Generating Natural Answers by Incorporating Copying and Retrieving Mechanisms in Sequence-to-Sequence Learning

Shizhu He[1], **Cao Liu**[1,2], **Kang Liu**[1] and **Jun Zhao**[1,2]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
{shizhu.he, cao.liu, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Generating answer with natural language sentence is very important in real-world question answering systems, which needs to obtain a right answer as well as a coherent natural response. In this paper, we propose an end-to-end question answering system called COREQA in sequence-to-sequence learning, which incorporates copying and retrieving mechanisms to generate natural answers within an encoder-decoder framework. Specifically, in COREQA, the semantic units (words, phrases and entities) in a natural answer are dynamically predicted from the vocabulary, copied from the given question and/or retrieved from the corresponding knowledge base jointly. Our empirical study on both synthetic and real-world datasets demonstrates the efficiency of COREQA, which is able to generate correct, coherent and natural answers for knowledge inquired questions.

1 Introduction

Question answering (QA) systems devote to providing exact answers, often in the form of phrases and entities for natural language questions (Woods, 1977; Ferrucci et al., 2010; Lopez et al., 2011; Yih et al., 2015), which mainly focus on analyzing questions, retrieving related facts from text snippets or knowledge bases (KBs), and finally predicting the answering semantic units-**SU** (words, phrases and entities) through ranking (Yao and Van Durme, 2014) and reasoning (Kwok et al., 2001).

However, in real-world environments, most people prefer the correct answer replied with a more natural way. For example, most existing commercial products such as Siri[1] will reply a natural answer "*Jet Li is 1.64m in height.*" for the question "*How tall is Jet Li?*", rather than only answering one entity "*1.64m*". Basic on this observation, we define the "natural answer" as the natural response in our daily communication for replying factual questions, which is usually expressed in a complete/partial natural language sentence rather than a single entity/phrase. In this case, the system needs to not only parse question, retrieve relevant facts from KB but also generate a proper reply. To this end, most previous approaches employed message-response patterns. Figure 1 schematically illustrates the major steps and features in this process. The system first needs to recognize the topic entity "*Jet Li*" in the question and then extract multiple related facts <*Jet Li, gender, Male*>, <*Jet Li, birthplace, Beijing*> and <*Jet Li, nationality, Singapore*> from KB. Based on the chosen facts and the commonly used message-response patterns "*where was %entity from?*" - "*%entity was born in %birthplace, %pronoun is %nationality citizen.*"[2], the system could finally generate the natural answer (McTear et al., 2016).

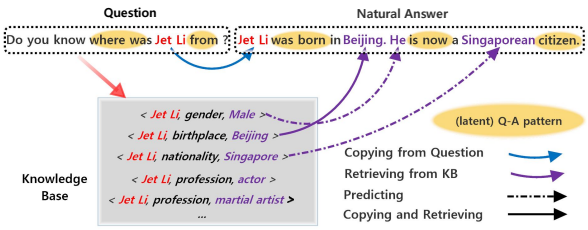

Figure 1: Incorporating copying and retrieving mechanisms in generating a natural answer.

In order to generate natural answers, typical

[1] http://www.apple.com/ios/siri/

[2] In this pattern, *%entity* indicates the placeholder of the topic entity, *%property* indicates the property value of the topic entity.

products need lots of Natural Language Processing (NLP) tools and pattern engineering (McTear et al., 2016), which not only suffers from high costs of manual annotations for training data and patterns, but also have low coverage that cannot flexibly deal with variable linguistic phenomena in different domains. Therefore, this paper devotes to develop an end-to-end paradigm that generates natural answers without any NLP tools (e.g. POS tagging, parsing, etc.) and pattern engineering. This paradigm tries to consider question answering in an end-to-end framework. In this way, the complicated QA process, including analyzing question, retrieving relevant facts from KB, and generating correct, coherent, natural answers, could be resolved jointly.

Nevertheless, generating natural answers in an end-to-end manner is not an easy task. The key challenge is that the words in a natural answer may be generated by different ways, including: *1)* the common words usually are predicted using a (conditional) language model (e.g. "*born*" in Figure 1); *2)* the major entities/phrases are selected from the source question (e.g. "*Jet Li*"); *3)* the answering entities/phrases are retrieved from the corresponding KB (e.g. "*Beijing*"). In addition, some words or phrases even need to be inferred from related knowledge (e.g. "*He*" should be inferred from the value of "*gender*"). And we even need to deal with some morphological variants (e.g. "*Singapore*" in KB but "*Singaporean*" in answer). Although existing end-to-end models for KB-based question answering, such as GenQA (Yin et al., 2016), were able to retrieve facts from KBs with neural models. Unfortunately, they cannot copy SUs from the question in generating answers. Moreover, they could not deal with complex questions which need to utilize multiple facts. In addition, existing approaches for conversational (Dialogue) systems are able to generate natural utterances (Serban et al., 2016; Li et al., 2016) in sequence-to-sequence learning (Seq2Seq). But they cannot interact with KB and answer information-inquired questions. For example, CopyNet (Gu et al., 2016) is able to copy words from the original source in generating the target through incorporating copying mechanism in conventional Seq2Seq learning, but they cannot retrieve SUs from external memory (e.g. KBs, Texts, etc.).

Therefore, facing the above challenges, this paper proposes a neural generative model called COREQA with Seq2Seq learning, which is able to reply an answer in a natural way for a given question. Specifically, we incorporate **CO**pying and **RE**trieving mechanisms within Seq2Seq learning. COREQA is able to analyze the question, retrieve relevant facts and generate a sequence of SUs using a hybrid method with a completely end-to-end learning framework. We conduct experiments on both synthetic data sets and real-world datasets, and the experimental results demonstrate the efficiency of COREQA compared with existing end-to-end QA/Dialogue methods.

In brief, our main contributions are as follows:

- We propose a new and practical question answering task which devotes to generating natural answers for information inquired questions. It can be regarded as a fusion task of QA and Dialogue.
- We propose a neural network based model, named as COREQA, by incorporating copying and retrieving mechanism in Seq2Seq learning. In our knowledge, it is the first end-to-end model that could answer complex questions in a natural way.
- We implement experiments on both synthetic and real-world datasets. The experimental results demonstrate that the proposed model could be more effective for generating correct, coherent and natural answers for knowledge inquired questions compared with existing approaches.

2 Background: Neural Models for Sequence-to-Sequence Learning

2.1 RNN Encoder-Decoder

Recurrent Neural Network (RNN) based Encoder-Decoder is the backbone of Seq2Seq learning (Cho et al., 2014). In the Encoder-Decoder framework, an encoding RNN first transform a source sequential object $X = [x_1, ..., x_{L_X}]$ into an encoded representation $\mathbf{c}$. For example, we can utilize the basic model: $\mathbf{h}_t = f(x_t, \mathbf{h}_{t-1}); \mathbf{c} = \phi(\mathbf{h}_1, ..., \mathbf{h}_{L_X})$, where $\{\mathbf{h}_t\}$ are the RNN hidden states, $\mathbf{c}$ is the context vector which could be assumed as an abstract representation of X. In practice, gated RNN variants such as LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Chung et al., 2014) are commonly used for learning long-term dependencies. And the another encoding

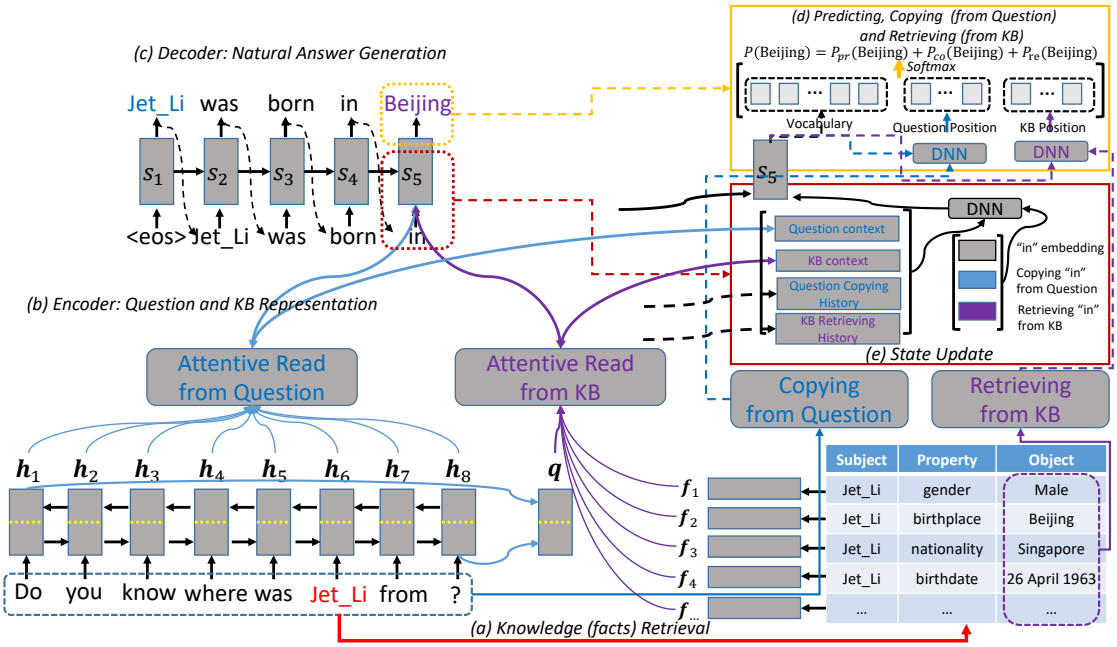

Figure 2: The overall diagram of COREQA.

tricks is Bi-directional RNN, which connect two hidden states of positive time direction and negative time direction. Once the source sequence is encoded, another decoding RNN model is to generate a target sequence $Y = [y_1, ..., y_{L_Y}]$, through the following prediction model: $\mathbf{s}_t = f(y_{t-1}, \mathbf{s}_{t-1}, \mathbf{c})$; $p(y_t|y_{<t}, X) = g(y_{t-1}, \mathbf{s}_t, \mathbf{c})$, where $\mathbf{s}_t$ is the RNN hidden state at time t, the predicted target word y_t at time t is typically performed by a $softmax$ classifier over a settled vocabulary (e.g. 30,000 words) through function g.

2.2 The Attention Mechanism

The prediction model of classical decoders for each target word y_i share the same context vector $\mathbf{c}$. However, a fixed vector is not enough to obtain a better result on generating a long targets.The attention mechanism in the decoding can dynamically choose context $\mathbf{c}_t$ at each time step (Bahdanau et al., 2014), for example, representing $\mathbf{c}_t$ as the weighted sum of the source states $\{\mathbf{h}_i\}$,

$$\mathbf{c}_t = \sum_{i=1}^{L_X} \alpha_{ti}\mathbf{h}_i; \quad \alpha_{ti} = \frac{e^{\rho(\mathbf{s}_{t-1}, \mathbf{h}_i)}}{\sum_{i'} e^{\rho(\mathbf{s}_{t-1}, \mathbf{h}'_i)}} \quad (1)$$

where the function ρ use to compute the attentive strength with each source state, which usually adopts a neural network such as multi-layer perceptron (MLP).

2.3 The Copying Mechanism

Seq2Seq learning heavily rely on the "meaning" for each word in source and target sequences, however, some words in sequences are "no-meaning" symbols and it is improper to encode them in encoding and decoding processes. For example, generating the response *"Of course, read"* for replying the message *"Can you read the word 'read'?"* should not consider the meaning of the second *"read"*. By incorporating the copying mechanism, the decoder could directly copy the sub-sequences of source into the target (Vinyals et al., 2015). The basic approach is to jointly predict the indexes of the target word in the fixed vocabulary and/or matched positions in the source sequences (Gu et al., 2016; Gulcehre et al., 2016).

3 COREQA

To generate natural answers for information inquired questions, we should first recognize key topics in the question, then extract related facts from KB, and finally fusion those instance-level knowledge with some global-level "smooth" and "glue" words to generate a coherent reply. In this section, we present COREQA, a differentiable Seq2Seq model to generate natural answers, which is able to analyze the question, retrieve relevant facts and predict SUs in an end-to-end fashion, and the predicted SUs may be predicted from the vo-

cabulary, copied from the given question, and/or retrieved from the corresponding KB.

3.1 Model Overview

As illustrated in Figure 2, COREQA is an encoder-decoder framework plugged with a KB engineer. A knowledge retrieval module is firstly employed to retrieve related facts from KB by question analysis (see Section 3.2). And then the input question and the retrieved facts are transformed into the corresponding representations by **Encoders** (see Section 3.3). Finally, the encoded representations are feed to **Decoder** for generating the target natural answer (see Section 3.4).

3.2 Knowledge (facts) Retrieval

We mainly focus on answering the information inquired questions (factual questions, and each question usually contains one or more topic entities). This paper utilizes the gold topic entities for simplifying our design. Given the topic entities, we retrieve the related facts from the corresponding KB. KB consists of many relational data, which usually are sets of inter-linked subject-property-object (*SPO*) triple statements. Usually, question contains the information used to match the *subject* and *property* parts in a fact triple, and answer incorporates the *object* part information.

3.3 Encoder

The encoder transforms all discrete input symbols (including words, entities, properties and properties' values) and their structures into numerical representations which are able to feed into neural models (Weston et al., 2014).

3.3.1 Question Encoding

Following (Gu et al., 2016), a bi-directional RNN (Schuster and Paliwal, 1997) is used to transform the question sequence into a sequence of concatenated hidden states with two independent RNNs. The forward and backward RNN respectively obtain $\{\overrightarrow{\mathbf{h}}_1, ..., \overrightarrow{\mathbf{h}}_{L_X}\}$ and $\{\overleftarrow{\mathbf{h}}_{L_X}, ..., \overleftarrow{\mathbf{h}}_1\}$. The concatenated representation is considered to be the short-term memory of question ($\mathbf{M}_Q = \{\mathbf{h}_t\}, \mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_{L_X-t+1}]$). $\mathbf{q} = [\overrightarrow{\mathbf{h}}_{L_X}, \overleftarrow{\mathbf{h}}_1]$ is used to represent the entire question, which could be used to compute the similarity between the question and the retrieved facts.

3.3.2 Knowledge Base Encoding

We use s, p and o denote the subject, property and object (value) of one fact f, and $\mathbf{e}_s$, $\mathbf{e}_p$ and $\mathbf{e}_o$ to denote its corresponding embeddings. The fact representation $\mathbf{f}$ is then defined as the concatenation of $\mathbf{e}_s$, $\mathbf{e}_p$ and $\mathbf{e}_o$. The list of all related facts' representations, $\{\mathbf{f}\} = \{\mathbf{f}_1, ..., \mathbf{f}_{L_F}\}$ (refer to $\mathbf{M}_{KB}$, L_F denotes the maximum of candidate facts), is considered to be a short-term memory of KB while answering questions about the topic entities.

In addition, given the distributed representation of question and candidate facts, we define the matching scores function between question and facts as $S(q, f_j) = DNN_1(\mathbf{q}, \mathbf{f}_j) = tanh(\mathbf{W}_2 \cdot tanh(\mathbf{W}_1 \cdot [\mathbf{q}, \mathbf{f}_j] + \mathbf{b}_1) + \mathbf{b}_2)$, , where DNN_1 is the matching function defined by a two-layer perceptron, $[\cdot, \cdot]$ denotes vector concatenation, and $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are the learning parameters. In fact, we will make a slight change of the matching function because it will also depend on the state of decoding process at different times. The modified function is $S(q, s_t, f_j) = DNN_1(\mathbf{q}, \mathbf{s}_t, \mathbf{f}_j)$ where s_t is the hidden state of decoder at time t.

3.4 Decoder

The decoder uses an RNN to generate a natural answer based on the short-term memory of question and retrieved facts which represented as $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$, respectively. The decoding process of COREQA have the following differences compared with the conventional decoder:

Answer words prediction: COREQA predicts SUs based on a mixed probabilistic model of three modes, namely the ***predict-mode***, the ***copy-mode*** and the ***retrieve-mode***, where the first mode predicts words with the vocabulary, and the two latter modes pick SUs from the questions and matched facts, respectively;

State update: the predicted word at step $t-1$ is used to update $\mathbf{s}_t$, but COREQA uses not only its word embedding but also its corresponding positional attention informations in $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$;

Reading short-Memory $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$: $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$ are fed into COREQA with two ways, the first one is the "meaning" with embeddings and the second one is the positions of different words (properties' values).

3.4.1 Answer Words Prediction

The generated words (entities) may come from vocabulary, source question and matched KB. Accordingly, our model use three correlative output

layer: *shortlist* prediction layer, *question location* copying layer and *candidate-facts location* retrieving layer, respectively. And we use the $softmax$ classifier of the above three cascaded output layers to pick SUs. We assume a vocabulary $\mathcal{V} = \{v_1, ..., v_N\} \cup \{\text{UNK}\}$, where UNK indicates any out-of-vocabulary (OOV) words. Therefore, we have adopted another two set of SUs $\mathcal{X}_Q$ and $\mathcal{X}_{KB}$ which cover words/entities in the source question and the partial KB. That is, we have adopted the instance-specific vocabulary $\mathcal{V} \cup \mathcal{X}_Q \cup \mathcal{X}_{KB}$ for each question. It's important to note that these three vocabularies $\mathcal{V}$, $\mathcal{X}_Q$ and $\mathcal{X}_{KB}$ may overlap.

At each time step t in the decoding process, given the RNN state $\mathbf{s}_t$ together with $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$, the probabilistic function for generating any target SU y_t is a "mixture" model as follow

$$\begin{aligned} &p(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{M}_Q, \mathbf{M}_{KB}) = \\ &p_{pr}(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t) \cdot p_m(pr|\mathbf{s}_t, y_{t-1}) + \\ &p_{co}(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{M}_Q) \cdot p_m(co|\mathbf{s}_t, y_{t-1}) + \\ &p_{re}(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{M}_{KB}) \cdot p_m(re|\mathbf{s}_t, y_{t-1}) \end{aligned} \tag{2}$$

where pr, co and re stand for the predict-mode, the copy-mode and the retrieve-mode, respectively, $p_m(\cdot|\cdot)$ indicates the probability model for choosing different modes (we use a $softmax$ classifier with two-layer MLP). The probability of the three modes are given by

$$\begin{aligned} p_{pr}(y_t|\cdot) &= \frac{1}{Z} e^{\psi_{pr}(y_t)} \\ p_{co}(y_t|\cdot) &= \frac{1}{Z} \sum\nolimits_{j:Q_j=y_t} e^{\psi_{co}(y_t)} \\ p_{re}(y_t|\cdot) &= \frac{1}{Z} \sum\nolimits_{j:KB_j=y_t} e^{\psi_{re}(y_t)} \end{aligned} \tag{3}$$

where $\psi_{pr}(\cdot)$, $\psi_{co}(\cdot)$ and $\psi_{re}(\cdot)$ are score functions for choosing SUs in predict-mode (from $\mathcal{V}$), copy-mode (from $\mathcal{X}_Q$) and retrieve-mode (from $\mathcal{X}_{KB}$), respectively. And Z is the normalization term shared by the three modes, $Z = e^{\psi_{pr}(v)} + \sum_{j:Q_j=v} e^{\psi_{co}(v)} + \sum_{j:KB_j=v} e^{\psi_{re}(v)}$. And the three modes could compete with each other through a $softmax$ function in generating target SUs with the shared normalization term (as shown in Figure 2. Specifically, the scoring functions of each mode are defined as follows:

***Pr*edict-mode**: Some generated words need reasoning (e.g. "He" in Figure 1) and morphological transformation (e.g. "Singaporean" in Figure 1). Therefore, we modify the function as $\psi_{pr}(y_t = v_i) = \mathbf{v}_i^T \mathbf{W}_{pr}[\mathbf{s}_t, \mathbf{c}_{q_t}, \mathbf{c}_{kb_t}]$, where $\mathbf{v}_i \in \mathcal{R}^{d_o}$ is the word vector at the output layer (not the input word embedding), $\mathbf{W}_{pr} \in \mathcal{R}^{(d_h+d_i+d_f)\times d_o}$ (d_i, d_h and d_f indicate the size of input word vector, RNN decoder hidden state and fact representation respectively), and $\mathbf{c}_{q_t}$ and $\mathbf{c}_{kb_t}$ are the temporary memory of reading $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$ at time t (see Section 3.4.3).

***Co*py-mode**: The score for "copying" the word x_j from question Q is calculated as $\psi_{co}(y_t = x_j) = DNN_2(\mathbf{h}_j, \mathbf{s}_t, \mathbf{hist}_Q)$, where DNN_2 is a neural network function with a two-layer MLP and $\mathbf{hist}_Q \in \mathcal{R}^{L_X}$ is an accumulated vector which record the attentive history for each word in question (similar with the coverage vector in (Tu et al., 2016)).

***Re*trieve-mode**: The score for "retrieving" the entity word v_j from retrieval facts ("Object" part) is calculated as $\psi_{re}(y_t = v_j) = DNN_3(\mathbf{f}_j, \mathbf{s}_t, \mathbf{hist}_{KB})$, where DNN_3 is also a neural network function and $\mathbf{hist}_{KB} \in \mathcal{R}^{L_F}$ is an accumulated vector which record the attentive history for each fact in candidate facts.

3.4.2 State Update

In the generic decoding process, each RNN hidden state $\mathbf{s}_t$ is updated with the previous state $\mathbf{s}_{t-1}$, the word embedding of previous predicted symbol y_{t-1}, and an optional context vector $\mathbf{c}_t$ (with attention mechanism). However, y_{t-1} may not come from vocabulary $\mathcal{V}$ and not owns a word vector. Therefore, we modify the state update process in COREQA. More specifically, y_{t-1} will be represented as concatenated vector of $[\mathbf{e}(y_{t-1}), \mathbf{r}_{q_{t-1}}, \mathbf{r}_{kb_{t-1}}]$, where $\mathbf{e}(y_{t-1})$ is the word embedding associated with y_{t-1}, $\mathbf{r}_{q_{t-1}}$ and $\mathbf{r}_{kb_{t-1}}$ are the weighted sum of hidden states in $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$ corresponding to y_{t-1} respectively.

$$\begin{aligned} \mathbf{r}_{q_t} = \sum\nolimits_{j=1}^{L_X} \rho_{tj}\mathbf{h}_j, \mathbf{r}_{kb_t} = \sum\nolimits_{j=1}^{L_F} \delta_{tj}\mathbf{f}_j \\ \rho_{tj} = \begin{cases} \frac{1}{K_1} p_{co}(x_j|\cdot), & x_j = y_t \\ \mathbf{0} & otherwise \end{cases} \\ \delta_{tj} = \begin{cases} \frac{1}{K_2} p_{re}(f_j|\cdot), & object(f_j) = y_t \\ \mathbf{0} & otherwise \end{cases} \end{aligned} \tag{4}$$

where $object(f)$ indicate the "object" part of fact f (see Figure 2), and K_1 and K_2 are the normalization terms which equal $\sum_{j':x'_j=y_t} p_{co}(x'_j|\cdot)$ and $\sum_{j':object(f'_j)=y_t} p_{re}(f'_j|\cdot)$, respectively, and it

could consider the multiple positions matching y_t in source question and KB.

3.4.3 Reading short-Memory $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$

COREQA employ the attention mechanism at decoding process. At each decoder time t, we selective read the context vector $\mathbf{c}_{q_t}$ and $\mathbf{c}_{kb_t}$ from the short-term memory of question $\mathbf{M}_Q$ and retrieval facts $\mathbf{M}_{KB}$ (alike to Formula 1). In addition, the accumulated attentive vectors $\mathbf{hist}_Q$ and $\mathbf{hist}_{KB}$ are able to record the positional information of SUs in the source question and retrieved facts.

3.5 Training

Although some target SUs in answer are copied and retrieved from the source question and the external KB respectively, COREQA is fully differential and can be optimized in an end-to-end manner using back-propagation. Given the batches of the source questions $\{X\}_M$ and target answers $\{Y\}_M$ both expressed with natural language (symbolic sequences), the objective function is to minimize the negative log-likelihood:

$$\mathcal{L} = -\frac{1}{N}\sum_{k=1}^{M}\sum_{t=1}^{L_Y} log[p(y_t^{(k)}|y_{<t}^{(k)}, X^{(k)}] \quad (5)$$

where the superscript (k) indicates the index of one question-answer (Q-A) pair. The network is no need for any additional labels for training models, because the three modes sharing the same $softmax$ classifier for predicting target words, they can learn to coordinate with each other by maximizing the likelihood of observed Q-A pairs.

4 Experiments

In this section, we present our main experimental results in two datasets. The first one is a small synthetic dataset in a restricted domain (only involving four properties of persons) (Section 4.1). The second one is a big dataset in open domain, where the Q-A pairs are extracted from community QA website and grounded against a KB with an Integer Linear Programming (ILP) method (Section 4.2). COREQA and all baseline models are trained on a NVIDIA TITAN X GPU using TensorFlow[3] tools, where we used the Adam (Kingma and Ba, 2014) learning rule to update gradients in all experimental configures. The sources codes and data will be released at the personal homepage of the first author[4].

4.1 Natural QA in Restricted Domain

Task: The QA systems need to answer questions involving 4 concrete properties of birthdate (including `year`, `month` and `day`) and `gender`). Through merely involving 4 properties, there are plenty of QA patterns which focus on different aspects of birthdate, for example, "*What year were you born?*" touches on "`year`", but "*When is your birthday?*" touches on "`month` and `day`".

Dataset: Firstly, 108 different Q-A patterns have been constructed by two annotators, one in charge of raising question patterns and another one is responsible for generating corresponding suitable answer patterns, e.g. `When is` $\%e$ `birthday?` $\rightarrow$ `She was born in` $\%m$ $\%d$`th.` where the variables $\%e$, $\%y$, $\%m$, $\%d$ and $\%g$ (deciding *she* or *he*) indicates the person's name, birth year, birth month, birth day and gender, respectively. Then we randomly generate a KB which contains 80,000 person entities, and each entity including four facts. Given KB facts, we can finally obtain specific Q-A pairs. And the sampling KB, patterns, and the generated Q-A pairs are shown in Table 1. In order to maintain the diversity, we randomly select 6 patterns for each person. Finally, we totally obtain 239,934 sequences pairs (half patterns may be unmatched because of "gender" property).

Q-A Patterns	**Examples** (e.g. KB facts (e2,year,1987);(e2,month,6); (e2,day,20);(e2,gender,male))
When is $\%e$ birthday? He was born in $\%m$ $\%d$th.	When is e2 birthday? He was born in June 20th.
What year were $\%e$ born? $\%e$ is born in $\%y$ year.	What year were e2 born? e2 is born in 1987 year.

Table 1: Sample KB facts, patterns and their generated Q-A pairs.

Experimental Setting: The total 239,934 Q-A pairs are split into training (90%) and testing set (10%). The baseline includes 1) generic RNN Encoder-Decoder (marked as *RNN*), 2) Seq2Seq with attention (marked as *RNN+atten*), 3) *CopyNet*, and 4) *GenQA*. For a fair comparison, we use bi-directional LSTM for encoder and another LSTM for decoder for all Seq2Seq models, with hidden layer size = 600 and word embedding dimen-

[3]https://www.tensorflow.org/

[4]http://www.nlpr.ia.ac.cn/cip/shizhuhe/publications.html

sion = 200. We set L_F as 5.

Metrics: We adopt (**automatic evaluation (AE)** to test the effects of different models. **AE** considers the precisions of the entire predicted answers and four specific properties, and the answer is complete correct only when all predicted properties' values is right. To measure the performance of the proposed method, we select following metrics, including P_g[5], P_y, P_m and P_d which denote the precisions for 'gender', 'year', 'month' and 'day' properties, respectively. And P_A, R_A and $F1_A$ indicate the precision, recall and F1 in the complete way.

Experimental Results: The **AE** experimental results are shown in Table 2. It is very clear from Table 2 that COREQA significantly outperforms all other compared methods. The reason of the GenQA's poor performance is that all synthetic questions need multiple facts, and GenQA will "safely" choose the most frequent property ("gender") for all questions. We also found the performances on "year" and "day" have a little worse than other properties such as "gender", it may because there have more ways to answer questions about "year" and "day".

Models	P_g	P_y	P_m	P_d	P_A	R_A	$F1_A$
RNN	72.2	0	1.1	0.2	0	27.5	0
RNN+atten	55.8	1.1	11.3	9.5	1.7	34	3.2
CopyNet	75.2	8.7	28.3	5.8	3.7	32.5	6.7
GenQA	73.4	0	0	0	0	27.1	0
COREQA	**100**	**84.8**	**93.4**	**81**	**87.4**	**94**	**90.6**

Table 2: The AE results (%) on synthetic test data.

Discussion: Because of the feature of directly "hard" copy and retrieve SUs from question and KB, COREQA could answer questions about unseen entities.To evaluate the effects of answering questions about unseen entities, we re-construct 2,000 new person entities and their corresponding facts about four known properties, and obtain 6,081 Q-A pairs through matching the sampling patterns mentioned above. The experimental results are shown in Table 3, it can be seen that the performance did not fall too much.

Entities	P_g	P_y	P_m	P_d	P_A	R_A	$F1_A$
Seen	100	84.8	93.4	81	87.4	94	90.6
Unseen	75.1	84.5	93.5	81.2	63.8	85.1	73.1

Table 3: The AE (%) for seen and unseen entities.

[5]The "gender" is right when the entity name (e.g. 'e2') or the personal pronoun (e.g. 'She') in answer is correct.

4.2 Natural QA in Open Domain

Task: To test the performance of the proposed approach in open domains, we modify the task of GenQA (Yin et al., 2016) for supporting multi-facts (a typical example is shown in Figure 1). That is, a natural QA system should generate a sequence of SUs as the natural answer for a given natural language question through interacting with a KB.

Dataset: GenQA have released a corpus[6], which contains a crawling KB and a set of ground Q-A pairs. However, the original Q-A pairs only matched with just one single fact. In fact, we found that a lot of questions need more than one fact (about 20% based on sampling inspection). Therefore, we crawl more Q-A pairs from Chinese community QA website (Baidu Zhidao[7]). Combined with the originally published corpus, we create a lager and better-quality data for natural question answering. Specifically, an Integral Linear Programming (ILP) based method is employed to automatically construct "grounding" Q-A pairs with the facts in KB (inspired by the work of adopting ILP to parse questions (Yahya et al., 2012)). In ILP, the main constraints and considered factors are listed below: 1) the "`subject`" entity and "`object`" entity of a triple have to match with question words/phrases (marked as *subject mention*) and answer words/phrases (marked as *object mention*) respectively; 2) any two *subject mentions* or *object mentions* should not overlap; 3) a *mention* can match at most one `entity`; 4) the edit distance between the Q-A pair and the matched candidate fact (use a space to joint three parts) is smaller, they are more relevant. Finally, we totally obtain 619,199 instances (an instance contains a question, an answer, and multiple facts), and the number of instances that can match one and multiple facts in KB are 499,809 and 119,390, respectively. Through the evaluation of 200 sampling instances, we estimate that approximate 81% matched facts are helpful for the generating answers. However, strictly speaking, only 44% instances are truly correct grounding. In fact, grounding the Q-A pairs from community QA website is a very challenge problem, we will leave it in the future work.

Experimental Setting: The dataset is split into training (90%) and testing set (10%). The sen-

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

[7]https://zhidao.baidu.com/

tences in Chinese are segmented into word sequences with Jieba[8] tool. And we use the words with the frequency larger than 3, which covering 98.4% of the word in the corpus. For a fair comparison, we use bi-directional LSTM for the encoder and another LSTM for decoder for all Seq2Seq models, with hidden layer size = 1024 and word embedding dimension = 300. We select CopyNet (more advanced Seq2Seq model) and GenQA for comparison. We set L_F as 10.

Metrics: Besides adopting the **AE** as a metric (same as GenQA (Yin et al., 2016)), we additionally use **manual evaluation (ME)** as another metric. **ME** considers three aspects about the quality of the generated answer (refer to (Asghar et al., 2016)): 1) **correctness**; 2) syntactical **fluency**; 3) **coherence** with the question. We employ two annotators to rate such three aspects of CopyNet, GenQA and COREQA. Specifically, we sample 100 questions, and conduct $C_3^2 = 3$ pair-wise comparisons for each question and count the winning times of each model (comparisons may both win or both lose).

Experimental Results: The **AE** and **ME** results are shown in Table 4 and Table 5, respectively. Meanwhile, we separately present the results according to the number of the facts which a question needs in KB, including just one single fact (marked as **Single**), multiple facts (marked as **Multi**) and all (marked as **Mixed**). In fact, we train two separate models for **Single** and **Multi** questions for the unbalanced data . From Table 4 and Table 5, we can clearly observe that COREQA significantly outperforms all other baseline models. And COREQA could generate a better natural answer in three aspects: correctness, fluency and coherence. CopyNet cannot interact with KB which is important to generate correct answers. For example, for "Who is the director of The_Little_Chinese_Seamstress?", if without the fact (The_Little_Chinese_Seamstress, director, Dai_Siji), QA systems cannot generate a correct answer.

Models	Single	Multi	Mixed
CopyNet	9.7	0.8	8.7
GenQA	47.2	28.9	45.1
COREQA	**58.4**	**42.7**	**56.6**

Table 4: The AE accuracies (%) on real world test data.

[8]https://github.com/fxsjy/jieba

Models	Correctness	Fluency	Coherence
CopyNet	0	13.3	3.3
GenQA	26.7	33.3	20
COREQA	**46.7**	**50**	**60**

Table 5: The ME results (%) on sampled mixed test data.

Case Study and Error Analysis: Table 6 gives some examples of generated by COREQA and the gold answers to the questions in test set. It is very clearly seen that the parts of generating SUs are predicted from the vocabulary, and other SUs are copied from the given question (marked as **bold**) and retrieved from the KB (marked as underline). And we analyze sampled examples and believe that there are several major causes of errors: **1)** did not match the right facts (ID 6); **2)** the generated answers contain some repetition of meaningless words (ID 7); **3)** the generated answers are not coherence natural language sentences (ID 8).

5 Related Work

Seq2Seq learning is to maximize the likelihood of predicting the target sequence Y conditioned on the observed source sequence X (Sutskever et al., 2014), which has been applied successfully to a large number of NLP tasks such as Machine Translation (Wu et al., 2016) and Dialogue (Vinyals and Le, 2015). Our work is partially inspired by the recent work of QA and Dialogue which have adopted Seq2Seq learning. CopyNet (Gu et al., 2016) and Pointer Networks (Vinyals et al., 2015; Gulcehre et al., 2016) which could incorporate copying mechanism in conventional Seq2Seq learning. Different from our application which deals with knowledge inquired questions and generates natural answers, CopyNet (Gu et al., 2016) and Pointer Networks (Gulcehre et al., 2016) can only copy words from the original input sequence. In contrast, COREQA is able to retrieve SUs from external memory. And GenQA (Yin et al., 2016) can only deal with the simple questions which could be answered by one fact, and it also did not incorporate the copying mechanism in Seq2Seq learning.

Moreover, our work is also inspired by Neural Abstract Machine (Graves et al., 2016; Yin et al., 2015; Liang et al., 2016) which could retrieve facts from KBs with neural models. Unlike natural answer, Neural Abstract Machine (Mou et al., 2016) is concentrating on obtaining concrete answer en-

ID	Question	Gold Answer	Generated Natural Answer
1	拜仁的哈格里夫斯是哪国人？ Which country did *Hargreaves* of Bayern comes from?	英国人！加拿大出生 British! born in Canada	加拿大人，英国人 Canadians, British
2	沙祖康是哪个族的？ Which ethnic groups is *Sha Zukang* from?	汉族现在是联合国副秘书长 Han, now he is the DSG of the UN	联合国副秘书长，是汉族 DSG of the UN, Han
3	罗伯特·舒曼是谁？ Who is *Robert Schumann*?	是一个作家、音乐家 a writer, musician	**罗伯特·舒曼**个作家 哲学家 **Robert Schuhmann** is a writer and philosopher.
4	马斯切拉诺是谁？ Who is *Mascherano*?	他是一名足球运动员 He is a football player	一个优秀的足球 运动员 An excellent football player
5	巴尔扎克与小裁缝的导演是谁？ Who is the director of *The Little Chinese Seamstress*?	戴思杰 Dai Sijie	**导演**是戴思杰 **Director** Dai Sijie
6	电影铁三角是谁拍的？ Who shot the movie *The Iron Triangle*?	徐克、林岭东、杜琪峰 Tsui Hark, Johnny To, Ringo Lam	冯小刚 Feng Xiaogang
7	谁知道席慕蓉这个人的一些详细资料？ Who knows some details of *Xi Murong*?	女诗人 poetess	是是是 yes, yes, yes
8	泡泡堂是哪个公司开发的？ Which company developed the game *Crazy Arcade*?	应该是上海盛大 should be the Shanda Group	玩上海盛大 playing Shanda Group

Table 6: Examples of the generated natural answers by COREQA.

tities with neural network based reasoning.

6 Conclusion and Future Work

In this paper, we propose an end-to-end system to generate natural answers through incorporating copying and retrieving mechanisms in sequence-to-sequence learning. Specifically, the sequences of SUs in the generated answer may be predicted from the vocabulary, copied from the given question and retrieved from the corresponding KB. And the future work includes: a) lots of questions cannot be answered directly by facts in a KB (e.g. "*Who is Jet Li's father-in-law?*"), we plan to learn QA system with latent knowledge (e.g. KB embedding (Bordes et al., 2013)); b) we plan to adopt memory networks (Sukhbaatar et al., 2015) to encode the temporary KB for each question.

Acknowledgments

The authors are grateful to anonymous reviewers for their constructive comments. The work was supported by the Natural Science Foundation of China (No.61533018) and the National High Technology Development 863 Program of China (No.2015AA015405).

References

Nabiha Asghar, Pascal Poupart, Jiang Xin, and Hang Li. 2016. Online sequence-to-sequence reinforcement learning for open-domain conversational agents. *arXiv preprint arXiv:1612.03929* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. pages 2787–2795.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

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

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine* 31(3):59–79.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538(7626):471–476.

Jiatao Gu, Zhengdong Lu, Hang Li, and O.K. Victor Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1631–1640.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 140–149.

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

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

Cody Kwok, Oren Etzioni, and Daniel S Weld. 2001. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS)* 19(3):242–262.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1192–1202.

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

Vanessa Lopez, Victoria Uren, Marta Sabou, and Enrico Motta. 2011. Is question answering fit for the semantic web?: a survey. *Semantic Web* 2(2):125–155.

Michael McTear, Zoraida Callejas, and David Griol. 2016. *The Conversational Interface: Talking to Smart Devices*. Springer Publishing Company, Incorporated, 1st edition.

Lili Mou, Zhengdong Lu, Hang Li, and Zhi Jin. 2016. Coupling distributed and symbolic execution for natural language queries. *arXiv preprint arXiv:1612.02741* .

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Coverage-based neural machine translation. *arXiv preprint arXiv:1601.04811* .

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

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

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR* abs/1410.3916.

William A Woods. 1977. Lunar rocks in natural english: Explorations in natural language question answering. In *Linguistic structures processing*. pages 521–569.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 379–390.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 956–966.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1321–1331.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

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