# Neural Generative Question Answering

**Jun Yin,[1]* Xin Jiang,[2] Zhengdong Lu,[2] Lifeng Shang,[2] Hang Li,[2] Xiaoming Li[1,3]**
[1]School of Electronic Engineering and Computer Science, Peking University
[2]Noah's Ark Lab, Huawei Technologies
[3]Collaborative Innovation Center of High Performance Computing, NUDT, Changsha, China
{jun.yin,lxm}@pku.edu.cn, {jiang.xin, lu.zhengdong, shang.lifeng, hangli.hl}@huawei.com

## Abstract

This paper presents an end-to-end neural network model, named Neural Generative Question Answering (GENQA), that can generate answers to *simple factoid questions*, based on the facts in a knowledge-base. More specifically, the model is built on the encoder-decoder framework for sequence-to-sequence learning, while equipped with the ability to enquire the knowledge-base, and is trained on a corpus of question-answer pairs, with their associated triples in the knowledge-base. Empirical study shows the proposed model can effectively deal with the variations of questions and answers, and generate right and natural answers by referring to the facts in the knowledge-base. The experiment on question answering demonstrates that the proposed model can outperform an embedding-based QA model as well as a neural dialogue model trained on the same data.

## 1 Introduction

Question answering (QA) can be viewed as a special case of single-turn dialogue: QA aims at providing correct answers to the questions in natural language, while dialogue emphasizes on generating relevant and fluent responses to the messages also in natural language [Shang *et al.*, 2015; Vinyals and Le, 2015]. Recent progress in deep learning has raised the possibility of realizing generation-based QA in a purely neutralized way. That is, the answer is generated by a neural network (e.g., recurrent neural network, or RNN) based on the question, which is able to handle the flexibility and diversity of language. More importantly, the model is trained in an end-to-end fashion, and thus there is no need in building the system using linguistic knowledge, e.g., creating a semantic parser.

There is however one serious limitation of this generation-based approach to QA. It is practically impossible to store all the knowledge in a neural network to achieve a desired precision and coverage in real world QA. This is a fundamental difficulty, rooting deeply in the way in which knowledge is acquired, represented and stored. The neural network, and more generally the fully distributed way of representation, is good at representing smooth and shared patterns, i.e., modeling the flexibility and diversity of language, but improper for representing discrete and isolated concepts, i.e., depicting the lexicon of language.

On the other hand, the recent success of memory-based neural network models has greatly extended the ways of storing and accessing text information, in both short-term memory (e.g., in [Bahdanau *et al.*, 2015]) and long-term memory (e.g., in [Weston *et al.*, 2015]). It is hence a natural choice to connect a neural model for QA with a neural model of knowledge-base on an external memory, which is also related to the traditional approach of template-based QA from knowledge-base.

In this paper, we report our exploration in this direction, with a proposed model called *Neural Generative Question Answering* (GENQA). The model can generate answers to *simple factoid questions* by accessing a knowledge-base. More specifically, the model is built on the encoder-decoder framework for sequence-to-sequence learning, while equipped with the ability to enquire a knowledge-base. Its specifically designed decoder, controlled by another neural network, can switch between generating a common word (e.g., `is`) and outputting a term (e.g., "`John Malkovich`" ) retrieved from knowledge-base with a certain probability. The model is trained on a dataset composed of real world question-answer pairs associated with triples in the knowledge-base, in which all components of the model are jointly tuned. Empirical study shows the proposed model can effectively capture the variation of language and generate right and natural answers to the questions by referring to the facts in the knowledge-base. The experiment on question answering demonstrates that the proposed model can outperform an embedding-based QA model as well as a neural dialogue model trained on the same data.

## 2 Task Description

### 2.1 The learning task

We formalize generative question answering as a supervised learning task or more specifically a sequence-to-sequence learning task. A generative QA system takes a sequence of words as input question and generates another sequence of

*The work is done when the first author worked as intern at Noah's Ark Lab, Huawei Technologies.

Table 1: Examples of training instances for generative QA. The KB-words in the training instances are underlined in the examples.

| Question & Answer | Triple (*subject*, *predicate*, *object*) |
|---|---|
| Q: *How tall is Yao Ming?*<br>A: *He is 2.29m and is visible from space.* | `(Yao Ming, height, 2.29m)` |
| Q: *Which country was Beethoven from?*<br>A: *He was born in what is now Germany.* | `(Ludwig van Beethoven, place of birth, Germany)` |
| Q: *Which club does Messi play for?*<br>A: *Lionel Messi currently plays for FC Barcelona in the Spanish Primera Liga.* | `(Lionel Messi, team, FC Barcelon)` |

Table 2: Statistics of the QA data and the knowledge-base.

| Community QA | Knowledge-base | |
|---|---|---|
| #QA pairs | #entities | #triples |
| 235,171,463 | 8,935,028 | 11,020,656 |

Table 3: Statistics of the training and test dataset for GENQA

| Training Data | | Test Data | |
|---|---|---|---|
| #QA pairs | #triples | #QA pairs | #triples |
| 696,306 | 58,019 | 23,364 | 1,974 |

words as output answer. In order to provide right answers, the system is connected with a knowledge-base (KB), which contains facts. During the process of answering, the system queries the KB, retrieves a set of candidate facts and generates a correct answer to the question using the right fact. The generated answer may contain two types of "words": one is common words for composing the answer (referred to as common word) and the other is specialized words in the KB denoting the answer (referred to as KB-word).

To learn a model for the task, we assume that each training instance consists of a question-answer pair with the KB-word specified in the answer. In this paper, we only consider the case of *simple factoid question*, which means each question-answer pair is associated with a single fact (i.e., one triple) of the KB. Without loss of generality, we focus on forward relation QA, where the question is on *subject* and *predicate* of the triple and the answer is from *object*. Tables 1 shows some examples of the training instances.

## 2.2 Data

To facilitate research on the task of generative QA, we create a new dataset by collecting data from the web. We first build a knowledge-base by mining from three Chinese encyclopedia web sites[1]. Specifically we extract entities and associated triples (*subject*, *predicate*, *object*) from the structured parts (e.g. HTML tables) of the web pages at the web sites. Then the extracted data is normalized and aggregated to form a knowledge-base. In this paper we sometimes refer to an item of a triple as a constituent of knowledge-base. Second, we collect question-answer pairs by extracting from two Chinese community QA sites[2]. Table 2 shows the statistics of the knowledge-base and QA-pairs.

We automatically and heuristically construct training and test data for generative QA by "grounding" the QA pairs with the triples in the knowledge-base. Specifically, for each QA pair, a list of candidate triples with the *subject* fields appearing in the question, is retrieved by using the Aho-Corasick string searching algorithm. The triples in the candidate list are then judged by a series of rules for relevance to the QA pair. The basic requirement for relevance is that the answer contains the *object* of the triple, which specifies the KB-word in the answer. Besides, we use additional scoring and filtering rules, attempting to find out the triple that truly matches the QA pair, if there is any. As the result of processing, 720K instances (tuples of question, answer, triple) are finally obtained with an estimated accuracy of 80%, i.e., 80% of instances have truly correct grounding. The data is publicly available online[3].

The data is further randomly partitioned into training dataset and test dataset by using triple as the partition key. In this way, all the questions in the test data are regarding to the unseen facts (triples) in the training data. Table 3 shows some statistics of the datasets. By comparing the numbers of triples in Table 2 and Table 3, we can see that a large portion of facts in the knowledge-base are not present in the training and test data, which demonstrates the necessity for the system to generalize to unseen facts.

The key challenge in learning of generative QA is to find a way to jointly train the neural network model in order to conduct understanding of question, generation of answer, and retrieval of relevant facts in KB, in a single and unified framework. To make things even harder, the data for training is noisy and informal, with typos, nonstandard expressions, and a wide range of language variations, which can block the system to acquire the right question-answer patterns.

# 3 The GENQA Model

Let $Q = (x_1, \ldots, x_{T_Q})$ and $Y = (y_1, \ldots, y_{T_Y})$ denote the natural language question and answer respectively. The knowledge-base is organized as a set of triples (*subject*, *predicate*, *object*), each denoted as $\tau = (\tau_s, \tau_p, \tau_o)$. Inspired by the work on the encoder-decoder framework for neural machine translation [Cho *et al.*, 2014b; Sutskever *et al.*, 2014; Bahdanau *et al.*, 2015] and neural natural language dialogue [Shang *et al.*, 2015; Vinyals and Le, 2015; Serban *et al.*, 2015], and the work on question answering with

[1] Baidu Baike, Baike.com, Douban.com

[2] Baidu Zhidao, Sogou Wenwen

[3] https://github.com/jxfeb/Generative_QA

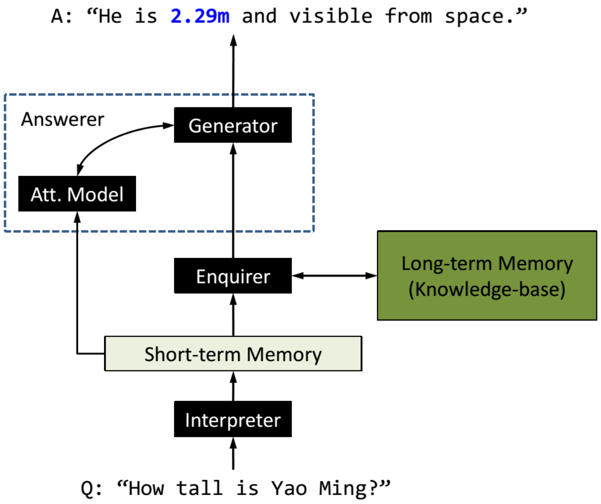


Figure 1: System diagram of GENQA.

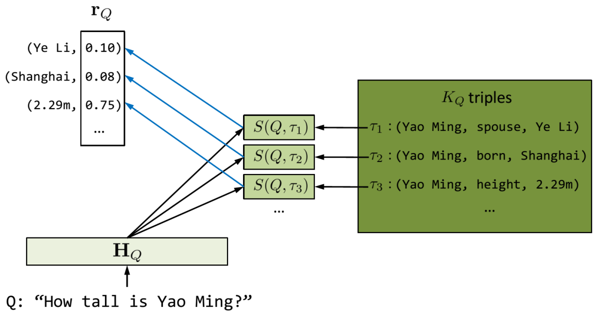

$\mathbf{r}_Q$
(Ye Li, 0.10)
(Shanghai, 0.08)
(2.29m, 0.75)
...
$S(Q,\tau_1)$
$S(Q,\tau_2)$
$S(Q,\tau_3)$
...
$K_Q$ triples
$\tau_1$ : (Yao Ming, spouse, Ye Li)
$\tau_2$ : (Yao Ming, born, Shanghai)
$\tau_3$ : (Yao Ming, height, 2.29m)
...
$\mathbf{H}_Q$
Q: "How tall is Yao Ming?"

Figure 2: Enquirer of GENQA.

knowledge-base embedding [Bordes *et al.*, 2014b; 2014a; 2015], we propose an end-to-end neural network model for generative QA, named GENQA, which is illustrated in Figure 1.

The GENQA model consists of **Interpreter**, **Enquirer**, **Answerer**, and an external knowledge-base. **Answerer** further consists of **Attention Model** and **Generator**. Basically, **Interpreter** transforms the natural language question $Q$ into a representation $\mathbf{H}_Q$ and saves it in the short-term memory. **Enquirer** takes $\mathbf{H}_Q$ as input to interact with the knowledge-base in the long-term memory, retrieves relevant facts (triples) from the knowledge-base, and summarizes the result in a vector $\mathbf{r}_Q$. The **Answerer** feeds on the question representation $\mathbf{H}_Q$ (through the **Attention Model**) as well as the vector $\mathbf{r}_Q$ and generates an answer with **Generator**. We elaborate each component hereafter.

### 3.1 Interpreter

Given the question represented as word sequence $Q = (x_1, \ldots, x_{T_Q})$, Interpreter encodes it to an array of vector representations. In our implementation, we adopt a bidirectional recurrent neural network (RNN) as in [Bahdanau *et al.*, 2015], which processes the sequence forward and backward by using two independent RNNs (here we use gated recurrent unit (GRU) [Chung *et al.*, 2014]). By concatenating the hidden states (denoted as $(\mathbf{h}_1, \cdots, \mathbf{h}_{T_Q})$), the embeddings of words (denoted as $(\mathbf{x}_1, \cdots, \mathbf{x}_{T_Q})$), and the one-hot representations of words, we obtain an array of vectors $\mathbf{H}_Q = (\tilde{\mathbf{h}}_1, \cdots, \tilde{\mathbf{h}}_{T_Q})$, where $\tilde{\mathbf{h}}_t = [\mathbf{h}_t; \mathbf{x}_t; x_t]$. This array of vectors is saved in the short-term memory, allowing for further processing by Enquirer and Answerer.

### 3.2 Enquirer

Enquirer "fetches" relevant facts from the knowledge-base with $\mathbf{H}_Q$ (as illustrated in Figure 2). Enquirer first performs term-level matching (similar to the method of associating question-answer pairs with triples described in Section 2) to retrieve a list of relevant candidate triples, denoted as $\mathcal{T}_Q = \{\tau_k\}_{k=1}^{K_Q}$. $K_Q$ is the number of candidate triples, which is at most several hundreds in our data. After obtaining $\mathcal{T}_Q$, Enquirer then evaluates the relevance of each candidate triple with the question in the embedded space[Bordes *et al.*, 2014b; 2014a].

More specifically Enquirer calculates the relevance (matching) scores between the question and the $K_Q$ triples. For question $Q$, the scores are represented in a $K_Q$-dimensional vector $\mathbf{r}_Q$ where the $k^{th}$ element of $\mathbf{r}_Q$ is defined as the probability

$$r_{Q_k} = \frac{e^{S(Q,\tau_k)}}{\sum_{k'=1}^{K_Q} e^{S(Q,\tau_{k'})}},$$

where $S(Q, \tau_k)$ denotes the matching score between question $Q$ and triple $\tau_k$.

The probability in $\mathbf{r}_Q$ will be further taken into the probabilistic model in Answerer for generating an answer. Since $\mathbf{r}_Q$ is of modest size, the number of triples involved in the matching score calculation is limited and the efficiency of the process is significantly enhanced. This is particularly true in the learning phase in which the parameters can be efficiently optimized from the supervision signals through back-propagation.

In this work, we provide two implementations for Enquirer to calculate the matching scores between question and triples.

**Bilinear Model** The first implementation simply takes the average of the word embedding vectors in $\mathbf{H}_Q$ as the representation of the question (with the result denoted as $\bar{\mathbf{x}}_Q$) . For each triple $\tau$ in the knowledge-base, it takes the average of the embeddings of its *subject* and *predicate* as the representation of the triple (denoted as $\mathbf{u}_\tau$). Then we define the matching score as

$$\bar{S}(Q, \tau) = \bar{\mathbf{x}}_Q^\top \mathbf{M} \mathbf{u}_\tau,$$

where $\mathbf{M}$ is a matrix parameterizing the matching between the question and the triple.

**CNN-based Matching Model** The second implementation employs a convolutional neural network (CNN) for modeling the matching score between the question and the triple, as in [Hu *et al.*, 2014] and [Shen *et al.*, 2014]. Specifically, the question is fed to a convolutional layer followed by a max-pooling layer, and summarized as a fixed-length vector, denoted as $\hat{\mathbf{h}}_Q$. Then $\hat{\mathbf{h}}_Q$ and $\mathbf{u}_\tau$ (again as the average of the embedding of the corresponding *subject* and *predicate*) are concatenated as input to a multi-layer perceptron (MLP) to produce their matching score

$$\hat{S}(Q,\tau) = f_{\text{MLP}}([\hat{\mathbf{h}}_Q; \mathbf{u}_\tau]).$$

For this model the parameters consist of those in the CNN and the MLP.

### 3.3 Answerer

Answerer uses an RNN to generate an answer based on the information of question saved in the short-term memory (represented as $\mathbf{H}_Q$) and the relevant facts retrieved from the long-term memory (indexed by $\mathbf{r}_Q$), as illustrated in Figure 3. The probability of generating the answer $Y = (y_1, y_2, \ldots, y_{T_Y})$ is defined as

$$p(y_1, \cdots, y_{T_Y} | \mathbf{H}_Q, \mathbf{r}_Q; \theta) = p(y_1 | \mathbf{H}_Q, \mathbf{r}_Q; \theta) \prod_{t=2}^{T_Y} p(y_t | y_1, \ldots, y_{t-1}, \mathbf{H}_Q, \mathbf{r}_Q; \theta)$$

where $\theta$ represents the parameters in the GENQA model. The conditional probability in the RNN model (with hidden states $\mathbf{s}_1, \cdots, \mathbf{s}_{T_Y}$) is specified by

$$p(y_t | y_1, \ldots, y_{t-1}, \mathbf{H}_Q, \mathbf{r}_Q; \theta) = p(y_t | y_{t-1}, \mathbf{s}_t, \mathbf{H}_Q, \mathbf{r}_Q; \theta).$$

In generating the $t^{th}$ word $y_t$ in the answer, the probability is given by the following mixture model

$$\begin{aligned} p(y_t | y_{t-1}, \mathbf{s}_t, \mathbf{H}_Q, \mathbf{r}_Q; \theta) = \\ & p(z_t = 0 | \mathbf{s}_t; \theta) p(y_t | y_{t-1}, \mathbf{s}_t, \mathbf{H}_Q, z_t = 0; \theta) + \\ & p(z_t = 1 | \mathbf{s}_t; \theta) p(y_t | \mathbf{r}_Q, z_t = 1; \theta), \end{aligned}$$

which sums the contributions from the "language" part and the "knowledge" part, with the coefficient $p(z_t | \mathbf{s}_t; \theta)$ being realized by a logistic regression model with $\mathbf{s}_t$ as input. Here the latent variable $z_t$ indicates whether the $t^{th}$ word is generated from a common vocabulary (for $z_t = 0$) or a KB vocabulary ($z_t = 1$). In this work, the KB vocabulary contains all the *objects* of the candidate triples associated with the particular question. For any word $y$ that is *only* in the KB vocabulary, e.g., "2.29m", we have $p(y_t | y_{t-1}, \mathbf{s}_t, \mathbf{H}_Q, z_t = 0; \theta) = 0$, while for $y$ that does not appear in KB, e.g., "and", we have $p(y_t | \mathbf{r}_Q, z_t = 1; \theta) = 0$. There are some words (e.g., "Shanghai") that appear in both common vocabulary and KB vocabulary, for which the probability contains nontrivial contributions from both bodies.

In generating common words, Answerer acts in the same way as the decoder of RNN in [Bahdanau *et al.*, 2015] with information from $\mathbf{H}_Q$ selected by the attention model. Specifically, the hidden state at $t$ step is computed as $\mathbf{s}_t = f_s(y_{t-1}, \mathbf{s}_{t-1}, c_t)$ and $p(y_t | y_{t-1}, \mathbf{s}_t, \mathbf{H}_Q, z_t = 0; \theta) = f_y(y_{t-1}, \mathbf{s}_t, c_t)$, where $c_t$ is the context vector computed as a weighted sum of the hidden states stored in the short-term memory $\mathbf{H}_Q$.

In generating KB-words via $p(y_t | \mathbf{r}_Q, z_t = 1; \theta)$, Answerer simply employs the model $p(y_t = k | \mathbf{r}_Q, z_t = 1; \theta) = r_{Q_k}$. The better a triple matched with the question, the more likely the *object* of the triple is selected.

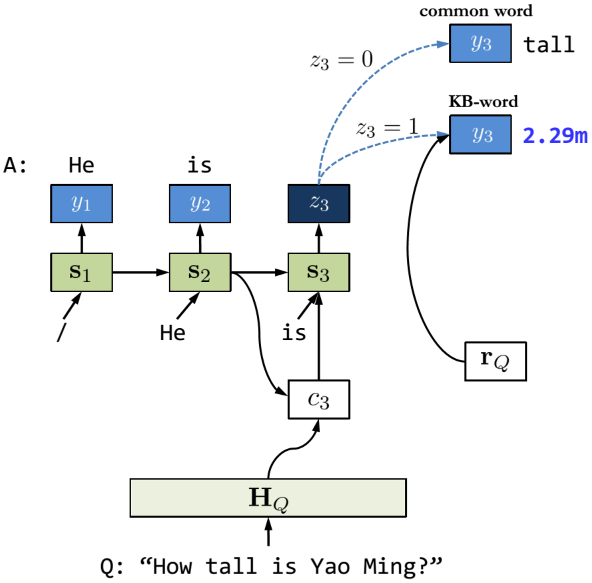


Figure 3: Answerer of GENQA.

### 3.4 Training

The parameters to be learned include the weights in the RNNs for Interpreter and Answerer, parameters in Enquirer (either the matrix $\mathbf{M}$ or the weights in the convolution layer and MLP), and the word-embeddings which are shared by the Interpreter RNN and the knowledge-base. GENQA, although essentially containing a retrieval operation, can be trained in an end-to-end fashion by maximizing the likelihood of observed data, since the mixture form of probability in Answerer provides a unified way to generate words from the common vocabulary and the KB vocabulary. Specifically, given the training data $\mathcal{D} = \{(Q^{(i)}, Y^{(i)}, \mathcal{T}_Q^{(i)})\}$, the optimal parameters are obtained by minimizing the negative log-likelihood with regularization on all the parameters

$$\ell(\mathcal{D}, \theta) = -\sum_{i=1}^{N_{\mathcal{D}}} \log(Y^{(i)} | Q^{(i)}, \mathcal{T}_Q^{(i)}) + \lambda \|\theta\|_F^2.$$

In practice the model is trained on machines with GPUs by using stochastic gradient-descent with mini-batch.

## 4 Experiments

### 4.1 Implementation details

The texts in Chinese in the data are converted into sequences of words using the Jieba Chinese word segmentor. Since the word distributions on questions and answers are different, we use different vocabularies for them. Specifically for questions, we use the most frequent 30K words in the questions and all the words in the *predicates* of the triples, covering 98.4% of the word usages in the questions. For answers, we use the most frequent 30K words in the answers with a coverage of 97.3%. All the out-of-vocabulary words are replaced by a special token "UNK". The dimensions of the hidden states of encoder and decoder are both set to 500, and the dimension of the word-embedding is set to 300. Our models are trained on an NVIDIA Tesla K40 GPU using Theano [Bastien *et al.*, 2012; Bergstra *et al.*, 2010], with the mini-batch size of 80. The training of each model takes about two or three days.

## 4.2 Comparison Models

To our best knowledge there is no previous work on generative QA, we choose three baseline methods: a neural dialogue model, a retrieval-based QA model, and an embedding based QA model, respectively corresponding to the generative aspect and the KB-retrieval aspect of GENQA:

**Neural Responding Machine (NRM)**: NRM [Shang *et al.*, 2015] is a neural network based generative model specially designed for short-text conversation. We train the NRM model with the question-answer pairs in the training data having the same vocabulary as GENQA. Since NRM does not access the knowledge-base during training and test, it actually remembers all the knowledge from the QA pairs in the model.

**Retrieval-based QA**: the knowledge-base is indexed by an information retrieval system (we use Apache Solr), in which each triple is deemed as a document. At the test phase, a question is used as the query and the top-retrieved triple is returned as the answer. Note that this method cannot generate natural language answers.

**Embedding-based QA**: as proposed by [Bordes *et al.*, 2014a; 2014b], the model is learnt from the question-triple pairs in the training data. The model learns to map questions and knowledge-base constituents into the same embedding space, where the similarity between question and triple is computed as the inner product of two embedding vectors. Different from the cross-entropy loss used in GENQA, this model uses a ranking loss function as follows:

$$\sum_{i=1}^{N_{\mathcal{D}}} \sum_{\tau,\tau' \in \mathcal{T}^{(i)}} \max(0, m - S(Q^{(i)}, \tau) + S(Q^{(i)}, \tau')),$$

where $\tau$ and $\tau'$ represent the positive and negative triples corresponding to the question. Similar to the retrieval-based QA, this model cannot generate natural language answers either.

Since we have two implementations of Enquirer of the GENQA model, we denote the one using the bilinear model as GENQA and the other using CNN and MLP as GENQA$_{\text{CNN}}$.

## 4.3 Results

We evaluate the performance of the models in terms of 1) accuracy, i.e., the ratio of correctly answered questions, and 2) the fluency of answers. In order to ensure an accurate evaluation, we randomly select 300 questions from the test set, and manually remove the nearly duplicate cases and filter out the mistaken cases (e.g., non-factoid questions).

**Accuracy** Table 4 shows the accuracies of the models on the test set. NRM has the lowest accuracy, showing the lack of ability to accurately remember the answers and generalize to questions unseen in the training data. For example, to question "`Which country does Xavi play for as a midfielder?`" (Translated from Chinese), NRM gives the wrong answer "`He plays for France`" (Translated from Chinese), since the athlete actually plays for Spain. The retrieval-based method achieves a moderate accuracy, but like most string-matching methods it suffers from word mismatch between the question and the triples in the KB. The embedding-based QA model achieves higher accuracy on test set, thanks to its generalization ability from distributed representations. GENQA and GENQA$_{\text{CNN}}$ are both better than the competitors, showing that GENQA can further benefit from the end-to-end training of sequence-to-sequence learning. We conjecture that the task of generating the appropriate answers may help the learning of word-embeddings of questions. Among the two GENQA variants, GENQA$_{\text{CNN}}$ achieves the best accuracy, getting over half of the questions right. An explanation for that is that the convolution layer helps to capture salient features in matching. The experiment results demonstrate the ability of GENQA models to find the right answers from the KB even with regard to new facts. For example, to the example question mentioned above, GENQA gives the correct answer "`He plays for Spain`".

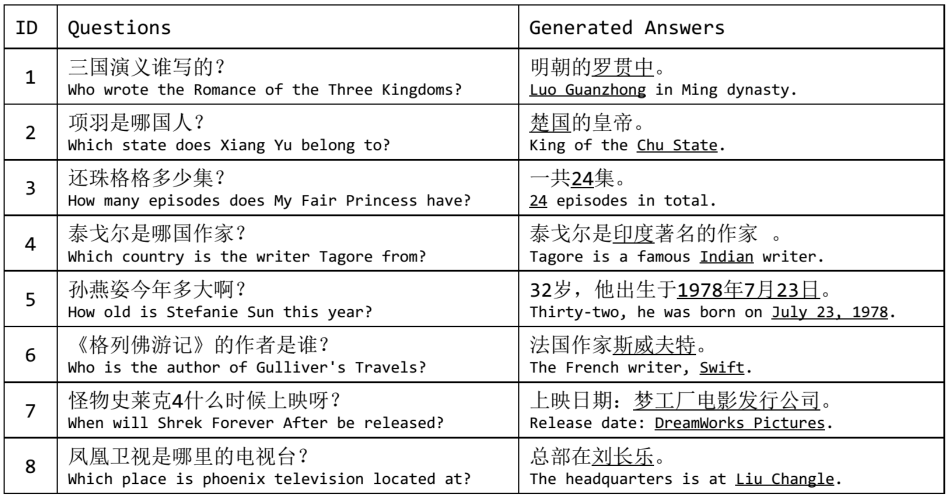
Table 4: Test accuracies

| **Models** | **Test** |
|---|---|
| Retrieval-based QA | 36% |
| NRM[Shang *et al.*, 2015] | 19% |
| Embedding-based QA [Bordes *et al.*, 2014b] | 45% |
| GENQA | 47% |
| GENQA$_{\text{CNN}}$ | **52%** |

**Fluency** We make some empirical comparisons and find no significant differences between NRM and GENQA in terms of the fluency of answers. In general, all the three generation-based models (two of them are ours) yield correct sentences in most of the time.

## 4.4 Case Study

Figure 4 gives some examples of generated answers to the questions in the test set by our GENQA models, with the underlined words generated from KB. Clearly it can smoothly blend KB-words and common words in the sentences, thanks to the unified neural model that can learn to determine the right time to place a KB-word or a common word. We notice that most of the generated answers are short sentences, for which there are two possible reasons: 1) many answers to the factoid questions in the dataset are usually short, and 2) we select the answer for each question by beam-searching the sequence with maximum log-likelihood normalized by its length, which generally prefers a short answer. Examples 1 to 4 show the correctly generated answers, for which the model not only matches the right triples (and thus generates the right KB-words), but also generates suitable common words surrounding them. However, in some cases like examples 5 and 6 even the right triples are found, the surrounding common words are improper or incorrect from the knowledge-base point of view (e.g., in example 6 the author



| ID | Questions | Generated Answers |
|---|---|---|
| 1 | 三国演义谁写的？<br>Who wrote the Romance of the Three Kingdoms? | 明朝的罗贯中。<br>Luo Guanzhong in Ming dynasty. |
| 2 | 项羽是哪国人？<br>Which state does Xiang Yu belong to? | 楚国的皇帝。<br>King of the Chu State. |
| 3 | 还珠格格多少集？<br>How many episodes does My Fair Princess have? | 一共24集。<br>24 episodes in total. |
| 4 | 泰戈尔是哪国作家？<br>Which country is the writer Tagore from? | 泰戈尔是印度著名的作家 。<br>Tagore is a famous Indian writer. |
| 5 | 孙燕姿今年多大啊？<br>How old is Stefanie Sun this year? | 32岁，他出生于1978年7月23日。<br>Thirty-two, he was born on July 23, 1978. |
| 6 | 《格列佛游记》的作者是谁？<br>Who is the author of Gulliver's Travels? | 法国作家斯威夫特。<br>The French writer, Swift. |
| 7 | 怪物史莱克4什么时候上映呀？<br>When will Shrek Forever After be released? | 上映日期：梦工厂电影发行公司。<br>Release date: DreamWorks Pictures. |
| 8 | 凤凰卫视是哪里的电视台？<br>Which place is phoenix television located at? | 总部在刘长乐。<br>The headquarters is at Liu Changle. |

Figure 4: Examples of the generated answers by GENQA.

"Jonathan Swift" is from Ireland rather than France). By investigating the correctly generated answers on test data, we find that roughly 8% of them having improper surrounding words. In some other cases, the model fails to match the correct triples with the questions, which produces completely wrong answers. For example 7, the question is about the release date of a movie, while the model finds its distributor and generates an answer incorrect both in terms of fact and language.

## 5 Related Work

Our work is inspired by recent work on neural machine translation and neural natural language dialogue. Most of neural translation models fall into the encoder-decoder framework [Cho *et al.*, 2014b; 2014a; Sutskever *et al.*, 2014], where the encoder summarizes the input sequence into a sequence of vector representations and the decoder generates the output sequence from the sequence of vector representations. Bahdanau et al. [2015] introduce the attention mechanism into the framework, and their system known as RNNsearch algorithm can jointly learn alignment and translation, and significantly improve the translation quality. This framework has also been used in natural language dialogue [Shang *et al.*, 2015; Vinyals and Le, 2015; Serban *et al.*, 2015; Wen *et al.*, 2015b; 2015a], where the end-to-end neural dialogue model is trained on a large amount of conversation data. Although promising, neural dialogue models still have problems and limitations, e.g., the lack of mechanism to incorporate knowledge.

Our work is also inspired by recent work on knowledge-base embedding and question answering from knowledge-base. TransE [Bordes *et al.*, 2013] is a method that learns the embedding vectors of the entities and the relations between entities by translating from *subject* entities to *object* entities. The model for question answering learns to embed questions and constituents in knowledge-base in the same low-dimensional space, where the similarity score between a question and a triple/subgraph is computed and the top ranked triples/subgraphs are selected as answers [Bordes *et al.*, 2014b; 2014a]. Yang et al. [Yang *et al.*, 2014] propose a method that transforms natural questions into their corresponding logical forms using joint relational embeddings, and conducts question answering by leveraging semantic associations between lexical representations and KB properties in the latent space.

Memory Networks [Weston *et al.*, 2015; Sukhbaatar *et al.*, 2015] is a recently proposed class of models that combine a large memory with a learning component that can read and write to the memory, in order to conduct reasoning for QA. Bordes et al. [2015] present an embedding-based question answering system developed under the framework of memory networks, which shows the perspective to involve more inference schemes in QA. Recently, Yin et al. [2015] propose an architecture, known as Neural Enquirer, to execute a natural language query on knowledge-base tables for question answering. It is a fully neural and end-to-end network that uses distributional representations of the query and the table, and realizes the execution of a compositional query through a series of differentiable operations.

## 6 Conclusion

In this paper we have proposed an end-to-end neural network model for generative question answering. The model is built on the encoder-decoder framework for sequence-to-sequence learning, while equipped with the ability to query a knowledge-base. Empirical studies show the proposed model is capable of generating natural and right answers to the questions by referring to the facts in the knowledgebase. In the future, we plan to continue the work on question answering and dialogue, which includes: 1) iterative question answering: a QA system that can interact with the user to confirm/clarify/answer her questions in a multi-turn dialogue; 2) question answering from complex knowledgebase: a QA system that has the ability of querying a complex-structured knowledge-base such as a knowledge graph.

## Acknowledgments

This work has been partially supported by 973 grant (2014CB340405) and NSFC grant (61272340).

## References

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[Bastien *et al.*, 2012] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

[Bergstra *et al.*, 2010] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3. Austin, TX, 2010.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2787–2795, 2013.

[Bordes *et al.*, 2014a] Antoine Bordes, Jason Weston, and Sumit Chopra. Question answering with subgraph embeddings. *EMNLP*, 2014.

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

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

[Cho *et al.*, 2014a] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[Cho *et al.*, 2014b] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*, 2014.

[Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[Hu *et al.*, 2014] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050, 2014.

[Serban *et al.*, 2015] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. *arXiv preprint arXiv:1507.04808*, 2015.

[Shang *et al.*, 2015] Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. In *Association for Computational Linguistics (ACL)*, pages 1577–1586, 2015.

[Shen *et al.*, 2014] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 373–374. International World Wide Web Conferences Steering Committee, 2014.

[Sukhbaatar *et al.*, 2015] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439, 2015.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[Vinyals and Le, 2015] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[Wen *et al.*, 2015a] Tsung-Hsien Wen, Milica Gasic, Dongho Kim, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. *arXiv preprint arXiv:1508.01755*, 2015.

[Wen *et al.*, 2015b] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*, 2015.

[Weston *et al.*, 2015] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *International Conference on Learning Representations (ICLR)*, 2015.

[Yang *et al.*, 2014] Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 645–650, 2014.

[Yin *et al.*, 2015] Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. Neural enquirer: Learning to query tables. *arXiv preprint arXiv:1512.00965*, 2015.