Enhancing Key-Value Memory Neural Networks for Knowledge Based Question Answering

Kun Xu[1], **Yuxuan Lai**[2], **Yansong Feng**[2,3], **Zhiguo Wang**[4]
[1]Tencent AI Lab
[2]ICST, Peking University
[3]The MOE Key Laboratory of Computational Linguistics, Peking University
[4]Amazon
syxu828@gmail.com
{erutan,fengyansong}@pku.edu.cn
zgw.tomorrow@gmail.com

Abstract

Traditional Key-value Memory Neural Networks (KV-MemNNs) are proved to be effective to support shallow reasoning over a collection of documents in domain specific Question Answering or Reading Comprehension tasks. However, extending KV-MemNNs to Knowledge Based Question Answering (KB-QA) is not trivia, which should properly decompose a complex question into a sequence of queries against the memory, and update the query representations to support multi-hop reasoning over the memory. In this paper, we propose a novel mechanism to enable conventional KV-MemNNs models to perform interpretable reasoning for complex questions. To achieve this, we design a new *query updating* strategy to mask previously-addressed memory information from the query representations, and introduce a novel *STOP* strategy to avoid invalid or repeated memory reading without strong annotation signals. This also enables KV-MemNNs to produce structured queries and work in a semantic parsing fashion. Experimental results on benchmark datasets show that our solution, trained with question-answer pairs only, can provide conventional KV-MemNNs models with better reasoning abilities on complex questions, and achieve state-of-art performances.

1 Introduction

Memory Neural Networks (MemNNs) [Weston *et al.*, 2014; Sukhbaatar *et al.*, 2015b] are a family of neural network models that aim to learn how to reason with a long-term memory component and various inference components. The memory component serves as a knowledge base to recall facts from the past. MemNNs have been successfully applied in many natural language processing applications such as question answering and reading comprehension (RC). Recently, Miller *et al.* [2016] proposed a variant of MemNNs, namely Key-Value Memory Neural Networks (KV-MemNNs), which generalizes the original MemNNs by storing facts in a key-value structured memory. Figure 1 illustrates the basic architecture of KV-MemNNs, which consists of five components. The question is first fed to the **Embedding** component and the **Hashing** component. The former converts the incoming question to an internal feature representation. The Hashing component uses the question to pre-select a list of facts to compose the key-value memory. The **Key Addressing** component takes the input question representation and the current memory to compute the relevance probability between the question and each key in the memory. The **Value Reading** component reads the values of all addressed memories by taking their weighted sum using the relevance probabilities. The obtained value representation is then added to the query representation to change the query focus for the next round of memory reading. After multiple hops of reasoning over the memories, the final value representation is treated as the answer representation to perform the final prediction over all candidate answers in the **Rank-**

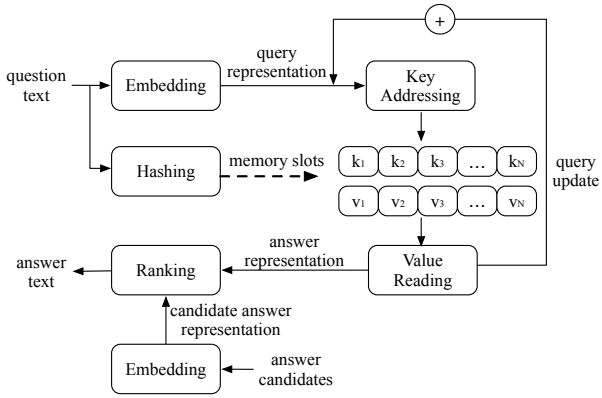

Figure 1: The Key-Value Memory Neural Network Architecture.

Proceedings of NAACL-HLT 2019, pages 2937–2947
Minneapolis, Minnesota, June 2 - June 7, 2019.

ing component.

The KV-MemNNs have been shown to support shallow reasoning in domain-specific knowledge based question answering (KB-QA) tasks such as MovieQA [Tapaswi *et al.*, 2016]. However, when applied to a more challenging scenario, e.g., open domain KB-QA, the KV-MemNNs models do not perform as well as expected, possibly due to two reasons. First of all, the focus of conventional KV-MemNNs is about understanding the facts in the memory rather than properly understanding the questions, where the latter requires incrementally decomposing a complex natural language question into a set of focused queries with the help of the facts in the memory. However, in open domain KB-QA, questions are usually more complicated, e.g., multi-relation questions such as *who does maggie grace play in taken*, where more than one entity and relation are mentioned. Secondly, as shown in Figure 1, KV-MemNNs usually work in an information retrieval (IR) fashion, which first retrieve a set of candidate answers from KB, then rank them by computing the similarity between the value representation and candidates, and finally select the top one or a fixed number of top candidates as the answer. We can imagine that it is not trivial for such IR-styled KV-MemNNs to properly resolve complex constraints from natural language questions, or to handle questions with multiple answers.

We believe that an ideal framework for open domain KB-QA should first understand the natural language questions, explicitly represent the meaning, and make the answer retrieval process more interpretable. To build such an interpretable KV-MemNN KB-QA model, we need to deal with the following challenges: (1) KV-MemNNs often read the memory repeatedly since they do not know when to stop; (2) during multiple memory readings, conventional KV-MemNNs often fail to precisely update the queries for multi-relation questions; (3) strong annotations are usually required to train an interpretable QA model, e.g., the supervision for the memory selection at each hop. To address the challenges, we propose a novel solution to make conventional KV-MemNNs feasible to open domain KB-QA. In particular, we introduce a flexible KV-MemNN solution that can work in both the IR and semantic parsing style with large-scale memory. To this end, we first present a novel query updating method that is able to decompose complex questions and precisely address a relevant key at each hop. Secondly, we introduce a new **STOP** strategy during memory readings, which imports a special key **STOP** into the memory and guides our model to avoid repeated or invalid memory readings. In addition, our proposed model can learn to reason over memory slots with weak supervision, e.g., question-answer pairs only, opposing the strong supervision that most current neural semantic parsers demand, which incurs high labor costs. Experimental results on two benchmark datasets show that our proposed model can not only enhance the reasoning capability of KV-MemNNs, but also be flexible enough to work as a semantic parser, with state-of-the-art performances.

2 Related Work

There are usually two main challenges in the open domain KB-QA task: (1) it often requires the ability to properly analyze and represent the natural language questions against knowledge bases, especially for those involving multiple entities and relations, which we also call as reasoning over the KBs; (2) training such interpretable question understanding models requires considerable strong annotations, which is expensive to obtain in practice. Existing works address these using either the information retrieval (IR) based solutions or the semantic paring (SP) based approaches. The IR-based models [Yao and Van Durme, 2014; Yao, 2015; Bast and Haussmann, 2015; Bordes *et al.*, 2015; Dong *et al.*, 2015; Jain, 2016; Lai *et al.*, 2019] tackle the KB-QA task by developing various ranking models towards the candidate answers, which implicitly meet the reasoning requirements during the candidate-searching step or in designing the ranking functions. In contrast, the SP-based approaches [Berant *et al.*, 2013; Kwiatkowski *et al.*, 2013; Berant and Liang, 2014; Reddy *et al.*, 2014; Yih *et al.*, 2015; Xu *et al.*, 2016] explicitly represent the meaning of questions as logical forms or structured queries that naturally support reasoning over structured KBs.

More recently, memory based reasoning solutions [Weston *et al.*, 2014; Miller *et al.*, 2016] are proposed to tackle the task. [Weston *et al.*, 2014] proposed the Memory Neural Networks (MemNNs), which enable the neural network models read/write on an external memory component, and are further extended into an End-to-End fashion [Sukhbaatar *et al.*, 2015a]. [Miller *et al.*,

2016] further proposed the Key-Value Memory Network, which generalizes the MemNN by storing facts in a key-value structured memory. Both of the two models could perform shallow reasoning over the memory, since they can find answers by consecutively making predictions over multiple memory slots. Compared to the flat memory slots in MemNNs, the Key-Value design can precisely accommodate more complex structured resources, e.g., discriminating subjects and objects in structured KBs, thus makes KV-MemNNs more flexible and better fit to different applications. These neural networks models are often trained in an end-to-end fashion, making the models relatively less interpretable.

Conceptually similar to our STOP strategy, Shen *et al.* [2017] propose the termination gate mechanism based on a random variable generated from the internal state for reading comprehension. In contrast, our model attempts to learn a general STOP key embedding based on the incrementally updated query representations, which can be learned from question-answer pairs only and lead to more explicit reasoning interpretations over structured KBs. This make our model potentially suit more real scenarios.

Our work is also related to [Jain, 2016; Bao *et al.*, 2016], which are designed to support reasoning for multi-relation questions by exploring the relation path and certain KB schema, e.g., CVT nodes, in the Freebase. The former also considers previously-addressed keys during query updating, but ignores the value representations. Thus, it still requires predefined rules and threshold to artificially add intermediate value representations to update the query. The latter also relies on a set of predefined rules to perform reasoning over Freebase. In contrast, our model incorporates both the key and value representations into the query representations, and update in a more uniform way, thus is more general and supports more reasoning scenarios in KB-QA.

3 Our Model

For a given question x, a knowledge base $\mathcal{KB}$ and the question's answer y, we aim to learn a model such that

$$\mathcal{F}(x, \mathcal{KB}) = \hat{y} \rightarrow y$$

where $\hat{y}$ is the predicted answer. In standard KV-MemNNs, the function $\mathcal{F}$ can be composed of five components, i.e., key hashing, key addressing, value reading, query updating and answer prediction. Next, we will introduce how we design a novel mechanism upon those components to equip KV-MemNNs with more powerful reasoning ability. The architecture of our model is shown in Figure 2.

3.1 Key Hashing

The knowledge facts in $\mathcal{KB}$ are usually organized in a triple <subject, *relation*, object>, such as <Maggie Grace, *fb:actor_character*, Kim>. In KV-MemNNs, these facts are stored in a key-value structured memory, where the key k is composed of the left-hand side entity (subject) and the relation, e.g., Maggie Grace *fb:actor_character*, and the value v is the right-hand side entity (object), e.g., Kim. Traditional KV-MemNNs first preselect a list of candidate KB facts $(k_1,v_1),...,(k_n,v_n)$ to compose the key-value memory. Particularly, one can first detect entity mentions in the question, and include all KB facts that contains with one of those entities as subject into the memory. In our experiments, we directly use the entity linking results of Xu *et al.* [2016] and filter out their relations that have more than 100 objects. In practice, we find this strategy could effectively avoid the memory exploding for popular entities. In order to help the model avoid repeated or invalid memory reading, we introduce a special key, **STOP**, into the memory for all questions. The corresponding value of the STOP key is a special symbol represented by an all-zero vector. The STOP key is designed to tell our model that we has already accumulated sufficient facts at hand to answer the question, so there is no need to find other knowledge facts from the memory in later hops.

3.2 Key Addressing & Value Reading

Key addressing is basically a matching process, aiming to find the most suitable key for a given query. It can be formulated as a function that computes the relevance probability p_i between the question x and each key k_i:

$$p_i = \text{Softmax}(A\Phi(x) \cdot A\Phi(k_i))$$

where Φ is a feature map of dimension D, A is a $d \times D$ matrix. The values of memories are then read by taking their weighted sum using the relevance probabilities, and the value representation o is returned to locate the answers or update the

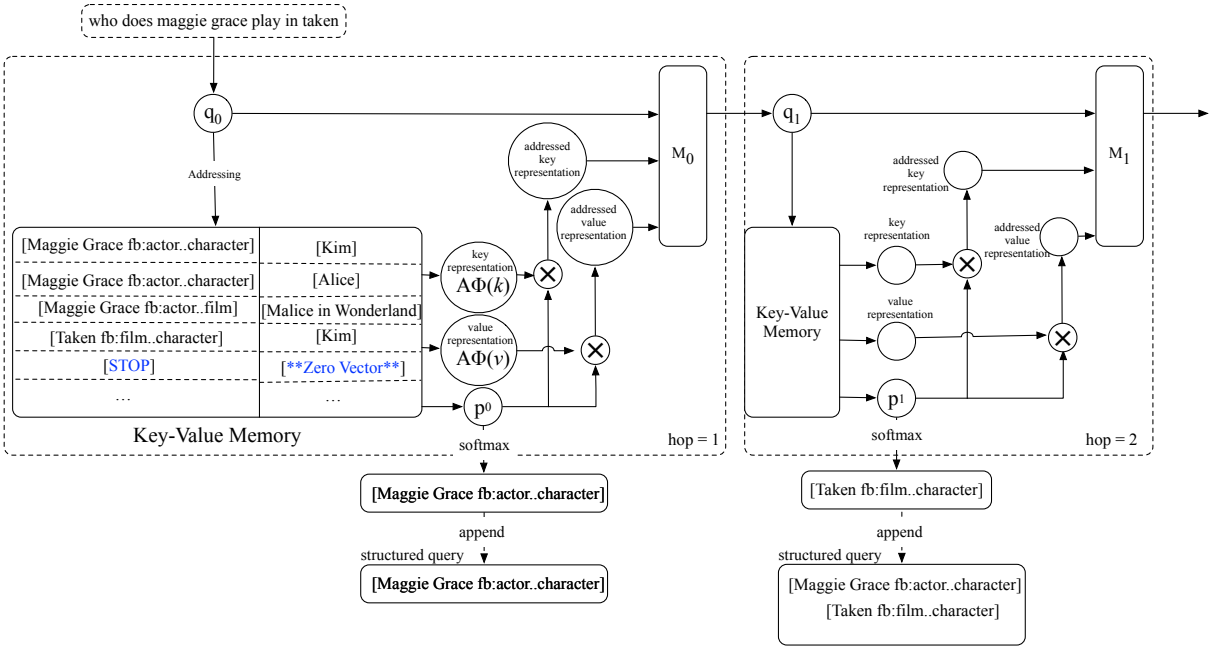

Figure 2: A running example of our key-value memory network model to answer the question *who does maggie grace play in taken*.

query for further memory addressing:

$$o = \sum_i p_i A\Phi(v_i)$$

There are many methods to represent questions and memory slots, including keys and values. Here we simply use the Bag-of-Words model to produce the representations, where we sum the embedding of each word in the question or memory slot together to obtain their vector representations.

3.3 Query Updating

After reading the addressed memory, the initial query representation $q = A\Phi(x)$ should be updated so that the new evidence o collected in the current hop can be properly considered to retrieve more pertinent information in later steps. Traditional KV-MemNNs simply add the initial query q and the returned value o, then perform a linear transformation to obtain the new query representation. This updating strategy is effective in the RC task, since the questions in RC are relatively simple, and their main emphasis is to select proper memory values to change the focus of the query, until reaching the answer. However, the questions in open domain KB-QA tasks are more complicated, usually involved with multiple relations or constraints.

Take the question "*who does maggie grace play in taken*" as an example, the expected answer should follow two constraints: (1) *Maggie_Grace* plays this answer; and (2) this answer is from the movie *Taken*. To answer this question, the model needs to perform two hops of inference consecutively, i.e., matching two keys `Maggie_Grace fb:actor..character` and `Taken fb:film..character` in the memory. Conventional query updating methods, e.g., adding q and o, may not be helpful in guiding the model to predict the other key in later hops, and possibly even hurt the performance, since it may introduce unrelated information into the new query. Intuitively, in KB-QA, masking previously-addressed keys from the query could benefit latter inference, since the model will be able to focus on the next hop, e.g., the movie *Taken*. Therefore, we take into account the query and addressed memories at the t-th hop when updating the query q^{t+1} for the next hop:

$$q^{t+1} = \mathbf{M}_t \cdot (q^t \oplus \underbrace{\sum_i p_i^t A\Phi(k_i)}_{addressed\ key} \oplus \underbrace{\sum_i p_i^t A\Phi(v_i)}_{addressed\ value})$$

where $\oplus$ denotes the concatenation of vectors. The query updating step is parameterized with a different matrix M_t on the t-th hop, which is designed to learn a proper way to combine these three representations.

3.4 Answer Prediction

Conventional KV-MemNNs use the value o at the final hop of inference to retrieve the answers, by simply computing the similarity between o and all candidate answers. This may be of risk for our task. First of all, many questions in the open domain KB-QA have multiple answers, but, KV-MemNNs are supposed to select the candidate with the highest similarity score as the only answer. Secondly, the value representation at the final step may not fully capture the answer information throughout the whole inference process. For example, for multi-constraint questions, the model may address different constraints at different hops, which requires the model to take the value representations in every hop into consideration in order to produce the final answer representation from a global view.

We therefore propose to accumulate the value representations of all hops to make the resulting answer representation lean more on satisfying all constraints. We compute the answer representation m at each hop by adding the value representations of both the current hop and the previous one: $m^{t+1} = o^{t+1} + o^t$, $m^0 = o^0$.

With the final m at hand, we could follow traditional IR-based methods to use the final **A**nswer **R**epresentation to find the best match over all possible candidate values in the memory, namely the **AR** approach. Instead, we can also collect all the best matched keys at every hop to construct a **S**tructured **Q**uery and execute it over the KB to obtain all qualified answers, namely the **SQ** approach. Specifically, the structured query can be constructed as: we select the keys that have the highest relevance probabilities in every hop, resulting a sequence of keys sk_0,.... Starting from sk_0, we append the key sk_i into the final structured query until we see the **STOP** key for the first time at the k-th hop, i.e., $SQ = \{sk_0, ..., sk_{k-1}\}$.

Apparently, the SQ approach can easily output all qualified answers through excuting the queries over the KB, while the AR approach still has difficulties in selecting multiple answers from a ranked list. However, keep in mind that, we do not have gold-standard structured queries for training. As a result, we have to adopt different strategies to find answers in the training and test phases. During **training**, after a fixed number H hops, we follow the AR approach to use the final m^H to compute a prediction over possible candidates, and we train the model by minimizing the cross-entropy between the prediction and gold-standard answers. During **testing**, we follow the SQ approach to collect the final answers by constructing and executing the structured queries over the KB to obtain all answers. As illustrated in Figure 2, for the example question *who does maggie grace play in taken*, our model selects three keys, i.e., [<maggie grace, fb:actor..character>, <taken, film..character>, and <STOP>]. We combine the previous two triples to construct the structured query, which is then executed over the KB to get the answer. We should point out that our model could still use the AR approach to predict the answers just like in the training phase. And as far as we know, our model is the first one that is suitable to tackle the KB-QA task in both the information retrieval and semantic parsing fashions.

3.5 Objective Function and Learning

Given an input question x, the network with parameters θ uses the answer representation m_x^h to perform the prediction over candidate answers at hop h, resulting a prediction vector a_x^h, where the i-th component is respect to the probability of candidate answer i. We denote t_x as the target distribution vector. We compute the standard cross-entropy loss between a_x^h and t_x, and further define the objective function over all training data:

$$L(\theta) = \sum_x \sum_{h=1}^{H} t_x \log a_x^h + \lambda ||\theta||^2$$

where λ is a vector of regularization parameters. Intuitively, this loss function makes our model generate shorter paths to reach answers from the question. On the other hand, it encourages the query updating method to mask the information already addressed in previous hops for the next query representation. This design, together with the query updating method, are the keys to learn the STOP strategy.

4 Experiments

We evaluate our model on two benchmark datasets to investigate whether our enhanced KV-MemNNs model can better perform reasoning over the memory in the open domain KB-QA task, and whether it can make the QA procedure more interpretable.

Semantic Paring Method	Answer F_1
Berant *et al.* [2013]	35.7%
[Berant and Liang, 2014]	39.9%
Yih *et al.* [2015]	52.5%
Bast and Haussmann [2015]	49.4%
Xu *et al.* [2016] (KB Only)	47.1%
Bao *et al.* [2016]	54.4%
CQU + AR (a)	42.0%
KVQU + AR (b)	43.2%
CQU + SQ (c)	39.6%
KVQU + SQ (d)	41.8%
STOP + CQU + AR (e)	43.3%
STOP + CQU + SQ (f)	45.8%
STOP + KVQU + AR (g)	48.6%
STOP + KVQU + SQ (h)	**54.6**%

Table 1: The performance of different models on the test set of WebQuestions.

4.1 Settings

We use the WebQuestions dataset [Berant *et al.*, 2013] as our main dataset, which contains 5,810 question-answer pairs. This dataset is built on Freebase [Bollacker *et al.*, 2008] and all answers are Freebase entities. We use the same training, development and test split as [Berant *et al.*, 2013], containing 3000, 778 and 2032 questions, respectively.

Our model is trained using the Adam optimizer [Kingma and Ba, 2014] with mini-batch size 60. The learning rate is set to 0.001. The complexity of model was penalized by adding L2 regularization to the cross entropy loss function. Gradients are clipped when their norm is bigger than 20. The hop size is set to 3. We initialize word embeddings using pre-trained word representations from Turian *et al.* [2010] and the dimension of word embedding is set to 50.

We use the average question-wise F_1 as our evaluation metric. We compare our model with representative IR based KB-QA models, and several state-of-the-art semantic parsing models. We also include different variants of our model for comparisons to shed light on the advantages of our proposed strategies, especially our three important building blocks - the STOP strategy, how to update a query, and how to obtain the answers.

CQU+AR: uses the **C**onventional **Q**uery **U**pdating method (CQU) which performs a linear transformation over the sum of the query and value representations. This method adopts the AR approach to predict answers where the one with highest probability is selected as the answer.

KVQU+AR: applies the approach introduced in this paper that additionally considers both the **K**ey and **V**alue representations in the **Q**uery **U**pdating (KVQU). This method updates the query representation after reading the memory values at each hop of inference. Note that this model can be seen as a variant of Jain [2016], which is essentially an IR-based approach.

CQU+SQ: uses the CQU method to update the query representations, and applies the SQ approach to obtain the answers.

KVQU+SQ: uses the KVQU method to update the query representations, and adopts the SQ approach to obtain the answers.

STOP+CQU+AR: introduces the STOP key into the memory, but still uses the conventional query updating method and answer representations to find the answers.

STOP+CQU+SQ: introduces the STOP key and uses the conventional query updating method, but uses the SQ approach to obtain the answers.

STOP+KVQU+AR: introduces the STOP key to the memory, uses the KVQU approach to update the query representations, and adopts the AR approach to retrieve the answers.

STOP+KVQU+SQ: is our **main model** that introduces the STOP key, applies the KVQU query updating method, and retrieves answers using the post-constructed structured queries.

4.2 Results and Discussion

Table 1 summarizes the performance of various methods on the test set of WebQuestions. We can see that our main model (**STOP+KVQU+SQ**) performs the best among all its variations, and significantly outperforms the state-of-the-art methods on WebQuestions (with one-tailed t-test significance of $p < 0.05$). We can also see that even the conventional KV-MemNN model (i.e., model (a)) could still outperform traditional semantic parsing models [Berant *et al.*, 2013; Berant and Liang, 2014], showing the effectiveness of memory networks in organizing and utilizing structured data. By replacing the CQU method with our proposed KVQU method, the KV-MemNN model (i.e., model (b)) could further gain an improvement of 1.2%, indicating a proper query representation updating method is critical to KV-MemNNs. After introducing the STOP strategy, the KV-MemNN model is capable to perform proper multi-hop reasoning over the memory, thus outperform most existing methods by a large margin except Bao *et al.* [2016].

Previously, Bao *et al.* [2016] achieves the state-

Question	Addressed Key	
who did armie hammer play in the social network	Armie Hammer fb:award_nominations..award_nominee Armie Hammer fb:award_nominations..award_nominee Armie Hammer fb:award_nominations..award_nominee	before
	Armie Hammer fb:actor..character Social Network fb:film..character **STOP**	after
who is the governor of India 2009	Indiana fb:governing_officials..office_holder Indiana fb:governing_officials..office_holder Indiana fb:governing_officials..office_holder	before
	Indiana fb:governing_officials..office_holder Indiana fb:governing_officials..from **STOP**	after
what team did david beckham play for in 2011	David Beckham fb:loans..borrowing_team David Beckham fb:player..team David Beckham fb:player..team	before
	David Beckham fb:player..team **STOP** **STOP**	after

Table 2: Running examples of addressed keys and corresponding relevance probabilities before and after introducing the STOP strategy. The question with **blue** color is correctly answered by introducing the STOP strategy while the **red** ones are still not resolved (see discussion in Session 4.8).

of-the-art performance on the WebQuestions by explicitly addressing the multi-constraint questions. Specifically, Bao *et al.* [2016] designs a multi-constraint graph for such questions based on the Freebase schema, and introduces a set of predefined rules to solve complex representations such as max, min, top-X and so on. On the other hand, our model only requires the KB facts to be stored in a Key-Value memory, independent of certain KB schemas. Although those complex expressions in WebQuestions could be easily covered by hand-crafted rules as many did, our model does not work upon such rules, and we think it is crucial to properly enhance KV-MemNNs with such more advanced reasoning abilities, which we leave for future work.

4.3 Impact of the STOP Key

As shown Table 1, we can see that when introducing the STOP strategy, almost all models improve by around 4%, not only in the SQ setting (e.g., from (d) to (f)), but also in the AR setting (e.g., from (a) to (e)). This is not surprising, since the STOP key is introduced to help our model learn to determine when it should stop reading the memory to avoid repeated or invalid addressing. This apparently will lead to more accurate key addressing at each hop, and produce more accurate structured queries. And, better key addressing can also help to generate better value representations at each hop, and finally better answer representation at the last hop. Keep in mind that we obtain such improvement in the case that we do not have gold-standard annotations for the STOP key, but with question-answer pairs only.

To better investigate how the STOP key works, we randomly select 200 questions from the test set and manually analyze the structured queries generated by our model. We find that, for 184 questions (92%) our model predicts the STOP key after one hop of inference and continuously predicts STOP keys in later hops. For the remaining questions, our model predicts two distinct keys before predicting the STOP key. Among these 184 questions, 178 questions (96.7%) can be resolved using exactly a one-triple query. For the remaining 6 questions, it requires two distinct triples to find the answers. This indicates that our model can successfully utilize the STOP strategy to avoid repeated or invalid reading over the memory, at least, for simple questions.

For the multi-relation questions which require at least two hops of inference to find the answers, we evaluate our model on 326 multi-constraint questions selected in Bao *et al.* [2016]. We find that for 283 questions (86.8%), our model performs two hops of inference before predicting the STOP key while for the remaining 13.2%, our model only performs one hop of inference. This demonstrates that even without strong annotations in terms of structured queries, our model still manages to recognize the multi-relation structure and properly stops the invalid reading process. Table 2 illustrates several examples before and after using the STOP strategy.

4.4 Query Updating

In KV-MemNNs, it is important to properly update the query representation after each hop, since the updated query will be used to address more focused information in the next hop, which is especially crucial to support multi-hop reasoning over the memory. We experimented with two query updating approaches, the traditional adding-based method, CQU, and our proposed KVQU, which additionally considers the addressed key representations in previous hops to update the query representation. From Table 1, we can see that no matter which techniques are used to retrieve the final answers, our model can always benefit from replacing the CQU with our KVQU. The main reason may be that KVQU learns to mask already addressed information and retain those untouched, which leads to more accurate key addressing and more expressive answer representations. But, we also observe that by switching from AR to SQ, STOP+KVQU+SQ achieves more improvement than STOP+CQU+SQ. One possible reason is that the KVQU method, together with the STOP strategy, can help address the most relevant keys (i.e., top#1 key) more accurately than CQU does at each hop, which results in more accurate structured queries.

We randomly select 50 multi-constraint questions from the test set of WebQuestions and analyze the addressed keys at each hop. We find that for 48 questions, the model with CQU (model (f)) repeatedly selects the same keys that have the highest relevance probabilities at the first two hops. However, when replacing CQU with KVQU, the model addresses different keys at the first two hops (examples shown in Table 2). One possible reason is that, in contrast to the CQU that only uses the value representation of current hop to update the query, KVQU additionally considers the addressed keys in the current hop, and aims to mask the already addressed information, so that in later hops, the model will focus on the remaining untouched part of the question.

4.5 Candidate Ranking v.s. Structured Query

We investigate two different answer retrieval methods, the IR styled method AR and the semantic parsing styled method SQ, in the experiments. By comparing CQU+AR (a) and CQU+SQ (c) in Table 1, we can see that replacing the answer retrieval method does not boost the performance, but actually hurts. This is not surprising, because with the vanilla CQU, the model does not know how to stop reading the memory, thus may select repeated/invalid keys after the first hop, which will produce incorrect structured queries. Similarly, the KVQU+SQ also suffers from the incorrect structured queries.

After introduce the STOP key into the memory, we find that no matter which query updating methods are used, the models using structured queries (STOP+*+SQ) significantly outperform their corresponding versions using the ranking based method (STOP+*+AR). Despite of the STOP strategy, the improvement may come from two aspects. First, the structured queries in SQ are built from the best key at every hop, thus have captured more global information throughout the reasoning procedure, while the AR method only uses the o in the last hop to retrieve answers. Secondly, the models with SQ can easily output multiple qualified answers from the KB, while the methods with AR can only output the top one from a ranked list in the memory. Moreover, from a practical perspective, the SQ methods provide more interpretability than the ranking based methods.

Hop Size	STOP + KVQU + SQ	STOP + KVQU + AR
1	38.7%	35.6%
2	45.9%	41.2%
3	**53.2%**	50.4%
4	53.1%	46.7%
5	53.0%	44.2%

Table 3: Performance (answer F_1) of STOP+KVQU+SQ and STOP+KVQU+AR with different hop sizes on the development set of WebQuestions.

4.6 Impact of the Hop Size

To investigate the impact of the hop size to the model performance, we evaluate the model STOP+KVQU+SQ and STOP+KVQU+AR with various hop size settings on the development set of WebQuestions. We evaluate both of the two models since we wonder which answer retrieval method is more sensitive to the hop size. As shown in Table 3, the model with AR answer retrieval method achieves the best performance with hop size of 3. Then with the hop size increasing, the performance drops. In contrast, the model with the SQ method also achieves the best performance when the hop size is 3. When the hop size increases, the performance does not drop signifi-

cantly, but almost remains unchanged. We think the reason may be that the model with the AR method keeps predicting the keys as the hop size increasing, which inevitably introduces noise to the answer representations. On the other hand, the model with the SQ method is less sensitive to the hop size, since once it sees the STOP key, the latter predictions do not affect the resulted structured query.

4.7 Results on QALD-4

To further examine our model under different conditions, besides WebQuestions, we also evaluate our main model on the QALD-4 dataset [Unger *et al.*], which is built upon another KB, DBpedia [Auer *et al.*, 2007]. Like Freebase, DBpedia stores real world facts in the triple format, making it easier for our model to adapt to this new KB. The QALD-4 dataset consists of three QA datasets, namely, Multilingual QA, Biomedical QA and Hybrid QA. Here we evaluate our model on the multilingual QA dataset which contains 250 English question-SPARQL pairs, where 200 questions (80%) are used for training and 50 questions for testing. These questions are more complicated than those of WebQuestions, e.g., all QALD-4 questions require at least two hops of inference over the KB to answer. Table 4 lists the results of our model and other participated systems [Unger *et al.*]. We can see that our model can achieve the best performance on this dataset, without importing extra rules.

4.8 Error Analysis

We analyze the errors of our main model on the development set of WebQuestions. Around 80% errors are caused by incorrect key addressing. The key addressing errors are mainly due to (1) the entities in the addressed keys are incorrect. Entity linking itself is a challenging task and we do not employ any existing entity linking tools to incorporate the linking confidence score, which may further improve our model; (2) the relations in the addressed keys are incorrect, which is mainly caused by insufficient context in the question. For example, in *what is duncan bannatyne*, the information from the question is quite limited for our model to predict the correct key *fb:profession*. Although the STOP strategy proves to be effective on most questions, there are still some questions where the STOP strategy does not work, e.g., *what team did david beckham play for in 2011* shown in Table 2. We failed to perform reasoning over the time constraint *2011* due to limited context left for *2011* after we addressed *play for* in earlier hops. Also due to this same reason, our model can not correctly answer the second question in Table 2, *who is the governor of India 2009*. Keep in mind that, we actually do not have annotated addressed keys at each hop to explicitly teach our model. The remaining cases also include failures in deep analysis according to specific KBs. For example, in *who is the mother of prince michael jackson*, our model only matches the triple *michael jackson fb:parents*, but fail to spot *female* in the next hop.

Method	Recall	Precision	F-measure
Xser	0.71	0.72	0.72
gAnswer	0.37	0.37	0.37
CASIA	0.40	0.32	0.36
our model	**0.78**	**0.82**	**0.81**

Table 4: Results of the models on the test set of QALD.

5 Conclusions

In this paper, we propose to apply the KV-MemNNs as a semantic parsing module to approach the open-domain KB-QA task. We introduce a novel STOP strategy to derive structured queries with flexible number of query triples during multi-hop memory reading, and present a new query updating method, which considers the already-addressed keys in previous hops as well as the value representations. Experimental results show that the STOP strategy not only enables multi-hop reasoning over the memory, but also acts as the key to construct the structured queries, which help our model achieve the state-of-the-art performances on two benchmark datasets.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments and valuable suggestions. We also thank Yao Fu for his constructive comments on the early version of the draft. This work is supported by National High Technology R&D Program of China (Grant No.2018YFC0831905), Natural Science Foundation of China (Grant No. 61672057, 61672058). For any correspondence, please contact Yansong Feng.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*, 2007.

Jun-Wei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. Constraint-based question answering with knowledge graph. In *Proceedings of the International Conference on Computational Linguistics (COLING 2016)*, pages 2503–2514, 2016.

Hannah Bast and Elmar Haussmann. More accurate question answering on freebase. In *CIKM*, 2015.

Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, 2014.

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

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075, 2015.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015.

Sarthak Jain. Question answering over knowledge base using factual memory networks. In *Proceedings of the Student Research Workshop, SRW@HLT-NAACL 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 109–115, 2016.

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

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

Yuxuan Lai, Yansong Feng, Xiaohan Yu, Zheng Wang, Kun Xu, and Dongyan Zhao. Lattice cnns for matching based chinese question answering. *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, 2019.

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

Siva Reddy, Mirella Lapata, and Mark Steedman. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association of Computational Linguistics*, pages 377–392, 2014.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM, 2017.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NIPS*, 2015.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Movieqa: Understanding stories in movies through question-answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 384–394, 2010.

Christina Unger, Corina Forascu, Vanessa López, Axel-Cyrille Ngonga Ngomo, Elena Cabrio, Philipp Cimiano, and Sebastian Walter. Question answering over linked data (QALD-4). In *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014.*

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

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, 2014.

Xuchen Yao. Lean question answering over freebase from scratch. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2015)*, 2015.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015.