Open Domain Question Answering via Semantic Enrichment

Huan Sun[†][*], Hao Ma[#], Wen-tau Yih[#], Chen-Tse Tsai[+], Jingjing Liu[#], Ming-Wei Chang[#]
huansun@cs.ucsb.edu
{haoma, scottyih, jingjl, minchang}@microsoft.com
ctsai12@illinois.edu
[†]University of California, Santa Barbara
[#]Microsoft Research
[+] University of Illinois at Urbana-Champaign

ABSTRACT

Most recent question answering (QA) systems query large-scale knowledge bases (KBs) to answer a question, after parsing and transforming natural language questions to KBs-executable forms (e.g., *logical forms*). As a well-known fact, KBs are far from complete, so that information required to answer questions may not always exist in KBs. In this paper, we develop a new QA system that mines answers directly from the Web, and meanwhile employs KBs as a significant auxiliary to further boost the QA performance.

Specifically, to the best of our knowledge, we make the first attempt to link answer candidates to entities in Freebase, during answer candidate generation. Several remarkable advantages follow: (1) Redundancy among answer candidates is automatically reduced. (2) The types of an answer candidate can be effortlessly determined by those of its corresponding entity in Freebase. (3) Capitalizing on the rich information about entities in Freebase, we can develop semantic features for each answer candidate after linking them to Freebase. Particularly, we construct answer-type related features with two novel probabilistic models, which directly evaluate the appropriateness of an answer candidate's types under a given question. Overall, such semantic features turn out to play significant roles in determining the true answers from the large answer candidate pool. The experimental results show that across two testing datasets, our QA system achieves an 18% $\sim$ 54% improvement under F_1 metric, compared with various existing QA systems.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval] Retrieval models; I.2.3 [Artificial Intelligence]Answer/reason extraction

Keywords: Question Answering; Knowledge Bases; Web Search

[*]This work was done when the first author was on a summer internship at Microsoft Research.

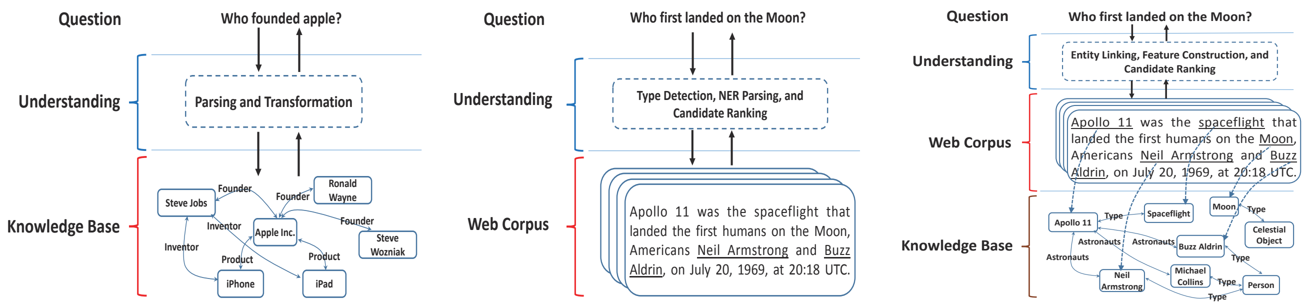
WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741651.

1. INTRODUCTION

Open-domain question answering (QA), which returns exact answers to natural language questions issued by users, is a challenging task and has been advocated as the key problem for advancing web search [15]. Based on the information sources used to find answers, QA systems can be majorly categorized into *knowledge base (KB)-based* and *corpus-based.* A KB-based QA system answers a question by directly querying structured knowledge bases such as Freebase [6], whereas a corpus-based QA system mines answers from an unstructured corpus, such as news articles or other diversified forms of documents available on the Web.

Large-scale knowledge bases, such as Freebase [6], DBpedia [40], YAGO [38], Google's Knowledge Graph and Microsoft's Satori, contain a wealth of valuable information, stored in the form of relation triples, e.g., (*Obama*, *Place-of-Birth*, *Honolulu*). Recent blossom of such large-scale knowledge bases have enabled numerous QA systems to directly extract answers from KBs. For example, Berant et al. [2, 3] develop semantic parsing techniques that map natural language utterances into logical form queries, to execute on a knowledge base. The Paralex system [17] extracts relation tuples from general web corpora via information extraction tools (e.g., ReVerb [16]) and stores them as extracted KBs; during QA, it maps open-domain questions to queries over the extracted KBs. QA systems developed in [18] resort to both curated KBs such as Freebase and extracted KBs from general corpora, to answer a question. Figure 1(a) briefly illustrates the scheme of a KB-based QA system, where a question gets answered by being parsed and transformed to a specific form, suitable to execute on KBs.

However, despite their large size, existing knowledge bases are still far from complete and not updated in a timely fashion [14, 32, 42]. As a result, information required to answer a question may not always exist in KBs. Furthermore, although semantic parsing [2, 3] has been a hot research topic recently, the problem of mapping natural language utterances to logical-form queries is still considered largely unsolved, which limits the practical use of a KB-based QA system. In contrast, interesting or important facts and statements can often appear repeatedly in the rich web corpus including news articles, Wikipedia-like pages, community QA sites, blogs and forums. Driven by this observation of *web redundancy* [7], in this paper, we study the second category of QA systems, i.e., a corpus-based QA system, with the focus on directly mining answers from the Web.


Question
Who founded apple?
Understanding
Parsing and Transformation
Knowledge Base
Steve Jobs
Founder
Founder
Ronald Wayne
Inventor
Apple Inc.
Founder
Steve Wozniak
Inventor
Product
Product
iPhone
iPad
Question
Who first landed on the Moon?
Understanding
Type Detection, NER Parsing, and Candidate Ranking
Web Corpus
Apollo 11 was the spaceflight that landed the first humans on the Moon, Americans Neil Armstrong and Buzz Aldrin, on July 20, 1969, at 20:18 UTC.
Question
Who first landed on the Moon?
Understanding
Entity Linking, Feature Construction, and Candidate Ranking
Web Corpus
Apollo 11 was the spaceflight that landed the first humans on the Moon, Americans Neil Armstrong and Buzz Aldrin, on July 20, 1969, at 20:18 UTC.
Knowledge Base
Apollo 11
Type
Spaceflight
Moon
Type
Celestial Object
Astronauts
Astronauts
Buzz Aldrin
Astronauts
Type
Neil Armstrong
Michael Collins
Type
Person
Type

(a) KB-based QA Systems (b) Web-based QA Systems (c) Our Proposed QuASE

Figure 1: Process Diagrams of Different QA systems

Prior to the blossom of KBs, QA systems are generally viewed as a variation of information retrieval systems. This view can be exemplified by the TREC QA tracks [41], where each participant system is required to extract a small piece of text from a large collection of documents, as the answer to a natural language query. While most QA systems conduct sophisticated NLP analysis and try to integrate syntactic, semantic and pragmatic knowledge for achieving better performance (e.g., [22, 24]), systems like Mulder [26] and AskMSR [7] take a different path by leveraging the *crowd knowledge* from the Web. Without deep natural language analysis, such systems issue simple reformulations of the questions as queries to a search engine, and rank the repeatedly occurring N-grams in the top snippets as answers, based on named entity recognition (NER) [31] and heuristic answer type checking. Despite the simplicity of this strategy, the corpus-based QA systems, particularly, Web-based QA systems, are highly scalable and are among the top performing systems in TREC-10 [8]. A high-level view of such systems is illustrated in Figure 1(b).

One main weakness of such Web-based QA systems is its insufficient knowledge about the generated answer candidates. For instance, different mentions of the same entity such as "*President Obama*" and "*Barack Obama*" are viewed as different answer candidates, and will not be grouped together in most cases. Answer type checking, which verifies whether the type of an answer candidate matches the question, relies on a generic named entity recognition component that provides a small set of crude type labels. As a result, such systems are typically limited to answering questions in only a handful of categories.

To address this issue, we propose a new QA system framework, named **QuASE**, (i.e., question answering via semantic enrichment) in this work. Our system extends the traditional Web-based QA system by linking answer candidates in the search texts to a knowledge base. Figure 1(c) briefly illustrates how our system works, in contrast to the existing KB-based and Web-based QA systems respectively in Figure 1(a) and Figure 1(b). Specifically, given a question, QuASE first selects a set of most prominent sentences from web resources. Then from those sentences, we utilize entity linking tools [13] to detect answer candidates and link them to entities in Freebase. Once each answer candidate is mapped to the corresponding entity in Freebase, abundant information, such as their description texts and Freebase types, can be utilized for feature generation and modeling. A ranking algorithm is subsequently trained based on such features to rank correct answers as top choices.

By incorporating KBs as an important auxiliary in this manner, our system not only maintains the scalability of Web-based QA systems, but also will significantly enhance the QA performance: (1) Redundancy among answer candidates is automatically reduced. In the previous example, "*President Obama*" and "*Barack Obama*" shall be the same answer candidate, as both can be linked to the entity "*Barack Obama*" in Freebase; (2) Entity types stored in KBs can be naturally used to determine the types of an answer candidate. Freebase types vary at the thousands scale, allowing us to deal with more types of questions. (3) KBs contain a wealth of information about entities, which can be adopted for featuring answer candidates. Particularly, we utilize two kinds of information to develop semantic features: (a) description texts of an entity for evaluating whether an answer candidate contextually matches a question, and (b) Freebase entity types of an answer candidate for evaluating their type-matching degree with a question. Specifically, unlike existing QA systems, we build novel probabilistic models to directly evaluate the appropriateness of an answer candidate's Freebase types under a question, and treat the evaluation scores as features for downstream ranking.

Our main contributions in this work are three-fold:

(1) **New QA Framework** – We make the first attempt to incorporate entity linking in QA systems to ground answer candidates on entities in knowledge bases. Then we develop semantic features for each answer candidate based on their rich semantics in KBs, including entity description texts and types, when evaluating their possibility as true answers. Our proposed new architecture is very effective. Compared to existing QA systems, QuASE achieves an 18%∼54% improvement in F_1 and 5% ∼ 20% in MRR, across different datasets.

(2) **Answer Type Checking Models** – In question answering, checking whether an answer candidate meets the expected answer types of a question is a crucial step. We develop two novel probabilistic models to directly evaluate the appropriateness of an answer candidate's types (available in KBs) given a question. To train the models, acquiring large-scale manually labeled data is generally very expensive; here, we propose a creative way to obtain labels from users' implicit behavioral data in query click logs. Our evaluations show that these two models can boost the QA performance by around 2% under all metrics. Combining these two models with other semantic features based on entity description texts, the performance can be boosted by 5%, indicating the advantages of leveraging rich semantics from KBs in question answering.

(3) **Extensive Experimental Evaluation** – In terms of testing data, we not only test our system on the well-known TREC dataset, which consists of well-formed questions, but also build a new testing question set, composed of free-form questions extracted from search engine query logs. Moreover, questions in this new dataset are from real online users instead of editors, which reflects more realistic information need than existing QA datasets. In addition to traditional Web-based QA systems, we also compare our system to the state-of-the-art KB-based QA systems. The results suggest that KB-based QA systems do not perform well, which coincides with our motivation that lots of knowledge is missing from current KBs. This observation also indicates the great importance of the Web-based QA system we are studying in this paper.

The rest of paper is organized as follows: we present our QA system framework in Section 2. Section 3 elaborates the features we have developed for ranking answer candidates. Two types of probabilistic models for extracting anwer-type related features are developed in Section 4 and 5. We conduct detailed experiments to verify the effectiveness of our system in Section 6. Section 7 reviews the related works. We finally conclude this work in Section 8.

2. METHODOLOGY

Figure 2 shows an end-to-end pipeline of our QA framework, which contains the following components in order: (1) Web Sentence Selection via Search Engine; (2) Answer Candidate Generation via Entity Linking; (3) Feature Generation and Ranking. We elaborate the details of each component as follows:

(1) **Web Sentence Selection via Search Engine**. Given a question, in order to find high-quality answer candidates, we design the following mechanism to retrieve highly relevant sentences from the Web that can potentially answer the question. We first submit the question as a query to a commercial search engine, and collect the top-50 returned snippets, as well as the top-50 documents. Since a query itself is generally short and contains only a few words, we compute the word count vector based on the returned snippets to represent the information for the query, denoted as w_q. For each sentence we parsed from the top-50 returned documents, we compute its word count vector w_s, and select those sentences with a high $\cos(w_s, w_q)$ into the high-quality sentence set. If the word vector w_s deviates far from w_q, the corresponding sentence can be regarded as irrelevant and thereby discarded.

(2) **Answer Candidate Generation via Entity Linking**. Once we obtain the sentence set, one of the state-of-the-art entity linking systems [13] is applied to identify answer candidates linked to Freebase. This system achieves the best scores at TAC-KBP 2013, by several novel designs such as postponing surface form boundary detections and discriminating concepts and entities in Wikipedia pages. S-ince the major target of this work is to verify that incorporating rich information from KBs will greatly boost the QA performance, we do not focus on constructing new entity linking tools in this paper. Moreover, for questions whose answers are not entities in Freebase, such as questions starting with "*when*", our system can be reduced to traditional Web-based QA systems without the auxiliary of KBs. In this paper, without loss of generality, we primarily focus on those questions targeted at certain entities in KBs.

(3) **Feature Generation and Ranking**. For each answer candidate, Freebase contains a wealth of information, such as their description texts and entity types. A set of semantic features shall be developed based on such rich information, and subsequently utilized in a ranking algorithm to evaluate the appropriateness of each candidate as the true answer.

Now we use an example to show how our system works. Given a question "*Who was the first American in space?*", we submit it to a search engine to return a set of relevant sentences {1. *On May 5, 1961, Shepard piloted the Freedom 7 mission...* ; 2. *Alan Shepard became the first American in space when the Freedom 7...*; ... }. On this sentence set, we apply entity linking to extract entities , such as *Freedom 7*, *Alan Shepard*, and *Sally Ride*, and link them to Freebase. Such linked entities are treated as answer candidates to the given question. For each answer candidate, semantic features are developed based on their rich information in Freebase, and subsequently integrated into a ranking algorithm, so that the true answer "*Alan Shepard*" will be ranked at the top of the candidate list.

Our QA system distinguishes itself from existing ones, in that it not only mines answers directly from the large-scale web resources, but also employs Freebase as a significant auxiliary to boost the performance. Freebase plays a significant role in both answer candidate generation and feature generation. As discussed in Section 1, by linking answer candidates to Freebase, our system is entitled with several unique advantages, such as reducing the redundancy among answer candidates and effortlessly granting answer candidates with Freebase entity types. Moreover, two kinds of rich information in KBs, entity description texts and entity types, will be naturally utilized to develop semantic features for downstream answer candidate ranking.

3. FEATURE DEVELOPMENT

Upon the generation of an answer candidate pool, effective features shall be developed in order to rank true answer-s as top choices. In this section, we elaborate the features developed for answer candidate ranking. Given a question, totally three categories of features are computed for each answer candidate. The features include both (1) non-semantic features: frequency that an answer candidate occurs in the retrieved sentence set, and (2) semantic features: Since we have linked each answer candidate to Freebase via entity linking, we are able to utilize their rich information in KBs to develop semantic features.

3.1 Count

The sentence set, returned by the sentence selection component, is considered quite related to the given question. The more frequent an answer candidate occurs in the sentence set, the more related it is to the question. Therefore, the *count* or *frequency* of each answer candidate serves as a significant indicator of being the correct answer or not. We compute the count of each answer candidate as one feature.

3.2 Textual Relevance

Given a question, the context where the true answer occurs and its descriptions in KBs should match the question. To evaluate the relevance of an answer candidate to a question, we first extract textual information from both the question side and the answer candidate side.

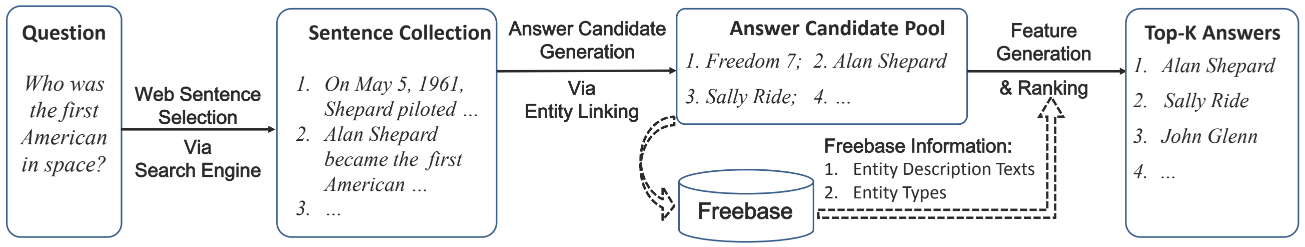


Figure 2: System Framework

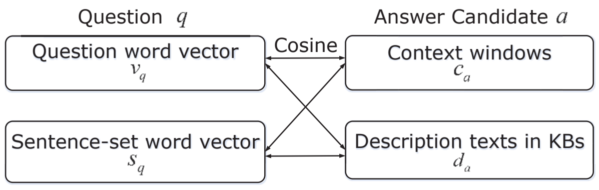

Question q
Answer Candidate a
Question word vector v_q
Cosine
Context windows c_a
Sentence-set word vector s_q
Description texts in KBs d_a

Figure 3: QA Textual Relevance Features

Textual Information on Question Side

(1) The set of words in question q;
(2) The relevant sentence set returned for question q.

Textual Information on Answer Candidate Side

(3) The context windows where an answer candidate a appears in retrieved sentences. Due to the general short length of a sentence, the size of a context window is set at 2, i.e., 2 words before and after answer candidate a compose the context where a occurs in a sentence. We collect all the context windows for each answer candidate in all the sentences.

(4) The description texts of answer candidate a in KBs. For example, Freebase describes entity "*Alan Shepard*", as "*Alan Bartlett 'Al' Shepard, Jr., was an American naval officer and aviator, test pilot, flag officer, one of the original NASA Mercury Seven astronauts ...*".

Based on the textual information for both questions and answer candidates, there can be many methods to measure the matching degree between a question and an answer candidate. We utilize a most intuitive method: For each piece of information (1) to (4), we compute the word frequency vector denoted as v_q, s_q, c_a, and d_a respectively. Then we apply cosine similarity measure between the textual information on the question side and that on the answer candidate side. As shown in Figure 3, we totally compute 4 features based on textual information for each <question, answer candidate> pair. Despite their simplicity, the features turn out to be very effective in improving QA performance. In fact, as future work, more complicated features can be developed and incorporated into our framework, such as deep semantic features learnt via deep learning [23].

3.3 Answer Type Related Features

Given a question "*the first American in space*", in the sentence set we selected, both the entity "*Alan Shepard*" (the astronaut) and the entity "*Freedom 7*" (the spaceflight) occur frequently. Both of them would be textually related to the question measured by features in Section 3.2. Therefore, the above count and textual relevance features turn out to be insufficient for such questions. However, by further checking the expected answer type of the question, it is obvious that the question is looking for a person instead of a spaceflight. Therefore, in order to find correct answers, there is a significant request for us to build answer-type related features, which evaluate the appropriateness of an answer candidate's types under a question.

There have been many works studying the expected answer types of a question [1, 27, 28, 33, 34, 35]. Directly applying their methodology to our setting is not trivial. First, previous type-prediction methods adopt a supervised learning methodology where they classify questions into a small set of types such as person and location. Such methods can hardly scale to thousands of entity types in Freebase, especially when the expected answer of a question is associated with multiple types in Freebase, e.g., entity "*Barack Obama*" associated with multiple types such as "*government.president*", "*people.person*", and "*celebrities.celebrity*". On the other hand, it is quite challenging, if not impossible, to build a mapping between the small set of types and an answer candidate's Freebase types, since Freebase contains thousands of fine-grained types while the types studied in previous methods are quite general and limited. In this paper, we directly handle thousands of Freebase types, and propose probabilistic models to directly measure the matching degree between a question and an answer candidate's Freebase types. The intuition behind such models is that words in a question should correlate with its answer types. Given a question q, we try to model the probability $P(t_a|q)$ or $P(q, t_a)$, where t_a is the set of Freebase types associated with answer candidate a. Answer candidate a with correct types should correspond to a higher $P(t_a|q)$ or $P(q, t_a)$. We will discuss two perspectives to model $P(t_a|q)$ and $P(q, t_a)$ respectively, and build our answer-type related features based on them. In Section 4, we first consider the type predictive power of a single word by modeling $P(t|w)$, where t is a single freebase type and w is a single word. Then different integration models based on $P(t|w)$ are explored, to predict $P(t_a|q)$. In Section 5, we simultaneously capture the joint associations among all the words in q and all the types in t_a, by building a unified generative model for $P(q, t_a)$.

4. WORD TO ANSWER TYPE MODEL

Now we first model the type predictive power of a single word, and then explore different models to integrate the

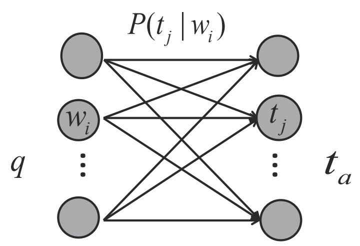


Figure 4: Word to Answer Type (WAT) Model

predictive power of each word in a question. To emphasize the separate consideration of each word in the first step, the underlying methods are named *word to answer type* (WAT) models.

In WAT, we define $P(t|w)$ as the conditional probability of observing Freebase type t as one answer type, if word w occurs in the question. To learn $P(t|w)$, we rely on a training set of questions, each of which is paired with its expected answer types. We will give more details on such training sets in our experiments. Given such a training set, we model $P(t|w)$ as follows:

$$P(t|w) = \frac{\#(w,t)}{\sum_t \#(w,t)} \tag{1}$$

where $\#(w,t)$ represents the co-occurrence frequency of word w and type t in the <question, answer types> pairs. The more frequently a type t co-occurs with a word w, the more likely the answer contains type t, if word w is in the question.

Given a question q with a set of words $\{w_i\}$ and answer candidate a with a set of Freebase types $t_a = \{t_j\}$, Figure 4 shows the point-wise word-to-type conditional probability, between the word set in q and the type set of a. Based on this point-wise conditional probability, the probability $P(t_a|q)$, can be estimated using the following different models.

1. Best Word-to-Type: Choose the highest point-wise conditional probability as $P(t_a|q)$.

$$P(t_a|q) = \max_{w_i \in q, t_j \in t_a} P(t_j|w_i) \tag{2}$$

2. Pivot Word: Choose the word in q that generates the highest productive conditional probability for all the types in t_a

$$P(t_a|q) = \max_{w_i \in q} \prod_{t_j \in t_a} P(t_j|w_i) \tag{3}$$

3. Pivot Word-to-Type: Choose the best set of point-wise conditional probabilities that generate the highest productive conditional probability for all the types in t_a.

$$P(t_a|q) = \prod_{t_j \in t_a} \max_{w_i \in q} P(t_j|w_i) \tag{4}$$

Now we define the WAT feature for answer candidate a based on the perplexity [5] of its type set t_a as:

$$\text{Perplexity}(t_a) = \exp(-\frac{\log(P(t_a|q))}{|t_a|}) \tag{5}$$

Where $|t_a|$ is the number of types in t_a. For Best Word-To-Type, since only one type is considered in the calculation, $|t_a| = 1$.

Perplexity has been applied for the evaluation of different topic models such as LDA [5], whereas in our work we use it as a matching measure between an answer candidate's Freebase types and words in a question. WAT model works based on the assumption that words in a question are predictive of the expected answer types. Based on golden pairs of questions and their expected answer types, we extract the distribution pattern of different types given a specific word in a question. For a new question and one answer candidate, if the answer candidate's types are expected, they should be better explained under the WAT models, i.e., associated with a higher $P(t_a|q)$, than otherwise. Correspondingly, WAT features for answer candidates with expected types should be lower than those with unmatched types. Overall three WAT features can be extracted, with $P(t_a|q)$ respectively instantiated by one of the three proposed models.

5. JOINT <QUESTION, ANSWER TYPE> ASSOCIATION

In this section, we consider the question "How likely can we observe a question and an entity with certain Freebase types, as a question-answer pair?" Different from WAT, we consider the predictive power of all the words simultaneously in a question. Given a question-answer pair, where answer a is associated with multiple Freebase types t_a, we build a generative model of the joint likelihood $P(q, t_a)$, to measure the matching of t_a with the question.

We assume the observation of a question q and its associated answer types t_a, can be explained by latent association patterns. One of such latent associations might be, words such as "*city*", "*place*", "*where*" will occur frequently in a question, if the expected answer type of the question is "*location*" in Freebase. The interplay of multiple latent associations can be captured by a generative model, named joint <question, answer type> association (JQA) model. Figure 5 shows the graphical representation of our generative model for JQA. We first clarify the notations in the figure as follows: (1) $\mathcal{D}$ is the set of <question, answer types> pairs while $|\mathcal{D}|$ denotes the set size. A plate means replicating a process for multiple times. (2) θ_i is the $K \times 1$ mixture weights of K latent association patterns, for the i-th <question, answer types> pair. (3) α, a $K \times 1$ vector, is parameters in a Dirichlet prior, and serves as a constraint of the mixture weights θ_i's. A Dirichlet prior for the mixture weights tends to alleviate over-fitting problems [5]. (4) Z_i is the hidden pattern label that can explain the joint observation of the current question and its answer types. Here we assume all the words in q and types in t_a are generated from the same latent association, since the number of words in a question, together with its associated types, is usually very small. (5) q^i and t_a^i respectively refer to the i-th question and its answer types. (6) β^Q, a $K \times |V^Q|$ matrix, defines the probability distribution over the word vocabulary V^Q, under K hidden patterns. β^T, a $K \times |V^T|$ matrix, defines the probability distribution over the Freebase type vocabulary V^T, under K hidden patterns. (7) The shaded variable w indicates one word observed in q^i while t represents one type in t_a^i.

Figure 5 conveys the generative process of a question and its answer types under multiple latent associations. Now we formally describe the generative process as follows:

For the i-th <question, answer types> pair in $\mathcal{D}$,

– Draw the mixture weights of K hidden association patterns: $\theta_i \sim \mathbf{Dir}(\alpha)$.

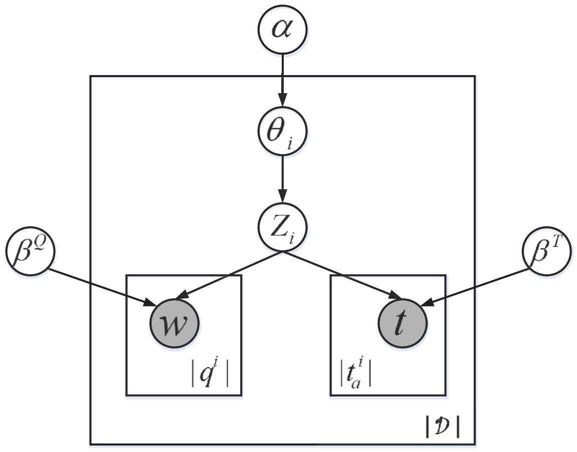


Figure 5: JQA Generative Model

- Draw a pattern label: $Z_i \sim \textbf{Mult}(\theta_i)$.
  - Draw a word for the question q^i from V^Q: $w \sim \beta^Q_{Z_i,:}$.
  - Draw a type for the answer a^i from V^T: $t \sim \beta^T_{Z_i,:}$.

We formulate the likelihood of observing all the question and its associated answer types as follows:

$$\begin{aligned}\mathcal{L} &= \prod_{i\in\mathcal{D}} P(q^i, t_a^i|\alpha, \beta^Q, \beta^T) \\ &= \prod_{i\in\mathcal{D}} \int_{\theta_i} P(\theta_i|\alpha) P(q^i, t_a^i|\theta_i, \beta^Q, \beta^T)\, d\theta_i\end{aligned} \tag{6}$$

Here,

$$\begin{aligned}&P(q^i, t_a^i|\theta_i, \beta^Q, \beta^T) \\ &= \sum_{Z_i} P(Z_i|\theta_i) \prod_{w\in q^i} P(w|Z_i, \beta^Q) \prod_{t\in t_a^i} P(t|Z_i, \beta^T)\end{aligned} \tag{7}$$

Finally, we resort to the maximum likelihood estimation approach to optimize the parameters in the model:

$$\arg\max_{\alpha,\beta^Q,\beta^T} \log \mathcal{L} \tag{8}$$

5.1 Model Solution

5.1.1 Variational Inference

Due to the interdependence of the latent variables, their true posterior distributions are computationally intractable. We introduce a variational distribution Q [4] in which the latent variables are independent of each other to approximate their true posterior distribution, *i.e.*, $Q(\theta, Z) = Q(\theta)Q(Z)$, where $\theta = \{\theta_i, \forall i \in \mathcal{D}\}$ and $Z = \{Z_i, \forall i \in \mathcal{D}\}$. According to the variational distribution, $Q(\theta_i) \sim \textbf{Dir}(\gamma_i)$, $Q(Z_i) \sim \textbf{Mult}(\phi_i)$, where γ_i and ϕ_i are $K \times 1$ variational parameters.

Instead of directly maximizing $\mathcal{L}$ which is intractable, we can maximize the lower bound of the log likelihood under the variational distribution and Jensen's inequality:

$$\log \mathcal{L} \geq E_Q \log P(\mathcal{D}, \theta, Z|\alpha, \beta^T, \beta^Q) + H(Q) = \lfloor \log \mathcal{L} \rfloor \tag{9}$$

where $\mathcal{D}$ denotes all the questions and their associated answer types. We expand the lower bound of the log likelihood as follows:

$$\begin{aligned}\lfloor \log \mathcal{L} \rfloor = &\sum_{i\in\mathcal{D}} E_Q \log P(\theta_i|\alpha) + \sum_{i\in\mathcal{D}} E_Q \log P(Z_i|\theta_i) \\ &+ \sum_{i\in\mathcal{D}} E_Q \log P(w \in q^i, t \in t_a^i|Z_i, \beta^Q, \beta^T) \\ &+ H(Q(\theta, Z))\end{aligned} \tag{10}$$

Each term on the right-hand side of the above equation, is a function over the model parameters as shown in Eqn. 11 to Eqn. 14.

$$\begin{aligned}&\sum_{i\in\mathcal{D}} E_Q \log P(\theta_i|\alpha) \\ &= -|\mathcal{D}| \cdot B(\alpha) + \sum_{i\in\mathcal{D}} \sum_k (\alpha_k - 1)[\psi(\gamma_{i,k}) - \psi(\sum_k \gamma_{i,k})]\end{aligned} \tag{11}$$

where $B(\alpha) = \frac{\prod_k \Gamma(\alpha_k)}{\Gamma(\sum_k \alpha_k)}$ is the normalization constant of the Dirichlet distribution $\textbf{Dir}(\alpha)$.

$$\sum_{\mathcal{D}} E_Q \log P(Z_i|\theta_i) = \sum_{i\in\mathcal{D}} \sum_k \phi_{i,k}[\psi(\gamma_{i,k}) - \psi(\sum_k \gamma_{i,k})] \tag{12}$$

The third term

$$\begin{aligned}&\sum_{i\in\mathcal{D}} E_Q \log P(w \in q^i, t \in t_a^i|Z_i, \beta^Q, \beta^T) \\ &= \sum_{i\in\mathcal{D}} \sum_k \phi_{ik}(\sum_{w\in q^i} N_w^{q^i} \log \beta^Q_{k,w} + \sum_{t\in t_a^i} \log \beta^T_{k,t})\end{aligned} \tag{13}$$

where $N_w^{q^i}$ is the frequency of word w in question q^i. The entropy term

$$\begin{aligned}&H(Q(\theta, Z)) \\ &= -\sum_{i\in\mathcal{D}} E_Q \log Q(\theta_i|\gamma_i) + \sum_{i\in\mathcal{D}} E_Q \log Q(Z_i|\phi_i) \\ &= \sum_{i\in\mathcal{D}} [\log B(\gamma_i) - \sum_k (\gamma_{i,k} - 1)(\psi(\gamma_{i,k}) - \psi(\sum_k \gamma_{i,k}))] \\ &- \sum_{i\in\mathcal{D}} \sum_k \phi_{ik} \log \phi_{ik}\end{aligned} \tag{14}$$

5.1.2 Parameter Estimation

The model parameters are estimated by using the variational expectation-maximization (EM) algorithm. In E-step, we update the variational parameters $\{\gamma\text{'s}, \phi\text{'s}\}$ while in M-step, we update the model parameters α, β^Q, and β^T so that $\lfloor \log \mathcal{L} \rfloor$ is maximized.

Specifically, E-step updates the variational parameters according to Eqn. 15 and 16.

$$\begin{aligned}\phi_{i,k} \sim \exp(&\sum_{w\in q^i} N_w^{q^i} \log \beta^Q_{k,w} + \sum_{t\in t_a^i} \log \beta^T_{k,t} \\ &+ \psi(\gamma_{i,k}) - \psi(\sum_k \gamma_{i,k}) - 1)\end{aligned} \tag{15}$$

$$\gamma_{i,k} = \alpha_k + \phi_{i,k} \tag{16}$$

During M-step, we maximize the lower bound over the parameter α, β^Q, and β^T, by utilizing the classic L-BFGS optimization algorithm [29]. The derivatives over the parameter α are calculated in Eqn. 17.

$$\frac{\partial \lfloor \log \mathcal{L} \rfloor}{\partial \alpha_k} = |\mathcal{D}|[-\psi(\alpha_k) + \psi(\sum_k \alpha_k)] + \sum_{i \in \mathcal{D}} [\psi(\gamma_{i,k}) - \psi(\sum_k \gamma_{i,k})] \quad (17)$$

We solve β^Q and β^T by $\beta^Q_{k,w} \propto \sum_{i\in\mathcal{D}} \phi_{i,k} N^{q^i}_w$ and $\beta^T_{k,t} \propto \sum_{i\in\mathcal{D}} \phi_{i,k}$.

We conduct E-step and M-step iteratively until the algorithm converges, indicating the current model parameters fit the observed training data.

5.2 JQA Feature Extraction

Now we discuss how to apply the learnt JQA model to evaluate the appropriateness of an answer candidate's types *w.r.t* a given question. Given a new question q_{new} and an answer candidate a_{new} with types $t_{a_{\text{new}}}$, we evaluate $P(q_{\text{new}}, t_{a_{\text{new}}}|\alpha, \beta^Q, \beta^T)$ using Eqn. 10. A standard inference procedure is employed to a new <question, answer types> pair [5]: α, β^Q, β^T are fixed as learnt from training data while variational parameters γ and ϕ are specifically estimated for the new pair according to Eqn. 15 and 16.

Similar to WAT, perplexity of observing question q_{new} and answer types $t_{a_{\text{new}}}$ is defined as:

$$\text{Perplexity}(q_{\text{new}}, t_{a_{\text{new}}}) = \exp(-\frac{\log(P(q_{\text{new}}, t_{a_{\text{new}}}|\alpha, \beta^Q, \beta^T))}{|q_{\text{new}}| + |t_{a_{\text{new}}}|})$$

This perplexity is named JQA feature for answer candidate ranking. The rationale behind JQA is similar to WAT. JQA assumes words in a question are associated with its expected answer types. We try to capture such associations by training JQA on golden pairs of questions and their expected answer types.

6. EXPERIMENTS

In this section, we are interested in evaluating QuASE in terms of the following aspects: (1) How do different feature combinations affect QuASE's performance? (2) How does QuASE compare to the state-of-the-art question answering systems? (3) What are the advantages of incorporating rich semantics in KBs into Web-based QA systems? (4) What are the advantages of our answer-type related features JQA and WAT? We further provide a detailed error analysis of different QA systems on questions they fail to answer.

6.1 Experimental Setup

QA Evaluation Datasets

We evaluate different question answering systems on two datasets: TREC questions and Bing queries. Table 1 shows statistics and example questions from each set.

TREC. The Text REtrieval Conference (TREC) had a QA track [41] since 1999. In the competition, editors first prepared some questions, and each participant system then finds answers from a big collection of news articles. TREC data have been publicly available, and become popular benchmarks for evaluating QA systems. We used factoid questions from TREC 8-12 as the TREC dataset in this work. Example questions are listed in Table 1. For questions to which answers are not entities in KBs, such as those starting with "*when*", QuASE can be reduced to traditional Web-based QA systems without incorporating KBs. Without loss of

Datasets	Example Questions
TREC	What are pennies made of?
1700 training	What is the tallest building in Japan?
202 testing	Who sang "Tennessee Waltz"?
Bing query	the highest flying bird
4725 training	indiana jones named after
1164 testing	designer of the golden gate bridge

Table 1: Two Question Sets in Our Experiments

generality, we thus eliminate those questions from the original dataset. Among the remaining 1902 questions, 202 questions from TREC 12 are used for testing and 1700 from TREC 8-11 for training. Although answers to these questions are provided by TREC, they are incomplete or sometimes incorrect for two reasons. First, the provided answers were detected from the given corpus. It is possible that some correct answers do not occur in the corpus, and therefore not included. Second, the correct answers to some questions may have changed, such as "*Who is the prime minister of France?*". In order to have a fair evaluation, the answers were revised using Amazon MTurk (see [39] for detail).

Bing query. Although roughly 10% of the queries submitted to a search engine are with specific informational intent, only one fifth of them are formulated as well-formed questions (e.g., lack of Wh-words) [43]. Bing query dataset in Table 1 shows several such examples. We created Bing query dataset by selecting queries from Bing users: queries are not well-formed questions, but targeted at certain entities in Freebase. Questions in this dataset are from real search engine users and reflect more realistic information need than existing QA benchmark datasets. We crowdsourced each question to at least three experienced human labelers for collecting correct entity answers in Freebase. Once all the labelers reach an agreement on the correct answers, we save the question paired with the correct answers. In the end, we gathered approximately 6000 question-answer pairs in total, from which we randomly select around 20% for testing and 80% for training.

Training Dataset for JQA and WAT

To train JQA and WAT proposed in Section 4 and 5, a sufficiently large training dataset with golden <question, answer types> pairs is indispensable. In reality, we do not have purified data available for training. Instead, we adopt a novel alternative way to obtain labels by joining users' implicit behavioral data in query click logs and the Freebase data. Specifically, we can obtain the <query, clicked url> pairs from the query click logs. Moreover, each entity in Freebase is also linked to some urls that are related to this entity (mostly Wikipedia pages or official web sites of this entity). Hence once a user issued a query and clicked on an entity related url, we can form a <question, answer types> pair: The question is the query given by the user, while we use the Freebase types of the entity corresponding to the clicked url as the answer types. Although such collected dataset is noisy in the sense that the clicked url might not be what the user truly look for, we will show that useful answer-type related features can still be learnt from the large amount of data to benefit the ultimate QA performance. Overall we collect a dataset of around **1.3 million** <question, answer types> pairs based on Bing query logs. We also tried directly using the training portion of each dataset in Table 1 to train the models. However, the training portions contain too limited questions, and features learnt from them can not perform as well as those learnt from the big query log dataset.

Answer Candidate Ranking

For each input question, our question answering pipeline produces an answer candidate pool. To rank these candidates, we first extract all the features discussed in Section 3 ~ Section 5, and map an answer candidate to a feature vector representation *w.r.t* the question. Our ranker then assigns a score to each feature vector and orders the answer candidates accordingly. In our experiments, we use an inhouse fast implementation of the MART gradient boosting decision tree algorithm [9, 21] to learn our ranker using the training set of our data. This algorithm learns an ensemble of regression trees and has shown great performance in various search ranking tasks [10].

Evaluation Measures

We compare different QA systems on each dataset using the following metrics:

(1) Precision, Recall & F_1: As in [18], we treat the top ranked answer candidate as the answer returned to a question. Notice that because the answer candidate pool might be empty, it is possible that no answer is returned by a QA system. As usual, precision and recall are defined as #(*correct answers*)/#(*questions with answers returned*) and #(*correct answers*)/#*questions.* We also compute the F_1 score, which is the harmonic mean of precision and recall.

(2) Mean Reciprocal Rank (MRR). Given a question, Reciprocal Rank (RR) is the reciprocal of the highest ranking position of a correct answer [36]. MRR is the average of the reciprocal ranks over questions:

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{r_i}, \tag{18}$$

where N is the number of questions and r_i is the highest ranking position of an answer to question i. If the true answer to a question is not detected, the RR for the question is 0.

Alternative QA systems

We compare QuASE with existing Web-based and KB-based QA systems. Due to unavailability of most existing QA systems, we select one representative from each category to compare.

(1) Web-based QA system: AskMSR+ [39]. As mentioned briefly in Section 1, early Web-based systems like Mulder [26] and AskMSR [7] have demonstrated that by leveraging the data redundancy in the Web, a simple system can be very competitive and outperform systems conducting sophisticated linguistic analysis of either questions or answer candidates[1]. We compare QuASE with AskMSR+ [39], an advanced version of AskMSR with two main changes. First, instead of reformulating a question according to some statement-like patterns as query terms to a search engine, AskMSR+ issues the question directly, as [39] found out that question reformulation no longer helps retrieve high-quality snippets. This may be due to the better performance of modern search engines, as well as the increased coverage of various community QA sites. Second, instead of using only N-grams extracted from snippets as answer candidates, AskMSR+ requires candidates to be specific types of named entities that can match the question. For instance, only a location entity can be a candidate to the "*where*" questions. With these design changes and other enhancements detailed in [39], AskMSR+ increases MRR by roughly 25% on the TREC dataset, compared to the original AskMSR system, and is thus a solid baseline.

(2) KB-based QA system: Sempre[2, 3]. Sempre[2] and ParaSempre[3] develop semantic parsers to parse natural language questions to logical forms, which are subsequently executed against knowledge bases. They have shown great performance for questions that are directly coined based on relation tuples in KBs. Here we test them on questions that are not necessarily answerable in KBs. The implementation of their systems is publicly available[2], as well as the pretrained systems. We applied the pre-trained systems to our evaluation datasets, and also re-trained the systems on our training datasets. We finally show the best performance we could obtain by these two systems.

6.2 Experimental Results

Results of different feature combinations in QuASE are summarized in Table 2. We have made the following observations: (1) As we discussed in Section 3.1, Count is a significant indicator of the true answer to a question. Although the simplest feature, it performs the best compared with other separated features. Such observation is consistent with the conclusion in [7] that based on data redundancy, simple techniques without complicated linguistic analyses can work well in QA systems. (2) Combined with the Textual Relevance (TR) features, Count+TR can further boost the performance by around 3% under F_1 and MRR, on both datasets. TR can be effective when some wrong answer candidates appear very frequently but their context information does not match the question. For example, given a question "*Who first landed on the moon?*", entity "*American*", which is the nationality of the astronaut, occurs quite frequently in the retrieved texts. However, the textual information related to "*American*" in Freebase can distinguish itself from the true answer, as its description is about American people, rather than people first landing on the moon. (3) Comparing Count+TR+WAT+JQA with Count+TR, our answer-type related features WAT and JQA turn out to be effective: they further improve the performance by around 2% on TREC and by 2% ~ 5% on Bing query across all the metrics. (4) The advantages of incorporating rich semantics in KBs into QuASE can be observed by comparing Count+TR+WAT+JQA with Count+$\text{TR}^{-\text{KBs}}$, where $\text{TR}^{-\text{KBs}}$ denote the two textual relevance features without involving entity description texts from KBs, , i.e., $\cos(v_q, c_a) + \cos(s_q, c_a)$ in Figure 3. With around 5% improvements on both datasets, utilizing rich semantics in KBs can obviously benefit the QA performance.

Table 3 shows the performance of three QA systems. Compared with AskMSR+ system, which also utilized the count of an answer candidate for ranking, the single feature Count in QuASE, as shown in Table 2, performs better by at least 10% in terms of F_1. The potential reason is that through linking answer candidates to KBs, same answer candidates

[1]For instance, AskMSR is one of the top systems in TREC-10 [8].

[2]https://github.com/percyliang/sempre

	Bing query				TREC			
Features in QuASE	Precision	Recall	F_1	MRR	Precision	Recall	F_1	MRR
Count	0.5513	0.5262	0.5384	0.6111	0.5446	0.5446	0.5446	0.6224
TR	0.5243	0.5004	0.5121	0.5880	0.4554	0.4554	0.4554	0.5740
JQA	0.1358	0.1296	0.1326	0.2737	0.1980	0.1980	0.1980	0.3579
WAT	0.1709	0.1631	0.1669	0.3100	0.2624	0.2624	0.2624	0.4049
Count+TR	0.5674	0.5416	0.5512	0.6239	0.5644	0.5644	0.5644	0.6425
Count+TR+WAT	0.5926	0.5657	0.5788	0.6370	0.5693	0.5693	0.5693	0.6513
Count+TR+JQA	0.5764	0.5502	0.5630	0.6296	0.5743	0.5743	0.5743	0.6476
Count+TR+WAT+JQA	**0.5962**	**0.5691**	**0.5823**	**0.6402**	**0.5792**	**0.5792**	**0.5792**	**0.6532**
Count+TR$^{-\text{KBs}}$	0.5638	0.5382	0.5507	0.6187	0.5495	0.5495	0.5495	0.6281

Table 2: Comparison among Different Feature Combinations in QuASE

	Bing query				TREC			
QA Systems	Precision	Recall	F_1	MRR	Precision	Recall	F_1	MRR
QuASE	**0.5962**	**0.5691**	**0.5823**	**0.6402**	**0.5792**	**0.5792**	**0.5792**	**0.6532**
AskMSR+ [39]	0.3782	0.3760	0.3771	0.5337	0.4925	0.4901	0.4913	0.6223
Sempre [2, 3]	0.2646	01940	0.2239	0.2372	0.1567	0.1040	0.1250	0.1437

Table 3: Comparison among Different QA Systems

with different surface forms can be automatically merged, and therefore, redundancy and noise among answer candidates can be significantly reduced. On Bing query, QuASE with all the features can obtain around **54**% improvement on F_1 and **20**% improvement under MRR, while on TREC, it can achieve about **18**% improvement under F_1 and **5**% improvement under MRR. The great improvement further verifies the advantages of employing KBs as an auxiliary into Web-based QA systems. The improvement on F_1 measure is generally higher. MRR takes into account the entire candidate list while F_1 focuses on the first ranking position, implying AskMSR+ has a poor ability to rank the true answer at the top position. We will give more detailed analysis on QuASE and AskMSR+ in Section 6.3. Sempre systems formulate a question into logical forms to be executed against KBs. They are among the state-of-the-art KB-based QA systems on questions that are guaranteed answerable in KBs. However as shown in Table 2, the performance of Sempre is generally much lower than our system under all the measures. Similar performance of Sempre on TREC has also been reported in [18]. There are potentially two reasons why Sempre systems cannot perform well on Bing query and TREC dataset. First, the knowledge required to answer questions might not exist in KBs. For example, given one example question in TREC "*What color is indigo?*", there is no relationship between the true answer "*blue*" and "*indigo*" corresponding to "*alias*" or "*synonym*" although both entities "*blue*" and "*indigo*" exist in Freebase. Second, many questions, especially those not well-formed ones in Bing query, are generally hard to be transformed to logical forms and thereby unsuccessful to obtain answers by executing logical forms against KBs. Since Sempre turns out to be much less effective on both datasets, we will not include it in the following experiments.

To examine the sensitivity of the parameter K in JQA, we conducted 5-fold cross validation on each training set. Figure 6 shows that MRR does not vary too much as we vary K in JQA. The cross-validation result under F_1 observes a similar pattern, which is omitted here due to space constraints. Since larger K implies more numerical computations and larger complexities, we fixed $K = 3$ in all of our experiments.

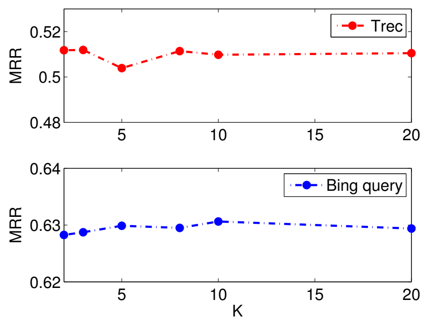


Figure 6: Cross Validation of Different K

6.3 System Comparison on Failed Questions

In this section, we provide detailed analysis of QuASE and AskMSR+ on their respective failed questions. We first define the failed questions for each system: AskMSR+'s failed questions are those where AskMSR+ ranks the true answer lower than QuASE, whereas QuASE's failed questions as those where QuASE ranks the true answer lower than AskMSR+. If there are multiple answers to a question, we consider the highest-ranked answer in defining failed questions.

Table 4 shows the rank distribution of the true answer given by each system on their failed questions. $r = \infty$ means the true answer is not in the candidate list. For those questions where QuASE performs worse than AskMSR+, in most cases, we do not have the answer in the candidate list, i.e., $r = \infty$. However, for those AskMSR+ performs worse, in most cases, AskMSR+ did find the true answer, but ranks it lower than QuASE. This result verifies that compared with AskMSR+, features in QuASE for ranking are effective as long as true answers are in the candidate list.

The distribution of QuASE's rank on failed questions					
Dataset	$r = \infty$	r=2	r=3	r=4	$r \geq 5$
Bing	**63.76%**	12.86%	7.62%	2.86%	11.90%
Trec	**53.57%**	14.29%	12.50%	3.57%	16.07%
The distribution of AskMSR+'s rank on failed questions					
Dataset	$r = \infty$	r=2	r=3	$r = 4$	$r \geq 5$
Bing	13.82%	**27.88%**	21.66%	11.75%	**24.88%**
Trec	19.64%	**35.71%**	14.29%	10.71%	**19.64%**

Table 4: Error Analysis of QuASE and AskMSR+

Due to the imperfectness of any answer candidate generation technique, including entity linking or named entity detection, true answers to many questions might not be detected and included in the candidate list. Therefore, no matter how effective a ranking feature is, the performance on questions without true answers in the candidate list, cannot be boosted. For each method, we define the answerable question set as those questions with true answers included in their candidate list. Table 5 shows the performance of QuASE and AskMSR+ on their respective answerable question sets. Although the answerable questions by QuASE are slightly fewer than those by AskMSR+ system, the performance of QuASE is significantly better than AskMSR+ with a **17% ∼ 38%** improvement on MRR and a **31% ∼ 74%** improvement on F_1. For QuASE, how to improve entity linking performance, in order to include true answers in the candidate list, can be important to further improve QA performance. However the study of that problem is beyond the scope of this paper and we leave it as future work.

	Bing AQS		
Systems	MRR	F_1	|AQS|
QuASE	**0.8583**	**0.7629**	869
AskMSR+ [39]	0.6230	0.4388	998
	Trec AQS		
QuASE	**0.8195**	**0.7267**	161
AskMSR+ [39]	0.7023	0.5530	179

Table 5: Results on Answerable Question Sets

7. RELATED WORK

Our work is mainly related to previous studies in three categories: (1) KB-based QA systems; (2) Web-based QA systems; (3) Answer type prediction.

KB-based QA systems. The advances of large-scale knowledge bases (KBs) have enabled numerous QA systems to directly query KBs. Examples of the most recent KB-based QA systems include [2, 3, 18, 17, 40, 44, 45, 46]. Apart from [2, 3, 18, 17] discussed in Section 1, Unger et al. [40] relies on parsing a question to produce a SPARQL template, which mirrors the internal structure of the question. This template is then instantiated using statistical entity identification and predicate detection. Similarly, Yahya et al. [44] presents a methodology for translating natural language questions into structured SPARQL queries based on an integer linear program. Zou et al. [46] propose to represent a natural language question using a semantic graph query to be matched with a subgraph in KBs and reduce question answering to a subgraph matching problem. Yao et al. [45] proposes to associate questions patterns with answer patterns described by Freebase with the help of a web-scale corpus. Different from such KB-based QA systems, we take an alternative path to question answering by mining answers directly from the rich web corpus, instead of querying either curated KBs such as Freebase or extracted KBs from the Web as in [45].

Web-based QA systems. Prior to the availability and popularity of knowledge bases, most of early QA systems such as [7, 12, 25, 20, 37, 41], mine answers from TREC [41] document collections or the rich web corpus. Brill et al. [7] constructed the AskMSR QA system by utilizing the rich web resources. In [25], Ko et al. focus on a general model to estimate the correctness of answer candidates, instead of developing an entire QA system. Ferrucci et al. [20] gives an overview of IBM Watson system framework including question analysis, search, hypothesis generation, and hypothesis scoring. Most of existing web-based QA systems detect answer candidates via exploring N-grams or named entity recognition. However, we advocate linking answer candidates to Freebase via entity linking, which induces several remarkable advantages. Our system can be regarded as a fundamental and expandable QA system with the new concept to incorporate rich information in KBs into Web-based QA systems.

Answer Type Prediction. In our QA system, we develop features that evaluate the appropriateness of an answer candidate's types under a question. Previous works such as [1, 11, 27, 28, 30, 33, 34, 35] on predicting the expected answer types of a question are related. In [28], Li et al. classify questions into a hierarchy of classes. Question analysis and answer type validation in Watson QA system are detailed in [27, 33]. [34] utilizes WordNet [19] as the answer type taxonomy to do question classification while [1] addresses automatically annotating queries with target types from the DBpedia ontology. [30] develop answer type related features by finding what matters in a question. Researchers in [11, 35] model the answer types as latent variables to avoid the requirement of the predefined type taxonomies. Such methods simultaneously capture the "clustering" information of questions and model the answer relevance conditioned on latent variables. The significant differences between our models for answer-type related features and previous methods have been discussed in Section 3.3. Overall we directly work with thousands of freebase types and propose probabilistic models to measure the matching degree between a question and the Freebase types of an answer candidate.

8. CONCLUSION AND FUTURE WORK

In this paper, we develop a novel Web-based question answering framework with KBs as a significant auxiliary. Driven by the incompleteness problem of KBs, our system directly mines answers from the Web and shows great advantages on questions not necessarily answerable by KBs. Unlike existing Web-based QA systems, our system links answer candidates to KBs during answer candidate generation, after which, rich semantics of entities, such as their description texts and entity types in KBs, are utilized to develop effective features for downstream answer candidate ranking. Compared with various QA systems, our system framework has achieved an 18%∼54% improvement under F_1. As future work, other information reserved in KBs such as various relationships among entities, can also be explored as semantic features to be incorporated in our system.

9. REFERENCES

[1] K. Balog and R. Neumayer. Hierarchical target type identification for entity-oriented queries. In *CIKM*, pages 2391–2394. ACM, 2012.

[2] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on Freebase from question-answer pairs. In *EMNLP*, pages 1533–1544, 2013.

[3] J. Berant and P. Liang. Semantic parsing via paraphrasing. In *ACL*, 2014.

[4] C. Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[5] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[6] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250. ACM, 2008.

[7] E. Brill, S. Dumais, and M. Banko. An analysis of the AskMSR question-answering system. In *EMNLP*, pages 257–264, 2002.

[8] E. Brill, J. J. Lin, M. Banko, S. T. Dumais, and A. Y. Ng. Data-intensive question answering. In *TREC*, 2001.

[9] C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. *Learning*, 11:23–581, 2010.

[10] C. Burges, K. M. Svore, P. N. Bennett, A. Pastusiak, and Q. Wu. Learning to rank using an ensemble of lambda-gradient models. In *Yahoo! Learning to Rank Challenge*, pages 25–35, 2011.

[11] S. Chaturvedi, V. Castelli, R. Florian, R. M. Nallapati, and H. Raghavan. Joint question clustering and relevance prediction for open domain non-factoid question answering. In *WWW*, pages 503–514, 2014.

[12] J. Chu-Carroll, J. Prager, C. Welty, K. Czuba, and D. Ferrucci. A multi-strategy and multi-source approach to question answering. Technical report, DTIC Document, 2006.

[13] S. Cucerzan and A. Sil. The msr systems for entity linking and temporal slot filling at TAC 2013. In *Text Analysis Conference*, 2013.

[14] X. Dong, K. Murphy, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A Web-scale approach to probabilistic knowledge fusion. In *SIGKDD*, pages 601–610, 2014.

[15] O. Etzioni. Search needs a shake-up. *Nature*, 476(7358):25–26, 2011.

[16] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545, 2011.

[17] A. Fader, L. Zettlemoyer, and O. Etzioni. Paraphrase-driven learning for open question answering. In *ACL*, pages 1608–1618, 2013.

[18] A. Fader, L. Zettlemoyer, and O. Etzioni. Open question answering over curated and extracted knowledge bases. In *SIGKDD*. ACM, 2014.

[19] C. Fellbaum. WordNet: An electronic lexical database. 1998. *http://www. cogsci. princeton. edu/wn*, 2010.

[20] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. Murdock, E. Nyberg, J. Prager, et al. Building watson: An overview of the DeepQA project. *AI magazine*, 31(3):59–79, 2010.

[21] J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[22] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. FALCON: Boosting knowledge for answer engines. In *TREC*, volume 9, pages 479–488, 2000.

[23] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[24] E. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C. Lin. Question answering in Webclopedia. In *TREC*, volume 9, 2000.

[25] J. Ko, E. Nyberg, and L. Si. A probabilistic graphical model for joint answer ranking in question answering. In *SIGIR on Rearch and Development in IR*, pages 343–350. ACM, 2007.

[26] C. Kwok, O. Etzioni, and D. Weld. Scaling question answering to the Web. *TOIS*, 19(3):242–262, 2001.

[27] A. Lally, J. Prager, M. McCord, B. Boguraev, S. Patwardhan, J. Fan, P. Fodor, and J. Chu-Carroll. Question analysis: How watson reads a clue. *IBM Journal of Research and Development*, 56(3.4):2–1, 2012.

[28] X. Li and D. Roth. Learning question classifiers. In *ICCL*, pages 1–7, 2002.

[29] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[30] X. Luo, H. Raghavan, V. Castelli, S. Maskey, and R. Florian. Finding what matters in questions. In *HLT-NAACL*, pages 878–887, 2013.

[31] E. Marsh and D. Perzanowski. MUC-7 evaluation of ie technology: Overview of results. In *MUC-7*, volume 20, 1998.

[32] B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *HLT-NAACL*, pages 777–782, 2013.

[33] J. W. Murdock, A. Kalyanpur, C. Welty, J. Fan, D. A. Ferrucci, D. Gondek, L. Zhang, and H. Kanayama. Typing candidate answers using type coercion. *IBM Journal of Research and Development*, 56(3.4):7–1, 2012.

[34] S. Na, I. Kang, S. Lee, and J. Lee. Question answering approach using a WordNet-based answer type taxonomy. In *TREC*, 2002.

[35] C. Pinchak and D. Lin. A probabilistic answer type model. In *EACL*, 2006.

[36] S. Robertson and H. Zaragoza. On rank-based effectiveness measures and optimization. *Information Retrieval*, 10(3):321–339, 2007.

[37] N. Schlaefer, P. Gieselmann, T. Schaaf, and A. Waibel. A pattern learning approach to question answering within the ephyra framework. In *Text, speech and dialogue*, pages 687–694. Springer, 2006.

[38] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: a core of semantic knowledge. In *WWW*, pages 697–706. ACM, 2007.

[39] C. Tsai, W. Yih, and C. Burges. Web-based question answering: Revisiting AskMSR. Technical Report MSR-TR-2015-20, Microsoft Research, 2015.

[40] C. Unger, L. Bühmann, J. Lehmann, A. Ngonga Ngomo, D. Gerber, and P. Cimiano. Template-based question answering over RDF data. In *WWW*, pages 639–648. ACM, 2012.

[41] E. M. Voorhees and D. M. Tice. Building a question answering test collection. In *SIGIR on Rearch and Development in IR*, pages 200–207. ACM, 2000.

[42] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, and D. Lin. Knowledge base completion via search-based question answering. In *WWW*, pages 515–526, 2014.

[43] R. W. White, M. Richardson, and W. Yih. Questions vs. queries in informational search tasks. Technical Report MSR-TR-2014-96, Microsoft Research, 2014.

[44] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. Natural language questions for the Web of data. In *EMNLP-CoNLL*, pages 379–390, 2012.

[45] X. Yao and B. Van Durme. Information extraction over structured data: Question answering with Freebase. In *ACL*, 2014.

[46] L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, and D. Zhao. Natural language question answering over RDF: a graph data driven approach. In *SIGMOD*, pages 313–324. ACM, 2014.