Knowledge-Based Question Answering as Machine Translation

Junwei Bao[†] *, **Nan Duan**[‡] , **Ming Zhou**[‡] , **Tiejun Zhao**[†]
[†]Harbin Institute of Technology
[‡]Microsoft Research
baojunwei001@gmail.com
{nanduan, mingzhou}@microsoft.com
tjzhao@hit.edu.cn

Abstract

A typical knowledge-based question answering (KB-QA) system faces two challenges: one is to transform natural language questions into their meaning representations (MRs); the other is to retrieve answers from knowledge bases (KBs) using generated MRs. Unlike previous methods which treat them in a cascaded manner, we present a translation-based approach to solve these two tasks in one unified framework. We translate questions to answers based on CYK parsing. Answers as translations of the span covered by each CYK cell are obtained by a question translation method, which first generates formal triple queries as MRs for the span based on question patterns and relation expressions, and then retrieves answers from a given KB based on triple queries generated. A linear model is defined over derivations, and minimum error rate training is used to tune feature weights based on a set of question-answer pairs. Compared to a KB-QA system using a state-of-the-art semantic parser, our method achieves better results.

1 Introduction

Knowledge-based question answering (KB-QA) computes answers to natural language (NL) questions based on existing knowledge bases (KBs). Most previous systems tackle this task in a cascaded manner: First, the input question is transformed into its meaning representation (MR) by an independent semantic parser (Zettlemoyer and Collins, 2005; Mooney, 2007; Artzi and Zettlemoyer, 2011; Liang et al., 2011; Cai and Yates, 2013; Poon, 2013; Artzi et al., 2013; Kwiatkowski et al., 2013; Berant et al., 2013); Then, the answers are retrieved from existing KBs using generated MRs as queries.

Unlike existing KB-QA systems which treat semantic parsing and answer retrieval as two cascaded tasks, this paper presents a unified framework that can integrate semantic parsing into the question answering procedure directly. Borrowing ideas from machine translation (MT), we treat the QA task as a translation procedure. Like MT, CYK parsing is used to parse each input question, and answers of the span covered by each CYK cell are considered the translations of that cell; unlike MT, which uses offline-generated translation tables to translate source phrases into target translations, a semantic parsing-based question translation method is used to translate each span into its answers on-the-fly, based on question patterns and relation expressions. The final answers can be obtained from the root cell. Derivations generated during such a translation procedure are modeled by a linear model, and minimum error rate training (MERT) (Och, 2003) is used to tune feature weights based on a set of question-answer pairs.

Figure 1 shows an example: the question *director of movie starred by Tom Hanks* is translated to one of its answers *Robert Zemeckis* by three main steps: (i) translate *director of* to *director of*; (ii) translate *movie starred by Tom Hanks* to one of its answers *Forrest Gump*; (iii) translate *director of Forrest Gump* to a final answer *Robert Zemeckis*. Note that the updated question covered by Cell[0, 6] is obtained by combining the answers to question spans covered by Cell[0, 1] and Cell[2, 6].

The contributions of this work are two-fold: (1) We propose a translation-based KB-QA method that integrates semantic parsing and QA in one unified framework. The benefit of our method is that we don't need to explicitly generate complete semantic structures for input questions. Be-

*This work was finished while the author was visiting Microsoft Research Asia.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 967–976,
Baltimore, Maryland, USA, June 23-25 2014.

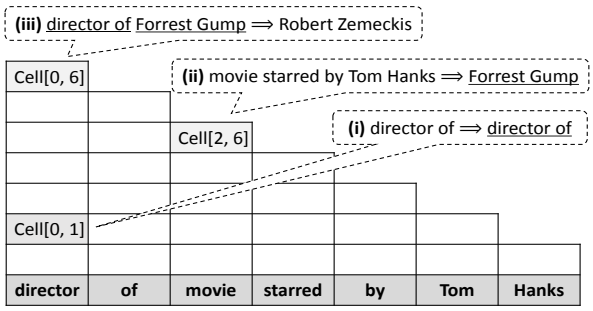

Figure 1: Translation-based KB-QA example

sides which, answers generated during the translation procedure help significantly with search space pruning. (2) We propose a robust method to transform single-relation questions into formal triple queries as their MRs, which trades off between transformation accuracy and recall using question patterns and relation expressions respectively.

2 Translation-Based KB-QA

2.1 Overview

Formally, given a knowledge base $\mathcal{KB}$ and an N-L question $\mathcal{Q}$, our KB-QA method generates a set of formal triples-answer pairs $\{\langle \mathcal{D}, \mathcal{A} \rangle\}$ as derivations, which are scored and ranked by the distribution $P(\langle \mathcal{D}, \mathcal{A} \rangle | \mathcal{KB}, \mathcal{Q})$ defined as follows:

$$\frac{exp\{\sum_{i=1}^{M} \lambda_i \cdot h_i(\langle \mathcal{D}, \mathcal{A} \rangle, \mathcal{KB}, \mathcal{Q})\}}{\sum_{\langle \mathcal{D}', \mathcal{A}' \rangle \in \mathcal{H}(\mathcal{Q})} exp\{\sum_{i=1}^{M} \lambda_i \cdot h_i(\langle \mathcal{D}', \mathcal{A}' \rangle, \mathcal{KB}, \mathcal{Q})\}}$$

- $\mathcal{KB}$ denotes a knowledge base[1] that stores a set of assertions. Each assertion $t \in \mathcal{KB}$ is in the form of $\{e_{sbj}^{ID}, p, e_{obj}^{ID}\}$, where p denotes a predicate, e_{sbj}^{ID} and e_{obj}^{ID} denote the subject and object entities of t, with unique IDs[2].

- $\mathcal{H}(\mathcal{Q})$ denotes the search space $\{\langle \mathcal{D}, \mathcal{A} \rangle\}$. $\mathcal{D}$ is composed of a set of *ordered* formal triples $\{t_1, ..., t_n\}$. Each triple $t = \{e_{sbj}, p, e_{obj}\}_i^j \in \mathcal{D}$ denotes an assertion in $\mathcal{KB}$, where i and j denotes the beginning and end indexes of the question span from which t is transformed. The order of triples in $\mathcal{D}$ denotes the order of translation steps from $\mathcal{Q}$ to $\mathcal{A}$. E.g., $\langle$*director of, Null, director of*$\rangle_0^1$, $\langle$*Tom Hanks, Film.Actor.Film, Forrest Gump*$\rangle_2^6$ and $\langle$*Forrest Gump, Film.Film.Director, Robert Zemeckis*$\rangle_0^6$ are three ordered formal triples corresponding to the three translation steps in Figure 1. We define the task of transforming question spans into formal triples as *question translation*. $\mathcal{A}$ denotes one final answer of $\mathcal{Q}$.

- $h_i(\cdot)$ denotes the i^{th} feature function.

- λ_i denotes the feature weight of $h_i(\cdot)$.

According to the above description, our KB-QA method can be decomposed into four tasks as: (1) search space generation for $\mathcal{H}(\mathcal{Q})$; (2) question translation for transforming question spans into their corresponding formal triples; (3) feature design for $h_i(\cdot)$; and (4) feature weight tuning for $\{\lambda_i\}$. We present details of these four tasks in the following subsections one-by-one.

2.2 Search Space Generation

We first present our translation-based KB-QA method in Algorithm 1, which is used to generate $\mathcal{H}(\mathcal{Q})$ for each input NL question $\mathcal{Q}$.

Algorithm 1: Translation-based KB-QA

```
for l = 1 to |Q| do
    for all i, j s.t. j − i = l do
        H(Q_i^j) = ∅;
        T = QTrans(Q_i^j, KB);
        foreach formal triple t ∈ T do
            create a new derivation d;
            d.A = t.e_obj;
            d.D = {t};
            update the model score of d;
            insert d to H(Q_i^j);
        end
    end
end
for l = 1 to |Q| do
    for all i, j s.t. j − i = l do
        for all m s.t. i ≤ m < j do
            for d_l ∈ H(Q_i^m) and d_r ∈ H(Q_{m+1}^j) do
                Q_update = d_l.A + d_r.A;
                T = QTrans(Q_update, KB);
                foreach formal triple t ∈ T do
                    create a new derivation d;
                    d.A = t.e_obj;
                    d.D = d_l.D ⋃ d_r.D ⋃ {t};
                    update the model score of d;
                    insert d to H(Q_i^j);
                end
            end
        end
    end
end
return H(Q).
```

[1] We use a large scale knowledge base in this paper, which contains 2.3B entities, 5.5K predicates, and 18B assertions. A 16-machine cluster is used to host and serve the whole data.

[2] Each KB entity has a unique ID. For the sake of convenience, we omit the ID information in the rest of the paper.

The first half (from Line 1 to Line 13) generates a formal triple set T for each unary span $\mathcal{Q}_i^j \in \mathcal{Q}$, using the question translation method $QTrans(\mathcal{Q}_i^j, \mathcal{KB})$ (Line 4), which takes $\mathcal{Q}_i^j$ as the input. Each triple $t \in T$ returned is in the form of $\{e_{sbj}, p, e_{obj}\}$, where e_{sbj}'s mention occurs in $\mathcal{Q}_i^j$, p is a predicate that denotes the meaning expressed by the context of e_{sbj} in $\mathcal{Q}_i^j$, e_{obj} is an answer of $\mathcal{Q}_i^j$ based on e_{sbj}, p and $\mathcal{KB}$. We describe the implementation detail of $QTrans(\cdot)$ in Section 2.3.

The second half (from Line 14 to Line 31) first updates the content of each bigger span $\mathcal{Q}_i^j$ by concatenating the answers to its any two consecutive smaller spans covered by $\mathcal{Q}_i^j$ (Line 18). Then, $QTrans(\mathcal{Q}_i^j, \mathcal{KB})$ is called to generate triples for the updated span (Line 19). The above operations are equivalent to answering a simplified question, which is obtained by replacing the answerable spans in the original question with their corresponding answers. The search space $\mathcal{H}(\mathcal{Q})$ for the entire question $\mathcal{Q}$ is returned at last (Line 31).

2.3 Question Translation

The purpose of question translation is to translate a span $\mathcal{Q}$ to a set of formal triples T. Each triple $t \in T$ is in the form of $\{e_{sbj}, p, e_{obj}\}$, where e_{sbj}'s mention[3] occurs in $\mathcal{Q}$, p is a predicate that denotes the meaning expressed by the context of e_{sbj} in $\mathcal{Q}$, e_{obj} is an answer to $\mathcal{Q}$ retrieved from $\mathcal{KB}$ using a triple query $q = \{e_{sbj}, p, ?\}$. Note that if no predicate p or answer e_{obj} can be generated, $\{\mathcal{Q}, Null, \mathcal{Q}\}$ will be returned as a special triple, which sets e_{obj} to be $\mathcal{Q}$ itself, and p to be $Null$. This makes sure the un-answerable spans can be passed on to the higher-level operations.

Question translation assumes each span $\mathcal{Q}$ is a *single-relation question* (Fader et al., 2013). Such assumption simplifies the efforts of semantic parsing to the minimum question units, while leaving the capability of handling multiple-relation questions (Figure 1 gives one such example) to the outer CYK-parsing based translation procedure. Two question translation methods are presented in the rest of this subsection, which are based on question patterns and relation expressions respectively.

2.3.1 Question Pattern-based Translation

A question pattern $\mathcal{QP}$ includes a pattern string $\mathcal{QP}_{pattern}$, which is composed of words and a slot symbol $[Slot]$, and a KB predicate $\mathcal{QP}_{predicate}$, which denotes the meaning expressed by the context words in $\mathcal{QP}_{pattern}$.

Algorithm 2: $\mathcal{QP}$-based Question Translation

```
1  T = ∅;
2  foreach entity mention e_Q ∈ Q do
3      Q_pattern = replace e_Q in Q with [Slot];
4      foreach question pattern QP do
5          if Q_pattern == QP_pattern then
6              E = Disambiguate(e_Q, QP_predicate);
7              foreach e ∈ E do
8                  create a new triple query q;
9                  q = {e, QP_predicate, ?};
10                 {A_i} = AnswerRetrieve(q, KB);
11                 foreach A ∈ {A_i} do
12                     create a new formal triple t;
13                     t = {q.e_sbj, q.p, A};
14                     t.score = 1.0;
15                     insert t to T;
16                 end
17             end
18         end
19     end
20 end
21 return T.
```

Algorithm 2 shows how to generate formal triples for a span $\mathcal{Q}$ based on question patterns ($\mathcal{QP}$-based question translation). For each entity mention $e_{\mathcal{Q}} \in \mathcal{Q}$, we replace it with $[Slot]$ and obtain a pattern string $\mathcal{Q}_{pattern}$ (Line 3). If $\mathcal{Q}_{pattern}$ can match one $\mathcal{QP}_{pattern}$, then we construct a triple query q (Line 9) using $\mathcal{QP}_{predicate}$ as its predicate and one of the KB entities returned by $Disambiguate(e_{\mathcal{Q}}, \mathcal{QP}_{predicate})$ as its subject entity (Line 6). Here, the objective of $Disambiguate(e_{\mathcal{Q}}, \mathcal{QP}_{predicate})$ is to output a set of disambiguated KB entities $\mathcal{E}$ in $\mathcal{KB}$. The name of each entity returned equals the input entity mention $e_{\mathcal{Q}}$ and occurs in some assertions where $\mathcal{QP}_{predicate}$ are the predicates. The underlying idea is to use the context (predicate) information to help entity disambiguation. The answers of q are returned by $AnswerRetrieve(q, \mathcal{KB})$ based on q and $\mathcal{KB}$ (Line 10), each of which is used to construct a formal triple and added to T for $\mathcal{Q}$ (from Line 11 to Line 16). Figure 2 gives an example.

Question patterns are collected as follows: First, *5W* queries, which begin with What, Where, Who, When, or Which, are selected from a large scale query log of a commercial search engine; Then, a cleaned entity dictionary is used to annotate each query by replacing all entity mentions it contains with the symbol $[Slot]$. Only high-frequent query patterns which contain one $[Slot]$ are maintained;

[3]For simplicity, a cleaned entity dictionary dumped from the entire $\mathcal{KB}$ is used to detect entity mentions in $\mathcal{Q}$.

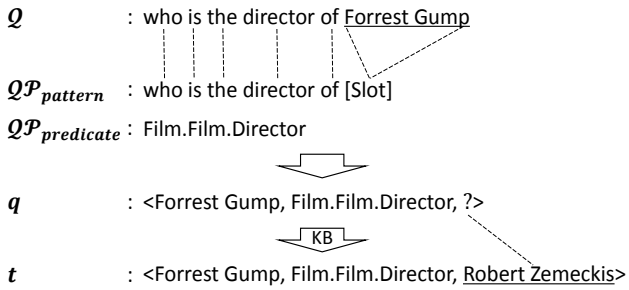

Figure 2: $\mathcal{QP}$-based question translation example

Lastly, annotators try to manually label the most-frequent 50,000 query patterns with their corresponding predicates, and 4,764 question patterns with single labeled predicates are obtained.

From experiments (Table 3 in Section 4.3) we can see that, question pattern based question translation can achieve high end-to-end accuracy. But as human efforts are needed in the mining procedure, this method cannot be extended to large scale very easily. Besides, different users often type the questions with the same meaning in different NL expressions. For example, although the question *Forrest Gump was directed by which moviemaker* means the same as the question $\mathcal{Q}$ in Figure 2, no question pattern can cover it. We need to find an alternative way to alleviate such coverage issue.

2.3.2 Relation Expression-based Translation

Aiming to alleviate the coverage issue occurring in $\mathcal{QP}$-based method, an alternative relation expression ($\mathcal{RE}$) -based method is proposed, and will be used when the $\mathcal{QP}$-based method fails.

We define $\mathcal{RE}_p$ as a relation expression set for a given KB predicate $p \in \mathcal{KB}$. Each relation expression $\mathcal{RE} \in \mathcal{RE}_p$ includes an expression string $\mathcal{RE}_{expression}$, which must contain at least one content word, and a weight $\mathcal{RE}_{weight}$, which denotes the confidence that $\mathcal{RE}_{expression}$ can represent p's meaning in NL. For example, *is the director of* is one relation expression string for the predicate *Film.Film.Director*, which means it is usually used to express this relation (predicate) in NL.

Algorithm 3 shows how to generate triples for a question $\mathcal{Q}$ based on relation expressions. For each possible entity mention $e_{\mathcal{Q}} \in \mathcal{Q}$ and a KB predicate $p \in \mathcal{KB}$ that is related to a KB entity e whose name equals $e_{\mathcal{Q}}$, $Sim(e_{\mathcal{Q}}, \mathcal{Q}, \mathcal{RE}_p)$ is computed (Line 5) based on the similarity between question context and $\mathcal{RE}_p$, which measures how likely $\mathcal{Q}$ can be transformed into a triple query

Algorithm 3: $\mathcal{RE}$-based Question Translation

```
1  T = ∅;
2  foreach entity mention e_Q ∈ Q do
3      foreach e ∈ KB s.t. e.name==e_Q do
4          foreach predicate p ∈ KB related to e do
5              score = Sim(e_Q, Q, RE_p);
6              if score > 0 then
7                  create a new triple query q;
8                  q = {e, p, ?};
9                  {A_i} = AnswerRetrieve(q, KB);
10                 foreach A ∈ {A_i} do
11                     create a new formal triple t;
12                     t = {q.e_sbj, q.p, A};
13                     t.score = score;
14                     insert t to T;
15                 end
16             end
17         end
18     end
19 end
20 sort T based on the score of each t ∈ T;
21 return T.
```

$q = \{e, p, ?\}$. If this score is larger than 0, which means there are overlaps between $\mathcal{Q}$'s context and $\mathcal{RE}_p$, then q will be used as the triple query of $\mathcal{Q}$, and a set of formal triples will be generated based on q and $\mathcal{KB}$ (from Line 7 to Line 15). The computation of $Sim(e_{\mathcal{Q}}, \mathcal{Q}, \mathcal{RE}_p)$ is defined as follows:

$$\sum_{n} \frac{1}{|\mathcal{Q}| - n + 1} \cdot \{ \sum_{\omega_n \in \mathcal{Q}, \omega_n \bigcap e_{\mathcal{Q}} = \phi} P(\omega_n | \mathcal{RE}_p) \}$$

where n is the n-gram order which ranges from 1 to 5, ω_n is an n-gram occurring in $\mathcal{Q}$ without overlapping with $e_{\mathcal{Q}}$ and containing at least one content word, $P(\omega_n|\mathcal{RE}_p)$ is the posterior probability which is computed by:

$$P(\omega_n|\mathcal{RE}_p) = \frac{Count(\omega_n, \mathcal{RE}_p)}{\sum_{\omega'_n \in \mathcal{RE}_p} Count(\omega'_n, \mathcal{RE}_p)}$$

$Count(\omega, \mathcal{RE}_p)$ denotes the weighted sum of times that ω occurs in $\mathcal{RE}_p$:

$$Count(\omega, \mathcal{RE}_p) = \sum_{\mathcal{RE} \in \mathcal{RE}_p} \{\#_{\omega}(\mathcal{RE}) \cdot \mathcal{RE}_{weight}\}$$

where $\#_{\omega}(\mathcal{RE})$ denotes the number of times that ω occurs in $\mathcal{RE}_{expression}$, and $\mathcal{RE}_{weight}$ is decided by the relation expression extraction component.

Figure 3 gives an example, where n-grams with rectangles are the ones that occur in both $\mathcal{Q}$'s context and the relation expression set of a given predicate $p = Film.Film.Director$. Unlike the $\mathcal{QP}$-based method which needs a perfect match, the

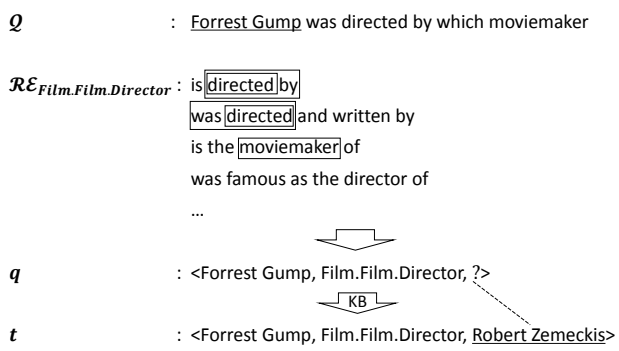

Figure 3: $\mathcal{RE}$-based question translation example

$\mathcal{RE}$-based method allows fuzzy matching between $\mathcal{Q}$ and $\mathcal{RE}_p$, and records this (Line 13) in generated triples, which is used as features later.

Relation expressions are mined as follows: Given a set of KB assertions with an identical predicate p, we first extract all sentences from English Wiki pages[4], each of which contains at least one pair of entities occurring in one assertion. Then, we extract the shortest path between paired entities in the dependency tree of each sentence as an $\mathcal{RE}$ candidate for the given predicate. The intuition is that any sentence containing such entity pairs occur in an assertion is likely to express the predicate of that assertion in some way. Last, all relation expressions extracted are filtered by heuristic rules, i.e., the frequency must be larger than 4, the length must be shorter than 10, and then weighted by the pattern scoring methods proposed in (Gerber and Ngomo, 2011; Gerber and Ngomo, 2012). For each predicate, we only keep the relation expressions whose pattern scores are larger than a pre-defined threshold. Figure 4 gives one relation expression extraction example. The statistics and overall quality of the relation expressions are listed in Section 4.1.

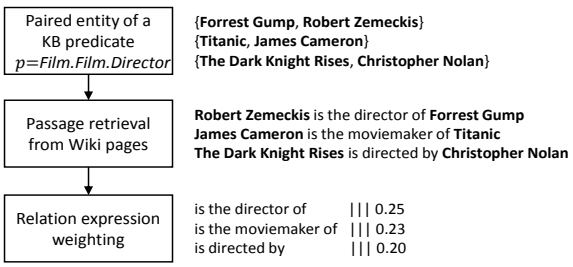

Figure 4: $\mathcal{RE}$ extraction example

[4]http://en.wikipedia.org/wiki/Wikipedia:Database_download

2.3.3 Question Decomposition

Sometimes, a question may provide multiple constraints to its answers. *movie starred by Tom Hanks in 1994* is one such question. All the films as the answers of this question should satisfy the following two constraints: (1) starred *by Tom Hanks*; and (2) released *in 1994*. It is easy to see that such questions cannot be translated to single triples.

We propose a dependency tree-based method to handle such multiple-constraint questions by (i) decomposing the original question into a set of *sub-questions* using syntax-based patterns; and (ii) intersecting the answers of all sub-questions as the final answers of the original question. Note, question decomposition only operates on the original question and question spans covered by complete dependency subtrees. Four syntax-based patterns (Figure 5) are used for question decomposition. If a question matches any one of these patterns, then sub-questions are generated by collecting the paths between n_0 and each $n_i(i > 0)$ in the pattern, where each n denotes a complete subtree with a noun, number, or question word as its root node, the symbol $*$ above $prep^*$ denotes this preposition can be skipped in matching. For the question mentioned at the beginning, its two sub-questions generated are *movie starred by Tom Hanks* and *movie starred in 1994*, as its dependency form matches pattern (a). Similar ideas are used in IBM Watson (Kalyanpur et al., 2012) as well.

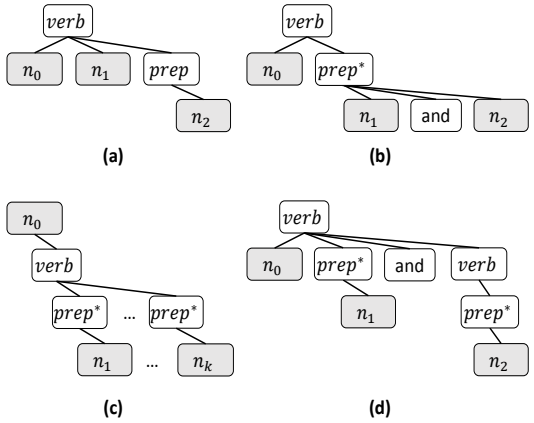

Figure 5: Four syntax-based patterns for question decomposition

As dependency parsing is not perfect, we generate single triples for such questions without considering constraints as well, and add them to the search space for competition. $h_{syntax\_constraint}(\cdot)$

is used to boost triples that are converted from sub-questions generated by question decomposition. The more constraints an answer satisfies, the better. Obviously, current patterns used can't cover all cases but most-common ones. We leave a more general pattern mining method for future work.

2.4 Feature Design

The objective of our KB-QA system is to seek the derivation $\langle \hat{\mathcal{D}}, \hat{\mathcal{A}} \rangle$ that maximizes the probability $P(\langle \mathcal{D}, \mathcal{A} \rangle | \mathcal{KB}, \mathcal{Q})$ described in Section 2.1 as:

$$\begin{aligned} \langle \hat{\mathcal{D}}, \hat{\mathcal{A}} \rangle &= \underset{\langle \mathcal{D}, \mathcal{A} \rangle \in \mathcal{H}(\mathcal{Q})}{argmax} P(\langle \mathcal{D}, \mathcal{A} \rangle | \mathcal{KB}, \mathcal{Q}) \\ &= \underset{\langle \mathcal{D}, \mathcal{A} \rangle \in \mathcal{H}(\mathcal{Q})}{argmax} \sum_{i=1}^{M} \lambda_i \cdot h_i(\langle \mathcal{D}, \mathcal{A} \rangle, \mathcal{KB}, \mathcal{Q}) \end{aligned}$$

We now introduce the feature sets $\{h_i(\cdot)\}$ that are used in the above linear model:

- $h_{question\_word}(\cdot)$, which counts the number of original question words occurring in $\mathcal{A}$. It penalizes those partially answered questions.
- $h_{span}(\cdot)$, which counts the number of spans in $\mathcal{Q}$ that are converted to formal triples. It controls the granularity of the spans used in question translation.
- $h_{syntax\_subtree}(\cdot)$, which counts the number of spans in $\mathcal{Q}$ that are (1) converted to formal triples, whose predicates are not $Null$, and (2) covered by complete dependency subtrees at the same time. The underlying intuition is that, dependency subtrees of $\mathcal{Q}$ should be treated as units for question translation.
- $h_{syntax\_constraint}(\cdot)$, which counts the number of triples in $\mathcal{D}$ that are converted from sub-questions generated by the question decomposition component.
- $h_{triple}(\cdot)$, which counts the number of triples in $\mathcal{D}$, whose predicates are not $Null$.
- $h_{triple_{weight}}(\cdot)$, which sums the scores of all triples $\{t_i\}$ in $\mathcal{D}$ as $\sum_{t_i \in \mathcal{D}} t_i.score$.
- $h_{\mathcal{QP}_{count}}(\cdot)$, which counts the number of triples in $\mathcal{D}$ that are generated by $\mathcal{QP}$-based question translation method.
- $h_{\mathcal{RE}_{count}}(\cdot)$, which counts the number of triples in $\mathcal{D}$ that are generated by $\mathcal{RE}$-based question translation method.
- $h_{staticrank_{sbj}}(\cdot)$, which sums the static rank scores of all subject entities in $\mathcal{D}$'s triple set as $\sum_{t_i \in \mathcal{D}} t_i.e_{sbj}.static\_rank$.
- $h_{staticrank_{obj}}(\cdot)$, which sums the static rank scores of all object entities in $\mathcal{D}$'s triple set as $\sum_{t_i \in \mathcal{D}} t_i.e_{obj}.static\_rank$.
- $h_{confidence_{obj}}(\cdot)$, which sums the confidence scores of all object entities in $\mathcal{D}$'s triple set as $\sum_{t \in \mathcal{D}} t.e_{obj}.confidence$.

For each assertion $\{e_{sbj}, p, e_{obj}\}$ stored in $\mathcal{KB}$, $e_{sbj}.static\_rank$ and $e_{obj}.static\_rank$ denote the static rank scores[5] for e_{sbj} and e_{obj} respectively; $e_{obj}.confidence\_rank$ represents the probability $p(e_{obj}|e_{sbj}, p)$. These three scores are used as features to rank answers generated in QA procedure.

2.5 Feature Weight Tuning

Given a set of question-answer pairs $\{\mathcal{Q}_i, \mathcal{A}_i^{ref}\}$ as the development (dev) set, we use the minimum error rate training (MERT) (Och, 2003) algorithm to tune the feature weights λ_i^M in our proposed model. The training criterion is to seek the feature weights that can minimize the accumulated errors of the top-1 answer of questions in the dev set:

$$\hat{\lambda}_1^M = \underset{\lambda_1^M}{argmin} \sum_{i=1}^{N} Err(\mathcal{A}_i^{ref}, \hat{\mathcal{A}}_i; \lambda_1^M)$$

N is the number of questions in the dev set, $\mathcal{A}_i^{ref}$ is the correct answers as references of the i^{th} question in the dev set, $\hat{\mathcal{A}}_i$ is the top-1 answer candidate of the i^{th} question in the dev set based on feature weights λ_1^M, $Err(\cdot)$ is the error function which is defined as:

$$Err(\mathcal{A}_i^{ref}, \hat{\mathcal{A}}_i; \lambda_1^M) = 1 - \delta(\mathcal{A}_i^{ref}, \hat{\mathcal{A}}_i)$$

where $\delta(\mathcal{A}_i^{ref}, \hat{\mathcal{A}}_i)$ is an indicator function which equals 1 when $\hat{\mathcal{A}}_i$ is included in the reference set $\mathcal{A}_i^{ref}$, and 0 otherwise.

3 Comparison with Previous Work

Our work intersects with two research directions: semantic parsing and question answering.

Some previous works on semantic parsing (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2006; Zettlemoyer and Collins, 2007; Wong and Mooney,

[5]The static rank score of an entity represents a general indicator of the overall quality of that entity.

2007; Kwiatkowski et al., 2010; Kwiatkowski et al., 2011) require manually annotated logical forms as supervision, and are hard to extend resulting parsers from limited domains, such as GEO, JOBS and ATIS, to open domains. Recent works (Clarke and Lapata, 2010; Liang et al., 2013) have alleviated such issues using question-answer pairs as weak supervision, but still with the shortcoming of using limited lexical triggers to link NL phrases to predicates. Poon (2013) has proposed an unsupervised method by adopting grounded-learning to leverage the database for indirect supervision. But transformation from NL questions to MRs heavily depends on dependency parsing results. Besides, the KB used (ATIS) is limited as well. Kwiatkowski et al. (2013) use Wiktionary and a limited manual lexicon to map POS tags to a set of predefined CCG lexical categories, which aims to reduce the need for learning lexicon from training data. But it still needs human efforts to define lexical categories, which usually can not cover all the semantic phenomena.

Berant et al. (2013) have not only enlarged the KB used for Freebase (Google, 2013), but also used a bigger lexicon trigger set extracted by the open IE method (Lin et al., 2012) for NL phrases to predicates linking. In comparison, our method has further advantages: (1) Question answering and semantic parsing are performed in an joint way under a unified framework; (2) A robust method is proposed to map NL questions to their formal triple queries, which trades off the mapping quality by using question patterns and relation expressions in a cascaded way; and (3) We use domain independent feature set which allowing us to use a relatively small number of question-answer pairs to tune model parameters.

Fader et al. (2013) map questions to formal (triple) queries over a large scale, open-domain database of facts extracted from a raw corpus by ReVerb (Fader et al., 2011). Compared to their work, our method gains an improvement in two aspects: (1) Instead of using facts extracted using the open IE method, we leverage a large scale, high-quality knowledge base; (2) We can handle multiple-relation questions, instead of single-relation queries only, based on our translation based KB-QA framework.

Espana-Bonet and Comas (2012) have proposed an MT-based method for factoid QA. But MT in there work means to translate questions into n-best translations, which are used for finding similar sentences in the document collection that probably contain answers. Echihabi and Marcu (2003) have developed a noisy-channel model for QA, which explains how a sentence containing an answer to a given question can be rewritten into that question through a sequence of stochastic operations. Compared to the above two MT-motivated QA work, our method uses MT methodology to translate questions to answers directly.

4 Experiment

4.1 Data Sets

Following Berant et al. (2013), we use the same subset of WEBQUESTIONS (3,778 questions) as the development set (Dev) for weight tuning in MERT, and use the other part of WEBQUESTIONS (2,032 questions) as the test set (Test). Table 1 shows the statistics of this data set.

Data Set	# Questions	# Words
WEBQUESTIONS	5,810	6.7

Table 1: Statistics of evaluation set. # Questions is the number of questions in a data set, # Words is the averaged word count of a question.

Table 2 shows the statistics of question patterns and relation expressions used in our KB-QA system. As all question patterns are collected with human involvement as we discussed in Section 2.3.1, the quality is very high (98%). We also sample 1,000 instances from the whole relation expression set and manually label their quality. The accuracy is around 89%. These two resources can cover 566 head predicates in our KB.

	# Entries	Accuracy
Question Patterns	4,764	98%
Relation Expressions	133,445	89%

Table 2: Statistics of question patterns and relation expressions.

4.2 KB-QA Systems

Since Berant et al. (2013) is one of the latest work which has reported QA results based on a large scale, general domain knowledge base (Freebase), we consider their evaluation result on WEBQUESTIONS as our baseline.

Our KB-QA system generates the k-best derivations for each question span, where k is set to 20.

The answers with the highest model scores are considered the best answers for evaluation. For evaluation, we follow Berant et al. (2013) to allow partial credit and score an answer using the F1 measure, comparing the predicted set of entities to the annotated set of entities.

One difference between these two systems is the KB used. Since Freebase is completely contained by our KB, we disallow all entities which are not included by Freebase. By doing so, our KB provides the same knowledge as Freebase does, which means we do not gain any extra advantage by using a larger KB. But we still allow ourselves to use the static rank scores and confidence scores of entities as features, as we described in Section 2.4.

4.3 Evaluation Results

We first show the overall evaluation results of our KB-QA system and compare them with baseline's results on Dev and Test. Note that we do not re-implement the baseline system, but just list their evaluation numbers reported in the paper. Comparison results are listed in Table 3.

	Dev (Accuracy)	Test (Accuracy)
Baseline	32.9%	31.4%
Our Method	42.5% **(+9.6%)**	37.5% **(+6.1%)**

Table 3: Accuracy on evaluation sets. $Accuracy$ is defined as the number of correctly answered questions divided by the total number of questions.

Table 3 shows our KB-QA method outperforms baseline on both Dev and Test. We think the potential reasons of this improvement include:

- Different methods are used to map NL *phrases* to KB predicates. Berant et al. (2013) have used a lexicon extracted from a subset of ReVerb triples (Lin et al., 2012), which is similar to the relation expression set used in question translation. But as our relation expressions are extracted by an in-house extractor, we can record their extraction-related statistics as extra information, and use them as features to measure the mapping quality. Besides, as a portion of entities in our KB are extracted from Wiki, we know the one-to-one correspondence between such entities and Wiki pages, and use this information in relation expression extraction for entity disambiguation. A lower disambiguation error rate results in better relation expressions.
- Question patterns are used to map NL *context* to KB predicates. Context can be either continuous or discontinues phrases. Although the size of this set is limited, they can actually cover head questions/queries[6] very well. The underlying intuition of using patterns is that those high-frequent questions/queries should and can be treated and solved in the QA task, by involving human effort at a relative small price but with very impressive accuracy.

In order to figure out the impacts of question patterns and relation expressions, another experiment (Table 4) is designed to evaluate their independent influences, where $\mathcal{QP}_{only}$ and $\mathcal{RE}_{only}$ denote the results of KB-QA systems which only allow question patterns and relation expressions in question translation respectively.

Settings	Test (Accuracy)	Test (Precision)
$\mathcal{QP}_{only}$	11.8%	97.5%
$\mathcal{RE}_{only}$	32.5%	73.2%

Table 4: Impacts of question patterns and relation expressions. $Precision$ is defined as the number of correctly answered questions divided by the number of questions with non-empty answers generated by our KB-QA system.

From Table 4 we can see that the accuracy of $\mathcal{RE}_{only}$ on Test (32.5%) is slightly better than baseline's result (31.4%). We think this improvement comes from two aspects: (1) The quality of the relation expressions is better than the quality of the lexicon entries used in the baseline; and (2) We use the extraction-related statistics of relation expressions as features, which brings more information to measure the confidence of mapping between NL phrases and KB predicates, and makes the model to be more flexible. Meanwhile, $\mathcal{QP}_{only}$ perform worse (11.8%) than $\mathcal{RE}_{only}$, due to coverage issue. But by comparing the precisions of these two settings, we find $\mathcal{QP}_{only}$ (97.5%) outperforms $\mathcal{RE}_{only}$ (73.2%) significantly, due to its high quality. This means how to extract high-quality question patterns is worth to be studied for the question answering task.

As the performance of our KB-QA system relies heavily on the k-best beam approximation, we evaluate the impact of the beam size and list the comparison results in Figure 6. We can see that as

[6]Head questions/queries mean the questions/queries with high frequency and clear patterns.

we increase k incrementally, the accuracy increase at the same time. However, a larger k (e.g. 200) cannot bring significant improvements comparing to a smaller one (e.g., 20), but using a large k has a tremendous impact on system efficiency. So we choose $k = 20$ as the optimal value in above experiments, which trades off between accuracy and efficiency.

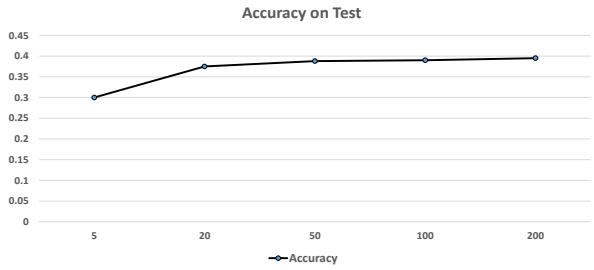

Figure 6: Impacts of beam size on accuracy.

Actually, the size of our system's search space is much smaller than the one of the semantic parser used in the baseline.This is due to the fact that, if triple queries generated by the question translation component cannot derive any answer from KB, we will discard such triple queries directly during the QA procedure. We can see that using a small k can achieve better results than baseline, where the beam size is set to be 200.

4.4 Error Analysis

4.4.1 Entity Detection

Since named entity recognizers trained on Penn TreeBank usually perform poorly on web queries, We instead use a simple string-match method to detect entity mentions in the question using a cleaned entity dictionary dumped from our KB. One problem of doing so is the entity detection issue. For example, in the question *who was Esther's husband ?*, we cannot detect *Esther* as an entity, as it is just part of an entity name. We need an ad-hoc entity detection component to handle such issues, especially for a web scenario, where users often type entity names in their partial or abbreviation forms.

4.4.2 Predicate Mapping

Some questions lack sufficient evidences to detect predicates. *where is Byron Nelson 2012 ?* is an example. Since each relation expression must contain at least one content word, this question cannot match any relation expression. Except for *Byron Nelson* and *2012*, all the others are non-content words.

Besides, ambiguous entries contained in relation expression sets of different predicates can bring mapping errors as well. For the following question *who did Steve Spurrier play pro football for?* as an example, since the unigram *play* exists in both *Film.Film.Actor* and *American_Football.Player.Current_Team* 's relation expression sets, we made a wrong prediction, which led to wrong answers.

4.4.3 Specific Questions

Sometimes, we cannot give exact answers to superlative questions like *what is the first book Sherlock Holmes appeared in?*. For this example, we can give all book names where *Sherlock Holmes* appeared in, but we cannot rank them based on their publication date , as we cannot learn the alignment between the constraint word *first* occurred in the question and the predicate *Book.Written_Work.Date_Of_First_Publication* from training data automatically. Although we have followed some work (Poon, 2013; Liang et al., 2013) to handle such special linguistic phenomena by defining some specific operators, it is still hard to cover all unseen cases. We leave this to future work as an independent topic.

5 Conclusion and Future Work

This paper presents a translation-based KB-QA method that integrates semantic parsing and QA in one unified framework. Comparing to the baseline system using an independent semantic parser with state-of-the-art performance, we achieve better results on a general domain evaluation set.

Several directions can be further explored in the future: (i) We plan to design a method that can extract question patterns automatically, using existing labeled question patterns and KB as weak supervision. As we discussed in the experiment part, how to mine high-quality question patterns is worth further study for the QA task; (ii) We plan to integrate an ad-hoc NER into our KB-QA system to alleviate the entity detection issue; (iii) In fact, our proposed QA framework can be generalized to other intelligence besides knowledge bases as well. Any method that can generate answers to questions, such as the Web-based QA approach, can be integrated into this framework, by using them in the question translation component.

References

Yoav Artzi and Luke S. Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *EMNLP*, pages 421–432.

Yoav Artzi, Nicholas FitzGerald, and Luke S. Zettlemoyer. 2013. Semantic parsing with combinatory categorial grammars. In *ACL (Tutorial Abstracts)*, page 2.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL*, pages 423–433.

James Clarke and Mirella Lapata. 2010. Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441.

Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *ACL*.

Cristina Espana-Bonet and Pere R. Comas. 2012. Full machine translation for factoid question answering. In *EACL*, pages 20–29.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545.

Anthony Fader, Luke S. Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL*, pages 1608–1618.

Daniel Gerber and Axel-Cyrille Ngonga Ngomo. 2011. Bootstrapping the linked data web. In *ISWC*.

Daniel Gerber and Axel-Cyrille Ngonga Ngomo. 2012. Extracting multilingual natural-language patterns for rdf predicates. In *ESWC*.

Google. 2013. Freebase. In *http://www.freebase.com*.

Aditya Kalyanpur, Siddharth Patwardhan, Branimir Boguraev, Adam Lally, and Jennifer Chu-Carroll. 2012. Fact-based question decomposition in deepqa. *IBM Journal of Research and Development*, 56(3):13.

Tom Kwiatkowski, Luke S. Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *EMNLP*, pages 1223–1233.

Tom Kwiatkowski, Luke S. Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *EMNLP*, pages 1512–1523.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP*, pages 1545–1556.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *ACL*, pages 590–599.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Thomas Lin, Mausam, and Oren Etzioni. 2012. Entity linking at web scale. In *AKBC-WEKEX*, pages 84–88.

Raymond J. Mooney. 2007. Learning for semantic parsing. In *CICLing*, pages 311–324.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*, pages 160–167.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *ACL*, pages 933–943.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *HLT-NAACL*.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *ACL*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI, Vol. 2*, pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pages 658–666.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*, pages 678–687.