Abstract

The existing factoid QA systems often lack a post-inspection component that can help models recover from their own mistakes. In this work, we propose to cross-check the corresponding KB relations behind the predicted answers and identify potential inconsistencies. Instead of developing a new model that accepts evidences collected from these relations, we choose to plug them back to the original questions directly and check if the revised question makes sense or not. A bidirectional LSTM is applied to encode revised questions. We develop a scoring mechanism over the revised question encodings to refine the predictions of a base QA system. This approach can improve the $F_1$ score of STAGG (Yih et al., 2015), one of the leading QA systems, from 52.5% to 53.9% on WEBQUESTIONS data.

1 Introduction

With the recent advances in building large scale knowledge bases (KB) like Freebase (Bollacker et al., 2008), DBpedia (Auer et al., 2007), and YAGO (Suchanek et al., 2007) that contain the world’s factual information, KB-based question answering receives attention of research efforts in this area. Traditional semantic parsing is one of the most promising approaches that tackle this problem by mapping questions onto logical forms using logical languages CCG (Kwiatkowski et al., 2013; Reddy et al., 2014; Choi et al., 2015; Reddy et al., 2016), DCS (Berant et al., 2013; Berant and Liang, 2014, 2015), or directly query graphs (Yih et al., 2015) with predicates closely related to KB schema. Recently, neural network based models have been applied to question answering (Bordes et al., 2015; Yih et al., 2015; Xu et al., 2016a,b).

While these approaches yielded successful results, they often lack a post-inspection component that can help models recover from their own mistakes. Table 1 shows the potential improvement we can achieve if such a component exists. Can we leverage textual evidences related to the predicted answers to recover from a prediction error? In this work, we show it is possible.

Our strategy is to cross-check the corresponding KB relations behind the predicted answers and identify potential inconsistencies. As an intermediate step, we define question revision as a tailored transformation of the original question using textual evidences collected from these relations in a knowledge base, and check if the revised questions make sense or not. Figure 1 illustrates the idea over an example question “what did Mary Wollstonecraft fight for?”. Obviously, “what [area of activism] did [activist] fight for?” looks more consistent over “what [profession] did [person] fight for?”. We shall build a model that prefers the former one. This model shall be specialized for comparing the revised questions and checking which one makes better sense, not for answering the revised questions. This strategy differentiates
our work from many existing QA studies.

Given a question, we first create its revisions with respect to candidate KB relations. We encode question revisions using a bidirectional LSTM. A scoring mechanism over these encodings is jointly trained with LSTM parameters with the objective that the question revised by a correct KB relation has a higher score than that of other candidate KB relations. We have modified QA-LSTM and ATTENTIVE-LSTM (Tan et al., 2016) accordingly (See Section 4). However, so far the performance is not as good as the question revision approach.

## 2 Question Revisions

We formalize three kinds of question revisions, namely entity-centric, answer-centric, and relation-centric that revise the question with respect to evidences from topic entity type, answer type, and relation description. As illustrated in Figure 2, we design revisions to capture generalizations at different granularities while preserving the question structure.

Let \( s_r \) (e.g., Activist) and \( o_r \) (e.g., ActivismIssue) denote the subject and object types of a KB relation \( r \) (e.g., AreaOfActivism), respectively. Let \( \alpha \) (type.object.name) denote a function returning the textual description of a KB element (e.g., relation, entity, or type). Assuming that a candidate answer set is retrieved by executing a KB relation \( r \) from a topic entity in question, we can uniquely identify the types of topic entity and answer for the hypothesis by \( s_r \) and \( o_r \), respectively. It is also possible that a chain of relations \( r = r_1r_2\ldots r_k \) is used to retrieve an answer set from a topic entity. When \( k = 2 \), by abuse of notation, we define \( s_{r_1r_2} = s_{r_1} \), \( o_{r_1r_2} = o_{r_2} \), and \( \alpha(r_1r_2) = \text{concat}(\alpha(r_1), \alpha(r_2)) \).

Let \( m : (q, r) \rightarrow q' \) denote a mapping from a given question \( q = [w_1, w_2, \ldots, w_L] \) and a KB relation \( r \) to revised question \( q' \). We denote the index span of wh-words (e.g., “what”) and topic entity (e.g., “Mary Wollstonecraft”) by \([i_s, i_e]\) and \([j_s, j_e]\).

### Entity-Centric (EC)

Entity-centric question revision aims a generalization at the entity level. We construct it by replacing topic entity tokens with its second top-ranked candidate. For the running example, it becomes “what did [activist] fight for”. Formally, \( m_{EC}(q, r) = [w_{[1:j_s-1]}; \alpha(s_r); w_{[j_s+1:L]}] \).

### Answer-Centric (AC)

It is constructed by augmenting the wh-words of entity-centric question revision with the answer type. The running example is revised to “[what activism issue] did [activist] fight for”. Formally, we define it as \( m_{AC}(q, r) = [w_{[1:j_s-1]}; \alpha(o_r); w_{[j_e+1:L]}] \), where \( w_i's \) are the tokens of entity-centric revised question.

### Relation-Centric (RC)

Here we augment the wh-words with the relation description instead. This form of question revision has the most expressive power in distinguishing between the KB relations in question context, but it can suffer more from the training data sparsity. For the running example, it maps to “[what area of activism] did [activist] fight for”. Formally, it is defined as \( m_{RC}(q, r) = [w_{[1:j_s-1]}; \alpha(r); w_{[j_e+1:L]}] \).

## 3 Model

### 3.1 Task Formulation

Given a question \( q \), we first run an existing QA system to answer \( q \). Suppose it returns \( r \) as the top predicted relation and \( r' \) is a candidate relation that is ranked lower. Our objective is to decide if

<table>
<thead>
<tr>
<th>Refinement</th>
<th>( F_1 )</th>
<th># Refined Qs</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAGG w/ Best Alternative</td>
<td>58.9</td>
<td>639</td>
</tr>
</tbody>
</table>

Table 1: What if we know the questions on which the system makes mistakes? Best alternative is computed by replacing the predictions of incorrectly answered questions by STAGG with its second top-ranked candidate.
there is a need to replace \( r \) with \( r' \). We formulate this task as finding a scoring function \( s(q, r) \) and a confidence margin threshold \( t \),

\[
\text{replace}(r, r', q) = \begin{cases} 1, & \text{if } s(q, r') - s(q, r) \geq t \\ 0, & \text{otherwise,} \end{cases}
\]  

which makes the replacement decision.

### 3.2 Encoding Question Revisions

Let \( q' = (w'_1, w'_2, \ldots, w'_d) \) denote a question revision. We first encode all the words into a \( d \)-dimensional vector space using an embedding matrix. Let \( e_i \) denote the embedding of word \( w'_i \). To obtain the contextual embeddings for words, we use bi-directional LSTM

\[
\overrightarrow{h}_i = \text{LSTM}_{fwd}(\overrightarrow{h}_{i-1}, e_i) \tag{2}
\]

\[
\overleftarrow{h}_i = \text{LSTM}_{bwd}(\overleftarrow{h}_{i+1}, e_i) \tag{3}
\]

with \( \overrightarrow{h}_0 = 0 \) and \( \overleftarrow{h}_{t+1} = 0 \). We combine forward and backward contextual embeddings by

\[ h_i = \text{concat}(\overrightarrow{h}_i, \overleftarrow{h}_i). \]

We then generate the final encoding of revised question \( q' \) by \( \text{enc}(q') = \text{concat}(h_1, h_t) \).

### 3.3 Training Objective

**Score Function.** Given a question revision mapping \( m \), a question \( q \), and a relation \( r \), our scoring function is defined as \( s(q, r) = \text{enc}(m(q, r)) \) where \( \text{enc} \) is a model parameter that is jointly learnt with the LSTM parameters.

**Loss Function.** Let \( T = \{(q, a_q)\} \) denote a set of training questions paired with their true answer set. Let \( U(q) \) denote the set of all candidate KB relations for question \( q \). Let \( f(q, r) \) denote the \( F_1 \) score of an answer set obtained by relation \( r \) when compared to \( a_q \). For each candidate relation \( r \in U(q) \) with a positive \( F_1 \) value, we define

\[
N(q, r) = \{r' \in U(q) : f(q, r) > f(q, r')\} \tag{4}
\]

as the set of its negative relations for question \( q \). Similar to a hinge-loss in (Bordes et al., 2014), we define the objective function \( J(\theta, w, E) \) as

\[
\sum_{(q,r,r')} \max(0, \delta\lambda(q, r, r') - (s(q, r) - s(q, r'))) \tag{5}
\]

where the sum is taken over all valid \( \{(q, r, r')\} \) triplets and the penalty margin is defined as \( \delta\lambda(q, r, r') = \lambda(f(q, r) - f(q, r')) \).

We use this loss function because: i) it allows us to exploit partially correct answers via \( F_1 \) scores, and ii) training with it updates the model parameters towards putting a large margin between the scores of correct \( (r) \) and incorrect \( (r') \) relations, which is naturally aligned with our prediction refinement objective defined in Equation 1.

### 4 Alternative Solutions

Our approach directly integrates additional textual evidences with the question itself, which can be processed by any sequence oriented model, and benefit from its future updates without significant modification. However, we could also design models taking these textual evidences into specific consideration, without even appealing to question revision. We have explored this option and tried two methods that closely follow QA-LSTM and ATTENTIVE-LSTM (Tan et al., 2016). The latter model achieves the state-of-the-art for passage-level question answer matching. Unlike our approach, they encode questions and evidences for candidate answers in parallel, and measure the semantic similarity between them using cosine distance. The effectiveness of these architectures has been shown in other studies (Neculaiu et al., 2016; Hermann et al., 2015; Chen et al., 2016; Mueller and Thyagarajan, 2016) as well.

We adopt these models in our setting as follows: (1) Textual evidences \( \alpha(s_e) \) (equiv. of EC revision), \( \alpha(o_e) \) (equiv. of AC revision) or \( \alpha(r) \) (equiv. of RC revision) of a candidate KB relation \( r \) is used in place of a candidate answer \( a \) in the original model, (2) We replace the entity mention with a universal \#entity# token as in (Yih et al., 2015) because individual entities are rare and uninformative for semantic similarity, (3) We train the score function \( \text{sim}(q, r) \) using the objective defined in Eq. 5. Further details of the alternative solutions can be found in Appendix.

### 5 Experiments

**Datasets.** For evaluation, we use the WEBQUESTIONS (Berant et al., 2013), a benchmark dataset for QA on Freebase. It contains 5,810 questions whose answers are annotated from Freebase using Amazon Mechanical Turk. We also use SIMPLEQUESTIONS (Bordes et al., 2015), a collection of 108,442 question/Freebase-fact pairs, for training data augmentation in some of our experiments, which is denoted by +SimpleQ, in results.
Training Data Preparation. We generate candidates only provides question-answer pairs along with annotated topic entities. We generate candidates $U(q)$ for each question $q$ by retrieving 1-hop and 2-hop KB relations $r$ from annotated topic entity $e$ in Freebase. For each relation $r$, we query $(e, r, ?)$ against Freebase and retrieve the candidate answers $r_a$. Then, we compute $f(q, r)$ by comparing the answer set $r_a$ with the annotated answers.

5.1 Implementation Details

Word embeddings are initialized with pretrained GloVe (Pennington et al., 2014) vectors¹, and updated during the training. We take the dimension of word embeddings and the size of LSTM hidden layer equal and experiment with values in \{50, 100, 200, 300\}. We apply dropout regularization on both input and output of LSTM encoder with probability 0.5. We hand tuned penalty margin scalar $\lambda$ as 1. The model parameters are optimized using Adam (Kingma and Ba, 2015) with batch size of 32. We implemented our models in tensorflow (Abadi et al., 2016).

To refine predictions of a base QA system, we take its second top ranked prediction as the refinement candidate $r'$ and employ the decision mechanism in Equation 1. Confidence margin threshold $t$ is tuned by grid search on the training data after the score function is trained. QUESREV-AC + RC model is obtained by a linear combination of QUESREV-AC and QUESREV-RC models, which is formally defined in Appendix B. To evaluate the alternative solutions for prediction refinement, we apply the same decision mechanism $replace(r, r', q)$ with the trained $sim(q, r)$ in Section 4 as the score function.

We use a dictionary² to identify wh-words in a question. We find topic entity spans using Stanford NER tagger (Manning et al., 2014). If there are multiple matches, we use the first matching span for both.

5.2 Results

Table 2 presents the main result of our prediction refinement model using Stagg’s results. Our approach improves the performance of a strong base QA system by 1.4% and achieves 53.9% in $F_1$ measure, which is slightly better than the state-of-the-art KB-QA system (Xu et al., 2016a). However, it is important to note here that Xu et al. (2016a) uses DBPedia knowledge base in addition to Freebase and the Wikipedia corpus that we do not utilize. Moreover, applying our approach on the Stagg predictions reranked by (Yavuz et al., 2016), referred as Stagg-Rank in Table 2, leads to a further improvement over a strong ensemble baseline. These suggest that our system captures orthogonal signals to the ones exploited in the base QA models. Improvements of QUESREV over both Stagg and Stagg-Rank are statistically significant.

In Table 3, we present variants of our approach. We observe that AC model yields to best refinement results when trained only on WebQuestions data (e.g., WebQ, column). This empirical observation is intuitively expected because it has more generalization power than RC, which might make AC more robust to the training data sparsity. This intuition is further justified by observing that augmenting the training data with SimpleQuestions improves the performance of RC model most as it has more expressive power.

<table>
<thead>
<tr>
<th>Method</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Berant et al., 2013)</td>
<td>35.7</td>
</tr>
<tr>
<td>(Yao and Van Durme, 2014)</td>
<td>33.0</td>
</tr>
<tr>
<td>(Berant and Liang, 2014)</td>
<td>39.9</td>
</tr>
<tr>
<td>(Bao et al., 2014)</td>
<td>35.5</td>
</tr>
<tr>
<td>(Bordes et al., 2014)</td>
<td>39.2</td>
</tr>
<tr>
<td>(Yang et al., 2014)</td>
<td>41.3</td>
</tr>
<tr>
<td>(Dong et al., 2015)</td>
<td>40.8</td>
</tr>
<tr>
<td>(Yao, 2014)</td>
<td>44.3</td>
</tr>
<tr>
<td>(Berant and Liang, 2015)</td>
<td>49.7</td>
</tr>
<tr>
<td>Stagg (Yih et al., 2015)</td>
<td>52.5</td>
</tr>
<tr>
<td>(Reddy et al., 2016)</td>
<td>50.3</td>
</tr>
<tr>
<td>(Xu et al., 2016b)</td>
<td>53.3</td>
</tr>
<tr>
<td>(Xu et al., 2016a)</td>
<td>53.8</td>
</tr>
<tr>
<td>QUESREV on Stagg</td>
<td>53.9</td>
</tr>
<tr>
<td>Ensemble</td>
<td></td>
</tr>
<tr>
<td>Stagg-Rank (Yavuz et al., 2016)</td>
<td>54.0</td>
</tr>
<tr>
<td>QUESREV on Stagg-Rank</td>
<td>54.3</td>
</tr>
</tbody>
</table>

Table 3: $F_1$ performance of variants of our model QUESREV and alternative solutions on base QA system Stagg.

¹http://nlp.stanford.edu/projects/glove/
²what, who, where, which, when, how
Table 4: Example questions and corresponding predictions of \textsc{stagg} (Yih et al., 2015) before and after using replacements proposed by \textsc{quesrev}. The colors red and blue indicate wrong and correct, respectively. Domain names of KB relations are dropped for brevity. \texttt{person.(education).institution} is used as a shorthand for the 2-hop relation \texttt{person.education-education.institution} in Example 1.

Although both QA-LSTM and ATTENTIVE-LSTM lead to successful prediction refinements on \textsc{stagg}, question revision approach consistently outperforms both of the alternative solutions. This suggests that our way of incorporating the new textual evidences by naturally blending them in the question context leads to a better mechanism for checking the consistency of KB relations with the question. It is possible to argue that part of the improvements of refinement models over \textsc{stagg} in Table 3 may be due to model ensembling. However, the performance gap between \textsc{quesrev} and the alternative solutions enables us to isolate this effect for query revision approach.

### 7 Conclusion

We present a prediction refinement approach for question answering over knowledge bases. We introduce question revision as a tailored augmentation of the question via various textual evidences from KB relations. We exploit revised questions as a way to reexamine the consistency of candidate KB relations with the question itself. We show that our method improves the quality of answers produced by \textsc{stagg} on the \textsc{webquestions} dataset.

### Acknowledgements

The authors would like to thank the anonymous reviewers for their thoughtful comments. 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.

### Appendix

See supplementary notes.
References


Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnndaily mail reading comprehension task. In Annual Meeting of the Association for Computational Linguistics (ACL).


Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics (TACL)*.


Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.


Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016a. Hybrid question answering over knowledge base and free text. In *International Conference on Computational Linguistics (COLING)*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016b. Question answering on freebase via relation extraction and textual evidence. 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 knowledge-based question answering. In *Empirical Methods on Natural Language Processing (EMNLP)*.

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

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)*.
