# Integrating Subject, Type, and Property Identification for Simple Question Answering over Knowledge Base

**Wei-Chuan Hsiao, Hen-Hsen Huang** and **Hsin-Hsi Chen**
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
weichuanhsiao@gmail.com; hhhuang@nlg.csie.ntu.edu.tw; hhchen@ntu.edu.tw

## Abstract

This paper presents an approach to identify subject, type and property from knowledge base for answering simple questions. We propose new features to rank entity candidates in KB. Besides, we split a relation in KB into type and property. Each of them is modeled by a bi-directional LSTM. Experimental results show that our model achieves the state-of-the-art performance on the SimpleQuestions dataset. The hard questions in the experiments are also analyzed in detail.

## 1 Introduction

With the popularity of the Internet, more and more new information is generated every day. The information may be stored in unstructured data, such as Wikipedia, which is presented as an article or web page, or in the form of structured data. Knowledge base (KB), such as Freebase (Bollacker et al., 2008) and DBpedia (Lehmann et al., 2015), is very popular. The information in the KB is a description of the relationship between two entities and is stored in the form of (subject, relation, object) triple. In KB, each entity is represented by a unique id, for example, J.K. Rowling's mid (Machine ID) in Freebase is "m.042xh".

With these large-scale open-domain KBs, it is important to access the knowledge efficiently and effectively to meet what users need. The most direct and close to people's life is question answering (QA) system in natural language. People can ask any questions in their familiar languages, and then use the QA system to get answers from the web resources. Current QA research is often based on KB to find appropriate triples for answering question, which is called QA over KB.

QA systems can be classified by the form of the question. There are two categories of questions in QA system, i.e., simple questions and complex questions. Simple questions can be answered by exactly one triple in the KB. For example, the question "Who is the author of Harry Potter?" can be answered by triple "(Harry Potter, author, J.K. Rowling)". Although this category is "simple" question, retrieving a triple from the KB is not a trivial task due to the billions of facts in the KB. Complex questions contain more restrictions. These questions may involve two or more triples in the KB, or have other semantic constraints to restrict the answers to a smaller set. For example, "the first" in the question "What is the name of the first Harry Potter novel?" restricts that there is only one answer. Previous researches showed that simple questions are the more common category in community QA websites (Fader et al., 2013). This paper focuses on factoid simple question-answering over Freebase.

Simple question can be answered with the object of one KB triple. Thus, the systems only need to find the subject and relation of the triple which can describe the question properly. The issues of simple QA are the identification of the subject entity in a question, and the resolution of the gap between the natural language expression in the question and the relation description in the KB. After a QA system receives users' questions, it needs to transform a question into a KB query, e.g. SPARQL. The question can then be transformed into the KB query, which can access KB to get the answer.



*Proceedings of the The 8th International Joint Conference on Natural Language Processing*, pages 976–985,
Taipei, Taiwan, November 27 – December 1, 2017 

Previous simple QA model often adopts a two-step paradigm. Entity-linking step identifies the subject entities in questions, and forms the candidate entity set and relation set. The candidate relation set is formed by all the relations which have connections with any entity in candidate entity set. Relation-finding step further identifies a proper relation from the candidate relation set. Dai et al. (2016) propose a neural-network based two-step approach to simple QA over Freebase, and formulate the task into a probabilistic form. Given a question $q$, the first step is to find the candidate relation $r$ with high probability $P(r|q)$. The second step is to find the subject $s$ with high $P(s|r,q)$. As a result, the object in the KB triple which contains the subject $s$ and the relation $r$ with the highest $P(s,r|q)$ is the answer.

The relation in KB triple has hierarchical structure: domain-type-property. For example, in "people.person.place_of_birth", "place_of_birth" is the property used to present the birth place of a person. The type of this relation is "person", and the domain is "people". In Freebase, many relations have the same property, but are in different types. For example, both "film.film.genre" and "music.artist.genre" have the property "genre", but they are under different domains and types. These two relations are regarded as the same if we only consider the property. The major issue is: they are different relations although they have the same meaning. On the other hand, if the whole relation is considered as a class, the distribution of relations is getting sparser. The past two-step approaches cannot distinguish the subtlety in the structured relation.

In this paper, we will propose a novel three-step approach, including subject, type and property identification steps, to deal with the hierarchical structure of the relation. Our approach introduces new features in subject identification to distinguish similar entities from different aspects. Moreover, splitting relation into type and property predicts relation more precisely. Experimental results show our model outperforms the existing models and achieves a state-of-the-art accuracy of 76.7%.

This paper is organized as follows. Section 2 introduces related works of QA system. Section 3 presents our three-step paradigm. Section 4 shows the experiments on the SimpleQuestions dataset (Bordes et al., 2015) and compares ours with previous works. We also discuss the importance of each component of our system. Section 5 analyzes the errors in the experiments. Section 6 concludes the remarks.

## 2 Related Works

Previous simple QA models can be divided into two categories. The first one is based on semantic parsing, which maps a question to its logical form. Then, the logical form can be transformed to SPARQL for KB retrieval.

Berant et al. (2013) present a semantic parser that does not need to be trained through the annotated logical form. They construct a lexicon that maps natural language phrases to KB relations by aligning large text corpus with Freebase. Candidate logical forms can be obtained by this lexicon and the other bridging operations.

Berant and Liang (2014) propose a semantic parser via paraphrasing. They use the intermediate question to deal with the problem of the mismatch between input question and its logical forms.

Yih et al. (2015) treat a question as a query graph, which can be directly mapped to its logical form. Semantic parsing is then equivalent to finding a sub-graph of the KB which can represent the question.

The semantic parsing approach which often requires the human annotated logical form may increase the cost of obtaining training data. Although the use of rule-based method to generate logical forms can reduce the use of annotated data, it limits the application domain. The second approach is information extraction, which needs only question-answer pairs for training. This method retrieves some candidate answers from KB for each question, and ranks the candidates through several features. Deep neural network models are often employed for deriving the vector representations of questions and the KB elements.

The Memory Network based QA system is proposed by Bordes et al. (2015). By embedding all of the KB elements and questions in the same vector space, the system can deal with the relationship between input language and the KB language.

Glob and He (2016) propose a character-level, attention-based encoder-decoder QA model. Their

model embeds the questions and KB elements by a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) encoder and a CNN-based encoder, respectively. And a LSTM decoder with attention mechanism is used to predict the appropriate answer entity and relation.

Dai et al. (2016) formulate the QA problem into a probabilistic form. Given a question, the answer is the triple in KB containing the subject and relation with the highest conditional probability. They first use a focused pruning method to tag the span in question which is most probable to be the subject entity and gets the candidate answer triples. Then they use a relation network and a subject network, both are a two-layer bidirectional-GRU model, to get the similarity scores between candidate triples and question. We modify their probabilistic form with the splitting of the relation part $r$ into type $t$ and property $p$.

Dong et al. (2015) introduce a multi-column CNNs (MCCNNs) to analyze the question in three different aspects: answer path, answer context and answer type. The system represents the question in three low-dimensional vectors, each of them then matches to one of the answer aspect to derive the scores of candidates.

Yin et al. (2016) also use the CNN-based approach to implement the QA system. They use a word-level CNN with attentive max-pooling to model the relationship between KB relations and question pattern. They also add an active linker, which is similar to the focused pruning method in Dai et al. (2016), to reduce the number of candidates and improve the performance significantly.

## 3 Methodology

Our method includes subject identification, type identification, and property identification steps, as shown in Figure 1. We first give an overall picture and then describe each of them in deep in the following sections.

### 3.1 A Three-Step Paradigm

The relation in a Freebase triple has a hierarchical structure in three levels: domain, type, and property. Many relations have the same property but they are in different types and domains. For example, both

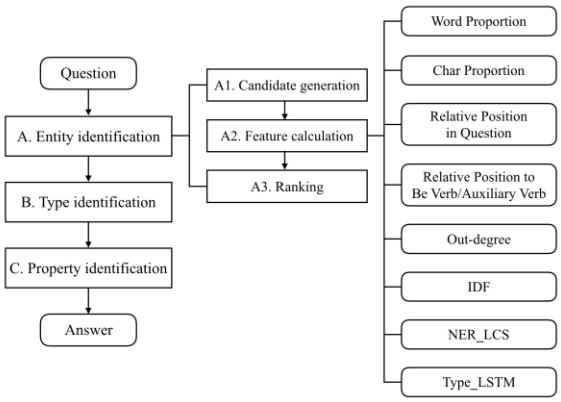


Figure 1: System Overview.

the relations "wine.wine.color" and "roses.roses.color" are used to describe the color of things, but they are in type "wine" and "roses", respectively. We separate a relation into type and property parts to understand the question meaning more precisely.

Given a KB $\mathcal{K}$ and a question $q$, the three-step paradigm aims at finding a KB triple containing *s*, *t*, and $p$ with the highest probability $P(s,t,p|q)$, where *s* is a subject, and *t* and $p$ are type and property of a structured relation, respectively. The process is formulated by Equations (1) and (2). The answer of the question is the object *o* of the triple (*s*, *r*, *o*) in $\mathcal{K}$.

$$P(s,t,p|q) = P(s|q) \cdot P(t|s,q) \cdot P(p|s,t,q) \quad (1)$$

$$s^*,t^*,p^* = \underset{s,t,p\in\mathcal{K}}{\operatorname{argmax}} P(s,t,p|q) \quad (2)$$

### 3.2 Entity Identification

The first step is to find potential entities in the question, and link them to KB. We use the Freebase subset FB5M in the SimpleQuestions dataset as the KB. We retrieve all candidate subject entities in question from the KB and calculate their entity linking scores on account of eight features, as in steps A1-A3 of Figure 1.

To find candidate entities for a question, we extract mentions in the question by maximum string matching. A mention that matches an entity name or an entity alias is extracted. When multiple matches are found in an overlapped span, the longest one is taken. And the mention is filtered if it is a stop word or a number that contains less than four

digits. All the entities in $\mathcal{K}$ that have the same names or aliases as the mentions form the candidate entity set.

After obtaining all the candidate entities, we assign each of the candidates a linking score. The score is measured by a learning-to-rank model with eight features in different aspects. The eight features are described as follows.

**Word Proportion** (Yin et al., 2016): We compute the proportion as the length of candidate entity divided by the length of the question. The length is measured by the number of words. The longer the matched entity name is, the more likely it is a subject.

**Char Proportion**: Similar to word proportion, the length is computed by the number of characters instead.

**Relative Position in Question** (Yin et al., 2016): The relative position of a candidate entity is the position of its last token divided by the length of the question (in words). That models an observation: most entity is far from the beginning of the question.

**Relative Position to Be Verb/Auxiliary Verb**: Subject entity tends to be close to and behind Be verb/auxiliary verb in a question. We take the subtraction of the Be verb/auxiliary verb position and the first token of candidate entity position as $\alpha$. If the candidate entity position is before the Be verb/auxiliary verb, this feature is $\alpha$, which is a negative number. Otherwise, this feature is $1/\alpha$, which is a positive number. If no such verbs exist, this feature is set to 0

**Out-degree**: The number of out-going links of the entity in the KB is taken as Out-degree feature. The more the number of links the entity has, the higher the feature value is and the entity in the KB is more informative. The direction of links from subject to object represents the impact of the entity.

**IDF**: We take each question in SimpleQuesions training set as a document, and compute the inverse document frequency of the entity. The higher the value is, the more specific the entity is.

**NER_LCS**: We use the LSTM-CRF named entity recognition (NER) tagger[1] (Lample et al., 2016) to find the span from the question that is most likely to be a subject entity, and compute the length of the longest common subsequence (in characters) between the candidate entity and the words in the tagged span. The tagger consists of an embedding layer, a bidirectional-LSTM layer, and a conditional random field layer to predict the label for each word. The higher the NER_LCS value is, the more possible the candidate is a subject entity.

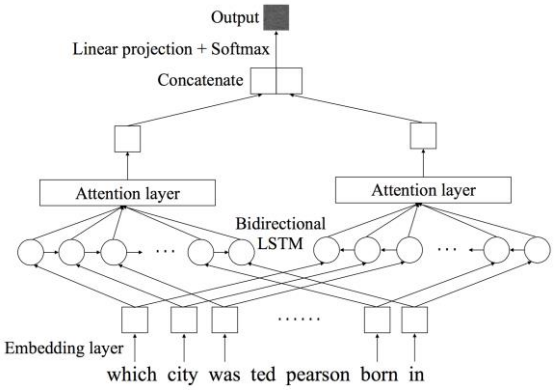


Figure 2: Structure of Type_LSTM in entity identification.

**Type_LSTM:** Each entity has entity types in Freebase to describe its characteristics, e.g., entity "Alex Golfis" belongs to types "person", "actor", and "deceased_person". There are total 500 types for entities in FB5M. For a question, we estimate the types of its subject. A bidirectional-LSTM with attention is trained to derive the probability distribution of each question over the 500 types. The structure of the network is shown in Figure 2. The training input of the LSTM is the whole questions and the types of their ground truth entities, and the training objective is categorical cross entropy. The test input is a question, and the output is a probability distribution of the question over the 500 types. We can get the type information of the candidate entity from Freebase, and sum over all dimensions corresponding to the types to get this feature. It means how probable the question is in some types.

With the above eight features, support vector machine for ranking ($SVM^{rank}$) (Joachims, 2006) is trained to combine these features and derive a score $u_1(s, q)$ for each candidate entity $s$ in the question $q$. Then each entity has the probability $P(s|q)$:

[1] https://github.com/glample/tagger .

$$P(s|q) = \frac{\exp(u_1(s,q))}{\sum_{s'} \exp(u_1(s',q))} \quad (3)$$

where $s'$ is an entity in the candidate entity set. For the entities in the KB but not in the candidate entity set, their $P(s|q) = 0$.

### 3.3 Type Identification

Given a question $q$, the type network determines the type of the relation most likely to be answered. We use $E(t)$ to represent the embedding of the type $t$, and use $g_1(q)$ to represent the vector of question $q$. Our goal is to make the cosine similarity between $g_1(q)$ and the correct type embedding $E(t^*)$ higher than the cosine similarities between $g_1(q)$ and the other types.

The network structure is shown in Figure 3. First, the question vector $g_1(q)$ is generated by a bidirectional-LSTM model. We change the words in $q$ to lowercase and remove the punctuation. Then we put the words into the embedding layer. After the bidirectional-LSTM layer, the two vectors in left and right directions are concatenated together. The concatenated vector is put into a linear projection layer with sigmoid, and the question vector $g_1(q)$ is generated. After that, we calculate the similarity between $g_1(q)$ and different type embeddings $E(t)$. The embedding $E(t)$ is trained with the bidirectional-LSTM model, and has the same dimension as $g_1(q)$. $E(t)$ and $g_1(q)$ are in the same space so that the cosine similarity $u_2(t,q)$ can be computed by Equation (4).

$$u_2(t,q) = \cos(g_1(q), E(t)) \quad (4)$$

The probability of type $t$ given the question $q$ and the candidate entity set is defined as follows:

$$P(t|s,q) = \begin{cases} \frac{\exp(u_2(t,q))}{\sum_{t'} \exp(u_2(t',q))}, & if\ t\ has\ link\ with\ s \\ 0, & otherwise \end{cases} \quad (5)$$

where $t'$ is a type in the candidate type set. To reduce the computation and the number of candidate types, only those types which have links with an entity in the candidate entity set is chosen into the candidate type set. $P(t|s,q)$ is set to 0 for those types which are impossible to be the correct answer due to the lack of links with any candidate entity.

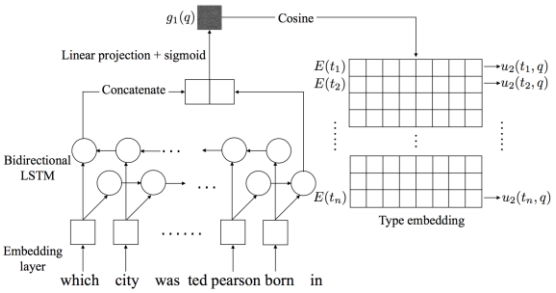


Figure 3: Type identification network.

For training the type network, the hinge loss with negative samples is the objective function to be minimized.

$$\ell(\theta_t) = \sum_{i=1}^{N_t} \max(0, m_t - u_2(t^*,q) + u_2(t_i,q)) \quad (6)$$

where $\theta_t$ is the parameters to learn, $N_t$ is the number of negative samples, $m_t$ is the margin, $t^*$ is the correct type, and $t_i$ is the type randomly sampled from all types except $t^*$.

### 3.4 Property Identification

Given a question $q$, property identification determines the property of the relation most likely to be answered. We use $E(p)$ to represent the embedding of the property $p$, and $g_2(q)$ to represent the vector of question $q$. Similar to the type network, our goal is to make the cosine similarity between $g_2(q)$ and the correct property embedding $E(p^*)$ higher than the cosine similarities between $g_2(q)$ and the other properties.

The structure of the property network is the same as that of the type network, which is shown in Figure 3, but all the weights and embedding matrixes are not shared with the type network. After deriving $g_2(q)$ and $E(p)$, we compute the cosine similarity $u_3(p,q)$:

$$u_3(p,q) = \cos(g_2(q), E(p)) \quad (7)$$

The probability of property $p$ given question $q$, candidate entity $s$, and candidate type $t$ is defined as follows:

$$P(p|s,t,q) = \begin{cases} \frac{\exp(u_3(p,q))}{\sum_{p'} \exp(u_3(p',q))}, & if\ p\ has\ link\ with\ s,t \\ 0, & otherwise \end{cases} \quad (8)$$

where $p'$ is a property in the candidate property set. To reduce the computation and the number of candidate properties, only the properties which have links with an entity in the candidate entity set and

| Method | Accuracy (%) |
|---|---|
| (Bordes et al., 2015) | 63.9 |
| (Dai et al., 2016) | 75.7 |
| (Yin et al., 2016) | 75.9 |
| (Golub and He, 2016) | 70.3 |
| Our approach (probability) | 76.1 |
| Our approach (sum) | **76.7** |

Table 1: Results on the SimpleQuestions test data.

| N | (Yin et al., 2016) | | Our approach |
|---|---|---|---|
| | Passive | Active | |
| 1 | 56.6 | 73.6 | 80.9 |
| 5 | 71.1 | 85.0 | 90.2 |
| 10 | 75.2 | 87.4 | 92.2 |
| 20 | 81.0 | 88.8 | 93.7 |
| 50 | 85.7 | 90.4 | 95.1 |
| 100 | 87.9 | 91.6 | 96.0 |

Table 2: Hit rates of the ground-truth entity.

belongs to a type in the candidate type set are chosen into the candidate property set. $P(p|s,t,q)$ is set to 0 for those properties which are impossible to be the correct answer due to the lack of links with any candidate entity and type.

For training property network, the hinge loss with negative samples is the objective function to be minimized.

$$\ell(\theta_p) = \sum_{i=1}^{N_p} \max(0, m_p - u_3(p^*, q) + u_3(p_i, q)) \quad (9)$$

where $\theta_p$ is the parameters to learn, $N_p$ is the number of negative samples, $m_p$ is the margin, $p^*$ is the correct property, and $p_i$ is the property randomly sampled from all properties except $p^*$.

## 4 Experiments

The SimpleQuestions[2] dataset (Bordes et al., 2015), which contains 75,910 training data, 10,845 validation data, and 21,687 test data, is adopted in the experiments. The evaluation is the same as in Bordes et al. (2015). The predicted answer is correct when the subject-relation pair is the same as the correct answer. We train our model on the training set. The validation set is used for early stop and parameter tuning. The test set is used for evaluation.

### 4.1 Experimental Setup

The number of negative samples used in SVM$^{rank}$ is set to 5. Other parameters for SVM$^{rank}$ are C = 0.1, epsilon = 0.01, and loss function option = 2. The word embeddings used in each neural network is initialized with the pre-trained GloVe (Pennington et al., 2014) with the dimension of 300. All the networks are optimized by mini-batch and Adam (Kingma et al., 2014) with the learning rate 0.001.

The TYPE_LSTM in entity identification step has a drop rate of 0.2. The hidden size of LSTM is 500. Batch size is 128. The parameters of type network and property network are the same. Maximal question length is 25. Mini-batch size is 100. The type and property embeddings are 500-dimensional with randomly initialized. The hidden size of LSTM is 500. Both hinge loss margins $m_t$ and $m_p$ are 0.4. The numbers of negative samples $N_t$ and $N_p$ are 65.

### 4.2 Overall Results

Table 1 shows the performances of our model compared with the other four methods. Bordes et al. (2015) use a memory network. Dai et al. (2016) employ a conditional focused neural-network based approach. Yin et al. (2016) apply attentive convolutional neural network with the passive or active linker. Golub and He (2016) use the character-level encoder-decoder framework. In our approach, "probability" means the outcome is the triple with the highest probability computed by Equation (1). We find that the subject entity and the property are more important than the type. The approach "sum" combines the scores $u_1(s,q)$, $u_2(t,q)$, and $u_3(p,q)$ by weighted summation, and selects the triple with the highest weighted sum. The weights are entity:type:property = 4:1:3, which are tuned on the validation data.

Our "probability" approach outperforms all previous models on FB5M, including the previous best model by Yin et al. (2016). The "sum" approach even significantly outperforms the "probability" approach on McNemar's test ($p < 0.01$).

### 4.3 Entity Identification Results

This section discusses the performance of the entity identification step. The hit rates of the top $N$ entities are shown, which means the coverage of

[2] https://research.fb.com/downloads/babi/ .

the ground truth entity by top-$N$ results. $N \in \{1,10,20,50,100\}$. Entity linker proposed by Yin et al. (2016) has two versions, passive and active linker. The difference between them is that they use the longest consecutive common subsequence between question and KB entity names to get candidates in the passive linker. On the other hand, the active linker first gets the span from the question that is most likely to be a subject entity by sequential labeling, and the linker uses the labeled span to search the candidate entity from KB. The number of the candidates in the active linker is less than the passive linker, because the active linker uses specific span in question to search candidates, and entities not in this span are filtered out.

Table 2 shows our performance in entity identification compares to Yin et al. (2016). Our entity linker outperforms their approach by over 7% in top-1 result using FB5M as background KB. We have six new features to score the entities in different aspects. The features "Char Proportion" and "Relative Position to Be Verb/Auxiliary Verb" can analyze the entities by their surface form in question, e.g., name and relative position. The features "Out-degree" and "Type_LSTM" can distinguish entities even if they have the same name. The "IDF" feature can kick out common entities and keep the more important ones. And "NER_LCS" feature has the similar effect to the active linker of Yin et al. (2016), but our feature can withstand the wrong subject entity prediction in sequential labeling, because we do not filter out any candidates, we give them the lower score instead.

## 4.4 Importance of Entity Identification Features

In this section, we discuss the importance of each feature in entity identification in three ways. We first show the performances with a single feature. And then, we consider the performances with one feature or a group of features being removed.

### Performance with Single Feature

Table 3 shows the hit rates of the top $N$ entities generated by a single feature. The feature "NER_LCS" has the best performance because this feature contains a part of information from the four surface

| Fetures \ N | Hit rates @ N | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 5 | 10 | 20 | 50 | 100 |
| Word Prop. | 41.0 | 62.7 | 70.5 | 77.6 | 85.8 | 90.3 |
| Char Prop. | 55.5 | 73.7 | 78.9 | 83.4 | 88.9 | 91.7 |
| Rel. Position in Q | 16.7 | 36.5 | 47.7 | 60.9 | 77.0 | 86.4 |
| Rel. Position to BeV./AuxV. | 30.2 | 44.2 | 52.6 | 61.4 | 75.7 | 84.2 |
| Out-degree | 15.0 | 39.6 | 52.5 | 67.2 | 82.6 | 89.5 |
| IDF | 43.4 | 68.9 | 76.4 | 82.5 | 88.5 | 91.6 |
| NER_LCS | 59.5 | 77.0 | 81.5 | 85.5 | 90.1 | 92.6 |
| Type_LSTM | 44.1 | 68.8 | 77.0 | 85.3 | 91.6 | 94.4 |

Table 3: Hit rates generated by a single feature in entity linker.

form features: "Word Proportion", "Char Proportion", "Relative Position in Question", and "Relative Position to Be Verb/Auxiliary Verb". The NER tagger labels the span of possible position of the subject entity in a question, and thus "NER_LCS" contains the position information. Moreover, the length of the longest common subsequence between the candidate entity and the words in the span gives the length proportion information.

The feature "Out-degree" has the lowest performance in $N=1$ hit rate, because the entities with many out-going links sometimes represent that they are general entities. For example, the entity "album (m.02lx2r)" in question "Which genre of album is harder……faster?" has 323,467 out-going links in FB5M, but the subject entity "harder……faster (m.01jp8ww)" has only 5 links. Although this feature sometimes highlights the general entities, it can distinguish entities that have the same names with the help of other features.

Table 3 also shows that the performance of "Relative Position to Be Verb/Auxiliary Verb" is almost twice the performance of "Relative Position in Question". It shows that the observation about position "subject entity tends to be close to and behind Be verb/auxiliary verb in a question" is more appropriate, and the Be verb/auxiliary verb plays an important role in identifying entities.

### Performance without One of the Features

Table 4 shows the hit rates of the top $N$ entities generated by removing one of the eight features from the entity linker. First, we can see that the performances without one of the first four features are

only reduced by about 1%, because the feature "NER_LCS" contains a part of the information from them, and it can make up for the removal of them. And then, we can find that although the performance of the feature "Out-degree" has the lowest accuracy in Table 3, its removal affects the performance by more than 2%.

The removal of "Type_LSTM" has the greatest impact. The result with this feature outperforms the result without it by 6%. The comparison shows the importance of the entity types which can help us get deeper meanings of an entity. And the "Type_LSTM" is difficult to make up by other features.

### Performance without a Group of Features

Table 5 shows the hit rates and overall accuracy of our "probability" approach generated by removing a group of features from the entity linker. We group the features with similar effect together. The first four features in Table 3 are about surface forms of entities, and thus they are put together. The "Individual features" group contains features "Out-degree" and "Type_LSTM", which can distinguish entities with the same name. The features "IDF" and "NER_LCS" are grouped into "features of specificity", which can identify the more likely subject entities in the question and kick out common entities.

The removal of individual features reduces the performance of entity identification by more than 20%. This result shows the importance of identifying entities with the same name. Otherwise, these entities would have the same score. However, the removal of these features only reduces the overall accuracy of 0.7%, because type identification step can make up a part of the removed "Type_LSTM" information. For example, the entity with name "Estonia" can be a country or a book, if one "Estonia" is a "book", it may not have a property in type "location", and thus we can distinguish these two types of "Estonia" a bit.

The removal of features of specificity affects the overall accuracy by 2.7%. These two features focus on more specific words and make the scores of the general entities, such as "album" or "movie", become lower. This behavior is important and cannot be replaced by the rest of the system.

## 4.5 Importance of Type Identification

Table 6 shows the importance of our type identification step. The models with type identification significantly outperform the models without type network on McNemar's test ($p < 0.01$), and increases the accuracies by at least 1%, no matter the final score is by probability or weighted sum. This shows that the type identification step can effectively handle the hierarchical structure of Freebase relations, and can better understand the semantics of the question.

| *N* / Fetures | Hit rates @ *N* | | | | | |
|---|---|---|---|---|---|---|
| | **1** | **5** | **10** | **20** | **50** | **100** |
| All features | 80.9 | 90.2 | 92.2 | 93.7 | 95.1 | 96.0 |
| w/o Word Prop. | 79.8 | 89.6 | 91.7 | 93.3 | 95.0 | 96.0 |
| w/o Char Prop. | 79.4 | 89.3 | 91.4 | 93.2 | 94.9 | 95.9 |
| w/o Rel. Position | 79.6 | 89.6 | 91.8 | 93.4 | 95.0 | 96.0 |
| w/o Rel. Position to BeV./AuxV. | 79.7 | 89.6 | 91.7 | 93.3 | 95.0 | 96.0 |
| w/o Out-degree | 78.8 | 88.6 | 90.9 | 92.8 | 94.6 | 95.7 |
| w/o IDF | 79.1 | 89.3 | 91.4 | 93.1 | 94.9 | 95.9 |
| w/o NER_LCS | 78.4 | 88.8 | 91.1 | 92.9 | 94.8 | 95.8 |
| w/o Type_LSTM | 74.9 | 88.0 | 90.7 | 92.7 | 94.4 | 95.6 |

Table 4: Hit rates generated by removing one feature from entity linker. "w/o" means the removal of the feature.

| *N* / Feature | Entity identification (Hit rates @ *N*) | | | | Overall (prob.) |
|---|---|---|---|---|---|
| | **1** | **5** | **10** | **100** | **Acc.** |
| All features | 80.9 | 90.2 | 92.2 | 96.0 | 76.2 |
| w/o Surface form features | 78.8 | 89.1 | 91.4 | 96.0 | 75.1 |
| w/o Individual features | 60.7 | 77.9 | 82.5 | 92.8 | 75.5 |
| w/o Features of specificity | 75.8 | 87.1 | 89.9 | 95.6 | 73.5 |

Table 5: Hit rates generated by removing group of features from entity linker.

| Settings | Accuracy (%) |
|---|---|
| Probability w/ type identification | 76.1 |
| Probability w/o type identification | 75.1 |
| Sum w/ type identification | 76.7 |
| Sum w/o type identification | 75.2 |

Table 6: Gain by the addition of the type identification step.

## 5 Error Analysis

We categorize some errors into the following types.

- **Entities with the same name**: The subject entity of the question "What is the place of birth of Sam Edwards?" is "Sam Edwards". Both entities "m.03kt3y" (an actress) and "m.042gjt" (a physicist) have the same name. This question does not have enough information to distinguish the two entities.
- **Deleted entity in Freebase**: Some questions' subject entities are deleted in the Freebase dump.
- **Similar properties**: Some relations are very similar, e.g., both "music.release.track" and "music.release.track_list" indicate tracks in an album.
- **Incomplete question**: Some questions in the dataset are not complete. For example, the question "What production company produced?" does not provide any entities.
- **Question from object-relation pair**: There are some questions formed by object-relation pairs of the triple. E.g., the question "Name a lawyer." is from triple (Charlie Herschel, people.person.profession, lawyer). We cannot find the subject of the triple from the question.
- **Typo**: Some questions contain typo. For example, the question "What is Roger Molliens gender?" should be "What is Roger Mollien's gender?"
- **Wrong answer**: Some answers are wrong. For example, the answer of the question "Where was David Armstrong born?" is the triple (Undisputed Comedy Series: Lil Rel, film.film.language, English). Obviously, it is not a correct answer to the question.
- **Controversial answer**: E.g., the relation of the question "Leo Bertos was born in what country?" is "people.person.nationality", but the more appropriate relation should be "people.person.place_of_birth", because people can have the naturalized citizenship.

## 6 Conclusion

We propose a three-step approach to identify subject, type and property from knowledge base (KB) for answering simple questions. Our ranking model with additional features outperforms the previous models in subject entity identification. The experiments also show that all entity features are important. Besides, by splitting the structured relation into type and property, our model benefits from understanding the question meaning more precisely. Our model achieves the state-of-the-art performance on the SimpleQuestions dataset. Error analysis shows that most errors come from the problematic questions in the dataset. Extension to complex questions will be explored.

## Acknowledgments

This research was partially supported by Ministry of Science and Technology, Taiwan, under grants MOST-104-2221-E-002-061-MY3, MOST-105-2221-E-002-154-MY3 and MOST-106-2923-E-002-012-MY3, and National Taiwan University under grant NTUCCP-106R891305.

## References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1533–1544, Seattle, Washington, USA.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD 2008)*, pages 1247–1249, Vancouver, BC, Canada.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL 2014)*, volume 7, page 92–102, Baltimore, USA.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075.

Zihang Dai, Lei Li, and Wei Xu. 2016. CFO: Conditional focused neural question answering with large-scale knowledge bases. In *Association for Computational Linguistics (ACL 2016)*, pages 800–810, Berlin, Germany.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of ACL-IJCNLP*, volume 1, pages 260–269, Beijing, China.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Association for Computational Linguistics (ACL 2013)*, pages 1608–1618. Sofia, Bulgaria.

David Golub and Xiaodong He. 2016. Character-level question answering with attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1598–1607, Austin, Texas.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Thorsten Joachims. 2006. Training Linear SVMs in Linear Time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD 2006)*, Philadelphia, Pennsylvania, USA.

Diederik P. Kingma and Jimmy Lei Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT 2016*, pages 260–270, San Diego, California.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal, 6(2):167–195.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543, Doha, Qatar.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Association for Computational Linguistics (ACL 2015),* pages 1321–1331, Beijing, China.

Wenpeng Yin, MoYu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING*, pages 1746–1756, Osaka, Japan.