NEURAL GENERATIVE QUESTION ANSWERING

Jun Yin[1] **Xin Jiang**[2] **Zhengdong Lu**[2]
Lifeng Shang[2] **Hang Li**[2] **Xiaoming Li**[1]
[1]School of Electronic Engineering and Computer Science, Peking University
{jun.yin, lxm}@pku.edu.cn
[2]Noah's Ark Lab, Huawei Technologies
{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*, both in natural language. More specifically, the model is built on the encoder-decoder framework for sequence-to-sequence learning, while equipped with the ability to access an embedded knowledge-base through an attention-like mechanism. The model is trained on a corpus of question-answer pairs, with their associated triples in the given knowledge-base. Empirical study shows the proposed model can effectively deal with the language variation of the question and generate a right answer by referring to the facts in the knowledge-base. The experiment on question answering demonstrates that the proposed model can outperform the embedding-based QA model as well as the neural dialogue models 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 asked in natural language, while dialogue models often emphasize on generating relevant and fluent responses in natural language in a conversation. Recent progress in neural dialogue (Shang et al., 2015; Vinyals & Le, 2015) has raised the intriguing possibility of having a generation-based model for QA. That is, the answer is generated by a neural network (e.g., Recurrent Neural Network, or RNN) based on a proper representation of the question, hence enjoying the flexibility of language in the answer. More importantly, since it can be trained in an end-to-end fashion, there is no need for extra effort on building a semantic parser for analyzing the question. There is however one serious limitation of this generation-based approach to QA. It is practically impossible to store all the knowledge in the weights of neural network with the desired precision and coverage for real world QA. This is a fundamental difficulty, rooting deeply in the way in which knowledge of different forms and abstract levels are acquired, represented and stored. The weights of neural network, and more generally the fully distributed way of representation, are good for representing smooth and shared patterns, e.g., in language modeling, but poor for representing discrete and isolated concepts, e.g., a particular year, city name, or the height of a person. On the other hand, the recent success of memory-based neural network models has greatly extended the current scheme of representing text information in both short-term memory and long short-term memory, offering much richer ways to store and access the information of one or more sentences (Bahdanau et al., 2015; Weston et al., 2015; Yin et al., 2015). It is hence a natural choice to connect the neural model for QA with an external memory with *long-term* knowledge residing in it, which also approaches the more traditional line of research on template-based QA equipped with a knowledge-base (KB) from a different angle. In this paper, we report our exploration in this direction, with a proposed model called *Neural Generative Question Answering* (GENQA).

Learning Task: We formalize generative question answering (GENQA) as a supervised learning task or more specifically a sequence-to-sequence learning task. A GENQA system takes a sequence of words as input question and generates another sequence of words as 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 GENQA model, 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.

Dataset: We build a knowledge-base by extracting triples from three Chinese encyclopedia web sites (Baidu Baike, Hudong Baike and Douban), and collect QA pairs from two Chinese community QA sites (Baidu Zhidao and Sougou Wenwen). Training and test data for GENQA are then constructed by "grounding" the QA pairs with the triples in knowledge-base using some heuristic rules. As the result, 720K instances (tuples of question, answer, triple) are finally obtained with an estimated 80% of instances being truly positive.

2 THE NEURAL 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 $\tau = (\tau_s, \tau_p, \tau_o)$. We propose an end-to-end neural network model for GENQA, which is illustrated in Figure 1. The GENQA model consists of **Interpreter**, **Enquirer**, **Answerer**, and an external knowledge-base. Basically, **Interpreter** transforms the natural language question Q into a representation $\mathcal{H}_Q$ and saves it in the short-term memory. **Enquirer** takes $\mathcal{H}_Q$ 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 $\mathcal{H}_Q$ (through the **Attention Model**) as well as the vector $\mathbf{r}_Q$ and generates the answer with **Generator**.

Interpreter: Interpreter encodes the question Q to the array of vector representations. We adopt a bi-directional RNN as in Bahdanau et al. (2015), which processes the sequence in forward and reverse order by using two independent RNNs. By concatenating the hidden states ($\mathbf{h}_t$) and the embedding of the words ($\mathbf{x}_t$), we obtain an array of vectors $\mathcal{H}_Q = \{\tilde{\mathbf{h}}_1, \cdots, \tilde{\mathbf{h}}_{T_Q}\}$, where $\tilde{\mathbf{h}}_t = [\mathbf{h}_t; \mathbf{x}_t]$. This array of vectors is saved in the short-term memory, allowing for further processing by Enquirer and Answerer for different purposes.

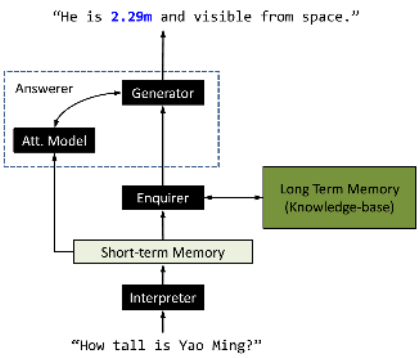

Figure 1: The diagram for GENQA.

Enquirer: Enquirer "fetches" the relevant facts from the knowledge-base with Q and $\mathcal{H}_Q$. It first performs term-level matching to retrieve a list of relevant candidate triples, denoted as $\mathcal{T}_Q = \{\tau_k\}_{k=1}^{K_Q}$. Then the task reduces to evaluating the relevance of each candidate triple with the question in the embedded space, which may be viewed as a kind of attention mechanism.

More specifically Enquirer calculates the 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 $r_{Q\,k} = \frac{e^{S(Q,\tau_k)}}{\sum_{k'=1}^{K_Q} e^{S(Q,\tau_{k'})}}$, where $S(Q, \tau_k)$ denotes the matching score between question Q and triple τ_k. The probability in $\mathbf{r}_Q$ will be further taken into the probabilistic model in Answerer for generating a particular answering sentence. In this work, we provide two implementations to calculate $S(Q, \tau_k)$ between question and triples: one is the bilinear matching model as in Bordes et al. (2014b;a), the other is the CNN-based matching model as in Hu et al. (2014) and Shen et al. (2014).

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

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

where the conditional probability in the RNN model (with hidden state $\mathbf{s}_1, \cdots, \mathbf{s}_{T_Y}$) is specified by $p(y_t|y_{t-1}, \mathbf{s}_t, \mathcal{H}_Q, \mathbf{r}_Q; \theta)$. In generating the t^{th} word y_t in the answer sentence, the probability is given by the following mixture model

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

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 $p(y_t|y_{t-1}, \mathbf{s}_t, \mathcal{H}_Q, 0; \theta) = 0$, while for y that does not appear in KB, e.g., "`and`", we have $p(y_t|\mathbf{r}_Q, 1; \theta) = 0$. There are some words (e.g., "`Shanghai`") that appear in both common vocabulary and KB vocabulary, for which the probability contains nontrivial contributions of both bodies.

In generating common words, Answerer acts in the same way as the decoder RNN in Bahdanau et al. (2015) with information from $\mathcal{H}_Q$ selected by the attention model. In generating KB words via $p(y_t|\mathbf{r}_Q, 1; \theta)$, Answerer simply employs the model $p(y_t = k|\mathbf{r}_Q, 1; \theta) = r_{Q_k}$. The better a triple matched with the question, the more likely the *object* of the triple is selected.

Training: The parameters to be learned include the weights in the RNNs for Interpreter and Answerer, parameters in the matching models of the Enquirer, 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 common vocabulary and (dynamic) KB vocabulary. In practice the model is trained on machines with GPUs by using stochastic gradient-descent with mini-batch.

3 EXPERIMENTS

To our best knowledge there is no previous work on generative QA, we choose three baseline methods: the **Neural Responding Machine (NRM)** (Shang et al., 2015), the **Retrieval-based QA** and the **Embedding-based QA** (Bordes et al., 2014a;b), respectively corresponding to the generative aspect and the KB-retrieval aspect of GENQA. For our models, we denote the one using the bilinear model as GENQA and the other based on CNN as $\text{GENQA}_{\text{CNN}}$.

Table 1: Training and test accuracies

Models	**Training**	**Test**
Retrieval-based	40%	36%
NRM	15%	19%
Embedding-based	36%	45%
GENQA	46%	47%
$\text{GENQA}_{\text{CNN}}$	59%	**52**%

We randomly select 300 questions from the training and test sets respectively, and evaluated the performance of the models in terms of answering accuracy and fluency. Table 1 shows the accuracies of the models in the training and test set respectively. NRM has the lowest accuracy on both training and test data, showing the lack of ability to remember the answers accurately 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 KB. The embedding-based QA model achieves higher accuracy on test sets, thanks to its generalization ability from distributed representations. GENQA and $\text{GENQA}_{\text{CNN}}$ are both better than the competitors, showing that GENQA can further benefit from the end-to-end training for sequence-to-sequence learning. For example, as we conjecture, the task of generating the appropriate answer may help the learning of word-embeddings of the question. Among the two GENQA variants, $\text{GENQA}_{\text{CNN}}$ 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 answer from KB even with regard to new facts. For example, to the example question mentioned above, GENQA gives the correct answer "`He plays for Spain`". 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 models based on sequence generation yield correct patterns in most of the time.

REFERENCES

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

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

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

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*, pp. 2042–2050, 2014.

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

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*, pp. 373–374. International World Wide Web Conferences Steering Committee, 2014.

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

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

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