Improving Semantic Parsing via Answer Type Inference

Semih Yavuz[1], **Izzeddin Gur**[1], **Yu Su**[1], **Mudhakar Srivatsa**[2] **and Xifeng Yan**[1]
[1]University of California, Santa Barbara, Department of Computer Science
[2]IBM Research
{syavuz,izzeddingur,ysu,xyan}@cs.ucsb.edu
msrivats@us.ibm.com

Abstract

In this work, we show the possibility of inferring the answer type before solving a factoid question and leveraging the type information to improve semantic parsing. By replacing the topic entity in a question with its type, we are able to generate an abstract form of the question, whose answer corresponds to the answer type of the original question. A bidirectional LSTM model is built to train over the abstract form of questions and infer their answer types. It is also observed that if we convert a question into a statement form, our LSTM model achieves better accuracy. Using the predicted type information to rerank the logical forms returned by AgendaIL, one of the leading semantic parsers, we are able to improve the F1-score from 49.7% to 52.6% on the WEBQUESTIONS data.

1 Introduction

Large scale knowledge bases (KB) like Freebase (Bollacker et al., 2008), DBpedia (Auer et al., 2007), and YAGO (Suchanek et al., 2007) that store the world's factual information in a structured fashion have become substantial resources for people to solve questions. KB-based factoid question answering (KB-QA) that attempts to find exact answers to natural language questions has gained much attention recently. KB-QA is a challenging task due to the representation variety between natural language and structural knowledge in KBs.

As one of the promising KB-QA techniques, semantic parsing maps a natural language question into its semantic representation (e.g., logical forms). It uses a logical language with predicates closely related to KB schema, and constructs a dictionary that maps relations to KB predicates. The problem then reduces to generating candidate logical forms, ranking them, and selecting one to derive the final answer.

Ranking	$F1$	# Improved Qs
AgendaIL	49.7	-
w/ Oracle Types@10	57.3	+234
w/ Oracle Types@20	58.7	+282
w/ Oracle Types@50	60.1	+331
w/ Oracle Types@All	60.5	+345

Table 1: What if the correct answer type is enforced? On WebQuestions, we remove those with incorrect answer types in the top-k logical forms returned by AgendaIL (Berant and Liang, 2015), a leading semantic parsing system, and report the new average F1 score as well as the number of questions with an improved F1 score.

In this work, we propose an answer type prediction model that can improve the ranking of the candidate logical forms generated by semantic parsing. The type of an entity, e.g., `person`, `organization`, `location`, carries very useful information for various down-stream natural language processing tasks such as co-reference resolution (Recasens et al., 2013), knowledge base population (Carlson et al., 2010), relation extraction (Yao et al., 2012), and question answering (Lin et al., 2012). Although the potential clues for answer type from the question has been employed in the recent work AgendaIL (Berant and Liang, 2015) at the lexical level, Table 1 suggests that there is yet a large room for further improvement by explicitly enforcing answer type. Inspired by this observation, we aim to directly predict the KB type of the answer from the question. In contrast to a small set of predefined types as used in previous answer type prediction methods (e.g., (Li and Roth, 2002)), KBs could have thousands of fine-grained types. Take "When did Shaq come into the NBA?" as a running example. We aim to predict the KB type of its answer as `SportsLeagueDraft`.[1]

The value of typing answers in a fine granularity can be appreciated from two perspectives: (1) Since each entity in a KB like Freebase has a few types, answer type could help prune answer candidates, (2) since each predicate in the KB has a unique type schema, answer type can help rank logical forms.

The key challenge of using answer types to rerank logic forms and hence their corresponding answers, is that it shall be done before the answer is found. Otherwise, there is no need to further infer its type. Inspired by the observation that the answer type of a question is invariant as long as the type of the topic entity (`Shaq`) remains the same (`DraftedAthlete`), we define **abstract question** as the question where the topic entity mention is replaced by its corresponding KB type. For the aforementioned example, the best candidate abstract question is "When did `DraftedAthlete` come into the NBA?" and the answer to this question is `SportsLeagueDraft`. Hence, we can reduce the answer type prediction task to abstract question answering.

The first step in our method is question abstraction, in which we generate candidate abstract questions based on the context of question and its candidate topic entities. We build a bidirectional LSTM network over the question that recursively computes vector representations for the past and future contexts of an entity mention. Based on these context representations, we predict the right type of the entity mention. Next, in order to better utilize the syntactic features of the question, we convert the question form into a normal statement form by using dependency tree of the question. For the running example, after performing the conversion, the abstract question becomes "`DraftedAthlete` come when into the NBA?" We then construct a bidirectional LSTM neural network over this final representation of the question and predict the type of the answer. Using the inferred answer type, we are able to improve the result of AgendaIL (Berant and Liang, 2015) on WebQuestions (Berant et al., 2013) from 49.7% to 52.6%.

2 Background

The knowledge base we work with consists of triples in subject-predicate-object form. It can be represented as $\mathcal{K} = \{(e_1, p, e_2) : e_1, e_2 \in \mathcal{E},\ p \in \mathcal{P}\}$, where $\mathcal{E}$ denotes the set of entities (e.g., `ShaquilleOneal`), and $\mathcal{P}$ denotes the set of binary predicates (e.g., `Drafted`). A knowledge base in this format can be visualized as a graph where entities are nodes, and predicates are directed edges between entities. Freebase is used in this work as the knowledge base. It has more than 41M entities, 596M facts, and 24K types.

Types are an integral part of the Freebase schema. Each entity e in Freebase has a set of categories (types) it belongs to, and this information can be obtained by checking the out-going predicates (`Type.Object.Type`) from e. For example, `ShaquilleOneal` has 20 Freebase types including `Person`, `BasketballPlayer`, `DraftedAthlete`, `Celebrity`, and `FilmActor`. For a specific question involving `ShaquilleOneal`, among these types, only a few will be relevant.

Each predicate in Freebase is from a subject entity to an object entity, and has a type signature. It has a unique expected types for its subject and object, independent of the individual subject and object entities themselves. For example, the predicate `People.Person.Profession` expects its subject to be of `Person` type and its object to be of `Profession` type.

3 Question Abstraction

The type of the topic entity rather than the entity itself is essential for inferring the answer type, which is invariant as the topic entity changes within the same class. For example, independent of which NBA player (with `DraftedAthlete` type) is the topic entity of this question "When did Shaq come into the NBA", the type of the answer is always go-

[1] KB type of answer ("1992 NBA Draft") in the context.

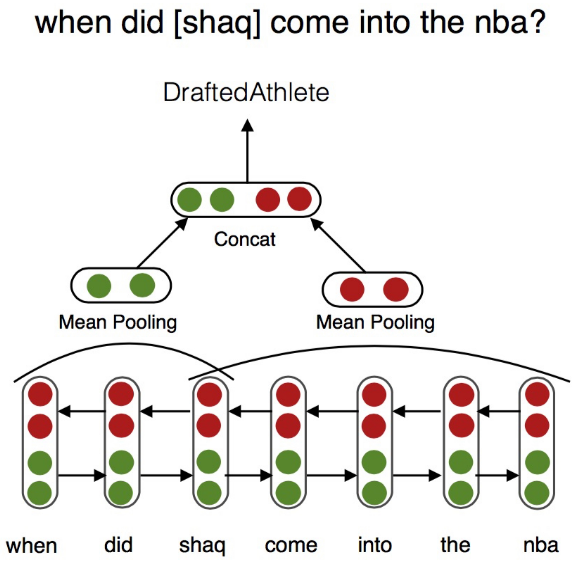

Figure 1: Bi-directional LSTM model for question abstraction. Green circles represent the forward sequence's hidden vectors, while the red circles denote the backward sequence's. `shaq` (the topic entity mention) is the single output node of the network.

ing to be `SportsLeagueDraft` in Freebase. Predicting this distinct type among the large number of candidate types in Freebase is a challenging task. We propose a two-step solution for this problem. In the first step, we compute a confidence score for each possible KB type for a given topic entity using a bidirectional LSTM network. The second step prunes candidate types using the entity type information in Freebase.

3.1 Formulation

Given a natural language question and its topic entity mention, question abstraction is to predict types of the mention in the question context. Formally, let $q = (x_1, x_2, \ldots, x_L)$ denote the question, m be the topic entity mention in q, and $T = \{t_1, t_2, \ldots, t_K\}$ the set of all types in KB. Given q and m, we compute a probability distribution $o \in \mathbb{R}^{K\times1}$ over T, where o_k denotes the likelihood of t_k being the correct type of m in q.

3.2 Scoring Topic Entity Types with LSTM

Model. We formulate question abstraction as a classification problem. A bidirectional LSTM network is built over q whose output is computed from the nodes that correspond to the words of m. Fig. 1 illustrates the model for the question "When did Shaq come into the NBA?"

Let $u(x) \in \mathbb{R}^{D\times1}$ denote the vector space embedding of word x. Forward and backward outputs $\overrightarrow{h}_l$, $\overleftarrow{h}_l \in \mathbb{R}^{D_h\times1}$ of bidirectional LSTM are recursively computed by

$$\overrightarrow{h}_l, \overrightarrow{c}_l = LSTM(u(x_l), \overrightarrow{h}_{l-1}, \overrightarrow{c}_{l-1}) \quad (1)$$

$$\overleftarrow{h}_l, \overleftarrow{c}_l = LSTM(u(x_l), \overleftarrow{h}_{l+1}, \overleftarrow{c}_{l+1}) \quad (2)$$

as described in Graves (2012), where $\overrightarrow{c_l}, \overleftarrow{c_l} \in \mathbb{R}^{D_h\times1}$ stand for LSTM cell states.

To encode the context of m to the final output, we apply an AVERAGE pooling layer when computing the output. For each output node $r \in [i, j]$ (i and j correspond to the starting and ending indices of m in q), we compute final forward and backward outputs by

$$\overrightarrow{v_r} = AVG(\overrightarrow{h_1}, \ldots, \overrightarrow{h_r}) \quad (3)$$

$$\overleftarrow{v_r} = AVG(\overleftarrow{h_r}, \ldots, \overleftarrow{h_n}), \quad (4)$$

where AVG stands for average pooling.

We take the average of outputs at each output node

$$\overrightarrow{v} = AVG(\overrightarrow{v_i}, \ldots, \overrightarrow{v_j}) \quad (5)$$

$$\overleftarrow{v} = AVG(\overleftarrow{v_i}, \ldots, \overleftarrow{v_j}) \quad (6)$$

as the forward and backward outputs of the whole network. The final representation v of the network is obtained by concatenating $\overrightarrow{v}$ and $\overleftarrow{v}$.

For question q, the probability distribution o over types is computed by

$$s(q) = W_{hy}v \quad (7)$$

$$o(q) = softmax(s(q)), \quad (8)$$

where $W_{hy} \in \mathbb{R}^{K\times(2D_h)}$ since v is the concatenation of two vectors of dimension D_h, where D_h is the hidden vector dimension.

Objective Function and Learning. Given an input question q with a topic entity mention m, LSTM network computes the probability distribution $o(q) \in \mathbb{R}^{K\times1}$ as in (8). Let $y(q) \in \mathbb{R}^{K\times1}$ denote the true target distribution over T for q, where

$y_k(q) = 1/n$ if t_k is a correct type, $y_k(q) = 0$ otherwise, and n is the number of correct types. We use the cross-entropy loss function between $y(q)$ and $o(q)$, and define the objective function over all training data as

$$J(\theta) = -\sum_q \sum_{k=1}^{K} y_k(q) \log o_k(q) + \frac{\lambda}{2} \|\theta\|^2 ,$$

where λ denotes the regularization parameter, and θ represents the set of all model parameters to be learned. We use stochastic gradient descent with RMSProp (Tieleman and Hinton, 2012) for minimizing the objective function.

3.3 Pruning

Let T_e represent the set of KB types for entity e. We define the set of candidate types for entity mention m as

$$C_m = \bigcup_e T_e,$$

where e is a possible match of m in KB. We only need to score the types in C_m. Once the hidden representation v is computed by LSTM, we use submatrix $W_{hy}[C_m]$ that consists of rows of W_{hy} corresponding to the types in C_m as the scoring matrix in (7). This returns the final scores for candidate types in C_m.

4 Conversion to Statement Form

The objective of the conversion is to canonicalize question form into declarative statement (subject-relation-object) form. We use a simple pattern-based method that relies on dependency tree[2] (Manning et al., 2014). It decides whether the sub-trees of the root need reordering based on their dependency relations[3].

Before obtaining the dependency tree, we retrieve named entity (NER) tags of the question tokens. We replace a group of question tokens corresponding a named entity with a special token, ENTITY, to simplify the parse tree. In Figure 2, the question is first transformed to "what boarding school did ENTITY go to?" Each question is represented by the root's dependency relations to its sub-trees in the original order, e.g., (dep, aux, nsubj, nmod). We cluster all these sequences and detect the patterns that appear at least 5 times in the training data. These patterns are then manually mapped to their corresponding conversion (pattern vs. mapping in Figure 2).

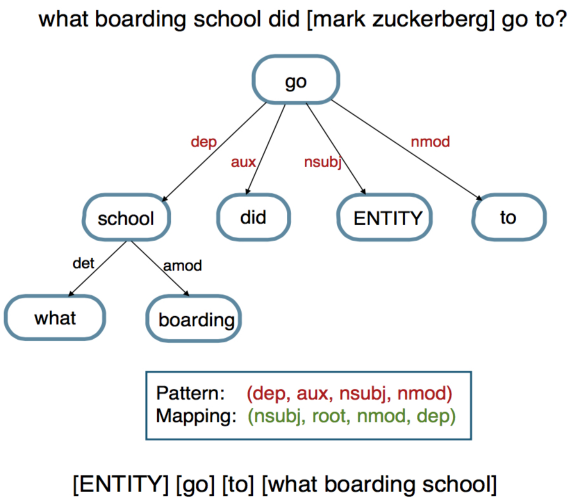
what boarding school did [mark zuckerberg] go to?
go
dep
aux
nsubj
nmod
school
did
ENTITY
to
det
amod
what
boarding
Pattern: (dep, aux, nsubj, nmod)
Mapping: (nsubj, root, nmod, dep)
[ENTITY] [go] [to] [what boarding school]

Figure 2: Conversion: red relations form the input pattern

Pattern	Conversion
(cop, nsubj) who was anakin skywalker?	(nsubj, root, cop) anakin skywalker was who
(dobj, aux, nsubj) what language does australians speak?	(nsubj, root, dobj) australians speak what language
(dobj, aux, nsubj, nmod) what did edward jenner do for a living?	(nsubj, root, dobj, nmod) edward jenner do what for a living
(nsubj, dobj) who played bilbo baggins?	(nsubj, root, dobj) who played bilbo baggins
(advmod, aux, nsubj) where did benjamin franklin died?	(nsubj, root, advmod) benjamin franklin died where

Table 2: Top-5 most common patterns with mappings.

Once the recomposition order of the sub-trees is determined by the conversion mapping, we finalize the reordering of the question tokens by keeping the order of words within the sub-trees same as the original order in the question. The example in Figure 2 becomes "ENTITY go to what boarding school" with its corresponding sub-tree conversion mapping (nsubj, root, nmod, dep). If no mapping is created for a pattern, we keep the order of the words exactly as they occur in the original question form.

The motivation behind conversion is to overcome the potential semantic confusion stemming from varities in syntactic structures. To exemplify, consider two hypothetical questions "who plays X in

[2] We use Stanford CoreNLP dependency parser

[3] http://universaldependencies.org

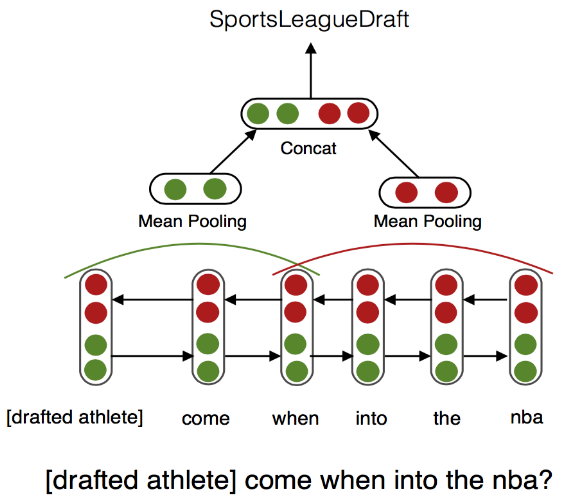

Figure 3: Bi-directional LSTM model over the final representation of the question. Green and red circles are corresponding to forward and backward hidden vectors, respectively. The output node is `when`.

Y?” and “who does Z play in Y?”, where X is a `FilmCharacter`, Y is a `Film`, and Z is a `FilmActor`, with answer types `FilmActor` and `FilmCharacter`, respectively. With conversion, we aim to transform second question into “Z play who in Y”, while leaving the first one as it is. Noting that the order of words affects the output of our answer type inference network, our intuition is to let the model distinguish better between such questions using their syntactic structure in this way.

5 Answer Type Prediction

Given a reordered question with topic entity mention m, and a topic entity type $t_e \in T$, our task is to predict a probability distribution $o \in \mathbb{R}^{K \times 1}$ over the answer types.

A topic entity type $t_e \in T$ is described as a set of words, $\{x_i\}$. Let $u(x_i) \in \mathbb{R}^{D \times 1}$ represent the vector space embedding of x_i, the representation of t_e is computed by the average encoding,

$$u(t_e) = \frac{1}{|\{x_i\}|} \sum_{x_i} u(x_i). \tag{9}$$

As the first step, we replace the words of entity mention m with topic entity type t_e, and obtain a new input word sequence r. t_e is treated as one word and encoded by Eq. 9. We construct a bidirectional LSTM network over this input sequence r, whose output node corresponds to the question word. The output of the network is a probability distribution over types denoting the likelihood of being the answer type. Figure 3 shows how the network is constructed for the running example. The same average pooling described in Section 3.2 is applied to obtain the final forward and backward output vectors $\overrightarrow{v}$ and $\overleftarrow{v}$ from the output node (this time, single output node) of network. The final output vector v for prediction is obtained by concatenating $\overrightarrow{v}$, and $\overleftarrow{v}$. The distribution o is computed by a standard softmax layer. The learning is performed by the same cross-entropy loss and objective function described in Section 3.2.

6 Reranking by Answer Type

In this section, we describe how to rerank logical forms based on our answer type prediction model.

Reranking Model. Let $l_1, l_2, \ldots, l_N$ be the logical forms generated for question q by a semantic parser, e.g., AgendaIL. Each logical form has a score from the semantic parser. Meanwhile, our answer type prediction model generates a score for the answer type of each logical form. Therefore, we can represent each logical form l_i using a pair of scores: the score from semantic parser and the score from our type prediction model. Suppose we know which logical forms are “correct”, using the two scores as input, we train a logistic regression model with cross-entropy loss to learn a binary classifier for predicting the correct logical forms. We rerank the top-k logical forms using their probability computed by the trained logistic regression model, and select the one with the highest probability. Finally, we run the selected logical form against KB to retrieve the answer. We select the optimal value of k from $[1, N]$ using the training data. For AgendaIL on WebQuestions, we find that $k = 80$ gives the best result.

Training Data Selection. We now discuss which logical forms are “correct”, i.e., how to select the positive examples to train the logistic regression model. Because a question can have more than one answer, we use the $F1$ score, the harmonic mean of *precision* and *recall*, to evaluate logical forms. We select all the logical forms with $F1 > 0$ as the set of positive examples. However, taking all the logical forms with $F1 = 0$ as negative examples will not work well. Even though the $F1$ score of a logical form is 0, its answer type could still be correct. Therefore, we use the following trick: If there is a positive example with answer type t, we do not treat any other logical form with answer type t as negative example. The logical forms having $F1 = 0$, with the aforementioned exception, are then selected as the final set of negative examples. Our empirical study shows this trick works well.

7 Experiments

In this section, we describe the datasets, model training, and experimental results.

7.1 Dataset and Evaluation Metrics

Datasets. To evaluate our method, we use the WebQuestions dataset (Berant et al., 2013), which contains 5,810 questions crawled via Google Suggest API. The answers to these questions are annotated from Freebase using Amazon Mechanical Turk. The data is split into training and test sets of size 3,778 and 2,032 questions, respectively. This dataset has been popularly used in question answering and semantic parsing.

The SimpleQuestions (Bordes et al., 2015) contains 108,442 questions written in natural language by English-speaking human annotators. This dataset is a collection of question/Freebase-fact pairs rather than question/answer pairs. The data[4] is split and provided as training(75,910), test(21,687), and validation(10,845) sets. Each question is mapped to the subject, relation, and object of the corresponding Freebase fact. This dataset is only used for training the question abstraction model.

Training Data Preparation. Since WebQuestions only provides question-answer pairs along with annotated topic entities, we need to figure out the type information, which can be used as training data. We obtain *simulated* types as follows: We retrieve 1-hop and 2-hop predicates r from/to annotated topic entity e in Freebase. For each relation r, we query $(e, r, ?)$ and $(?, r, e)$ against Freebase and retrieve the candidate answers r_a. The $F1$ value of each candidate answer r_a is computed with respect to the annotated answer. The subject and object types of the relation r with the highest $F1$ value is selected as the *simulated* type for the topic entity and the answer. When there are multiple such relations, we obtain multiple *simulated* types for topic entity and answer, one from each relation. We treat each of them as correct with equal probability.

Candidate Logical Forms for Evaluation. To obtain candidate logical forms, we train AgendaIL (Berant and Liang, 2015) on WebQuestions with beam size 200 using the publicly available code[5] by the authors.

Evaluation Metric. We report average $F1$ score of the reranked logical forms using the predicted answer types as the main evaluation metric. It is a common performance measure in question answering as questions might have multiple answers.

7.2 Experimental Setup

We use 50 dimensional word embeddings, which are initialized by the 50 dimensional pre-trained word vectors[6] from GloVe (Pennington et al., 2014), and updated in the training process. Hyperparameters are tuned on the development set. The size of the LSTM hidden layer is set at 50. We use RMSProp (Tieleman and Hinton, 2012) with a learning rate of 0.005 and mini-batch size of 32 for the optimization. We use a dropout layer with probability 0.5 for regularization. We implemented the LSTM networks using *Theano* (Theano Development Team, 2016).

Identifying Topic Entity. We use Stanford NER tagger (Manning et al., 2014) to identify topic entity span for both training and test data. For entity linking, annotated mention span is mapped to a ranked list of candidate Freebase entities using Freebase Search API for the test data. For the training data, we use the gold Freebase topic entity linkings of each question provided by WebQuestions, coming from its question generation process.

Question Abstraction. We first pre-train the LSTM model described in Section 3.2 on the SimpleQuestions dataset. Then, we update the pre-trained model on the training portion of WebQuestions data where the *simulated* topic entity types are used as true labels. We use the detected topic entity mentions to obtain candidate matching entities in the KB using Freebase Search API. We use top-

[4] http://fb.ai/babi.

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

[6] http://nlp.stanford.edu/projects/glove/

Model	F1
(Berant et al., 2013)	35.7
(Yao and Van Durme, 2014)	33.0
(Berant and Liang, 2014)	39.9
(Bao et al., 2014)	37.5
(Bordes et al., 2014)	39.2
(Yang et al., 2014)	41.3
(Dong et al., 2015b)	40.8
(Yao, 2015)	44.3
(Berant and Liang, 2015)	49.7
(Yih et al., 2015)	52.5
(Reddy et al., 2016)	50.3
(Xu et al., 2016)	**53.3**
(Yih et al., 2015) (w/ Freebase API)	48.4
(Yih et al., 2015) (w/o ClueWeb)	50.9
(Xu et al., 2016) (w/o Wikipedia)	47.1
Our Approach (w/o SimpleQuestions)	51.6
Our Approach	**52.6**

Table 3: Comparison of our reranking-by-type system with several existing works on WebQuestions.

3 entities returned for the pruning step of *Question Abstraction* on the test examples.

Answer Type Prediction. We train *Answer Type Prediction* model using the *simulated* topic entity and answer types for each question. We perform the answer type prediction on test data using the predicted topic entity type.

7.3 Results

Our main result is presented in Table 3. Our system adds 2.9% absolute improvement over AgendaIL, and achieves 52.6% in $F1$ measure. Yih et al. (2015) achieve 52.5% by leveraging ClueWeb and S-MART (Yang and Chang, 2015), an advanced entity linking system. Xu et al. (2016) achieve 53.3% by leveraging Wikipedia and S-MART. If tested without Clueweb/Wikipedia/S-MART, their F1 scores are 48.4% and 47.1%, respectively. When our method is tested without using SimpleQuestions data for pretraining question abstraction module, it attains F1 score of 51.6%.

In Table 4, we present some question examples where our method can select a better logical form. Take the question "who did [australia] fight in the first world war?" as an example. Our topic entity type prediction module returns `MilitaryCombatant`, `StatisticalRegion`, and `Kingdom` as the top-3 results for the type of "australia" in this question, which indicates that it exploits the context of this short question successfully. The abstract question is "[military combatant] fight who in the first world war?" for which our system returns `MilitaryCombatant`, `MilitaryConflict`, and `MilitaryCommander` as answer types with probabilities 0.73, 0.25, and 0.005, respectively, `MilitaryCombatant` is indeed the right answer type. This example shows the effect of abstraction in channeling the context in the most relevant direction to find the right answer type. In Table 5, we provide a comparison of the selected logical forms based on AgendaIL rankings and our rankings.

Method	F1	Gain	Loss
Base	50.3	69	47
Base + Conv	50.5	96	56
Base + Abs	52.2	184	87
Base + Abs + Conv	52.6	203	93
AgendaIL	49.7	-	-

Table 6: Ablation analysis of modules of our method. Gain/Loss columns denote the number of questions where the F1 score of our selected logical form is greater/less than that of the top ranked logical forms from AgendaIL.

7.4 Ablation Analysis

In this section, we evaluate the effect of individual components of our model. Note that the answer type prediction model described in Section 5 can work independently from question abstraction and form conversion. We develop the following variants i) Base, ii) Base + Conversion, iii) Base + Abstraction, iv) Base + Abstraction + Conversion, where Base corresponds to a model that infers answer types without employing abstraction or form conversion. We train/test each variant separately. Table 6 shows each component contributes and question abstraction does help boost the performance.

Suppose we perform answer type prediction without question abstraction, and feed "[australia] fight who in the first world war?" into the answer type prediction model (Base + Conversion). The predicted answer type is `Location`. Unfortunately, there is neither a 1-hop or 2-hop correct relation from/to `Australia` with the expected type `Location` nor a correct (with positive $F1$) candi-

Question	Topic Entity Type Prediction	Answer Type Prediction	AgendaIL Answer Type	F1 Gain
who inspired **obama**?	`InfluenceNode`	`InfluenceNode`	`UsVicePresident`	1.0
what are some books that **mark twain** wrote?	`Author`	`WrittenWork`	`InfluenceNode`	0.3
who won the **league cup** in 2002?	`SportsAwardType`	`SportsAwardWinner`	`SportsLeagueSeason`	1.0
what type of government does **france** use?	`Country`	`FormOfGovernment`	`Government`	1.0
where are the **new orleans hornets** moving to?	`SportsTeam`	`SportsFacility`	`Location`	1.0
who did **australia** fight in the first world war?	`MilitaryCombatant`	`MilitaryCombatant`	`MilitaryCommander`	0.4
what guitar does **corey taylor** play?	`Musician`	`MusicalInstrument`	`Organization`	0.33
what region is **turkey** considered?	`Location`	`AdministrativeDivision`	`Beer`	0.93
what country does **rafael nadal** play for?	`Athlete`	`Country`	`OlympicDiscipline`	1.0

Table 4: Example questions where our type prediction helps select a better logical form. The F1 gain shows the difference between the F1 score of the logical form we select and the top ranked logical form from AgendaIL.

Questions and Selected Logical Forms
1. what are some books that **mark twain** wrote? AgendaIL: (`MarkTwain - Influence.InfluenceNode.InfluencedBy - ?`) Ours: (`MarkTwain - Book.Author.WorksWritten - ?`)
2. what guitar does **corey taylor** play? AgendaIL: (`?  - Organization.Organization.Founders - CoreyTaylor`) Ours: (`CoreyTaylor - Music.GroupMember.InstrumentsPlayed - ?`)
3. what type of government does **france** use? AgendaIL: (`France - Government.GovernmentalJurisdiction.Government - ?`) Ours: (`France - Location.Country.FormOfGovernment - ?`)

Table 5: Comparison of selected logical forms for some examples. Logical forms are simplified and canonicalized into (subject - predicate - object) format for better readability, where `?` corresponds to answer nodes.

date logical form with the answer type `Location`. This shows that through question abstraction, a better logical form is selected for this question.

To exemplify another benefit of question abstraction, consider the question "where does [marta] play soccer?" The top 3 entity linkings via Freebase Search API for "marta" are `MetropolitanAtlantaRapidTransit-Authority`, `Marta`, and `SantaMarta`, where the correct entity is the second one. Our question abstraction system returns `FootballPlayer` as the top topic entity type prediction that is indeed corresponding to the correct entity. Utilizing the context via question abstraction we are able to recover useful information when the entity linking is uncertain.

Table 6 also shows that the conversion to statement form also helps, especially together with *Abstraction*. In the above example, the model without *Conversion* (Base + Abs) predicts the answer type for ''where does [football player] play soccer'' as `SportsFacility`, whereas the full model, considering `Conversion` as well, finds the answer type for ''[football player] play soccer where'' as `SportsTeam` which is the better type in this case.

7.5 Error Analysis

We present a further analysis of our approach by classifying the type inference errors made on randomly sampled 100 questions. 9% of the errors are due to inference at incorrect granularity (e.g., `City` instead of `Location`). 12% of the errors are the result of incorrect answer labels (hence incorrect answer types) or question ambiguity (e.g., "where is dwight howard now?"). 11% of them are incorrect, but acceptable inferences, e.g., `BookWrittenWork` instead of `BookEdition` for question "what dawkins book to read first?" 39% of the errors are due to the sparsity problem: They are made on questions whose answer type appears less than 5 times in the training data (e.g., `DayOfYear`). The remaining 29% of them are due to incorrect question abstraction. In most of the question abstraction errors, the predicted topic entity type is semantically close to the correct type. In other cases such as "what did joey jordison play in slipknot?" where we predict `FilmActor` as the topic entity type while `Musician` is the correct one. In these cases, the answer type inference is not able to correct the abstraction error. These 29% of errors also contain the entity linking errors.

8 Related Work

Freebase QA has been studied from two different perspectives: grounded QA systems that work directly on KBs and general purpose ungrounded QA systems. Kwiatkowski et al. (2013) generates KB agnostic intermediary CCG parses of questions which are grounded afterwards given a KB. Bordes et al. (2014) uses a vector space embedding approach to measure the semantic similarity between question and answers. Yao and Van Durme (2014), Bast and Haussmann (2015) and Yih et al. (2015) exploit a graph centric approach where a grounded subgraph query is generated from question and then executed against a KB. In this work, we propose a neural answer type inference method that can be incorporated in existing grounded semantic parsers as a complementary feature to improve ranking of the candidate logical forms.

Berant and Liang (2015) uses lambda DCS logical language with predicates from Freebase. In their approach, types are included as a part of unary lexicon for building the logical forms from natural language questions. However, no explicit type inference is exploited. We show that such information could indeed be useful for selecting logical forms.

There have been a series of studies investigating the expected answer type of a question in different contexts such as Li and Roth (2002), Lally et al. (2012), and Balog and Neumayer (2012). Most of these approaches classify the questions into a small set of types. Even when the set of classes is more fine-grained, e.g., 50 classes in Li and Roth (2002), they cannot be used for our purpose as it would require nontrivial mapping between these categories and a much larger number of KB types. Furthermore, these methods often rely on a rich set of hand crafted features and external resources.

Sun et al. (2015) uses Freebase types to learn the relevance of candidate answers to a given question via an association model. Their model directly ranks the answer candidates by utilizing types, whereas ours ranks the logical forms via predicting answer type. In this sense, we are able to take advantage of both logical form and type inference. Su et al. (2015) exploits answer typing to facilitate knowledge graph search, but their input is graph query instead of natural language question. They predict answer types using additional relevance feedback for graph queries, while our algorithm directly infers answer types from input questions. On the question abstraction side, our work is related to a recent study (Dong et al., 2015a) which classifies entity mentions into 22 types derived from DBpedia. They use a multilayer perceptron over a fixed size window and a recurrent neural network for the representations of context and entity mention, respectively. Instead, we use a bidirectional LSTM network to exploit the full context more flexibly.

9 Conclusion

In this paper, we present a question answer type inference framework and leverage it to improve semantic parsing. We define the notion of abstract question as the class of questions that can be answered by type instead of entity. Question answer type inference is then reduced to "question abstraction" and "abstract question answering", both of which are formulated as classification problems. Question abstraction is performed by exploiting the topic entity and its context in question via an LSTM network . A separate neural network is trained to exploit the abstraction to make the final question answer type inference. Our method improves the ranking of logical forms returned by AgendaIL on the WEBQUESTIONS dataset. In the future, we would like to investigate how the abstraction and explicit type inference can be incorporated in the early stage of semantic parsing for generating better candidate logical forms.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments, and Huan Sun for fruitful discussions. This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF09-2-0053, NSF IIS 1528175, and NSF CCF 1548848. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *International Semantic Web Conference (ISWC)*.

Krisztian Balog and Robert Neumayer. 2012. Hierarchical target type identification for entity-oriented queries. In *ACM International Conference on Information and Knowledge Management (CIKM)*.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *ACM International Conference on Information and Knowledge Management (CIKM)*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics (TACL)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD International Conference on Management of Data*.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. *ArXiv*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *ArXiv*.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *ACM International Conference on Web Search and Data mining (WSDM)*.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015a. A hybrid neural model for type classification of entity mentions. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015b. Question answering over freebase with multi-column convolutional neural networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alex Graves, 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 5–13. Springer Berlin Heidelberg.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Adam Lally, John M Prager, Michael C McCord, BK Boguraev, Siddharth Patwardhan, James Fan, Paul Fodor, and Jennifer Chu-Carroll. 2012. Question analysis: How watson reads a clue. *IBM Journal of Research and Development*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *International Conference on Computational Linguistics (COLING)*.

Thomas Lin, Mausam, and Oren Etzioni. 2012. No noun phrase left behind: Detecting and typing unlinkable entities. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Marta Recasens, Marie catherine De Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics (TACL)*.

Yu Su, Shengqi Yang, Huan Sun, Mudhakar Srivatsa, Sue Kase, Michelle Vanni, and Xifeng Yan. 2015. Exploiting relevance feedback in knowledge graph search. In *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *World Wide Web (WWW)*.

Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *World Wide Web (WWW)*.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *ArXiv*.

Tijmen Tieleman and Geoffrey E. Hinton. 2012. Lecture 6.5 - RMSProp, COURSERA: Neural networks for machine learning. *Technical Report*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structure learning algorithms applied to entity linking. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledged-based question answering. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *The North American Chapter of the Association for Computational Linguistics (NAACL)*.

Wen-tau Yih, MingWei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.