Bidirectional Attentive Memory Networks for Question Answering over Knowledge Bases

Yu Chen
Rensselaer Polytechnic Institute
cheny39@rpi.edu

Lingfei Wu
IBM Research
wuli@us.ibm.com

Mohammed J. Zaki
Rensselaer Polytechnic Institute
zaki@cs.rpi.edu

Abstract

When answering natural language questions over knowledge bases (KBs), different question components and KB aspects play different roles. However, most existing embedding-based methods for knowledge base question answering (KBQA) ignore the subtle interrelationships between the question and the KB (e.g., entity types, relation paths and context). In this work, we propose to directly model the two-way flow of interactions between the questions and the KB via a novel Bidirectional Attentive Memory Network, called BAMnet. Requiring no external resources and only very few hand-crafted features, on the WebQuestions benchmark, our method significantly outperforms existing information-retrieval based methods, and remains competitive with (hand-crafted) semantic parsing based methods. Also, since we use attention mechanisms, our method offers better interpretability compared to other baselines.

1 Introduction

With the rapid growth in large-scale knowledge bases (KBs) such as DBPedia (Auer et al., 2007) and FreeBase (Google, 2018), knowledge base question answering (KBQA) has drawn increasing attention over the past few years. *Given questions in natural language (NL), the goal of KBQA is to automatically find answers from the underlying KB*, which provides a more natural and intuitive way to access the vast underlying knowledge resources.

One of the most prominent challenges of KBQA is the lexical gap. For instance, the same question can be expressed in various ways in NL while a KB usually has a canonical lexicon. It is therefore non-trivial to map an NL question to a structured KB. The approaches proposed to tackle the KBQA task can be roughly categorized into two groups: semantic parsing (SP) and information retrieval (IR) approaches. SP-based approaches address the problem by constructing a semantic parser that converts NL questions into intermediate logic forms, which can be executed against a KB. Traditional semantic parsers (Wong and Mooney, 2007) require annotated logical forms as supervision, and are limited to narrow domains with a small number of logical predicates. Recent efforts overcome these limitations via the construction of hand-crafted rules or features (Abujabal et al., 2017; Hu et al., 2018) schema matching (Cai and Yates, 2013), and using weak supervision from external resources (Krishnamurthy and Mitchell, 2012).

Unlike SP-based approaches that usually assume a pre-defined set of lexical triggers or rules, which limit their domains and scalability, IR-based approaches directly retrieve answers from the KB in light of the information conveyed in the questions. These IR-based approaches usually do not require hand-made rules and can therefore scale better to large and complex KBs. Recently, deep neural networks have been shown to produce strong results on many NLP tasks. In the field of KBQA, under the umbrella of IR-based approaches, many embedding-based methods (Bordes et al., 2014b; Hao et al., 2017) have been proposed and have shown promising results. These methods adopt various ways to encode questions and KB subgraphs into a common embedding space and directly match them in that space, and can be typically trained in an end-to-end manner.

Compared to existing embedding-based methods that encode questions and KB subgraphs independently, we introduce a novel **B**idirectional **A**ttentive **M**emory **net**work, called **BAMnet** that captures the mutual interactions between questions and the underlying KB, which is stored in a content-addressable memory. We assume that the world knowledge (i.e., the KB) is helpful for better understanding the questions. Similarly, the questions

themselves can help us focus on important KB aspects. To this end, we design a *two-layered bidirectional attention network*. The *primary attention network* is intended to focus on important parts of a question in light of the KB and important KB aspects in light of the question. Built on top of that, the *secondary attention network* is intended to enhance the question and KB representations by further exploiting the two-way attention. Through this idea of *hierarchical two-way attention*, we are able to distill the information that is the most relevant to answering the questions on both sides of the question and KB.

We highlight the contributions of this paper as follows: 1) we propose a novel bidirectional attentive memory network for the task of KBQA which is intended to directly model the two-way interactions between questions and the KB; 2) by design, our method offers good interpretability thanks to the attention mechanisms; 3) on the WebQuestions benchmark, our method significantly outperforms previous information-retrieval based methods while remaining competitive with (hand-crafted) semantic parsing based methods.

2 Related work

Two broad classes of SP-based and IR-based approaches have been proposed for KBQA. The former attempts to convert NL questions to logic forms. Recent work focused on approaches based on weak supervision from either external resources (Krishnamurthy and Mitchell, 2012; Berant et al., 2013; Yao and Van Durme, 2014; Hu et al., 2018; Yih et al., 2015; Yavuz et al., 2016), schema matching (Cai and Yates, 2013), or using hand-crafted rules and features (Unger et al., 2012; Berant et al., 2013; Berant and Liang, 2015; Reddy et al., 2016; Bao et al., 2016; Abujabal et al., 2017; Hu et al., 2018; Bast and Haussmann, 2015; Yih et al., 2015). A thread of research has been explored to generate semantic query graphs from NL questions such as using coarse alignment between phrases and predicates (Berant et al., 2013), searching partial logical forms via an agenda-based strategy (Berant and Liang, 2015), pushing down the disambiguation step into the query evaluation stage (Hu et al., 2018), or exploiting rich syntactic information in NL questions (Xu et al., 2018a,b). Notably, another thread of SP-based approaches try to exploit IR-based techniques (Yao and Van Durme, 2014; Bast and Haussmann, 2015; Yang et al., 2014; Yih et al., 2015; Bao et al., 2016; Yavuz et al., 2016; Liang et al., 2016) by computing the similarity of two sequences as features, leveraging a neural network-based answer type prediction model, or training end-to-end neural symbolic machine via REINFORCE (Williams, 1992). However, most SP-based approaches more or less rely on hand-crafted rules or features, which limits their scalability and transferability.

The other line of work (the IR-based) has focused on mapping answers and questions into the same embedding space, where one could query any KB independent of its schema without requiring any grammar or lexicon. Bordes et al. (2014b) were the first to apply an embedding-based approach for KBQA. Later, Bordes et al. (2014a) proposed the idea of subgraph embedding, which encodes more information (e.g., answer path and context) about the candidate answer. In follow-up work (Bordes et al., 2015; Jain, 2016), memory networks (Weston et al., 2014) were used to store candidates, and could be accessed iteratively to mimic multi-hop reasoning. Unlike the above methods that mainly use a bag-of-words (BOW) representation to encode questions and KB resources, (Dong et al., 2015; Hao et al., 2017) apply more advanced network modules (e.g., CNNs and LSTMs) to encode questions. Hybrid methods have also been proposed (Feng et al., 2016; Xu et al., 2016; Das et al., 2017), which achieve improved results by leveraging additional knowledge sources such as free text. While most embedding-based approaches encode questions and answers independently, (Hao et al., 2017) proposed a cross-attention mechanism to encode questions according to various candidate answer aspects. Differently, in this work, our method goes one step further by modeling the bidirectional interactions between questions and a KB.

The idea of bidirectional attention proposed in this work is similar to those applied in machine reading comprehension (Wang and Jiang, 2016; Seo et al., 2016; Xiong et al., 2016). However, these previous works focus on capturing the interactions between two bodies of text, in this work, we focus on modeling the interactions between one body of text and a KB.

3 A modular bidirectional attentive memory network for KBQA

Given an NL question, the goal is to fetch answers from the underlying KB. Our proposed BAMnet

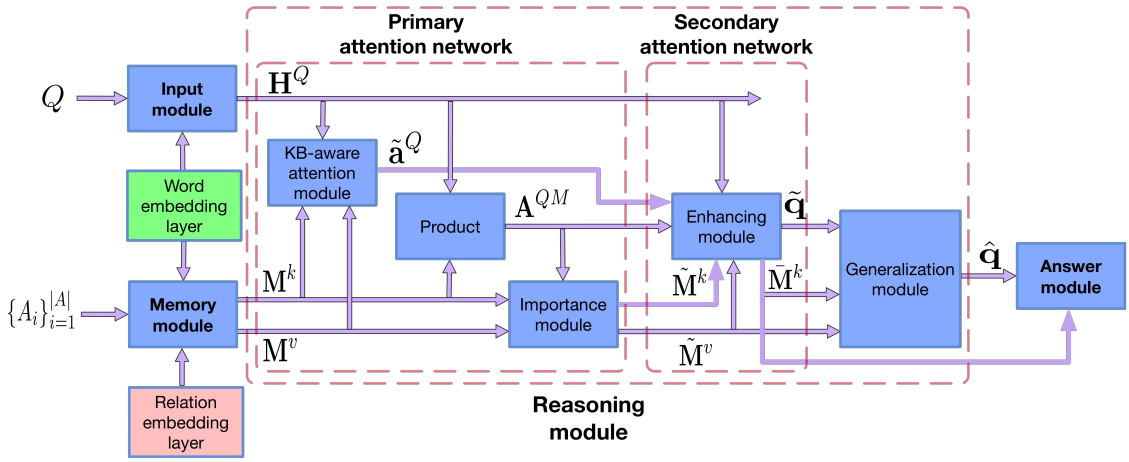

Figure 1: Overall architecture of the BAMnet model.

model consists of four components which are the *input module*, *memory module*, *reasoning module* and *answer module*, as shown in Fig. 1.

3.1 Input module

An input NL question $Q = \{q_i\}_{i=1}^{|Q|}$ is represented as a sequence of word embeddings (q_i) by applying a word embedding layer. We then use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to encode the question as $\mathbf{H}^Q$ (in $\mathbb{R}^{d \times |Q|}$) which is the sequence of hidden states (i.e., the concatenation of forward and backward hidden states) generated by the BiLSTM.

3.2 Memory module

Candidate generation Even though all the entities from the KB could in principle be candidate answers, this is computationally expensive and unnecessary in practice. We only consider those entities which are "close" to the main topic entity of a question. An answer is the text description (e.g., a name) of an entity node. For example, *Ohio* is the topic entity of the question "Who was the secretary of state of Ohio in 2011?" (see Fig. 2). After getting the topic entity, we collect all the entities connected to it within h hops as candidate answers, which we denote as $\{A_i\}_{i=1}^{|A|}$.

KB representation For each candidate answer from the KB, we encode three types of information: answer type, path and context.

Answer type Entity type information is an important clue in ranking answers. For example, if a question uses the interrogative word *where*, then candidate answers with types relevant to the concept of location are more likely to be correct. We use a BiLSTM to encode its text description to get a d-dimensional vector $\mathbf{H}_i^{t_1}$ (i.e., the concatenation of last forward and backward hidden states).

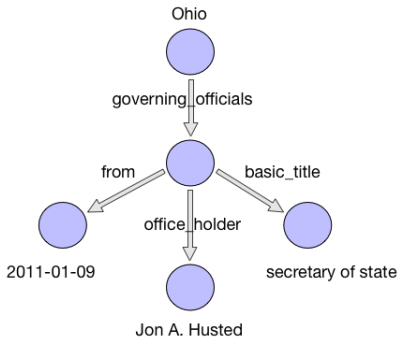

Figure 2: A working example from Freebase. Relations in Freebase have hierarchies where high-level ones provide too broad or even noisy information about the relation. Thus, we choose to use the lowest level one.

Answer path We define an answer path as a sequence of relations from a candidate answer to a topic entity. For example, for the *Ohio* question (see Fig. 2), the answer path of *Jon A. Husted* can be either represented as a sequence of relation ids [*office_holder, governing_officials*] or the text description [*office, holder, governing, officials*]. We thus encode an answer path as $\mathbf{H}_i^{p_1}$ via a BiLSTM, and as $\mathbf{H}_i^{p_2}$ by computing the average of its relation embeddings via a relation embedding layer.

Answer context The answer context is defined as the surrounding entities (e.g., sibling nodes) of a candidate which can help answer questions with constraints. For example, in Fig. 2, the answer context of *Jon A. Husted* includes the government position title *secretary of state* and starting date *2011-01-09*. However, for simple questions without constraints, the answer context is unnec-

essary and can potentially incorporate noise. We tackle this issue with two strategies: 1) we use a novel *importance module* (explained later) to focus on important answer aspects, and 2) we only consider those context nodes that have overlap with the question. Specifically, for each context node (i.e., a sequence of words) of a candidate, we first compute the longest common subsequence between it and the question, we then encode it via a BiLSTM only if we get a non-stopwords substring. Finally, the answer context of a candidate answer will be encoded as the average of all context node representations, which we denote as $\mathbf{H}_i^c$.

Key-value memory module In our model, we use a key-value memory network (Miller et al., 2016) to store candidate answers. Unlike a basic memory network (Weston et al., 2014), its addressing stage is based on the key memory while the reading stage uses the value memory, which gives greater flexibility to encode prior knowledge via functionality separation. Thus, after encoding the answer type, path and context, we apply linear projections on them as follows:

$$\begin{aligned} \mathbf{M}_i^{k_t} &= f_t^k(\mathbf{H}_i^{t_1}) & \mathbf{M}_i^{v_t} &= f_t^v(\mathbf{H}_i^{t_1}) \\ \mathbf{M}_i^{k_p} &= f_p^k([\mathbf{H}_i^{p_1}; \mathbf{H}_i^{p_2}]) & \mathbf{M}_i^{v_p} &= f_p^v([\mathbf{H}_i^{p_1}; \mathbf{H}_i^{p_2}]) \\ \mathbf{M}_i^{k_c} &= f_c^k(\mathbf{H}_i^c) & \mathbf{M}_i^{v_c} &= f_c^v(\mathbf{H}_i^c) \end{aligned} \quad (1)$$

where $\mathbf{M}_i^{k_t}$ and $\mathbf{M}_i^{v_t}$ are d-dimensional key and value representations of answer type A_i^t, respectively. Similarly, we have key and value representations for answer path and answer context. We denote $\mathbf{M}$ as a key-value memory whose row $\mathbf{M}_i = \{\mathbf{M}_i^k, \mathbf{M}_i^v\}$ (both in $\mathbb{R}^{d\times 3}$), where $\mathbf{M}_i^k = [\mathbf{M}_i^{k_t}; \mathbf{M}_i^{k_p}; \mathbf{M}_i^{k_c}]$ comprises the keys, and $\mathbf{M}_i^v = [\mathbf{M}_i^{v_t}; \mathbf{M}_i^{v_p}; \mathbf{M}_i^{v_c}]$ comprises the values. Here $[,]$ and $[;]$ denote row-wise and column-wise concatenations, respectively.

3.3 Reasoning module

The reasoning module consists of a generalization module, and our novel *two-layered bidirectional attention network* which aims at capturing the two-way interactions between questions and the KB. The *primary attention network* contains the KB-aware attention module which focuses on the important parts of a question in light of the KB, and the importance module which focuses on the important KB aspects in light of the question. The *secondary attention network* (*enhancing module* in Fig. 1) is intended to enhance the question and KB vectors by further exploiting the two-way attention.

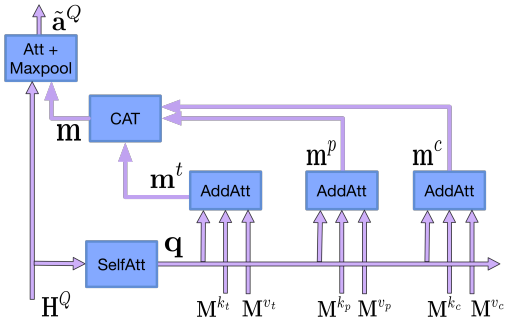

Figure 3: KB-aware attention module. CAT: concatenation, SelfAtt: self-attention, AddAtt: additive attention.

KB-aware attention module Not all words in a question are created equal. We use a KB-aware attention mechanism to focus on important components of a question, as shown in Fig. 3. Specifically, we first apply self-attention (SelfAtt) over all question word vectors $\mathbf{H}^Q$ to get a d-dimensional question vector $\mathbf{q}$ as follows

$$\begin{aligned} \mathbf{q} &= \text{BiLSTM}([\mathbf{H}^Q \mathbf{A}^{QQ^T}, \mathbf{H}^Q]) \\ \mathbf{A}^{QQ} &= \text{softmax}((\mathbf{H}^Q)^T \mathbf{H}^Q) \end{aligned} \quad (2)$$

where softmax is applied over the last dimension of an input tensor by default. Using question summary $\mathbf{q}$, we apply another attention (AddAtt) over the memory to obtain answer type $\mathbf{m}_t$, path $\mathbf{m}_p$ and context summary $\mathbf{m}_c$:

$$\begin{aligned} \mathbf{m}_x &= \sum_{i=1}^{|A|} a_i^x \cdot \mathbf{M}_i^{v_x} \\ \mathbf{a}^x &= \text{Att}_{\text{add}}(\mathbf{q}, \mathbf{M}^{k_x}) \end{aligned} \quad (3)$$

where $x \in \{t, p, c\}$, and $\text{Att}_{\text{add}}(\mathbf{x}, \mathbf{y}) = \text{softmax}(\tanh([\mathbf{x}^T, \mathbf{y}]\mathbf{W_1})\mathbf{W_2})$, with $\mathbf{W_1} \in \mathbb{R}^{2d\times d}$ and $\mathbf{W_2} \in \mathbb{R}^{d\times 1}$ being trainable weights.

So far, we have obtained the KB summary $\mathbf{m} = [\mathbf{m}_t; \mathbf{m}_p; \mathbf{m}_c]$ in light of the question. We proceed to compute the question-to-KB attention between question word q_i and KB aspects as formulated by $\mathbf{A}^{Qm} = \mathbf{H}^{Q^T}\mathbf{m}$. By applying max pooling over the last dimension (i.e., the KB aspect dimension) of $\mathbf{A}^{Qm}$, that is, $\mathbf{a}_i^Q = \max_j \mathbf{A}_{ij}^{Qm}$, we select the strongest connection between q_i and the KB. The idea behind it is that each word in a question serves a specific purpose (i.e., indicating answer type, path or context), and max pooling can help find out that purpose. We then apply a softmax over the resulting vector to obtain $\tilde{\mathbf{a}}^Q$ which is a *KB-aware question attention vector* since it indicates the importance of q_i in light of the KB.

Importance module The importance module focuses on important KB aspects as measured by their relevance to the questions. We start by computing a $|Q| \times |A| \times 3$ attention tensor $\mathbf{A}^{QM}$ which indicates the strength of connection between each pair of $\{q_i, A_j^x\}^{x=\{t,p,c\}}$. Then, we take the max of the question word dimension of $\mathbf{A}^{QM}$ and normalize it to get an attention matrix $\tilde{\mathbf{A}}^M$, which indicates the importance of each answer aspect for each candidate answer. After that, we proceed to compute question-aware memory representations $\tilde{\mathbf{M}}^k$. Thus, we have:

$$\begin{aligned}
\tilde{\mathbf{M}}^v &= \{\tilde{\mathbf{M}}_i^v\}_{i=1}^{|A|} \in \mathbb{R}^{|A|\times d} \quad \tilde{\mathbf{M}}_i^v = \sum_{j=1}^{3} \mathbf{M}_{ij}^v \\
\tilde{\mathbf{M}}^k &= \{\tilde{\mathbf{M}}_i^k\}_{i=1}^{|A|} \in \mathbb{R}^{|A|\times d} \quad \tilde{\mathbf{M}}_i^k = \tilde{\mathbf{A}}_i^M \mathbf{M}_i^k \\
\tilde{\mathbf{A}}^M &= \text{softmax}(\mathbf{A}^{M^T})^T \quad \mathbf{A}^M = \max_i \{\mathbf{A}_i^{QM}\}_{i=1}^{|Q|} \\
\mathbf{A}^{QM} &= \left(\mathbf{M}^k \mathbf{H}^Q\right)^T
\end{aligned} \tag{4}$$

Enhancing module We further enhance the question and KB representations by exploiting two-way attention. We compute the KB-enhanced question representation $\tilde{\mathbf{q}}$ which incorporates the relevant KB information by applying max pooling over the last dimension (i.e., the answer aspect dimension) of $\mathbf{A}^{QM}$, that is, $\mathbf{A}_M^Q = \max_k \{\mathbf{A}_{:,:,k}^{QM}\}_{k=1}^3$, and then normalizing it to get a question-to-KB attention matrix $\tilde{\mathbf{A}}_M^Q$ from which we compute the question-aware KB summary and incorporate it into the question representation $\tilde{\mathbf{H}}^Q = \mathbf{H}^Q + \tilde{\mathbf{a}}^Q \odot (\tilde{\mathbf{A}}_M^Q \tilde{\mathbf{M}}^v)^T$. Finally, we obtain a d-dimensional KB-enhanced question representation $\tilde{\mathbf{q}} = \tilde{\mathbf{H}}^Q \tilde{\mathbf{a}}^Q$.

Similarly, we compute a question-enhanced KB representation $\overline{\mathbf{M}}^k$ which incorporates the relevant question information:

$$\begin{aligned}
\overline{\mathbf{M}}^k &= \tilde{\mathbf{M}}^k + \tilde{\mathbf{a}}^M \odot (\tilde{\mathbf{A}}_Q^M (\tilde{\mathbf{H}}^Q)^T) \\
\tilde{\mathbf{a}}^M &= (\tilde{\mathbf{A}}_M^Q)^T \tilde{\mathbf{a}}^Q \in \mathbb{R}^{|A|\times 1} \\
\tilde{\mathbf{A}}_Q^M &= \text{softmax}(\mathbf{A}_M^{Q^T}) \in \mathbb{R}^{|A|\times|Q|}
\end{aligned} \tag{5}$$

Generalization module We add a one-hop attention process before answering. We use the question representation $\tilde{\mathbf{q}}$ to query over the key memory $\overline{\mathbf{M}}^k$ via an attention mechanism, and fetch the most relevant information from the value memory, which is then used to update the question vector using a GRU (Cho et al., 2014). Finally, we apply a residual layer (He et al., 2016) (i.e., $y = f(x) + x$) and batch normalization (BN) (Ioffe and Szegedy, 2015), which help the model performance in practice. Thus, we have

$$\begin{aligned}
\hat{\mathbf{q}} &= \text{BN}(\tilde{\mathbf{q}} + \mathbf{q}') \quad \mathbf{q}' = \text{GRU}(\tilde{\mathbf{q}}, \tilde{\mathbf{m}}) \\
\tilde{\mathbf{m}} &= \sum_{i=1}^{|A|} a_i \cdot \tilde{\mathbf{M}}_i^v \quad \mathbf{a} = \text{Att}_{\text{add}}^{\text{GRU}}(\tilde{\mathbf{q}}, \overline{\mathbf{M}}^k)
\end{aligned} \tag{6}$$

3.4 Answer module

Given the representation of question Q which is $\hat{\mathbf{q}}$ and the representation of candidate answers $\{A_i\}_{i=1}^{|A|}$ which is $\{\overline{\mathbf{M}}_i^k\}_{i=1}^{|A|}$, we compute the matching score $S(\hat{\mathbf{q}}, \overline{\mathbf{M}}_i^k)$ between every pair (Q, A_i) as $S(\mathbf{q}, \mathbf{a}) = \mathbf{q}^T \cdot \mathbf{a}$. The candidate answers are then ranked by their scores.

3.5 Training and testing

Training Intermediate modules such as the *enhancing module* generate "premature" representations of questions (e.g., $\tilde{\mathbf{q}}$) and candidate answers (e.g., $\overline{\mathbf{M}}^k$). Even though these intermediate representations are not optimal for answer prediction, we can still use them along with the final representations to jointly train the model, which we find helps the training probably by providing more supervision since we are directly forcing intermediate representations to be helpful for prediction. Moreover, we directly match interrogative words to KB answer types. A question Q is represented by a 16-dimensional interrogative word (we use "which", "what", "who", "whose", "whom", "where", "when", "how", "why" and "whether") embedding $\mathbf{q}^w$ and a candidate answer A_i is represented by entity type embedding $\mathbf{H}_i^{t_2}$ with the same size. We then compute the matching score $S(\mathbf{q}^w, \mathbf{H}_i^{t_2})$ between them. Although we only have weak labels (e.g., incorrect answers do not necessarily imply incorrect types) for the type matching task, and there are no shared representations between two tasks, we find in practice this strategy helps the training process as shown in Section 4.4.

Loss Function: In the training phase, we force positive candidates to have higher scores than negative candidates by using a triplet-based loss function:

$$\begin{aligned}
o = g(\mathbf{H}^Q \tilde{\mathbf{a}}^Q, \sum_{j=1}^{3} \mathbf{M}_{:,j}^k) + g(\tilde{\mathbf{q}}, \overline{\mathbf{M}}^k) \\
+ g(\hat{\mathbf{q}}, \overline{\mathbf{M}}^k) + g(\mathbf{q}^w, \mathbf{H}^{t_2})
\end{aligned} \tag{7}$$

where $g(\mathbf{q}, \mathbf{M}) = \sum_{\substack{a^+ \in A^+ \\ a^- \in A^-}} \ell(S(\mathbf{q}, \mathbf{M}_{a^+}), S(\mathbf{q}, \mathbf{M}_{a^-}))$, and $\ell(y, \hat{y}) = \max(0, 1 + \hat{y} - y)$ is a hinge loss

function, and A^+ and A^- denote the positive (i.e., correct) and negative (i.e., incorrect) answer sets, respectively. Note that at training time, the candidate answers are extracted from the KB subgraph of the gold-standard topic entity, with the memory size set to N_{max}. We adopt the following sampling strategy which works well in practice: if N_{max} is larger than the number of positive answers $|A^+|$, we keep all the positive answers and randomly select negative answers to fill up the memory; otherwise, we randomly select $\min(N_{max}/2, |A^-|)$ negative answers and fill up the remaining memory with random positive answers.

Testing At testing time, we need to first find the topic entity. We do this by using the top result returned by a separately trained topic entity predictor (we also compare with the result returned by the Freebase Search API). Then, the *answer module* returns the candidate answer with the highest scores as predicted answers. Since there can be multiple answers to a given question, the candidates whose scores are close to the highest score within a certain margin, θ, are regarded as good answers as well. Therefore, we formulate the inference process as follows:

$$\hat{A} = \{\hat{a} \quad | \, \hat{a} \in A \,\&\, \max_{a' \in A}\{S(\hat{\mathbf{q}}, \overline{\mathbf{M}}^k_{a'})\} - S(\hat{\mathbf{q}}, \overline{\mathbf{M}}^k_{\hat{a}}) < \theta\} \tag{8}$$

where $\max_{a' \in A}\{S(\hat{\mathbf{q}}, \overline{\mathbf{M}}^k_{a'})\}$ is the score of the best matched answer and $\hat{A}$ is the predicted answer set. Note that θ is a hyper-parameter which controls the degree of tolerance. Decreasing the value of θ makes the model become stricter when predicting answers.

3.6 Topic entity prediction

Given a question Q, the goal of a topic entity predictor is to find the best topic entity $\hat{c}$ from the candidate set $\{C_i\}_{i=1}^{|C|}$ returned by external topic entity linking tools (we use the Freebase Search API and S-MART (Yang and Chang, 2016) in our experiments). We use a convolutional network (CNN) to encode Q into a d-dimensional vector $\mathbf{e}$. For candidate topic entity C_i, we encode three types of KB aspects, namely, the entity name, entity type and surrounding relations where both entity name and type are represented as a sequence of words while surrounding relations are represented as a bag of sequences of words. Specifically, we use three CNNs to encode them into three d-dimensional vectors, namely, $\mathbf{C}_i^n$, $\mathbf{C}_i^t$ and $\mathbf{C}_i^{r_1}$. Note that for surrounding relations, we first encode each of the relations and then compute their average. Additionally, we compute an average of the relation embeddings via a relation embedding layer which we denote as $\mathbf{C}_i^{r_2}$. We then apply linear projections on the above vectors as follows:

$$\begin{aligned} \mathbf{P}_i^k &= f^k([\mathbf{C}_i^n; \mathbf{C}_i^t; \mathbf{C}_i^{r_1}; \mathbf{C}_i^{r_2}]) \\ \mathbf{P}_i^v &= f^v([\mathbf{C}_i^n; \mathbf{C}_i^t; \mathbf{C}_i^{r_1}; \mathbf{C}_i^{r_2}]) \end{aligned} \tag{9}$$

where $\mathbf{P}_i^k$ and $\mathbf{P}_i^v$ are d-dimensional key and value representations of candidate C_i, respectively. Furthermore, we compute the updated question vector $\hat{\mathbf{e}}$ using the generalization module mentioned earlier. Next, we use a dot product to compute the similarity score between Q and C_i. A triplet-based loss function is used as formulated by $o = g(\mathbf{e}, \mathbf{P}_i^k) + g(\hat{\mathbf{e}}, \mathbf{P}_i^k)$ where $g(.)$ is the aforementioned hinge loss function. When training the predictor, along with the candidates returned from external entity linking tools, we do negative sampling (using string matching) to get more supervision. In the testing phase, the candidate with the highest score is returned as the best topic entity and no negative sampling is applied.

4 Experiments

This section provides an extensive evaluation of our proposed BAMnet model against state-of-the-art KBQA methods. The implementation of BAMnet is available at https://github.com/hugochan/BAMnet.

4.1 Data and metrics

We use the Freebase KB and the WebQuestions dataset, described below:

Freebase This is a large-scale KB (Google, 2018) that consists of general facts organized as subject-property-object triples. It has 41M non-numeric entities, 19K properties, and 596M assertions.

WebQuestions This dataset (Berant et al., 2013) (nlp.stanford.edu/software/sempre) contains 3,778 training examples and 2,032 test examples. We further split the training instances into a training set and development set via a 80%/20% split. Approximately 85% of questions can be directly answered via a single FreeBase predicate. Also, each question can have multiple answers. In our experiments, we use a development version of the dataset (Baudis and Pichl, 2016), which additionally provides (potentially noisy) entity mentions for each question.

Following (Berant et al., 2013), macro F1 scores (i.e., the average of F1 scores over all questions) are reported on the WebQuestions test set.

4.2 Model settings

When constructing the vocabularies of words, entity types or relation types, we only consider those questions and their corresponding KB subgraphs appearing in the training and validation sets. The vocabulary size of words is $V = 100,797$. There are 1,712 entity types and 4,996 relation types in the KB subgraphs. Notably, in FreeBase, one entity might have multiple entity types. We only use the first one available, which is typically the most concrete one. For those non-entity nodes which are boolean values or numbers, we use "bool" or "num" as their types, respectively.

We also adopt a **query delexicalization** strategy where for each question, the topic entity mention as well as constraint entity mentions (i.e., those belonging to "date", "ordinal" or "number") are replaced with their types. When encoding KB context, if the overlap belongs to the above types, we also do this delexicalization, which will guarantee it matches up with the delexicalized question well in the embedding space.

Given a topic entity, we extract its 2-hop subgraph (i.e., $h = 2$) to collect candidate answers, which is sufficient for *WebQuestions*. At training time, the memory size is limited to $N_{max} = 96$ candidate answers (for the sake of efficiency). If there are more potential candidates, we do random sampling as mentioned earlier. We initialize word embeddings with pre-trained GloVe vectors (Pennington et al., 2014) with word embedding size $d_v = 300$. The relation embedding size d_p, entity type embedding size d_t and hidden size d are set as 128, 16 and 128, respectively. The dropout rates on the word embedding layer, question encoder side and the answer encoder side are 0.3, 0.3 and 0.2, respectively. The batch size is set as 32, and answer module threshold $\theta = 0.7$. As for the topic entity prediction, we use the same hyperparameters. For each question, there are 15 candidates after negative sampling in the training time. When encoding a question, we use a CNN with filter sizes 2 and 3. A linear projection is applied to merge features extracted with different filters. When encoding a candidate aspect, we use a CNN with filter size 3. Linear activation and max-pooling are used together with CNNs. In the training process, we use the Adam optimizer (Kingma and Ba, 2014) to train the model. The initial learning rate is set as 0.001 which is reduced by a factor of 10 if no improvement is observed on the validation set in 3 consecutive epochs. The training procedure stops if no improvement is observed on the validation set in 10 consecutive epochs. The hyper-parameters are tuned on the development set.

4.3 Performance comparison

As shown in Table 1, our method can achieve an F1 score of 0.557 when the gold topic entity is known, which gives an upper bound of our model performance. When the gold topic entity is unknown, we report the results using: 1) the Freebase Search API, which achieves a recall@1 score of 0.857 on the test set for topic entity linking, and 2) the topic entity predictor, which achieves a recall@1 score of 0.898 for entity retrieval.

As for the performance of BAMnet on WebQuestions, it achieves an F1 score of 0.518 using the topic entity predictor, which is significantly better than the F1 score of 0.497 using the Freebase Search API. We can observe that BAMnet significantly outperforms previous state-of-the-art IR-based methods, which conclusively demonstrates the effectiveness of modeling bidirectional interactions between questions and the KB.

It is important to note that unlike the state-of-the-art SP-based methods, BAMnet relies on no external resources and very few hand-crafted features, but still remains competitive with those approaches. Based on careful hand-drafted rules, some SP-based methods (Bao et al., 2016; Yih et al., 2015) can better model questions with constraints and aggregations. For example, (Yih et al., 2015) applies many manually designed rules and features to improve performance on questions with constraints and aggregations, and (Bao et al., 2016) directly models temporal (e.g., "after 2000"), ordinal (e.g., "first") and aggregation constraints (e.g., "how many") by adding detected constraint nodes to query graphs. In contrast, our method is end-to-end, with very few hand-crafted rules.

Additionally, (Yavuz et al., 2016; Bao et al., 2016) train their models on external Q&A datasets to get extra supervision. For a fairer comparison, we only show their results without training on external Q&A datasets. Similarly, for hyhrid systems (Feng et al., 2016; Xu et al., 2016), we only report results without using Wikipedia free text. It is in-

Methods (ref)	Macro F_1
SP-based	
(Berant et al., 2013)	0.357
(Yao and Van Durme, 2014)	0.443
(Wang et al., 2014)	0.453
(Bast and Haussmann, 2015)	0.494
(Berant and Liang, 2015)	0.497
(Yih et al., 2015)	**0.525**
(Reddy et al., 2016)	0.503
(Yavuz et al., 2016)	0.516
(Bao et al., 2016)	0.524
(Feng et al., 2016)	0.471
(Reddy et al., 2017)	0.495
(Abujabal et al., 2017)	0.510
(Hu et al., 2018)	0.496
IR-based	
(Bordes et al., 2014a)	0.392
(Yang et al., 2014)	0.413
(Dong et al., 2015)	0.408
(Bordes et al., 2015)	0.422
(Xu et al., 2016)	**0.471**
(Hao et al., 2017)	0.429
Our Method: BAMnet	
w/ gold topic entity	0.557
w/ Freebase Search API	0.497
w/ topic entity predictor	**0.518**

Table 1: Results on the WebQuestions test set. Bold: best in-category performance.

teresting to note that both (Yih et al., 2015) and (Bao et al., 2016) also use the ClueWeb dataset for learning more accurate semantics. The F1 score of (Yih et al., 2015) drops from 0.525 to 0.509 if ClueWeb information is removed. To summarize, BAMnet achieves state-of-the-art performance of 0.518 without recourse to any external resources and relies only on very few hand-crafted features. If we assume gold-topic entities are given then BAMnet achieves an F1 of 0.557.

4.4 Ablation study

We now discuss the performance impact of the different modules and strategies in BAMnet. Note that gold topic entity is assumed to be known when we do this ablation study, because the error introduced by topic entity prediction might reduce the real performance impact of a module or strategy. As shown in Table 2, significant performance drops were observed after turning off some key attention

Methods	Macro F_1
all	**0.557**
w/o two-layered bidirectional attn	0.534
w/o kb-aware attn (+self-attn)	0.544
w/o importance module	0.540
w/o enhancing module	0.550
w/o generalization module	0.542
w/o joint type matching	0.545
w/o topic entity delexicalization	0.529
w/o constraint delexicalization	0.554

Table 2: Ablation results on the WebQuestions test set. Gold topic entity is assumed to be known.

modules, which confirms that the real power of our method comes from the idea of hierarchical two-way attention. As we can see, when turning off the *two-layered bidirectional attention network*, the model performance drops from 0.557 to 0.534. Among all submodules in the attention network, the *importance module* is the most significant since the F1 score drops to 0.540 without it, thereby confirming the effectiveness of modeling the query-to-KB attention flow. On the flip side, the importance of modeling the KB-to-query attention flow is confirmed by the fact that replacing the *KB-aware attention module* with self-attention significantly degrades the performance. Besides, the secondary attention layer, the *enhancing module*, also contributes to the overall model performance. Finally, we find that the topic entity delexicalization strategy has a big influence on the model performance while the constraint delexicalization strategy only marginally boosts the performance.

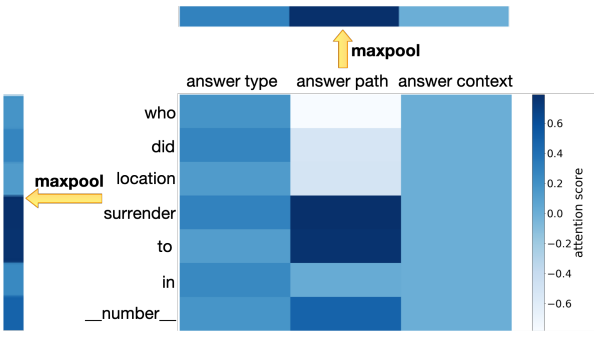

Figure 4: Attention heatmap generated by the reasoning module. Best viewed in color.

4.5 Interpretability analysis

Here, we show that our method does capture the mutual interactions between question words

KB Aspects	Questions	BAMnet w/o BiAttn.	BAMnet	Gold Answers
Answer Type	What degrees did Obama get in college?	Harvard Law School, Columbia University, Occidental College	Bachelor of Arts, Juris Doctor, Political Science	Juris Doctor, Bachelor of Arts
	What music period did Beethoven live in?	Austrian Empire, Germany, Bonn	Classical music, Opera	Opera, Classical music
Answer Path	Where did Queensland get its name from?	Australia	Queen Victoria	Queen Victoria
	Where does Delaware river start?	Delaware Bay	West Branch Delaware River, Mount Jefferson	West Branch Delaware River, Mount Jefferson
Answer Context	What are the major cities in Ukraine?	Kiev, Olyka, ... Vynohradiv, Husiatyn	Kiev	Kiev
	Who is running for vice president with Barack Obama 2012?	David Petraeus	Joe Biden	Joe Biden

Table 3: Predicted answers of BAMnet w/ and w/o bidirectional attention on the WebQuestions test set.

and KB aspects, by visualizing the attention matrix $\mathbf{A}^{QM}$ produced by the *reasoning module*. Fig. 4 shows the attention heatmap generated for a test question "who did location surrender to in __number__" (where "location" and "__number__" are entity types which replace the topic entity mention "France" and the constraint entity mention "ww2", respectively in the original question). As we can see, the attention network successfully detects the interactions between "who" and answer type, "surrender to" and answer path, and focuses more on those words when encoding the question.

To further examine the importance of the two-way flow of interactions, in Table 3, we show the predicted answers of BAMnet with and without the *two-layered bidirectional attention network* on samples questions from the WebQuestions test set. We divide the questions into three categories based on which kind of KB aspect is the most crucial for answering them. As we can see, compared to the simplified version which is not equipped with bidirectional attention, our model is more capable of answering all the three types of questions.

4.6 Error analysis

To better examine the limitations of our approach, we randomly sampled 100 questions on which our method performed poorly (i.e., with per-question F1 score less than 0.6), and categorized the errors. We found that around 33% of errors are due to label issues of gold answers and are not real mistakes. This includes incomplete and erroneous labels, and also alternative correct answers. Constraints are another source of errors (11%), with temporal constraints accounting for most. Some questions have implicit temporal (e.g., tense) constraints which our method does not model. A third source of error is what we term type errors (13%), for which our method generates more answers than needed because of poorly utilizing answer type information. Lexical gap is another source of errors (5%). Finally, other sources of errors (38%) include topic entity prediction error, question ambiguity, incomplete answers and other miscellaneous errors.

5 Conclusions and future work

We introduced a novel and effective bidirectional attentive memory network for the purpose of KBQA. To our best knowledge, we are the first to model the mutual interactions between questions and a KB, which allows us to distill the information that is the most relevant to answering the questions on both sides of the question and KB. Experimental results show that our method significantly outperforms previous IR-based methods while remaining competitive with hand-crafted SP-based methods. Both ablation study and interpretability analysis verify the effectiveness of the idea of modeling mutual interactions. In addition, our error analysis shows that our method actually performs better than what the evaluation metrics indicate.

In the future, we would like to explore effective ways of modeling more complex types of constraints (e.g., ordinal, comparison and aggregation).

Acknowledgments

This work is supported by IBM Research AI through the IBM AI Horizons Network. We thank the anonymous reviewers for their constructive suggestions.

References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *WWW*, pages 1191–1200.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *COLING*, pages 2503–2514.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *CIKM*, pages 1431–1440. ACM.

Petr Baudis and Jan Pichl. 2016. Dataset factoid webquestions. https://github.com/brmson/dataset-factoid-webquestions.

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

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *TACL*, 3:545–558.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *EMNLP*, pages 615—-620.

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

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *ECML-PKDD*, pages 165–180. Springer.

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

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. *arXiv preprint arXiv:1704.08384*.

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

Yansong Feng, Songfang Huang, Dongyan Zhao, et al. 2016. Hybrid question answering over knowledge base and free text. In *COLING*, pages 2397–2407.

Google. 2018. Freebase data dumps. https://developers.google.com/freebase.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *ACL*, volume 1, pages 221–231.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*, pages 770–778.

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

Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. 2018. Answering natural language questions by subgraph matching over knowledge graphs. *TKDE*, 30(5):824–837.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Sarthak Jain. 2016. Question answering over knowledge base using factual memory networks. In *NAACL-HLT*, pages 109–115.

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

Jayant Krishnamurthy and Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *EMNLP-CoNLL*, pages 754–765.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

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. *TACL*, 4:127–140.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *EMNLP*, pages 89–101.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. In *WWW*, pages 639–648. ACM.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Zhenghao Wang, Shengquan Yan, Huaming Wang, and Xuedong Huang. 2014. An overview of microsoft deep qa system on stanford webquestions benchmark. Technical Report MSR-TR-2014-121, Microsoft Research.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *ACL*, pages 960–967.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. *arXiv preprint arXiv:1603.00957*.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018a. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018b. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. *arXiv preprint arXiv:1808.07624*.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *EMNLP*, pages 645–650.

Yi Yang and Ming-Wei Chang. 2016. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. *arXiv preprint arXiv:1609.08075*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *ACL*, volume 1, pages 956–966.

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *EMNLP*, pages 149–159.

Scott 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 *ACL*, pages 1321–1331.