experiments, the linear regression method was more accurate than the other methods. Therefore, we used linear regression to combine the Web validation scores with the extractor scores.
- **Combination:** We also combined three baseline systems (Clustering, Filtering, and Web validation) using linear regression in order to see the extent to which combined approaches could improve answer ranking performance. Three combinations were tested: Clustering+Filtering(C+F), Clustering+Web(C+W) and Clustering+Filtering+Web (C+F+W).
- **MaxEnt with internal resources:** Maximum entropy has been used for the answer ranking task in several QA systems [10,9] using internal resources such as answer type, frequency of answers in the candidate list, etc. We implemented a maximum entropy reranker as another baseline with popularly used internal features: (1) frequency of answers in the candidate list, (2) answer type matching, (3) question word absent in the answer sentence, (4) inverse term frequency of question keywords in the answer sentence, (5) confidence of individual answer candidate provided by an extractor, and (6) the expected answer type. As this baseline is not using any external resources, the comparison of it with our framework can show the degree to which the external resources are useful in answer ranking.

Performance was measured by average top answer accuracy: the number of correct top answers divided by the number of questions where at least one correct answer exists in the candidate list provided by an extractor. As the performance of answer ranking depends on the quality of answer extraction (when there is no correct answer from extractor, we cannot evaluate the performance of the framework), we only used the number of questions for which an answer extractor finds at least one correct answer in its candidate list. In this way, we can focus on evaluating how much the framework can boost the rank of correct answers.

5-fold cross-validation was performed to evaluate our answer ranking framework. 5-fold cross-validation splits the data set into 5 groups, and uses the first group for testing and the others for training. Then it uses the second group for testing and the others for training. This procedure is repeated 5 times by switching training and test data. Finally the score is calculated by the average of scores produced from each test-training set pair. This is a very popular approach to avoid overfitting to a specific data set when applying machine learning techniques.

As the number of answer candidates affects answer ranking performance, we used only top N answer candidates for the experiments. As EPHYRA produced much more answer candidates than JAVELIN (as shown in Table 1 and Table 6), we used a different threshold for them: top 100 answers for JAVELIN and top 120 answers for EPHYRA.

5.2 Experiments with the JAVELIN QA System

A total of 1760 factoid questions from the TREC8-12 QA evaluations served as a data set. To better understand how the performance of our framework can vary for different extraction techniques, we tested our answer ranking framework with three JAVELIN answer extraction modules: FST, LIGHT and SVM. FST is an answer extractor based on finite state transducers that incorporate a set of extraction patterns (both manually-created and generalized patterns). LIGHT is an extractor that selects answer candidates using a non-linear distance heuristic between the keywords and an answer candidate. SVM is an extractor that uses Support Vector Machines to discriminate between correct and incorrect answers.

Table 1 compares extractor performance on the test questions. The second column shows the number of questions for which each extractor finds answer candidates. For example, FST can find answer candidates only for 837 questions among the 1760 questions. The third column in the table shows the number of questions which contain at least one correct answer in the candidate list. The fourth column shows the average number of answer candidates per question. As can be seen in the third column of Table 1, for many questions they did not find correct answers. Therefore, we only used the questions for which an answer extractor finds at least one correct answer in its candidate list to generate scores. This allows us to evaluate how much the framework can boost the rank of correct answers, which makes sense only when some correct answer(s) exist in the extractor outputs.

Table 2 shows the average top answer accuracy for the baseline systems and the answer ranking framework. The result shows that baseline systems improved

performance over **Extractor**. Among the three extractors, FST produced the best score. Although FST covers fewer questions than LIGHT and SVM, its answers are more accurate than answers from the other extractors. On the other hand, SVM score was lower than FST and LIGHT. As the SVM extractor was not well-tuned and did not have an intelligent way to break ties, it produced many answer candidates with the same confidence score. When evaluating the extractor performance, we just pick the first answer candidate in the extractor output even though its score is same to the second answer. We think this is why the score of the SVM extractor is much lower than the score of the FST and LIGHT extractors.

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 ranking for JAVELIN. As for **MaxEnt**, it improved performance over **Clustering**, but did not gain benefit from the use of external resources such as Web, gazetteers and WordNet because it used only internal resources.

When compared with the baseline systems, the answer ranking framework obtained the best performance gain for all three extractors. The highest gain was achieved for the SVM extractor mostly because SVM often produced multiple answer candidates with the same confidence score, and the framework could select the correct answer by considering additional relevance and similarity features.

Further analysis examined the degree to which the average top answer accuracy was affected by answer similarity features and answer relevance features. Table 3 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 ranking. Relevance features also significantly improved the performance, and the gain was larger than the gain from the similarity features.

When combining both types of features together, the answer selection performance increased for all three extractors. The biggest improvement was found with candidates produced by the SVM extractor: a 247% improvement over **Extractor**.

We also analyzed the average top answer accuracy when using individual features. Table 4 shows the effect of the individual answer relevance feature on different extraction outputs. The combination of all features significantly improved performance compared to answer ranking using a single feature. Comparing data-driven features with knowledge-based features, we note that the

data-driven features (such as Wikipedia and Web) 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 the Web, which provided a performance increase of an average of 74% over **Extractor**.

Table 5 shows the effect of individual similarity features on different extractors. Table 5 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, Cosine 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, Cosine and Jaccard performed well among the five string similarity metrics, we also compared the combination of each of the three metric with synonyms, and then chose Levenshtein and synonyms as the two best similarity features in the answer ranking framework.

5.3 Experiments with the EPHYRA QA system

We also evaluated the answer ranking framework with the answer candidates provided by the EPHYRA QA system. For this evaluation, 998 factoid questions from the TREC13-15 QA evaluations served as a data set.

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 6 shows the characteristics of the EPHYRA extractors.

Table 7 shows the performance of baseline systems and the answer ranking framework on the EPHYRA answer candidates. **Clustering** did not affect performance even though it was useful in the JAVELIN case. As EPHYRA already combined similar answer candidates, there were significantly fewer answers 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`. One interesting result is that `C+F+W` produced lower performance than `Web validation` because each individual method (`Clustering`, `Filtering`, `Web validation`, respectively) sometimes made a conflict decision. For example, given the question “Which city in China has the largest number of foreign financial companies?”, an extractor returns three “Beijing” and two “Shanghai” extracted from 5 different documents. In this case, `Clustering` produces a higher score for “Beijing” because it occurs more than “Shanghai”, but `Web` produces a higher score for “Shanghai” because there is a Web document which mentions Shanghai has largest number of foreign financial companies. This demonstrates that combination of multiple approaches is hard. However, our answer ranking framework made a small but significant improvement over one single baseline even though it merged multiple approaches. When compared with `Extractor`, the framework significantly improved performance.

6 Experiments on Answer Merging

We also evaluated the framework to merge multiple extractor outputs.

6.1 JAVELIN

The same data set used to evaluate each individual JAVELIN extractor was used to evaluate the answer ranking framework 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 and created a new list by combining them. Among 1760 questions in the new list, there were 978 questions for which at least one extractor identified a correct answer.

The performance was measured using the average answer accuracy, calculated by the number of correct top answers among the 978 questions for which at least one correct answer exists. As there are more answer candidates when merging multiple extractor outputs, we also used MRR, which is the average of the reciprocal rank of the top correct answer among the top N (here N=5) candidates.

Two baselines were used: `MaxScore` and `CombSum`. `MaxScore` picks the highest score among the scores provided by the three JAVELIN extractors. `CombSum`

sums the scores from the three extractors and then reranks the answers according to the sum.

Table 8 shows the experimental results of **MaxScore**, **CombSum**, **Framework**. When comparing **MaxScore** with **Combsum**, **CombSum** produced better performance. **Framework** combines answers using the framework. Answer merging with the framework significantly improved both average top answer accuracy and MRR over **CombSum**.

6.2 EPHYRA

We conducted similar experiments to merge answer lists produced by each individual EPHYRA extractor (**Extractor1** and **Extractor2**). The same 998 questions used to evaluate each individual EPHYRA extractor were used to evaluate answer merging. Again, we only considered the top 50 answer candidates produced by each individual extractor and created a new list by combining them. Among the total of 998 questions, 524 questions had at least one correct answer.

Table 9 shows the experimental results. Similar to the JAVELIN case, **Combsum** performed better than **MaxScore** and the framework produced the best performance. When comparing JAVELIN and EPHYRA, we had more 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.

7 Experiments with List Questions

List questions require a list of strings as answer (e.g, “Which countries produce coffee?”). As there are more than one correct answer, F-measure (harmonic mean of precision and recall) is used to evaluate the performance. To answer list questions, most QA systems first produce a list of answers, and then use a cutoff threshold to get top N answers whose score is higher than the threshold. As the performance highly depends on the cutoff threshold, it is hard to setup baseline, and we report the results from the latest TREC 2007 evaluation.

In the TREC 2007 QA track, we applied the answer ranking framework for list questions by merging the results from three extractors: answer type-based extractor, pattern-based extractor and semantic extractor. The first two extractors were used in our previous experiments in Section 6 (**Extractor 1** and **Extractor2**, respectively). The third extractor was recently developed for TREC

2007; it converts the question and answer candidates to semantic structures using ASSERT semantic role labeler and then compares their semantic similarity based on a fuzzy similarity metric. More details on the extractors can be found at (Schlaefter et al., 2008).

Table 10 shows the performance of our system which incorporated the answer ranking framework. It can be seen that our system worked much better than the median over all 51 runs and was the top 5 system.

As this was the first time we used the answer ranking framework for list questions and TREC allowed the participants to submit maximum three runs, we submitted one run without using the framework to answer list questions and another run using the framework. For the former, we applied heuristics which have been popularly used to find the best threshold for list questions. By exploiting the previous TREC list questions as training data, we learned the best threshold and used it to find answers for the TREC 2007 list questions. The performance of this run was 0.123. When comparing this run (score: 0.123) with the run which used the framework (score: 0.144), we achieved 17.1% improvement. This demonstrates the effectiveness of the answer ranking framework in list questions.

8 Discussion

In our experiments, we tested the framework using two QA systems: JAVELIN and EPHYRA. Each QA system is a representative of a different answering strategy. JAVELIN searches the given corpus (the TREC/AQUAINT corpora) to extract answer candidates. EPHYRA exploits larger corpus such as Web to get answer candidates and then conducts answer projection to find supporting documents from the TREC/AQUAINT corpus.

The experimental results demonstrate that the framework improved answer ranking performance in both systems. When comparing the performance gain in JAVELIN and EPHYRA, we had more performance gain in the JAVELIN case because of difference in the number of redundant answers and the variance of extractor scores. Therefore, the utility of individual features was different (e.g., similarity features played an important role in JAVELIN, but not in EPHYRA because answer candidates provided by EPHYRA tend to have many fewer redundancy). This tells that answer ranking performance is inherently system-dependent and applying the framework to other QA systems requires retraining. 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 framework 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 framework for another QA system, we need to obtain training data provided by the QA system. The training data should contain question keywords, answer type information, answer strings and their associated scores. When scores are not available, answer ranks can be used.
- Mapping of answer type classification: Each QA system has a different answer type hierarchy. As our answer ranking framework highly depends 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 framework uses.
- Tuning of the threshold for incorrect answers: Another important issue is how to set a threshold for list questions and how to decide NIL answers for factoid questions[38]. As a default strategy, we can do the binary answer classification using a default threshold of 0.5: if the probability of an answer candidate is lower than 0.5, it is considered to be a wrong answer and is filtered out of the answer list. However, this can be tuned for a specific QA system. In our experiments on list questions, we found that the candidates whose estimated probabilities are at least 25% of the probability of the top answer produced the best performance. It was because the variance of the extractor scores was high (e.g., the score of the top answer candidate was sometimes very low, which resulted in the estimated probabilities below a threshold of 0.5.). Therefore, it is often useful to tune the framework using cross-validation to find the best threshold for a specific system.
- Extractor score normalization: Related with the previous item, when the scores from extractors have a high variance, it is necessary to normalize the extractor scores for answer merging. Some simple score normalization approaches such as CombSum, CombMNZ[22] can be used as a preprocessing step.
- Learning weights for the framework: The weights $\vec{\alpha}, \vec{\beta}, \vec{\lambda}$ should be learned for a specific QA system. As our framework combines answer merging with answer ranking, the weight learning process is much easier. For example, IBM’s PIQUANT QA system [5] has multiple answering agents to extract answer candidates and an answer merger to combine the candidates provided by each answer agent. This requires 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

framework unifies answer validation and answer merging, it can be easily integrated into another QA system with only one trainable step to learn the weights. This is a novel approach to design a flexible and extensible system architecture for answer ranking and answer merging in question answering.

9 Conclusions and Future Work

In this paper, we proposed a answer ranking framework which combines answer relevance and similarity features. An extensive set of empirical results on TREC factoid and list questions show that the framework significantly improved answer ranking performance over extractor results and always produced better performance than other answer ranking algorithms.

Five different extractors were used for the evaluation: (1) an extractor based on finite state transducers, (2) an extractor that selects answer candidates using non-linear distance heuristics, (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, and (5) an extractor that uses answer types to extract associated named entities. Our experimental results show that the answer ranking framework improved answer ranking performance on answer candidates provided by different extraction techniques. This is evidence that our framework is robust and generalizable for different extraction techniques.

Furthermore, the framework was 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 framework was also effective in merging answer candidates in both systems.

We plan to extend the framework to support complex questions, which require longer answers representing facts or relations (e.g., “*What is the relationship between Alan Greenspan and Robert Rubin?*”). As answer candidates are long text snippets, different features should be used for answer ranking. Possible relevance features include question keyword inclusion and predicate structure match[27]. 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, we intend to explore novelty detection approaches evaluated in (Allan et al., 2003). Incorporating more resources to estimate answer relevance and answer similarity is another future work. We also plan to merge the results from answers provided by several QA

systems in order to have higher quality answers.

Acknowledgments

This work was supported in part by ARDA/DTO Advanced Question Answering for Intelligence (AQUAINT) program award number NBCHC040164.

References

- [1] J. Allan, C. Wade, and A. Bolivar. (2003). Retrieval and novelty detection at the sentence level. Proceedings of SIGIR.
- [2] M. Bilotti, P. Ogilvie, J. Callan. and E. Nyberg. (2007). Structured Retrieval for Question Answering. Proceedings of SIGIR.
- [3] D. Buscaldi and P. Rosso. (2006). Mining Knowledge from Wikipedia for the Question Answering task. Proceedings of the International Conference on Language Resources and Evaluation.
- [4] J. Chu-Carroll, K. Czuba, J. Prager, and A. Ittycheriah.(2003). In question answering, two heads are better than one. Proceedings of HLT/NAACL.
- [5] J. Chu-Carroll, K. Czuba, J. Prager, A. Ittycheriah, and S. Blair-Goldensohn. (2005). IBM's PIQUANT II in TREC2004. Proceedings of the Text REtrieval Conference.
- [6] C. Clarke, G. Cormack, and T. Lynam. (2001). Exploiting redundancy in question answering. Proceedings of SIGIR.
- [7] W. Cohen, P. Ravikumar and S. Fienberg. (2003). A Comparison of String Distance Metrics for Name-Matching Tasks Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web.
- [8] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. (1999). Probabilistic Networks and Expert Systems. Springer.
- [9] A. Echihabi, U. Hermjakob, E. Hovy, D. Marcu, E. Melz, and D. Ravichandran. (2004). How to select an answer string? T. Strzalkowski and S. Harabagiu, editors, Advances in Textual Question Answering. Kluwer, 2004.
- [10] D. Ravichandran, E. Hovy and F. J. Och. (2003). Statistical QA - Classifier vs. Re-ranker: What's the difference? Proceedings of the ACL Workshop on Multilingual Summarization and Question Answering.
- [11] C. Fellbaum. (1998). WordNet: An Electronic Lexical Database. MIT Press.

- [12] E. Fox and J. Shaw (1994). Combination of multiple searches Proceedings of the Text REtrieval Conference.
- [13] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunsecu, R. Girju, V. Rus, and P. Morarescu. (2000). Falcon: Boosting knowledge for answer engines. Proceedings of TREC.
- [14] V. Jijkoun, J. Kamps, G. Mishne, C. Monz, M. de Rijke, S. Schlobach and O. Tsur. (2004). The University of Amsterdam at TREC 2003. Proceedings of the Text REtrieval Conference.
- [15] V. Jijkoun and J. van Rantwijk and D. Ahn and E. Tjong Kim Sang and M. de Rijke (2006). The University of Amsterdam at CLEF@QA 2006. Working Notes CLEF.
- [16] J. Ko, L. Si, and E. Nyberg. (2007a) A Probabilistic Framework for Answer Selection in Question Answering. Proceedings of NAACL/HLT.
- [17] J. Ko, L. Si, and E. Nyberg. (2007b) A Probabilistic Graphical Model for Joint Answer Ranking in Question Answering. Proceedings of SIGIR.
- [18] J. Kupiec. (1993). MURAX: A Robust Linguistic Approach For Question-Answering Using An On-Line Encyclopedia. Proceedings of ACM SIGIR Conference on Research and Development on Information Retrieval.
- [19] C. Kwok, O. Etzioni, and D. Weld. (2001). Scaling Question Answering to the Web. Proceedings of the Text REtrieval Conference.
- [20] J. Lin and B. Katz. (2003). Question answering from the web using knowledge annotation and knowledge mining techniques. Proceedings of the Text REtrieval Conference.
- [21] B. Magnini, M. Negri, R. Pervete, and H. Tanev. (2002). Comparing statistical and content-based techniques for answer validation on the web. Proceedings of the VIII Convegno AI*IA.
- [22] R. Manmatha, H. Sever. (2002). A Formal Approach to Score Normalization for Metasearch Proceedings of HLT.
- [23] T. Minka. (2003). A Comparison of Numerical Optimizers for Logistic Regression. Unpublished draft.
- [24] D. Moldovan, D. Clark, S. Harabagiu, and S. Maiorano. (2003). Cogex: A logic prover for question answering. Proceedings of HLT-NAACL
- [25] 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. (2003). The JAVELIN Question-Answering System at TREC 2002. Proceedings of the Text REtrieval Conference.

- [26] 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. (2004). A multi-strategy approach with dynamic planning. Proceedings of the Text REtrieval Conference.
- [27] E. Nyberg, T. Mitamura, R. Frederking, M. Bilotti, K. Hannan, L. Hiyakumoto, J. Ko, F. Lin, V. Pedro, and A. Schlaikjer. (2006). JAVELIN I and II Systems at TREC 2005. Proceedings of Text REtrieval Conference.
- [28] E. Nyberg, T. Mitamura, R. Frederking, V. Pedro, M. Bilotti, A. Schlaikjer, and K. Hannan. (2005). Extending the javelin qa system with domain semantics. Proceedings of the 20th National Conference on Artificial Intelligence.
- [29] J. Prager, E. Brown, A. Coden, and D. Radev. (2000). Question answering by predictive annotation. Proceedings of SIGIR.
- [30] J. Prager, J. Chu-Carroll, K. Czuba, C. Welty, A. Ittycheriah, and R. Mahindru. (2004). IBM's PIQUANT in TREC2003. Proceedings of Text REtrieval Conference.
- [31] J. Prager, S. Luger, J. Chu-Carroll. (2007). Type Nanotheries: A Framework for Term Comparison. Proceedings of CIKM.
- [32] J. Savoy and P-Y Berger.(2004). Selection and Merging Strategies for Multilingual Information Retrieval. Proceedings of Cross-Language Evaluation Forum.
- [33] N. Schlaefler, P. Gieselmann and G. Sautter. (2007). The Ephyra QA System at TREC 2006. Proceedings of the Text REtrieval Conference.
- [34] N. Schlaefler, J. Ko, J. Betteridge, M. Pathak, E. Nyberg and G. Sautter. (2008). Semantic Extensions of The Ephyra QA System for TREC 2007 Proceedings of the Text REtrieval Conference.
- [35] S. Schlobach, M. Olsthoorn, and M. de Rijke. (2004). Type checking in open-domain question answering. Proceedings of European Conference on Artificial Intelligence.
- [36] L. Si, J. Callan, S. Cetintas and H. Yuan. (2008). An effective and efficient results merging strategy for multilingual information retrieval in federated search environments. Information Retrieval. 11(1) 1-24.
- [37] E. Voorhees. (2003). Overview of the TREC 2002 question answering track. Proceedings of Text REtrieval Conference.
- [38] E. Voorhees. (2004). Overview of the TREC 2003 question answering track. Proceedings of Text REtrieval Conference.
- [39] J. Xu, A. Licuanan, J. May, S. Miller, and R. Weischedel. (2003). TREC 2002 QA at BBN: Answer Selection and Confidence Estimation. Proceedings of Text REtrieval Conference.

[40] <http://www.50states.com>

[41] <https://www.cia.gov/library/publications/the-world-factbook/index.html>

[42] http://ir.dcs.gla.ac.uk/test_collections/wt10g.html

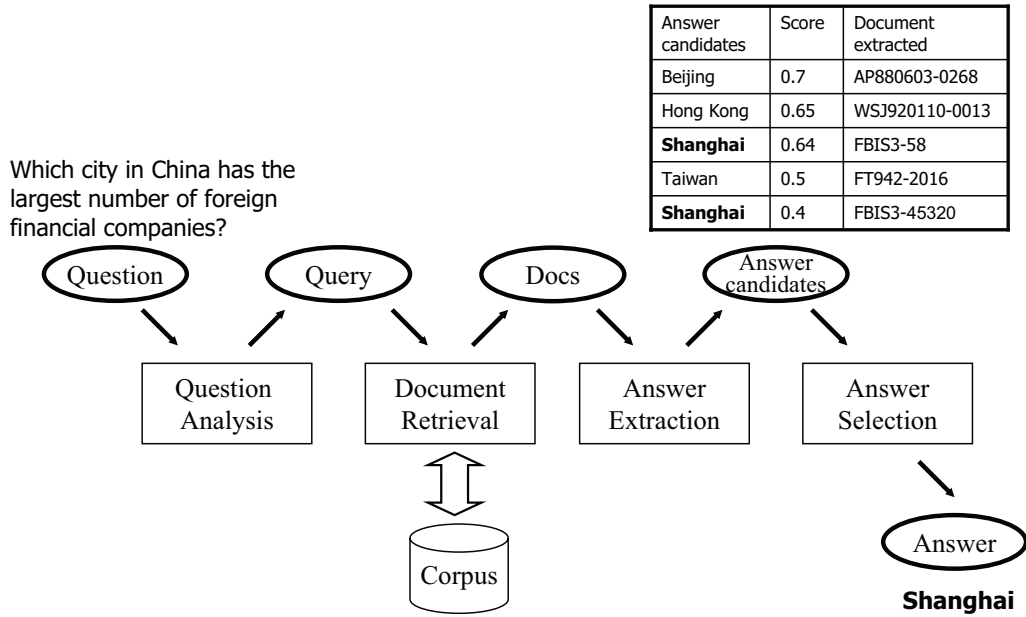


Fig. 1. A traditional QA pipeline architecture

- 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.

Fig. 2. Algorithm to generate an answer relevance score from gazetteers.

- 1) If the answer candidate matches the gloss of the most common senses (sense 1 and sense 2) in 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 because Montevideo is described as ‘capital of Uruguay’ in WordNet).
- 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.

Fig. 3. Algorithm to generate an answer relevance score from WordNet ontology.

Answer extractor	#Questions with any answers	#Questions with correct answers	Avg. size of answer sets
FST	837	301	4.19
LIGHT	1637	889	36.93
SVM	1553	871	38.70

Table 1

Performance characteristics of individual JAVELIN extractors on the 1760 TREC factoid questions.

	EX	CLU	FIL	WEB	C+F	C+W	C+F+W	ME	ARF
FST	0.691	0.774	0.731	0.797	0.787	0.837	0.844	0.797	0.880
LIGHT	0.404	0.456	0.420	0.520	0.462	0.544	0.560	0.467	0.624
SVM	0.282	0.402	0.315	0.489	0.425	0.505	0.519	0.443	0.584

Table 2

Performance of baseline systems and the answer ranking framework in JAVELIN: EX (Extractor), CLU(Clustering), FIL(Filtering), C+F(Clustering+Filtering), C+W(Clustering+Web), C+F+W(Clustering+Filtering+Web), ME (MaxEnt with internal resources), ARF (Answer ranking framework).

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, for each question keyword K_j in the document,

1.1. Compute tf and idf of K_j

1.1. Update the Wikipedia score:

$$ws(A_i) += (1+\log(\text{tf})) \times (1+\log(\text{idf}))$$

2. If not, for each question keyword K_j ,

2.1. Search for a Wikipedia document whose title is K_j

2.2. Calculate tf and idf of A_i in the document

2.3. Update the Wikipedia score:

$$ws(A_i) += (1+\log(\text{tf})) \times (1+\log(\text{idf}))$$

Fig. 4. Algorithm to generate an answer relevance score from Wikipedia (tf: term frequency, idf: inverse document frequency)

1. For each answer candidate A_i , retrieve the top 10 snippets from Google.

2. Initialize the Web score: $ws(A_i) = 0$

3. For each snippet s_i :

3.1. Initialize the snippet co-occurrence score: $cs(s_i) = 1$

3.2. For each question keyword k_j in s_i :

3.2.1. Compute distance d , the minimum number of words between k_j and A_i , excluding stopwords and other keywords

3.2.2. Update the snippet co-occurrence score:

$$cs(s_i) = cs(s_i) \times 2^{\frac{1}{(1+d)}}$$

3.3. Add the snippet score to the web score

4. Normalize the web score by dividing by a constant C

Fig. 5. Algorithm to generate a score from the Web.

	Extractor	Similarity	Relevance	All
FST	0.658	0.751	0.855	0.880
LIGHT	0.394	0.466	0.612	0.624
SVM	0.169	0.420	0.578	0.584

Table 3

Average top answer accuracy of features in the answer ranking framework (Extractor: performance of extractors, Similarity: merging similarity features, Relevance: merging relevance features, ALL: combination of all features)

	Extractor	Gazetteer	WordNet	Wikipedia	Web	ALL
FST	0.658	0.704	0.706	0.773	0.790	0.855
LIGHT	0.394	0.413	0.412	0.478	0.541	0.612
SVM	0.169	0.186	0.196	0.422	0.485	0.578

Table 4

Performance of individual answer relevance features (ALL: combination of all relevance features).

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

Table 5

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

Answer extractor	#Questions with any answers	#Questions with correct answers	Avg size of answers
Extractor 1	813	464	27
Extractor 2	535	305	104

Table 6

Performance characteristics of EPHYRA extractors

	EX	CLU	FIL	WEB	C+F	C+W	C+F+W	ME	ARF
Extractor1	0.508	0.508	0.512	0.577	0.512	0.577	0.575	0.531	0.581
Extractor2	0.497	0.497	0.493	0.597	0.493	0.597	0.587	0.553	0.603

Table 7

Performance of baseline systems and the answer ranking framework in EPHYRA: EX (Extractor), ARF (Answer ranking framework), ME (MaxEnt with internal resources).

	MaxScore	CombSum	Framework
TOP	0.290	0.404	0.603
MRR	0.478	0.598	0.761

Table 8

Answer merging performance in JAVELIN: **MaxScore** picks the highest score among the scores provided by the three JAVELIN extractors and **CombSum** sums the scores from the three extractors. **Framework** merges the results from the extractors using our framework.

	MaxScore	CombSum	Framework
TOP	0.481	0.506	0.542
MRR	0.655	0.678	0.715

Table 9

Answer merging performance in EPHYRA.

TOP1	TOP2	TOP3	TOP4	TOP5 (our system with ARF)	Our system without ARF	Median Score
0.479	0.324	0.147	0.145	0.144	0.123	0.085

Table 10

F-score of the top 5 systems on the TREC-2007 list questions. Our system with ARF was the 5th best system in TREC.