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

Shizhu He\textsuperscript{1}, Cao Liu\textsuperscript{1,2}, Kang Liu\textsuperscript{1} and Jun Zhao\textsuperscript{1,2}
\textsuperscript{1} National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
\textsuperscript{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\textsuperscript{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.”\textsuperscript{2}, the system could finally generate the natural answer (McTear et al., 2016).

In order to generate natural answers, typical

\textsuperscript{1}http://www.apple.com/ios/siri/
\textsuperscript{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 COpying and REtrrieving 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 copy 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 $X = [x_1, ..., x_L]$ into an encoded representation $e$. For example, we can utilize the basic model: $h_t = f(x_t, h_{t-1}); c = \phi(h_1, ..., h_{L_X})$, where $\{h_t\}$ are the RNN hidden states, $c$ 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
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 \( Y = [y_1, \ldots, y_{L_Y}] \), through the following prediction model:

\[
    s_t = f(y_{t-1}, s_{t-1}, c); \quad p(y_t | y < t, X) = g(y_{t-1}, s_t, c),
\]

where \( s_t \) 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 \( y_t \) share the same context vector \( c \). 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 \( c_t \) at each time step (Bahdanau et al., 2014), for example, representing \( c_t \) as the weighted sum of the source states \( \{h_i\} \),

\[
c_t = \sum_{i=1}^{L_X} \alpha_{ti} h_i; \quad \alpha_{ti} = \frac{e^{\rho(s_{t-1}, h_i)}}{\sum_{i'} e^{\rho(s_{t-1}, h_{i'})}} \quad (1)
\]

where the function \( \rho \) 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-

---

**Figure 2:** The overall diagram of COREQA.
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 \( \{ \tilde{h}_1, \ldots, \tilde{h}_L \} \) and \( \{ \tilde{h}_L, \ldots, \tilde{h}_1 \} \). The concatenated representation is considered to be the short-term memory of question \( (M_Q = \{ h_t \}) \), \( h_t = [ \tilde{h}_t, \tilde{h}_{L_X-t+1}] \). \( q = [ \tilde{h}_{L_X}, \tilde{h}_1] \) 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 \( e_s, e_p \) and \( e_o \) to denote its corresponding embeddings. The fact representation \( f \) is then defined as the concatenation of \( e_s, e_p \) and \( e_o \). The list of all related facts’ representations, \( \{ f \} = \{ f_1, \ldots, f_{L_F} \} \) (refer to \( M_{KB}, L_F \) 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 \( S(q, f_j) = DNN_1(q, f_j) = tanh(W_2 \cdot tanh(W_1[q, f_j] + b_1) + b_2) \), where \( DNN_1 \) is the matching function defined by a two-layer perceptron, \( [\cdot, \cdot] \) denotes vector concatenation, and \( W_1, W_2, b_1 \) and \( b_2 \) 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 \( S(q, s_t, f_j) = DNN_1(q, s_t, f_j) \) 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 \( M_Q \) and \( M_{KB} \), 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 \( s_t \), but COREQA uses not only its word embedding but also its corresponding positional attention informations in \( M_Q \) and \( M_{KB} \);

**Reading short-Memory** \( M_Q \) and \( M_{KB} \): \( M_Q \) and \( M_{KB} \) 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 $V = \{v_1, \ldots, v_N\} \cup \{\text{UNK}\}$, where UNK indicates any out-of-vocabulary (OOV) words. Therefore, we have adopted another two set of SUs $X_Q$ and $X_{KB}$ which cover words/entities in the source question and the partial KB. That is, we have adopted the instance-specific vocabulary $V \cup X_Q \cup X_{KB}$ for each question. It’s important to note that these three vocabularies $V$, $X_Q$ and $X_{KB}$ may overlap.

At each time step $t$ in the decoding process, given the RNN state $s_t$ together with $M_Q$ and $M_{KB}$, the probabilistic function for generating any target SU $y_t$ is a “mixture” model as follow

$$P(y_t|s_t, y_{t-1}, M_Q, M_{KB}) =$$

$$p_{pr}(y_t|s_t, y_{t-1}, c_t) \cdot p_{m}(p_{pr}|s_t, y_{t-1}) +$$

$$p_{co}(y_t|s_t, y_{t-1}, M_Q) \cdot p_{m}(p_{co}|s_t, y_{t-1}) +$$

$$p_{re}(y_t|s_t, y_{t-1}, M_{KB}) \cdot p_{m}(p_{re}|s_t, y_{t-1})$$

(2)

where $p_{pr}$, $p_{co}$ and $p_{re}$ stand for the predict-mode, the copy-mode and the retrieve-mode, respectively. $p_{m}(|\cdot|)$ 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

$$p_{pr}(y_t|\cdot) = \frac{1}{Z} \exp(\psi_{pr}(y_t))$$

$$p_{co}(y_t|\cdot) = \frac{1}{Z} \sum_j:Q_j=y_t \exp(\psi_{co}(y_t))$$

$$p_{re}(y_t|\cdot) = \frac{1}{Z} \sum_j:KB_j=y_t \exp(\psi_{re}(y_t))$$

(3)

where $\psi_{pr}(|\cdot|)$, $\psi_{co}(|\cdot|)$ and $\psi_{re}(|\cdot|)$ are score functions for choosing SUs in predict-mode (from $V$), copy-mode (from $X_Q$) and retrieve-mode (from $X_{KB}$), respectively. And $Z$ is the normalization term shared by the three modes, $Z = \exp(\psi_{pr}(|\cdot|)) + \sum_j:Q_j=y_t \exp(\psi_{co}(|\cdot|)) + \sum_j:KB_j=y_t \exp(\psi_{re}(|\cdot|))$.

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:

**Predict-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 $\psi_{pr}(y_t = v_i) = v_i^T W_{pr}[s_t, c_{q_i}, c_{kb}]$, where $v_i \in \mathbb{R}^d$ is the word vector at the output layer (not the input word embedding), $W_{pr} \in \mathbb{R}^{(d_i+d_f+d_f) \times d_r}$, and $d_i, d_f$ and $d_r$ indicate the size of input word vector, RNN decoder hidden state and fact representation respectively, and $c_{q_i}$ and $c_{kb}$ are the temporary memory of reading $M_Q$ and $M_{KB}$ at time $t$ (see Section 3.4.3).

**Copy-mode**: The score for “copying” the word $x_j$ from question Q is calculated as $\psi_{co}(y_t = x_j) = DN_{N2}(h_j, s_t, hist_Q)$, where $DN_{N2}$ is a neural network function with a two-layer MLP and $hist_Q \in \mathbb{R}^{L \times x}$ is an accumulated vector which record the attentive history for each word in question (similar with the coverage vector in (Tu et al., 2016)).

**Retrieve-mode**: The score for “retrieving” the entity word $v_j$ from retrieval facts (“Object” part) is calculated as $\psi_{re}(y_t = v_j) = DN_{N3}(f_j, s_t, hist_{KB})$, where $DN_{N3}$ is also a neural network function and $hist_{KB} \in \mathbb{R}^{L \times f}$ 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 $s_t$ is updated with the previous state $s_{t-1}$, the word embedding of previous predicted symbol $y_{t-1}$, and an optional context vector $c_t$ (with attention mechanism). However, $y_{t-1}$ may not come from vocabulary $V$ and not own a word vector. Therefore, we modify the state update process in COREQA. More specifically, $y_{t-1}$ will be represented as concatenated vector of $[e(y_{t-1}), r_{q_{t-1}}, r_{kb_{t-1}}]$, where $e(y_{t-1})$ is the word embedding associated with $y_{t-1}$, $r_{q_{t-1}}$ and $r_{kb_{t-1}}$ are the weighted sum of hidden states in $M_Q$ and $M_{KB}$ corresponding to $y_{t-1}$ respectively.

$$r_{q_i} = \sum_{j=1}^{L_X} \rho_{tj} h_j, r_{kb_t} = \sum_{j=1}^{L_F} \delta_{tj} f_j$$

$$\rho_{tj} = \begin{cases} \frac{1}{K_1} p_{co}(x_j|\cdot), & x_j = y_t \\ 0 & otherwise \end{cases}$$

$$\delta_{tj} = \begin{cases} \frac{1}{K_2} p_{re}(f_j|\cdot), & \text{object}(f_j) = y_t \\ 0 & otherwise \end{cases}$$

(4)

where $\text{object}(f)$ indicate the “object” part of fact $f$ (see Figure 2), and $K_1$ and $K_2$ are the normalization terms which equal $\sum_j: x_j = y_t \exp(\psi_{co}(|x_j|))$ and $\sum_j: \text{object}(f_j) = y_t \exp(\psi_{re}(|f_j|))$, respectively, and it
could consider the multiple positions matching $y_t$ in source question and KB.

3.4.3 Reading short-Memory $M_Q$ and $M_{KB}$

COREQA employ the attention mechanism at decoding process. At each decoder time $t$, we selective read the context vector $c_{q_t}$ and $c_{kb_t}$ from the short-term memory of question $M_Q$ and retrieval facts $M_{KB}$ (alike to Formula 1). In addition, the accumulated attentive vectors $\text{hist}_Q$ and $\text{hist}_{KB}$ 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:

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

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 %dth. 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).

<table>
<thead>
<tr>
<th>Q-A Patterns</th>
<th>Examples (e.g. KB facts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>When is %e birthday?</td>
<td>When is %e birthday?</td>
</tr>
<tr>
<td>He was born in %m %dth.</td>
<td>He was born in June 20th.</td>
</tr>
<tr>
<td>What year were %e born?</td>
<td>What year were %e born?</td>
</tr>
<tr>
<td>%e is born in %y year.</td>
<td>%e is born in 1987 year.</td>
</tr>
</tbody>
</table>

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
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.

### 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 $P_g$, $P_y$, $P_m$ 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”.

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

<table>
<thead>
<tr>
<th>Models</th>
<th>$P_g$</th>
<th>$P_y$</th>
<th>$P_m$</th>
<th>$P_d$</th>
<th>$P_A$</th>
<th>$R_A$</th>
<th>$F1_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>72.2</td>
<td>0</td>
<td>1.1</td>
<td>0.2</td>
<td>0</td>
<td>27.5</td>
<td>0</td>
</tr>
<tr>
<td>RNN+atten</td>
<td>55.8</td>
<td>1.1</td>
<td>11.3</td>
<td>9.5</td>
<td>1.7</td>
<td>34</td>
<td>3.2</td>
</tr>
<tr>
<td>CopyNet</td>
<td>75.2</td>
<td>8.7</td>
<td>28.3</td>
<td>5.8</td>
<td>3.7</td>
<td>32.5</td>
<td>17.3</td>
</tr>
<tr>
<td>GenQA</td>
<td>73.4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27.1</td>
<td>0</td>
</tr>
<tr>
<td>COREQA</td>
<td>100</td>
<td>84.8</td>
<td>93.4</td>
<td>81</td>
<td>87.4</td>
<td>94</td>
<td>90.6</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Entities</th>
<th>$P_g$</th>
<th>$P_y$</th>
<th>$P_m$</th>
<th>$P_d$</th>
<th>$P_A$</th>
<th>$R_A$</th>
<th>$F1_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seen</td>
<td>100</td>
<td>84.8</td>
<td>93.4</td>
<td>81</td>
<td>87.4</td>
<td>94</td>
<td>90.6</td>
</tr>
<tr>
<td>Unseen</td>
<td>75.1</td>
<td>84.5</td>
<td>93.5</td>
<td>81.2</td>
<td>63.8</td>
<td>85.1</td>
<td>73.1</td>
</tr>
</tbody>
</table>

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, 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 websites (Baidu Zhi-dao). Combined with the originally published corpus, we create a larger 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 question- s (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-

---

5 The “gender” is right when the entity name (e.g. ‘e2’) or the personal pronoun (e.g. ‘She’) in answer is correct.

6 https://github.com/sjfeb/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.

<table>
<thead>
<tr>
<th>Models</th>
<th>Single</th>
<th>Multi</th>
<th>Mixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyNet</td>
<td>9.7</td>
<td>0.8</td>
<td>8.7</td>
</tr>
<tr>
<td>GenQA</td>
<td>47.2</td>
<td>28.9</td>
<td>45.1</td>
</tr>
<tr>
<td>COREQA</td>
<td>58.4</td>
<td>42.7</td>
<td>56.0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Models</th>
<th>Correctness</th>
<th>Fluency</th>
<th>Coherence</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyNet</td>
<td>0</td>
<td>13.3</td>
<td>3.3</td>
</tr>
<tr>
<td>GenQA</td>
<td>26.7</td>
<td>33.3</td>
<td>20.0</td>
</tr>
<tr>
<td>COREQA</td>
<td>46.7</td>
<td>50.0</td>
<td>60.0</td>
</tr>
</tbody>
</table>

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-
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).

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

<table>
<thead>
<tr>
<th>ID</th>
<th>Question</th>
<th>Gold Answer</th>
<th>Generated Natural Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Which country did Hargreaves of Bayern comes from?</td>
<td>British! born in Canada</td>
<td>Hargreaves of Bayern is a British football player.</td>
</tr>
<tr>
<td>2</td>
<td>Which ethnic groups is Shu Zalong from?</td>
<td>Han, now he is the DSG of the UN</td>
<td>Shu Zalong is a Han Chinese football player, DSG of the UN</td>
</tr>
<tr>
<td>3</td>
<td>Who is Robert Schumann named?</td>
<td>an outstanding Hungarian musician</td>
<td>Robert Schumann is a writer and philosopher.</td>
</tr>
<tr>
<td>4</td>
<td>Who is Mascherano?</td>
<td>He is a football player</td>
<td>An excellent football player</td>
</tr>
<tr>
<td>5</td>
<td>Who is the director of The Little Chinese Seamstress?</td>
<td>Dai Sui</td>
<td>Director Dai Sui</td>
</tr>
<tr>
<td>6</td>
<td>Who shot the movie The Iron Triangle?</td>
<td>Tsui Hark, Johnny To, Ringo Lam</td>
<td>Feng Xiaogang</td>
</tr>
<tr>
<td>7</td>
<td>Who knows some details of Xi Muxing?</td>
<td>poetess</td>
<td>is the poet</td>
</tr>
<tr>
<td>8</td>
<td>Which company developed the game Crazy Arcade?</td>
<td>should be the Shanda Group</td>
<td>playing Shanda Group</td>
</tr>
</tbody>
</table>

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

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


