The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

# Variational Reasoning for Question Answering with Knowledge Graph

**Yuyu Zhang***
Georgia Institute of Technology
yuyu.zhang@cc.gatech.edu

**Hanjun Dai***
Georgia Institute of Technology
hanjun.dai@cc.gatech.edu

**Zornitsa Kozareva**
Amazon Web Services
kozareva@amazon.com

**Alexander J. Smola**
Amazon Web Services
smola@amazon.com

**Le Song**
Georgia Institute of Technology
lsong@cc.gatech.edu

**Abstract**

Knowledge graph (KG) is known to be helpful for the task of question answering (QA), since it provides well-structured relational information between entities, and allows one to further infer indirect facts. However, it is challenging to build QA systems which can learn to reason over knowledge graphs based on question-answer pairs alone. First, when people ask questions, their expressions are noisy (for example, typos in texts, or variations in pronunciations), which is non-trivial for the QA system to match those mentioned entities to the knowledge graph. Second, many questions require multi-hop logic reasoning over the knowledge graph to retrieve the answers. To address these challenges, we propose a novel and unified deep learning architecture, and an end-to-end variational learning algorithm which can handle noise in questions, and learn multi-hop reasoning simultaneously. Our method achieves state-of-the-art performance on a recent benchmark dataset in the literature. We also derive a series of new benchmark datasets, including questions for multi-hop reasoning, questions paraphrased by neural translation model, and questions in human voice. Our method yields very promising results on all these challenging datasets.

## 1 Introduction

Question answering (QA) has been a long-standing research problem in Machine Learning and Artificial Intelligence. Thanks to the creation of large-scale knowledge graphs such as DBPedia (Auer et al. 2007) and Freebase (Bollacker et al. 2008), QA systems can be armed with well-structured knowledge on specific and open domains. Many traditional approaches for KG-powered QA are based on semantic parsers (Clarke et al. 2010; Liang, Jordan, and Klein 2011; Berant et al. 2013; Yih et al. 2015), which first map a question to formal meaning representation (e.g. logical form) and then translate it to a KG query. The answer to the question can be retrieved by executing the query. One of the disadvantages of these approaches is that the model is not trained end-to-end and errors may be cascaded.

With the recent success of deep learning, some end-to-end solutions based on neural networks have been proposed and show very promising performance on benchmark datasets, such as Memory Networks (Weston, Chopra, and Bordes 2014), Key-Value Memory Networks (Miller, Fisch, and et. al. 2016) and Gated Graph Sequence Neural Networks (Li et al. 2015). However, these neural approaches treat the KG as a flattened big table of itemized knowledge records, making it hard to exploit the structure information in the graph and thus weak on logic reasoning. When the answer is not a direct neighbor of the topic entity in question (i.e. there are multiple hops between question and answer entities in the KG), which requires logic reasoning over the KG, the neural approaches usually perform poorly. For instance, it is easy to handle single-hop questions like "*Who wrote the paper titled ...?*" by querying itemized knowledge records in triples (*paper_title*, *authored_by*, *author_name*). However, logic reasoning on the KG is required for multi-hop questions such as "*Who have co-authored papers with ...?*". With the KG, we start from the mentioned author, and follow $author \xrightarrow{authored} paper \xrightarrow{authored\_by} author$ to find answers. A common remedy is the so-called knowledge graph completion: create new relations for non-neighbor entity pairs in the KG (Socher et al. 2013a; Dong et al. 2014; Guu, Miller, and Liang 2015). However, multi-hop reasoning is combinatorial in nature, i.e. the number of multi-hop relations grow explosively with the increase of hops. For example, if we create new relation types like *friend-of-friend* and *friend-of-friend-of-friend*, the number of edges in the KG will explode, which is intractable for both storage and computation.

Another key challenge is how to locate topic entities in the KG. Most existing works assume that the topic entity in question can be located by simple string matching (Miller, Fisch, and et. al. 2016; Dodge et al. 2015; Li et al. 2015; Berant et al. 2013), which is often not true. When people ask questions, either in text or speech, various noise can be introduced in the expressions. For example, people are likely to make typos or name ambiguity in question. In even harder case, audio questions, people may pronounce the same entity differently in different questions, even for the same person. Due to these noises, it is hard to do exact matching to locate topic entities. For text questions, broad matching techniques (e.g. hand-craft rules, regular expressions, edit distance, etc.) are.g.e widely used for entity recognition (Rao, McNamee, and Dredze 2013). However, they require domain experts and lots of human effort. For speech questions, it is even harder

*Both authors contributed equally to the paper.

to match topic entities directly. Most existing QA systems first do speech recognition, converting the audio to text, and then match entities in text. Unfortunately, the error rate is typically high for speech recognition system to recognize entities in voice, such as human names or street addresses. Since it is not end-to-end, the error of the speech recognition system may cascade to affect the downstream QA system.

Typically, the training data for QA system is provided as question-answer pairs, where fine-grained annotation of these pairs are not available, or only available for a few. More specifically, there are very few explicit annotations of the exact entity present in the question, the type of the questions, and the exact logic reasoning steps along the knowledge graph leading to the answer. Thus it is challenging to simultaneously learn to locate the topic KG entity in the question, and figure out the unknown reasoning steps pointing to the answer based on training question-answer pairs alone.

To address the challenges mentioned above, we propose an end-to-end learning framework for question answering with knowledge graph named variational reasoning network (VRN), which have the following new features:

- We build a probabilistic modeling framework for end-to-end QA system, which can simultaneously handle uncertain topic entity and multi-hop reasoning.
- We propose a novel propagation-like deep learning architecture over the knowledge graph to perform logic inference in the probabilistic model.
- We apply the REINFORCE algorithm with variance reduction technique to make the system end-to-end trainable.
- We derive a series of new challenging benchmark datasets METAQA[1] (MoviE Text Audio QA) intended for research on question-answering systems. These datasets contain over 400K questions for both single- and multi-hop reasoning. To test QA systems in more realistic (and more difficult) scenarios, METAQA also provides neural-translation-model-paraphrased datasets, and text-to-speech-based audio datasets.

Extensive experiments show that our method achieves state-of-the-art performance on both single- and multi-hop datasets, demonstrating the capability of multi-hop reasoning. Moreover, we obtain promising results on the challenging audio QA datasets, showing the effectiveness of end-to-end learning framework. With the rise of virtual assistant tools (e.g. Alexa, Cortana, Google Assistant and Siri), QA systems are now even closer to our daily life. This paper is one step towards more realistic QA systems, which can handle noisy question input in both text and speech, and learn from examples to reason over the knowledge graph.

## 2 Related Work

**QA with semantic parser:** Most traditional approaches for KG-powered QA are based on semantic parsers, which map the question to a certain meaning representation or logical form (Clarke et al. 2010; Liang, Jordan, and Klein 2011; Kwiatkowski et al. 2013; Berant et al. 2013; Yih et al. 2015; Marx et al. 2014; Höffner et al. 2016), or directly map the question to an executable program (Liang et al. 2016). These approaches require domain-specific grammars, rules, or fine-grained annotations. Also, they are not designed to handle noisy questions, and do not support end-to-end training since they use separate stages for question parsing and logic reasoning.

**Neural approaches for QA:** The family of memory networks achieves state-of-the-art performance in various kinds of QA tasks. Some of them are able to do reasoning within local context (Kumar, Irsoy, and et. al. 2015; Sukhbaatar et al. 2015) using attention mechanism (Yang et al. 2015). For QA with KG, Miller, Fisch, and et. al. achieves state-of-the-art performance, outperforming previous works (Bordes, Chopra, and Weston 2014; Weston, Chopra, and Bordes 2014) on benchmark datasets. Recent work (Neelakantan et al. 2016) uses neural programmer model for QA with single knowledge table. However, the multi-hop reasoning capability of these approaches depends on recurrent attentions and there is no explicit traversal over the KG.

**Graph embedding:** Recently, researchers have built deep architectures to embed structured data, such as trees (Socher et al. 2013b; Irsoy and Cardie 2014; Mou et al. 2016) or graphs (Duvenaud et al. 2015; Dai, Dai, and Song 2016; Atwood and Towsley 2016). Also some works (Li et al. 2015; Johnson 2017) extend it to sequential case like multi-step reasoning. However, these approaches only work on small instances like sentences or molecules. Instead, our work embeds the reasoning-graph from source entity to every target entity in large-scale knowledge graph.

**Multi-hop reasoning:** There are some other works on knowledge graph completion with traversal, which requires path sampling (Guu, Miller, and Liang 2015; Neelakantan, Roth, and McCallum 2015) or dynamic programming (Toutanova et al. 2016). Our work can handle QA with natural language or human speech, and the reasoning-graph embeddings can represent complicated reasoning rules.

In summary, most of the existing approaches have separate stages for entity locating, such as keyword matching, frequency-based method, and domain-specific methods (Yang and Chang 2016). Since they are not jointly trained with the reasoning part, the errors in entity locating (e.g. incorrectly recognized name entity from speech recognition system) will be cascaded to the downstream QA system.

## 3 Model

### 3.1 Problem definition

**Knowledge base/graph (KG):** A knowledge graph is a directed graph where the entities and their relations are represented by nodes and edges, respectively, i.e. $\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}))$. Furthermore, each edge from $E(\mathcal{G})$ is a triplet $(a_i^1, r_i, a_i^2)$, representing a directed relation $r_i$ between subject entity $a_i^1$ and object entity $a_i^2$ both from the node set $V(\mathcal{G})$. Each entity in the knowledge graph can also contain additional information such as type and text description. For instance, entity $a_i^1$ is described as actor *Jennifer Lawrence*, and entity $a_i^2$ is movie *Passengers*. Then a relation in the knowledge graph can be (*Jennifer Lawrence*, *acted_in*, *Passengers*), where the corresponding $r_i$ is *acted_in*.

[1]Our new benchmark dataset collections METAQA are publicly available at https://goo.gl/f3AmcY.

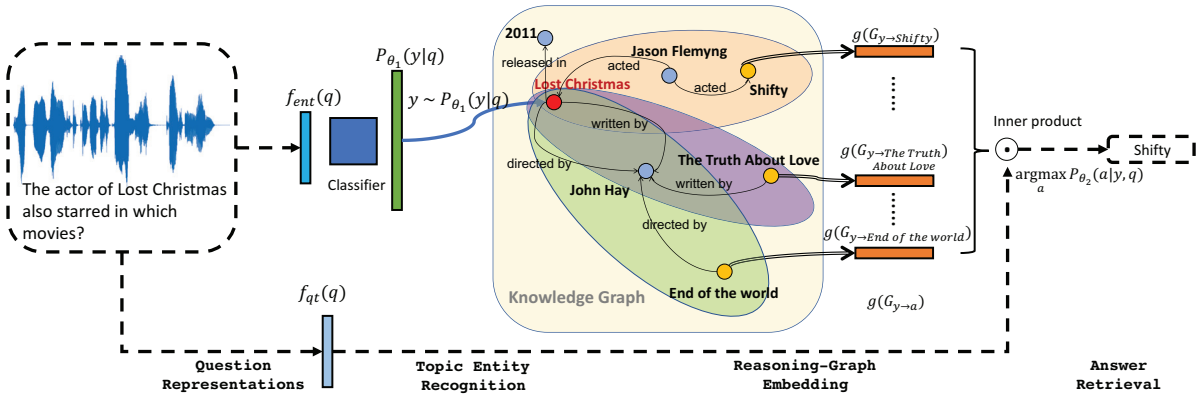


Figure 1: End-to-end architecture of the variational reasoning network (VRN) for question-answering with knowledge graph. The model consists of two probabilistic modules for topic entity recognition ($P(y|q)$) and logic reasoning over knowledge graph ($P(a|y,q)$) respectively. Inside the knowledge base plate, the scope of entity *Lost Christmas* (colored red) is illustrated, and each colored ellipsoid plate corresponds to the reasoning graph leading to a potential answer colored in yellow. The reasoning graphs are efficiently embedded and scored against the question embeddings to retrieve the best answer. During training, to handle the non-differentiable sampling operation $y \sim P(y|q)$, we use variational posterior with the REINFORCE algorithm.

**Question answering with KG:** Given a question $q$, the algorithm is asked to output an entity in the knowledge graph which properly answers the question. For example, $q$ can be a question like "*who acted in the movie Passengers*?", and one possible answer is *Jennifer Lawrence*, which is an entity in the KG. In a more challenging setting, $q$ can even be an audio segment reading the same question. The training set $D_{train} = \{(q_i, a_i)\}_{i=1}^N$ contains $N$ pairs of question and answers. Note that fine-grained annotation is not present, such as the exact entity present in the question, question type, or the exact logic reasoning steps along the knowledge graph leading to the answer. Thus, a QA system with KG should be able to handle noisy entity in questions and learn multi-hop reasoning directly from question-answer pairs.

## 3.2 Overall formulation

To address both key challenges in a unified probabilistic framework, we propose the variational reasoning network (VRN). The overall architecture is shown in Fig 1. VRN consists of two probabilistic modules, as described below.

**Module for topic entity recognition:** Recognizing the topic entity $y$ (or the entity mentioned in the question) is the first step in performing logic reasoning over the knowledge graph[2]. For example, the topic entity mentioned in Sec 3.1 is the movie *Passenger*. We denote the topic entity as $y$, and model the compatibility of this entity with the question $q_i$ as a probabilistic model $P_{\theta_1}(y|q_i)$, which shows the probability of the KG entity $y$ being mentioned in the question $q_i$. Depending on the question form (text or audio), the parameterization of $P_{\theta_1}(y|q_i)$ may be different and details can be found in Sec 3.3.

**Module for logic reasoning over knowledge graph:** Given the topic entity $y$ in question $q_i$, one need to reason over the knowledge graph to find out the answer $a_i$. As described in Sec 3.1, the algorithm should learn to use the reasoning rule $(y, \textit{acted\_by}, a_i)$ for that question. Since there is no annotations for such reasoning step, the QA system has to learn it only from question-answer pairs. Thus we model the likelihood of an answer $a_i$ being correct given entity $y$ and question $q_i$ as $P_{\theta_2}(a_i|y,q_i)$. The parameterization of $P_{\theta_2}(a_i|y,q_i)$ need to capture traversal or reasoning over knowledge graph, which is explained in detail in Sec 3.4.

Since the topic entity in question is not annotated, it is natural to formulate the problem by treating the topic entity $y$ as a latent variable. With the two probabilistic components above, we model the probability of answer $a_i$ being correct given question $q_i$ as $\sum_{y\in V(\mathcal{G})} P_{\theta_1}(y|q_i)P_{\theta_2}(a_i|y,q_i)$, which sums out all possibilities of the latent variable. Given a training set $D_{train}$ of $N$ question-answer pairs, the set of parameters $\theta_1$ and $\theta_2$ can be estimated by maximizing the log-likelihood of this latent variable model:

$$\max_{\theta_1,\theta_2} \frac{1}{N}\sum_{i=1}^{N} \log\left(\sum_{y\in V(\mathcal{G})} P_{\theta_1}(y|q_i)P_{\theta_2}(a_i|y,q_i)\right). \quad (1)$$

Next we will describe our parametrization of $P_{\theta_1}(y|q_i)$ and $P_{\theta_2}(a_i|y,q_i)$, and the algorithms for learning and inference based on that.

## 3.3 Probabilistic module for topic entity recognition

Most existing QA approaches assume that topic entities are annotated, or can be simply found via string matching. However, for more realistic questions or even audio questions, a more general approach is to build a recognizer that can be trained jointly with the logic reasoning engine.

To handle unlabeled topic entities, we notice that the full context of the question can be helpful. For example, *Michael* could either be the name of a movie or an actor. It is hard to tell which one relates to the question by merely looking

[2]In this paper, we consider the case with single topic entity in each question.

**Algorithm 1** Joint training of VRN

Initialize $\theta_1, \theta_2, \phi$ with small labeled set
**for** $i = 1$ **to** $n$ **do**
  Sample $(q_i, a_i)$ from the training data
  Sample $\{y_j\}_{j=1}^{M}$ using (8)
  Smoothing $\tilde{\mu}, \tilde{\sigma}$ with $\{A(y_j, q_i, a_i)\}_{j=1}^{M}$
  Update the baseline $b(\cdot)$ using least square
  $\psi \leftarrow \psi - \eta \nabla_\psi \mathcal{L}$ using (10)
  $\theta_1 \leftarrow \theta_1 - \eta \nabla_{\theta_1} \mathcal{L}$, $\theta_2 \leftarrow \theta_2 - \eta \nabla_{\theta_2} \mathcal{L}$
**end for**

at this entity name. However, we should be able to resolve the unique entity by checking the surrounding words in the question. Similarly, in the knowledge graph there could be multiple entities with the same name, but the connected edges (relations) of the entity nodes are different, which helps to resolve the unique entity. For example, as a movie name, *Michael* may be connected with a *directed_by* edge pointing to an entity of director; while as an actor name, *Michael* may be connected with *birthday* and *height* edges.

Specifically, we use a neural network $f_{\text{ent}}(\cdot) : q \mapsto \mathbb{R}^d$ which can represent the question $q$ in a $d$ dimensional vector. Depending on the question form (text or audio), this neural network can be a simple embedding network mapping bag-of-words to a vector, or a recurrent neural network to embed sentences, or a convolution neural network to embed audio questions. Thus the probability of having $y$ in $q$ is

$$P_{\theta_1}(y|q) = \text{softmax}\left(W_y^\top f_{\text{ent}}(q)\right) \quad (2)$$

$$= \frac{\exp(W_y^\top f_{\text{ent}}(q))}{\sum_{y' \in V(\mathcal{G})} \exp(W_{y'}^\top f_{\text{ent}}(q))}, \quad (3)$$

where $W_y \in \mathbb{R}^d, \forall y \in V(\mathcal{G})$ are the weights in the last classification layer. This parameterization avoids heuristic keyword matching for the entity as is done in previous work (Miller, Fisch, and et. al. 2016; Bordes, Chopra, and Weston 2014), and makes the entity recognition process differentiable and end-to-end trainable.

### 3.4 Probabilistic module for logic reasoning over knowledge graph

Parameterizing the reasoning model $P_{\theta_2}(a|y, q)$ is challenging, since 1) knowledge graph can be very large; 2) the required logic reasoning is unknown and can be multi-step. In other words, retrieving the answer requires multi-step traversal over a gigantic graph. Thus in this paper, we propose a *reasoning-graph* embedding architecture, where all the inference rules and their complex combinations are represented as nonlinear embeddings in vector space and will be learned.

**Scope of $y$.** More specifically, we assume the maximum number of steps (or hops), $T$, of the logic reasoning is known to the algorithm. Starting from a topic entity $y$, we perform topological sort (ignoring the original edge direction) for all entities within $T$ hops according to the knowledge graph. After that, we get an ordered list of entities $a_1, a_2, \ldots, a_m$ and their relations from the knowledge graph. We call this subgraph $\mathcal{G}_y$ with ordered nodes as the scope of $y$. Fig 2 shows

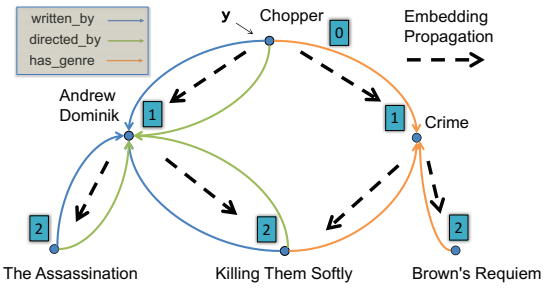


Figure 2: A question like "*movie sharing same genre and director*" would require two reasoning paths $y \to$ *Crime* $\to a$ and $y \to$ *Andrew Dominik* $\to a$. The vector representation should encode the information of the entire reasoning-graph, which can be computed recursively. Thus the embedding of *Andrew Dominik* can be reused by *The assassination* and *Killing Them Softly*.

an example of a 2-hop scope, where entities are labeled with their topological distance to the source entity.

**Reasoning graph to $a$.** Given a potential answer $a$ in the scope $\mathcal{G}_y$, we denote $\mathcal{G}_{y \to a}$ to be the minimum subgraph that contains all the paths from $y$ to $a$ in $\mathcal{G}_y$. The actual logic reasoning leading to answer $a$ for question $q$ is unknown but hidden in the reasoning graph. Thus we will learn a vector representation (or embedding) for $\mathcal{G}_{y \to a}$, denoted as $g(\mathcal{G}_{y \to a}) \in \mathbb{R}^d$, for scoring the compatibility of the question type and the hidden path in the reasoning graph.

More specifically, suppose the question is embedded using a neural network $f_{\text{qt}}(\cdot) : q \mapsto \mathbb{R}^d$, which captures the question type and implies the type of logic reasoning we need to perform over knowledge graph. Then the compatibility (or likelihood) of answer $a$ being correct can be computed using the embedded reasoning graph $\mathcal{G}_{y \to a}$ and the scope $\mathcal{G}_y$ as

$$P_{\theta_2}(a|y, q) = \text{softmax}\left(f_{\text{qt}}(q)^\top g(\mathcal{G}_{y \to a})\right) \quad (4)$$

$$= \frac{\exp(f_{\text{qt}}(q)^\top g(\mathcal{G}_{y \to a}))}{\sum_{a' \in V(\mathcal{G}_y)} \exp(f_{\text{qt}}(q)^\top g(G_{y \to a'}))}. \quad (5)$$

We note that the normalization in the likelihood requires the embedding of the reasoning graphs for all entities $a'$ in the scope $\mathcal{G}_y$. This may involve thousands of or even more reasoning graphs depending on the KG and the number of hops. Computing these embeddings separately can be very computationally expensive. Instead, we develop a neural architecture which can compute these embeddings jointly and share intermediate computations.

**Joint embedding reasoning graphs.** More specifically, we propose a "forward graph embedding" architecture, which is analogous to forward filtering in Hidden Markov Model or Bayesian Network. The embedding of the reasoning graph for $a$ is computed recursively using its parents' embeddings:

$$g(\mathcal{G}_{y \to a}) = \frac{1}{\#\text{Parent}(a)} \sum_{a_j \in \text{Parent}(a), (a_j, r, a) \text{ or } (a, r, a_j) \in \mathcal{G}_y} \sigma(V \times [g(\mathcal{G}_{y \to a_j}), \vec{e}_r]), \quad (6)$$

where $\vec{e}_r$ is the one-hot encoding of relation type $r \in \mathcal{R}$, $V \in \mathbb{R}^{d \times (d + |\mathcal{R}|)}$ are the model parameters, $\sigma(\cdot)$ is a nonlinear

function such as ReLU, and $\#\text{Parent}(a)$ counts the number of parents of $a$ in $\mathcal{G}_y$. The only boundary case is $g(G_{y\to y}) = \vec{0}$ when $y = a$. Overall, computing the embedding $g(\mathcal{G}_{y\to a})$ for all $a$ takes $O(|V(\mathcal{G}_y)|+|E(\mathcal{G}_y)|)$ time, which is proportional to the number of nodes and edges in the scope $\mathcal{G}_y$.

This formulation is able to capture various reasoning rules. Take Fig 2 as an example: the embedding of the entity *Killing Them Softly* sums up the two embeddings propagated from its parents. Thus it tends to match the reasoning paths from the parent entities. Note that this formulation is significantly different from the work in (Duvenaud et al. 2015; Dai, Dai, and Song 2016; Atwood and Towsley 2016), where embedding is computed for each small molecular graph separately. Furthermore, those graph embedding methods often contain iterative processes which visit each nodes multiple times.

# 4 End-to-end Learning

In this section, we describe the algorithm for learning the parameters in $P_{\theta_1}(y|q)$ and $P_{\theta_2}(a|y,q)$. The overall learning algorithm is described in Algorithm 1.

## 4.1 Variational method with inverse reasoning-graph embedding

EM algorithm is often used to learn latent variable models. However, performing exact EM updates for the objective in (1) is intractable since the posterior cannot be computed in closed form. Instead, we use variational inference and optimize the evidence lower bound:

$$\max_{\psi,\theta_1,\theta_2} \quad \mathcal{L}(\psi,\theta_1,\theta_2) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{Q_\psi(y|q_i,a_i)}[ \log P_{\theta_1}(y|q_i) + \log P_{\theta_2}(a_i|y,q_i) - \log Q_\psi(y|q_i,a_i)], \quad (7)$$

where the variational posterior $Q_\psi(y|q,a)$ is jointly learned with the model. Note that (7) is essentially optimizing the lower bound of (1). Thus to reduce the approximation error, a powerful set of posterior distributions is necessary.

**Variational posterior.** $Q_\psi$ computes the likelihood of the topic entity $y$ for a question $q$, with additional information of answer $a$. Thus besides the direct text or acoustic compatibility of $y$ and $q$, we can also introduce logic match with the help of $a$. Similar to the forward propagation architecture used in Sec 3.4, here we can define the scope $\mathcal{G}_a$ for answer $a$, the inverse reasoning graph $G_{a\to y}$, and the inverse embedding architecture to efficiently compute the embedding $\tilde{g}(G_{a\to y})$. Finally, the variational posterior consists of two parts:

$$Q_\psi(y|q,a) \propto \exp\left(\tilde{W}_y^\top \tilde{f}_{\text{ent}}(q) + \tilde{f}_{\text{qt}}(q)^\top \tilde{g}(G_{a\to y})\right), \quad (8)$$

where the normalization is done over all entities $y'$ in the scope $\mathcal{G}_a$. Furthermore, the embedding operators $\tilde{f}_{\text{ent}}, \tilde{f}_{\text{qt}}$ and parameters $\{\tilde{W}_y\}_{y\in V(\mathcal{G})}$ are defined in the same way as (4) and (6) but with different set of parameters. One can also share the parameter to obtain a more compact model.

## 4.2 REINFORCE with variance reduction

Since the latent variable $y$ in the variational objective (7) takes discrete values, which is not differentiable with respect to $\psi$, we use the REINFORCE algorithm (Williams 1992) with variance reduction (Mnih and Gregor 2014) to tackle this problem.

First, using the likelihood ratio trick, the gradient of $\mathcal{L}$ with respect to posterior parameters $\psi$ can be computed as (for simplicity of notation, we assume that there is only one training instance, i.e., $N = 1$):

$$\nabla_\psi \mathcal{L} = \mathbb{E}_{Q_\psi(y|q,a)}\Big[\nabla_\psi \log Q_\psi(y|q,a)\, A(y,q,a)\Big], \quad (9)$$

where $A(y,q,a) = \log P_{\theta_1}(y|q) + \log P_{\theta_2}(a|y,q) - \log Q_\psi(y|q,a)$ can be treated as the learning signal in policy gradient.

Second, to reduce the variance of gradient, we center and normalize the signal $A(y,q,a)$ and also subtract a baseline function $b(q,a)$ (Mnih and Gregor 2014). Finally, the gradient in (9) can be approximated by the Monte Carlo method using $K$ samples of the latent variable from $Q_\psi$:

$$\nabla_\psi \mathcal{L} \approx \frac{1}{K}\sum_{j=1}^{K}\nabla_\psi \log Q_\psi(y_j|q,a)\left(\frac{(A(y_j,q,a)-\tilde{\mu})}{\tilde{\sigma}} - b(q,a)\right), \quad (10)$$

where $\tilde{\mu}$ and $\tilde{\sigma}$ estimate the mean and standard deviation of $A(y_j,q,a)$ with moving average. $b(q,a)$ is another neural network that fits the expected normalized learning signal. In our experiments, we simply build a two-layer perceptron with concatenated one-hot answer and question features. Here $b(q,a)$ tries to fit $\tilde{A}(y_j,q,a) = \frac{(A(y_j,q,a)-\tilde{\mu})}{\tilde{\sigma}}$ by minimizing the square loss. For other parameters $\theta_1$ and $\theta_2$ in $P_{\theta_1}(y|q)$ and $P_{(\theta_2)}(a|y,q)$ respectively, the gradients are computed in the normal way.

# 5 Inference

During inference, we are only given the question $q$, and ideally we want to find the answer by computing $\arg\max_{y,a}\log\left(P_{\theta_1}(y|q)P_{\theta_2}(a|y,q)\right)$. However, this computation is quadratic in the number of entities and thus too expensive. Alternatively, we can approximate it via beam search. So we select $k$ candidate entities $y_1, y_2, \ldots, y_k$ with top scores from $P_{\theta_1}(y|q)$, and then the answer is given by

$$a^* = \operatorname*{argmax}_{a\in\mathcal{G}_y, y\in\{y_1,y_2,\ldots,y_k\}} \log P_{\theta_2}(a|y,q). \quad (11)$$

In our experiments, we found that $k = 1$ (equivalent as greedy inference) can already achieve good performance.

# 6 Experiments

## 6.1 The METAQA benchmark

There is an existing public QA dataset named WikiMovies[3], which consists of question-answer pairs in the domain of movies and provides a medium-sized knowledge graph (Miller, Fisch, and et. al. 2016). However, it has several limitations: 1) all questions in it are single-hop, thus it

[3]It is available at https://research.fb.com/downloads/babi.

Table 1: Test results (% hits@1) on Vanilla and Vanilla-EU datasets. EU stands for entity unlabeled.

| | Vanilla 1-hop | Vanilla 2-hop | Vanilla 3-hop | Vanilla-EU 1-hop | Vanilla-EU 2-hop | Vanilla-EU 3-hop |
|---|---|---|---|---|---|---|
| VRN | **97.5** | **89.9** | **62.5** | **82.0** | **75.6** | **38.3** |
| Bordes, Chopra, and Weston's QA system | 95.7 | 81.8 | 28.4 | 39.5 | 38.3 | 26.9 |
| KV-MemNN | 95.8 | 25.1 | 10.1 | 35.8 | 10.3 | 10.5 |
| Supervised embedding | 54.4 | 29.1 | 28.9 | 18.1 | 23.2 | 25.3 |

Table 2: Test results (% hits@1) on NTM-EU and Audio-EU datasets. EU stands for entity unlabeled.

| | NTM-EU 1-hop | NTM-EU 2-hop | NTM-EU 3-hop | Audio-EU 1-hop | Audio-EU 2-hop | Audio-EU 3-hop |
|---|---|---|---|---|---|---|
| VRN | **81.3** | **69.7** | **38.0** | **37.0** | **24.6** | **21.1** |
| Bordes, Chopra, and Weston's QA system | 32.5 | 32.3 | 25.3 | 18.5 | 19.3 | 15.3 |
| KV-MemNN | 33.9 | 8.7 | 10.2 | 4.3 | 7.0 | 15.3 |
| Supervised embedding | 16.1 | 22.8 | 24.2 | 4.1 | 6.1 | 12.1 |

is not able to evaluate the ability of reasoning; 2) there is no noise on the topic entity in question, so it can be easily located in the knowledge graph; 3) it is generated from very limited number of text templates, which is easy to be exploited by models and of limited practical value. Some small datasets like WebQuestions (Berant et al. 2013) are mostly for single-hop questions; while WikiTableQuestions (Pasupat and Liang 2015) involves tiny knowledge table for each question, instead of one large-scale knowledge graph shared among all questions.

Thus in this paper, we introduce a new challenging question-answer benchmark: METAQA (MoviE Text Audio QA). It contains more than 400K questions for both single and multi-hop reasoning, and provides more realistic text and audio versions. METAQA serves as a comprehensive extension of WikiMovies. Due to the page limit, we briefly list the datasets included in METAQA below, and put more details in Appendix A[4].

- **Vanilla:** We have the original WikiMovies as the Vanilla 1-hop dataset. For multi-hop reasoning, we design 21 types of 2-hop questions and 15 types of 3-hop questions, and generate them by random sampling from a text template pool. Details and question examples are in Appendix B.
- **NTM:** Thanks to the recent breakthrough in neural translation models (NTM), we can introduce more variations over the Vanilla datasets. We use a NTM trained by dual learning techniques (He et al. 2016) to paraphrase question by first translating it from English to French, and then sample translations back to English with beam search. The questions in the NTM dataset have different wordings but keep the same meaning. This dataset also contains 1-hop, 2-hop and 3-hop categories.
- **Audio:** To make it even more practical and challenging, we generate audio datasets with the help of text-to-speech (TTS) system. We use Google TTS service to read all the questions in Vanilla. We also provide extracted MFCC features for each question. The Audio dataset also contains 1-hop, 2-hop and 3-hop categories. Note that although the audio is machine-generated, it is still much less regulated compared to text-template-generated data, and have a lot of variations in waveforms. For example, even for the same word, the TTS system can have different intonations depending on the word position in question and other context words. Visualization of the audio data can be found in Appendix C.

[4]For Appendix, refer to the full version of our paper at https://arxiv.org/abs/1709.04071.

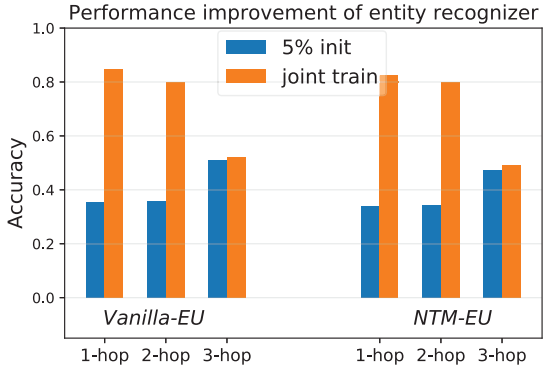


Figure 3: Improvement of the entity recognizer.

## 6.2 Competitor methods

We have three competitor methods: 1) as discussed in Sec 2, Miller, Fisch, and et. al. proposed Key-Value Memory Networks (KV-MemNN), and reported state-of-the-art results at that time on WikiMovies; 2) Bordes, Chopra, and Weston's QA system also tries to embed the inference subgraph for reasoning (Bordes, Chopra, and Weston 2014), but the representation is simply an unordered bag-of-relationships and neighbor entities; 3) the "supervised embedding" is considered as yet another baseline method, which is a simple approach but often works surprisingly well as reported in (Dodge et al. 2015).

We implement baseline methods with Tensorflow. Our results on Vanilla 1-hop are consistent with the reported performance in (Miller, Fisch, and et. al. 2016). We take

whichever higher and report it in Table 1. For example, our KV-MemNN obtains 95.8% test accuracy, while the original paper reports 93.9% on the same dataset, so we just report 95.8% in table.

When training KV-MemNN, we use the same number of "internal hops" as the hop number of that dataset. We also try to use more "internal hops" than the dataset hop number, but it is not helpful. Also, we insert knowledge items within 3 hops of the located topic entity to the memory slots, which ensures that if the topic entity is correctly matched, the answer is existing somewhere in the memory array.

### 6.3 Experimental settings

We use all the datasets in METAQA for experiments. We follow the same split of train/validation/test for all datasets. The number of questions in each part is listed in Appendix (Table 3). We tune hyperparameters on validation set for all methods. In both Vanilla and NTM, we use bag-of-words representation for entity name to parameterize $W_y$ in (3).

For Vanilla, we have two different settings: 1) provide the entity labels in all questions, so that we can compare with KV-MemNN under the same setting of Miller, Fisch, and et. al. on Vanilla 1-hop dataset; 2) only provide 5% entity labels among all questions, named as Vanilla-EU (EU stands for topic entity unlabeled). We make all the methods use bag-of-words representation of the question, and avoid hard entity matching. This setting is more of a sanity check of how much the method is dependent on labeled topic entities. In practice, hard matching can always be an option on text data, but it is not feasible for audio data.

To make task more realistic and challenging, we experiment with EU setting for NTM and Audio datasets. For NTM-EU, only 5% topic entity labels among all questions are provided. For Audio-EU, a higher labeled ratio 20% since it is much more difficult than text data. To handle the variant length of audio questions, we use a simple convolutional neural network (CNN) with three convolutional layers and three max-pooling layers to embed the audio questions into fixed-dimension vectors. We put more details about CNN embedding in Appendix D.

For all the EU setting above, the small set of entity labeled questions are used to initialize a topic entity recognizer. After that, all methods train on entire dataset but without the entity labels. For VRN, we show that this pretrained recognizer will also get improved with variational joint training; for other baselines, the entity recognizer will be fixed.

### 6.4 Results and discussions

The experimental results are listed in Table 1 and Table 2.

**Vanilla:** Since all the topic entities are labeled, Vanilla mainly evaluates the ability of logic reasoning. Note that Vanilla 1-hop is the same as WikiMovies, which is included for sanity check. All the baseline methods achieve similar performance as reported in the original papers (Miller, Fisch, and et. al. 2016; Bordes, Chopra, and Weston 2014), while our method performs the best. It is clear to see that 2- and 3-hop questions are harder, leading to significant accuracy drop on all methods. Nevertheless, our method still achieves promising results and lead competitors by a large margin. We notice that KV-MemNN is not performing well on multi-hop reasoning, perhaps due to explosion of relevant knowledge items.

**Vanilla-EU:** Without topic entity labels, all reasoning-based methods are getting worse on multi-hop questions. However, supervised embedding gets better in this case, since it just learns to remember the pair of question and answer entities. According to the statistics in Appendix (Table 4), a big portion of questions can be answered by just memorizing the pairs in training data. That explains why supervised embedding behaves differently on this dataset.

**NTM-EU:** The questions in this dataset are paraphrased by neural translation model, which increases the variety of wordings, and makes the task harder. It is reasonable that all methods are getting slightly worse results compared to Vanilla-EU. The same explanation applies to supervised embedding, which is not reasoning but memorizing all the pairs. This is indeed weak generalization and it takes advantage of the nature of this dataset, but it is not likely to perform well on new entity pairs.

**Audio-EU:** This audio dataset is the most challenging one. As mentioned in Sec 6.1, even the same word can be pronounced in a variety of intonations. It is hard to recognize the entity in audio data, also hard to tell the question type. It is not surprising that all methods perform worse compared to text data. Our method achieves 37% on 1-hop audio questions, which is very promising. For 2-hop and 3-hop questions, our method still outperforms other methods. Clearly, there is large room for improvement on audio QA. We leave it as future work, and hopefully the METAQA benchmark can facilitate more researchers working on QA systems.

### 6.5 Model ablation

Since our framework uses variational method to jointly learn the entity recognizer and reasoning graph embedding, we here do the model ablation to answer the following two questions: *1) is the reasoning graph embedding approach necessary for inference? 2) is the variational method helpful for joint training?*

**Importance of reasoning graph embedding:** As the results shown in Table 1, our proposed VRN outperforms all the other baselines, especially in 3-hop setting. Since this experiment only compares the reasoning ability, it clearly shows that simply representing the inference rule as linear combination of reasoning graph entities is not enough.

**Improvement of entity recognition with joint training:** In Fig 3 we show that using our joint training framework with variance reduction REINFORCE, we can improve the entity recognition performance further without the corresponding topic entity label supervision. For 1-hop and 2-hop questions, our model can improve greatly. While for 3-hop, since the inference task is much harder, we can only marginally improve the performance. For audio data, we've improved by 10% in 1-hop case, and it is hard to improve further for multi hops. In Table 1, the baselines perform significantly worse in the EU setting, due to the absence of joint training.

### 6.6 Inspection of learning and inference

We study the convergence of our learning algorithm in Appendix E.1. It shows variance reduction technique helps the

convergence significantly, while simpler tasks converge better. Also we present an example inference path with highest score in the reasoning graph in Appendix E.2. To answer "*What are the main languages in David Mandel films?*", the model learns to find the movie *EuroTrip* first through *directed* or *wrote* relationships, then follow *in_language* to get the correct answer *German*. For visualizing general multi-hop reasoning, attention mechanism in the aggregation operator of each node would be helpful.

**Acknowledgments** This project was supported in part by NSF IIS-1218749, NIH BIGDATA 1R01GM108341, NSF CAREER IIS-1350983, NSF IIS-1639792 EAGER, NSF CNS-1704701, ONR N00014-15-1-2340, Intel ISTC, NVIDIA and Amazon AWS.

## References

Atwood, J., and Towsley, D. 2016. Diffusion-convolutional neural networks. In *NIPS*.

Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; and Ives, Z. 2007. Dbpedia: A nucleus for a web of open data. *The semantic web*.

Berant, J.; Chou, A.; Frostig, R.; and Liang, P. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Bordes, A.; Chopra, S.; and Weston, J. 2014. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*.

Clarke, J.; Goldwasser, D.; Chang, M.-W.; and Roth, D. 2010. Driving semantic parsing from the world's response. In *Proceedings of the fourteenth conference on computational natural language learning*.

Dai, H.; Dai, B.; and Song, L. 2016. Discriminative embeddings of latent variable models for structured data. In *ICML*.

Dodge, J.; Gane, A.; Zhang, X.; Bordes, A.; Chopra, S.; Miller, A.; Szlam, A.; and Weston, J. 2015. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv*.

Dong, X.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K.; Strohmann, T.; Sun, S.; and Zhang, W. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*.

Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*.

Guu, K.; Miller, J.; and Liang, P. 2015. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*.

He, D.; Xia, Y.; Qin, T.; Wang, L.; Yu, N.; Liu, T.; and Ma, W.-Y. 2016. Dual learning for machine translation. In *NIPS*.

Höffner, K.; Walter, S.; Marx, E.; Usbeck, R.; Lehmann, J.; and Ngonga Ngomo, A.-C. 2016. Survey on challenges of question answering in the semantic web. *Semantic Web* (Preprint):1–26.

Irsoy, O., and Cardie, C. 2014. Deep recursive neural networks for compositionality in language. In *NIPS*.

Johnson, D. D. 2017. Learning graphical state transitions. In *ICLR*.

Kumar, A.; Irsoy, O.; and et. al. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Kwiatkowski, T.; Choi, E.; Artzi, Y.; and Zettlemoyer, L. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP*.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Liang, C.; Berant, J.; Le, Q.; Forbus, K. D.; and Lao, N. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Liang, P.; Jordan, M. I.; and Klein, D. 2011. Learning dependency-based compositional semantics. In *ACL*.

Marx, E.; Usbeck, R.; Ngomo, A.-C. N.; and et. al. 2014. Towards an open question answering architecture. In *ICSS*.

Miller, A.; Fisch, A.; and et. al. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Mnih, A., and Gregor, K. 2014. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*.

Mou, L.; Li, G.; Zhang, L.; Wang, T.; and Jin, Z. 2016. Convolutional neural networks over tree structures for programming language processing. In *AAAI*.

Neelakantan, A.; Le, Q. V.; Abadi, M.; McCallum, A.; and Amodei, D. 2016. Learning a natural language interface with neural programmer. *arXiv preprint arXiv:1611.08945*.

Neelakantan, A.; Roth, B.; and McCallum, A. 2015. Compositional vector space models for knowledge base completion. *arXiv preprint arXiv:1504.06662*.

Pasupat, P., and Liang, P. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Rao, D.; McNamee, P.; and Dredze, M. 2013. Entity linking: Finding extracted entities in a knowledge base. In *Multi-source, multilingual information extraction and summarization*. Springer.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013a. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*.

Socher, R.; Perelygin, A.; Wu, J. Y.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Sukhbaatar, S.; Weston, J.; Fergus, R.; et al. 2015. End-to-end memory networks. In *NIPS*.

Toutanova, K.; Lin, X. V.; Yih, W.-t.; Poon, H.; and Quirk, C. 2016. Compositional learning of embeddings for relation paths in knowledge bases and text. In *ACL*.

Weston, J.; Chopra, S.; and Bordes, A. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

Yang, Y., and Chang, M.-W. 2016. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. *arXiv preprint arXiv:1609.08075*.

Yang, Z.; He, X.; Gao, J.; Deng, L.; and Smola, A. 2015. Stacked attention networks for image question answering. *arXiv preprint arXiv:1511.02274*.

Yih, W.-t.; Chang, M.-W.; He, X.; and Gao, J. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*.