Harvesting Paragraph-Level Question-Answer Pairs from Wikipedia

Xinya Du and **Claire Cardie**
Department of Computer Science
Cornell University
Ithaca, NY, 14853, USA
{xdu, cardie}@cs.cornell.edu

Abstract

We study the task of generating from Wikipedia articles question-answer pairs that cover content beyond a single sentence. We propose a neural network approach that incorporates coreference knowledge via a novel gating mechanism. Compared to models that only take into account sentence-level information (Heilman and Smith, 2010; Du et al., 2017; Zhou et al., 2017), we find that the linguistic knowledge introduced by the coreference representation aids question generation significantly, producing models that outperform the current state-of-the-art. We apply our system (composed of an answer span extraction system and the passage-level QG system) to the 10,000 top-ranking Wikipedia articles and create a corpus of over one million question-answer pairs. We also provide a qualitative analysis for this large-scale generated corpus from Wikipedia.

1 Introduction

Recently, there has been a resurgence of work in NLP on reading comprehension (Hermann et al., 2015; Rajpurkar et al., 2016; Joshi et al., 2017) with the goal of developing systems that can answer questions about the content of a given passage or document. Large-scale QA datasets are indispensable for training expressive statistical models for this task and play a critical role in advancing the field. And there have been a number of efforts in this direction. Miller et al. (2016), for example, develop a dataset for open-domain question answering; Rajpurkar et al. (2016) and Joshi et al. (2017) do so for reading comprehension (RC); and Hill et al. (2015) and Hermann et al. (2015), for the related task of answering cloze questions (Winograd, 1972; Levesque et al., 2011). To create these datasets, either crowdsourcing or *(semi-)synthetic* approaches are used. The (semi-)synthetic datasets (e.g., Hermann et al. (2015)) are large in size and cheap to obtain; however, they do not share the same characteristics as explicit QA/RC questions (Rajpurkar et al., 2016). In comparison, high-quality *crowdsourced* datasets are much smaller in size, and the annotation process is quite expensive because the labeled examples require expertise and careful design (Chen et al., 2016).

Paragraph:
(1) *Tesla* was renowned for *his* achievements and showmanship, eventually earning *him* a reputation in popular culture as an archetypal "mad scientist". (2) *His* patents earned *him* a considerable amount of money, much of which was used to finance *his* own projects with varying degrees of success. (3) *He* lived most of his life in a series of New York hotels, through *his* retirement. (4) *Tesla* died on 7 January 1943. ...

Questions:
– What was Tesla's reputation in popular culture?
mad scientist
– How did Tesla finance his work?
patents
– Where did Tesla live for much of his life?
New York hotels

Figure 1: Example input from the fourth paragraph of a Wikipedia article on *Nikola Tesla*, along with the natural questions and their answers from the SQuAD (Rajpurkar et al., 2016) dataset. We show in italics the set of mentions that refer to Nikola Tesla — *Tesla*, *him*, *his*, *he*, etc.

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 1907–1917
Melbourne, Australia, July 15 - 20, 2018.

Thus, there is a need for methods that can automatically generate high-quality question-answer pairs. Serban et al. (2016) propose the use of recurrent neural networks to generate QA pairs from structured knowledge resources such as Freebase. Their work relies on the existence of automatically acquired KBs, which are known to have errors and suffer from incompleteness. They are also non-trivial to obtain. In addition, the questions in the resulting dataset are limited to queries regarding a single fact (i.e., tuple) in the KB.

Motivated by the need for large scale QA pairs and the limitations of recent work, we investigate methods that can automatically "harvest" (generate) question-answer pairs from raw text/unstructured documents, such as Wikipedia-type articles.

Recent work along these lines (Du et al., 2017; Zhou et al., 2017) (see Section 2) has proposed the use of attention-based recurrent neural models trained on the crowdsourced SQuAD dataset (Rajpurkar et al., 2016) for question generation. While successful, the resulting QA pairs are based on information from a single sentence. As described in Du et al. (2017), however, nearly 30% of the questions in the human-generated questions of SQuAD rely on information beyond a single sentence. For example, in Figure 1, the second and third questions require coreference information (i.e., recognizing that "His" in sentence 2 and "He" in sentence 3 both corefer with "Tesla" in sentence 1) to answer them.

Thus, our research studies methods for incorporating coreference information into the training of a question generation system. In particular, we propose gated **Coref**erence knowledge for **N**eural **Q**uestion **G**eneration (**CorefNQG**), a neural sequence model with a novel gating mechanism that leverages continuous representations of *coreference clusters* — the set of mentions used to refer to each entity — to better encode linguistic knowledge introduced by coreference, for paragraph-level question generation.

In an evaluation using the SQuAD dataset, we find that CorefNQG enables better question generation. It outperforms significantly the baseline neural sequence models that encode information from a single sentence, and a model that encodes *all* preceding context and the input sentence itself. When evaluated on only the portion of SQuAD that requires coreference resolution, the gap between our system and the baseline systems is even larger.

By applying our approach to the 10,000 top-ranking Wikipedia articles, we obtain a question answering/reading comprehension dataset with over one million QA pairs; we provide a qualitative analysis in Section 6. The dataset and the source code for the system are available at `https://github.com/xinyadu/HarvestingQA`.

2 Related Work

2.1 Question Generation

Since the work by Rus et al. (2010), question generation (QG) has attracted interest from both the NLP and NLG communities. Most early work in QG employed rule-based approaches to transform input text into questions, usually requiring the application of a sequence of well-designed general rules or templates (Mitkov and Ha, 2003; Labutov et al., 2015). Heilman and Smith (2010) introduced an overgenerate-and-rank approach: their system generates a set of questions and then ranks them to select the top candidates. Apart from generating questions from raw text, there has also been research on question generation from symbolic representations (Yao et al., 2012; Olney et al., 2012).

With the recent development of deep representation learning and large QA datasets, there has been research on recurrent neural network based approaches for question generation. Serban et al. (2016) used the encoder-decoder framework to generate QA pairs from knowledge base triples; Reddy et al. (2017) generated questions from a knowledge graph; Du et al. (2017) studied how to generate questions from sentences using an attention-based sequence-to-sequence model and investigated the effect of exploiting sentence- vs. paragraph-level information. Du and Cardie (2017) proposed a hierarchical neural sentence-level sequence tagging model for identifying question-worthy sentences in a text passage. Finally, Duan et al. (2017) investigated how to use question generation to help improve question answering systems on the sentence selection subtask.

In comparison to the related methods from above that generate questions from raw text, our method is different in its ability to take into account contextual information beyond the sentence-level by introducing coreference knowledge.

2.2 Question Answering Datasets and Creation

Recently there has been an increasing interest in question answering with the creation of many datasets. Most are built using crowdsourcing; they are generally comprised of fewer than 100,000 QA pairs and are time-consuming to create. WebQuestions (Berant et al., 2013), for example, contains 5,810 questions crawled via the Google Suggest API and is designed for knowledge base QA with answers restricted to Freebase entities. To tackle the size issues associated with WebQuestions, Bordes et al. (2015) introduce SimpleQuestions, a dataset of 108,442 questions authored by English speakers. SQuAD (Rajpurkar et al., 2016) is a dataset for machine comprehension; it is created by showing a Wikipedia paragraph to human annotators and asking them to write questions based on the paragraph. TriviaQA (Joshi et al., 2017) includes 95k question-answer authored by trivia enthusiasts and corresponding evidence documents.

(Semi-)synthetic generated datasets are easier to build to large-scale (Hill et al., 2015; Hermann et al., 2015). They usually come in the form of cloze-style questions. For example, Hermann et al. (2015) created over a million examples by pairing CNN and Daily Mail news articles with their summarized bullet points. Chen et al. (2016) showed that this dataset is quite noisy due to the method of data creation and concluded that performance of QA systems on the dataset is almost saturated.

Closest to our work is that of Serban et al. (2016). They train a neural triple-to-sequence model on SimpleQuestions, and apply their system to Freebase to produce a large collection of human-like question-answer pairs.

3 Task Definition

Our goal is to harvest high quality question-answer pairs from the paragraphs of an article of interest. In our task formulation, this consists of two steps: **candidate answer extraction** and **answer-specific question generation**. Given an input paragraph, we first identify a set of *question-worthy* candidate answers $ans = (ans_1, ans_2, ..., ans_l)$, each a span of text as denoted in color in Figure 1. For each candidate answer ans_i, we then aim to generate a question Q — a sequence of tokens $y_1, ..., y_N$ — based on the sentence S that contains candidate ans_i such that:

- Q asks about an aspect of ans_i that is of potential interest to a human;
- Q might rely on information from sentences that precede S in the paragraph.

Mathematically then,

$$Q = \arg\max_{Q} P\left(Q|S, C\right) \quad (1)$$

where $P(Q|S,C) = \prod_{n=1}^{N} P\left(y_n|y_{<n}, S, C\right)$ where C is the set of sentences that precede S in the paragraph.

4 Methodology

In this section, we introduce our framework for harvesting the question-answer pairs. As described above, it consists of the question generator CorefNQG (Figure 2) and a candidate answer extraction module. During test/generation time, we (1) run the answer extraction module on the input text to obtain answers, and then (2) run the question generation module to obtain the corresponding questions.

4.1 Question Generation

As shown in Figure 2, our generator prepares the feature-rich input embedding — a concatenation of (a) a refined coreference position feature embedding, (b) an answer feature embedding, and (c) a word embedding, each of which is described below. It then encodes the textual input using an LSTM unit (Hochreiter and Schmidhuber, 1997). Finally, an attention-copy equipped decoder is used to decode the question.

More specifically, given the input sentence S (containing an answer span) and the preceding context C, we first run a coreference resolution system to get the coref-clusters for S and C and use them to create a *coreference transformed* input sentence: for each pronoun, we append its most representative non-pronominal coreferent mention. Specifically, we apply the simple feed-forward network based mention-ranking model of Clark and Manning (2016) to the concatenation of C and S to get the coref-clusters for all entities in C and S. The C&M model produces a score/representation s for each mention pair (m_1, m_2),

$$s(m_1, m_2) = \mathbf{W}_m h_m(m_1, m_2) + b_m \quad (2)$$

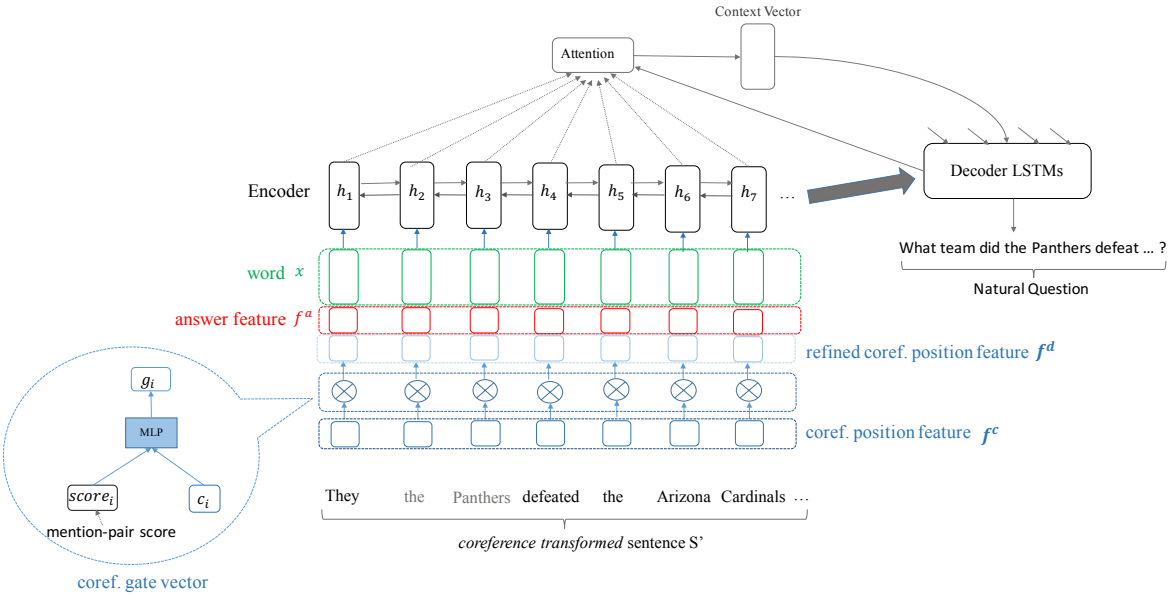

Figure 2: The gated **Coref**erence knowledge for **N**eural **Q**uestion **G**eneration (**CorefNQG**) Model.

word	they	the	panthers	defeated	the	arizona	cardinals	49	–	15	...
ans. feature	O	O	O	O	B_ANS	I_ANS	I_ANS	O	O	O	...
coref. feature	B_PRO	B_ANT	I_ANT	O	O	O	O	O	O	O	...

Table 1: Example input sentence with coreference and answer position features. The corresponding gold question is "What team did the Panthers defeat in the NFC championship game ?"

where $\mathbf{W}_m$ is a $1 \times d$ weight matrix and b is the bias. $h_m(m_1, m_2)$ is representation of the last hidden layer of the three layer feedforward neural network.

For each pronoun in S, we then heuristically identify the most "representative" antecedent from its coref-cluster. (Proper nouns are preferred.) We append the new mention after the pronoun. For example, in Table 1, "the panthers" is the most representative mention in the coref-cluster for "they". The new sentence with the appended coreferent mention is our *coreference transformed* input sentence S' (see Figure 2).

Coreference Position Feature Embedding For each token in S', we also maintain one position feature $\mathbf{f^c} = (c_1, ..., c_n)$, to denote pronouns (e.g., "they") and antecedents (e.g., "the panthers"). We use the BIO tagging scheme to label the associated spans in S'. "B_ANT" denotes the start of an antecedent span, tag "I_ANT" continues the antecedent span and tag "O" marks tokens that do not form part of a mention span. Similarly, tags "B_PRO" and "I_PRO" denote the pronoun span. (See Table 1, "coref. feature".)

Refined Coref. Position Feature Embedding Inspired by the success of gating mechanisms for controlling information flow in neural networks (Hochreiter and Schmidhuber, 1997; Dauphin et al., 2017), we propose to use a gating network here to obtain a refined representation of the coreference position feature vectors $\mathbf{f^c} = (c_1, ..., c_n)$. The main idea is to utilize the mention-pair score (see Equation 2) to help the neural network learn the importance of the coreferent phrases. We compute the refined (gated) coreference position feature vector $\mathbf{f^d} = (d_1, ..., d_n)$ as follows,

$$g_i = \text{ReLU}(\mathbf{W}_a c_i + \mathbf{W}_b score_i + b)$$
$$d_i = g_i \odot c_i \quad (3)$$

where $\odot$ denotes an element-wise product between two vectors and ReLU is the rectified linear activation function. $score_i$ denotes the mention-pair score for each antecedent token (e.g., "the" and "panthers") with the pronoun (e.g., "they"); $score_i$ is obtained from the trained model (Equation 2) of the C&M. If token i is not added later as an antecedent token, $score_i$ is set to zero. $\mathbf{W}_a$, $\mathbf{W}_b$ are weight matrices and b is the bias vector.

Answer Feature Embedding We also include an answer position feature embedding to generate answer-specific questions; we denote the answer span with the usual BIO tagging scheme (see,

e.g., "the arizona cardinals" in Table 1). During training and testing, the answer span feature (i.e., "B_ANS", "I_ANS" or "O") is mapped to its feature embedding space: $\mathbf{f^a} = (a_1, ..., a_n)$.

Word Embedding To obtain the word embedding for the tokens themselves, we just map the tokens to the word embedding space: $\mathbf{x} = (x_1, ..., x_n)$.

Final Encoder Input As noted above, the final input to the LSTM-based encoder is a concatenation of (1) the refined coreference position feature embedding (light blue units in Figure 2), (2) the answer position feature embedding (red units), and (3) the word embedding for the token (green units),

$$e_i = \text{concat}(d_i, a_i, x_i) \tag{4}$$

Encoder As for the encoder itself, we use bidirectional LSTMs to read the input $\mathbf{e} = (e_1, ..., e_n)$ in both the forward and backward directions. After encoding, we obtain two sequences of hidden vectors, namely, $\overrightarrow{\mathbf{h}} = (\overrightarrow{h_1}, ..., \overrightarrow{h_n})$ and $\overleftarrow{\mathbf{h}} = (\overleftarrow{h_1}, ..., \overleftarrow{h_n})$. The final output state of the encoder is the concatenation of $\overrightarrow{\mathbf{h}}$ and $\overleftarrow{\mathbf{h}}$ where

$$h_i = \text{concat}(\overrightarrow{h_i}, \overleftarrow{h_i}) \tag{5}$$

Question Decoder with Attention & Copy On top of the feature-rich encoder, we use LSTMs with attention (Bahdanau et al., 2015) as the decoder for generating the question $y_1, ..., y_m$ one token at a time. To deal with rare/unknown words, the decoder also allows directly copying words from the source sentence via pointing (Vinyals et al., 2015).

At each time step t, the decoder LSTM reads the previous word embedding w_{t-1} and previous hidden state s_{t-1} to compute the new hidden state,

$$s_t = \text{LSTM}(w_{t-1}, s_{t-1}) \tag{6}$$

Then we calculate the *attention distribution* α_t as in Bahdanau et al. (2015),

$$\begin{aligned} e_{t,i} &= h_i^T \mathbf{W}_c s_{t-1} \\ \alpha_t &= \text{softmax}(e_t) \end{aligned} \tag{7}$$

where $\mathbf{W}_c$ is a weight matrix and attention distribution α_t is a probability distribution over the source sentence words. With α_t, we can obtain the context vector h_t^*,

$$h_t^* = \sum_{i=1}^{n} \alpha_t^i h_i \tag{8}$$

Then, using the context vector h_t^* and hidden state s_t, the probability distribution over the target (question) side vocabulary is calculated as,

$$P_{vocab} = \text{softmax}(\mathbf{W}_d \text{concat}(h_t^*, s_t)) \tag{9}$$

Instead of directly using P_{vocab} for training/generating with the fixed target side vocabulary, we also consider *copying* from the source sentence. The copy probability is based on the context vector h_t^* and hidden state s_t,

$$\lambda_t^{copy} = \sigma\left(\mathbf{W}_e h_t^* + \mathbf{W}_f s_t\right) \tag{10}$$

and the probability distribution over the source sentence words is the sum of the attention scores of the corresponding words,

$$P_{copy}(w) = \sum_{i=1}^{n} \alpha_t^i * \mathbb{1}\{w == w_i\} \tag{11}$$

Finally, we obtain the probability distribution over the dynamic vocabulary (i.e., union of original target side and source sentence vocabulary) by summing over P_{copy} and P_{vocab},

$$P(w) = \lambda_t^{copy} P_{copy}(w) + (1 - \lambda_t^{copy}) P_{vocab}(w) \tag{12}$$

where σ is the sigmoid function, and $\mathbf{W}_d$, $\mathbf{W}_e$, $\mathbf{W}_f$ are weight matrices.

4.2 Answer Span Identification

We frame the problem of identifying candidate answer spans from a paragraph as a sequence labeling task and base our model on the BiLSTM-CRF approach for named entity recognition (Huang et al., 2015). Given a paragraph of n tokens, instead of directly feeding the sequence of word vectors $\mathbf{x} = (x_1, ..., x_n)$ to the LSTM units, we first construct the feature-rich embedding $\mathbf{x}'$ for each token, which is the concatenation of the word embedding, an NER feature embedding, and a character-level representation of the word (Lample et al., 2016). We use the concatenated vector as the "final" embedding $\mathbf{x}'$ for the token,

$$x_i' = \text{concat}(x_i, \text{CharRep}_i, \text{NER}_i) \tag{13}$$

where CharRep_i is the concatenation of the last hidden states of a character-based biLSTM. The intuition behind the use of NER features is that SQuAD answer spans contain a large number of named entities, numeric phrases, etc.

Then a multi-layer Bi-directional LSTM is applied to $(x_1', ..., x_n')$ and we obtain the output state

z_t for time step t by concatenation of the hidden states (forward and backward) at time step t from the last layer of the BiLSTM. We apply the softmax to $(z_1, ..., z_n)$ to get the normalized score representation for each token, which is of size $n \times k$, where k is the number of tags.

Instead of using a softmax training objective that minimizes the cross-entropy loss for each individual word, the model is trained with a CRF (Lafferty et al., 2001) objective, which minimizes the negative log-likelihood for the entire correct sequence: $-\log(p_{\mathbf{y}})$,

$$p_{\mathbf{y}} = \frac{\exp(q(\mathbf{x}^{'}, \mathbf{y}))}{\sum_{\mathbf{y}^{'} \in \mathbf{Y}^{'}} \exp(q(\mathbf{x}^{'}, \mathbf{y}^{'}))} \quad (14)$$

where $q(\mathbf{x}^{'}, \mathbf{y}) = \sum_{t=1}^{n} P_{t,y_t} + \sum_{t=0}^{n-1} A_{y_t,y_{t+1}}$, P_{t,y_t} is the score of assigning tag y_t to the t^{th} token, and $A_{y_t,y_{t+1}}$ is the transition score from tag y_t to y_{t+1}, the scoring matrix A is to be learned. $\mathbf{Y}^{'}$ represents all the possible tagging sequences.

5 Experiments

5.1 Dataset

We use the SQuAD dataset (Rajpurkar et al., 2016) to train our models. It is one of the largest general purpose QA datasets derived from Wikipedia with over 100k questions posed by crowdworkers on a set of Wikipedia articles. The answer to each question is a segment of text from the corresponding Wiki passage. The crowdworkers were users of Amazon's Mechanical Turk located in the US or Canada. To obtain high-quality articles, the authors sampled 500 articles from the top 10,000 articles obtained by Nayuki's Wikipedia's internal PageRanks. The question-answer pairs were generated by annotators from a paragraph; and although the dataset is typically used to evaluate reading comprehension, it has also been used in an open domain QA setting (Chen et al., 2017; Wang et al., 2018). For training/testing answer extraction systems, we pair each paragraph in the dataset with the gold answer spans that it contains. For the question generation system, we pair each sentence that contains an answer span with the corresponding gold question as in Du et al. (2017).

To quantify the effect of using predicted (rather than gold standard) answer spans on question generation (e.g., predicted answer span boundaries can be inaccurate), we also train the models on an augmented "Training set w/ noisy examples" (see Table 2). This training set contains all of the original training examples *plus* new examples for predicted answer spans (from the top-performing answer extraction model, bottom row of Table 3) that *overlap* with a gold answer span. We pair the new training sentence (w/ predicted answer span) with the gold question. The added examples comprise 42.21% of the noisy example training set.

For generation of our one million QA pair corpus, we apply our systems to the 10,000 top-ranking articles of Wikipedia.

5.2 Evaluation Metrics

For question generation evaluation, we use BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014).[1] BLEU measures average n-gram precision vs. a set of reference questions and penalizes for overly short sentences. METEOR is a recall-oriented metric that takes into account synonyms, stemming, and paraphrases.

For answer candidate extraction evaluation, we use precision, recall and F-measure vs. the gold standard SQuAD answers. Since answer boundaries are sometimes ambiguous, we compute *Binary Overlap* and *Proportional Overlap* metrics in addition to *Exact Match*. Binary Overlap counts every predicted answer that overlaps with a gold answer span as correct, and Proportional Overlap give partial credit proportional to the amount of overlap (Johansson and Moschitti, 2010; Irsoy and Cardie, 2014).

5.3 Baselines and Ablation Tests

For question generation, we compare to the state-of-the-art baselines and conduct ablation tests as follows: **Du et al. (2017)**'s model is an attention-based RNN sequence-to-sequence neural network (without using the answer location information feature). **Seq2seq + copy$_{\text{w/ answer}}$** is the attention-based sequence-to-sequence model augmented with a copy mechanism, with answer features concatenated with the word embeddings during encoding. **Seq2seq + copy$_{\text{w/ full context + answer}}$** is the same model as the previous one, but we allow access to the full context (i.e., all the preceding sentences and the input sentence itself). We denote it as **ContextNQG** henceforth for simplicity. **CorefNQG** is the coreference-based model proposed in this paper. **CorefNQG–gating** is an

[1] We use the evaluation scripts of Du et al. (2017).

Models	Training set			Training set w/ noisy examples		
	BLEU-3	BLEU-4	METEOR	BLEU-3	BLEU-4	METEOR
Baseline (Du et al., 2017) (w/o answer)	17.50	12.28	16.62	15.81	10.78	15.31
Seq2seq + copy (w/ answer)	20.01	14.31	18.50	19.61	13.96	18.19
ContextNQG: Seq2seq + copy (w/ full context + answer)	20.31	14.58	18.84	19.57	14.05	18.19
CorefNQG	**20.90**	**15.16**	**19.12**	**20.19**	**14.52**	18.59
- gating	20.68	14.84	18.98	20.08	14.40	**18.64**
- mention-pair score	20.56	14.75	18.85	19.73	14.13	18.38

Table 2: Evaluation results for question generation.

Models	Precision			Recall			F-measure		
	Prop.	Bin.	Exact	Prop.	Bin.	Exact	Prop.	Bin.	Exact
NER	24.54	25.94	12.77	**58.20**	**67.66**	**38.52**	34.52	37.50	19.19
BiLSTM	43.54	45.08	22.97	28.43	35.99	18.87	34.40	40.03	20.71
BiLSTM w/ NER	44.35	46.02	25.33	33.30	40.81	23.32	38.04	43.26	24.29
BiLSTM-CRF w/ char	**49.35**	**51.92**	**38.58**	30.53	32.75	24.04	37.72	40.16	29.62
BiLSTM-CRF w/ char w/ NER	45.96	51.61	33.90	41.05	43.98	28.37	**43.37**	**47.49**	**30.89**

Table 3: Evaluation results of answer extraction systems.

ablation test, the gating network is removed and the coreference position embedding is not refined. **CorefNQG–mention-pair score** is also an ablation test where all mention-pair $score_i$ are set to zero.

For answer span extraction, we conduct experiments to compare the performance of an off-the-shelf NER system and BiLSTM based systems.

For **training and implementation details**, please see the Supplementary Material.

6 Results and Analysis

6.1 Automatic Evaluation

Table 2 shows the BLEU-$\{3, 4\}$ and METEOR scores of different models. Our CorefNQG outperforms the seq2seq baseline of Du et al. (2017) by a large margin. This shows that the copy mechanism, answer features and coreference resolution all aid question generation. In addition, CorefNQG outperforms both Seq2seq+Copy models significantly, whether or not they have access to the full context. This demonstrates that the coreference knowledge encoded with the gating network explicitly helps with the training and generation: it is more difficult for the neural sequence model to learn the coreference knowledge in a latent way. (See input 1 in Figure 3 for an example.) Building end-to-end models that take into account coreference knowledge in a latent way is an interesting direction to explore. In the ablation tests, the performance drop of CorefNQG–gating shows that the gating network is playing an important role for getting *refined* coreference position feature embedding, which helps the model learn the importance of an antecedent. The performance drop of CorefNQG–mention-pair score shows the mention-pair score introduced from the external system (Clark and Manning, 2016) helps the neural network better encode coreference knowledge.

	BLEU-3	BLEU-4	METEOR
Seq2seq + copy (w/ ans.)	17.81	12.30	17.11
ContextNQG	18.05	12.53	17.33
CorefNQG	**18.46**	**12.96**	**17.58**

Table 4: Evaluation results for question generation on the portion that requires coreference knowledge (36.42% examples of the original test set).

To better understand the effect of coreference resolution, we also evaluate our model and the baseline models on just that portion of the test set that requires pronoun resolution (36.42% of the examples) and show the results in Table 4. The gaps of performance between our model and the baseline models are still significant. Besides, we see that all three systems' performance drop on this partial test set, which demonstrates the hardness of generating questions for the cases that require pronoun resolution (passage context).

We also show in Table 2 the results of the QG models trained on the training set augmented with noisy examples with predicted answer spans.

Input 1: The elizabethan navigator, sir francis drake was born in the nearby town of tavistock and was the mayor of plymouth. ... . he died of dysentery in 1596 off the coast of puerto rico.
Human: In what year did Sir Francis Drake die ?
ContextNQG: When did he die ?
CorefNQG: When did sir francis drake die ?

Input 2: american idol is an american singing competition ... . it began airing on fox on june 11 , 2002, as an addition to the idols format based on the british series pop idol and has since become one of the most successful shows in the history of american television.
Human: When did american idol first air on tv ?
ContextNQG: When did fox begin airing ?
CorefNQG: When did american idol begin airing ?

Input 3: ... the a38 dual-carriageway runs from east to west across the north of the city . within the city it is designated as ' the parkway ' and represents the boundary between the urban parts of the city and the generally more recent suburban areas .
Human: What is the a38 called inside the city ?
ContextNQG: What is another name for the city ?
CorefNQG: What is the city designated as ?

Figure 3: Example questions (with answers highlighted) generated by human annotators (ground truth questions), by our system CorefNQG, and by the Seq2seq+Copy model trained with full context (i.e., ContextNQG).

There is a consistent but acceptable drop for each model on this new training set, given the inaccuracy of predicted answer spans. We see that CorefNQG still outperforms the baseline models across all metrics.

Figure 3 provides sample output for input sentences that require contextual coreference knowledge. We see that ContextNQG fails in all cases; our model misses only the third example due to an error introduced by coreference resolution — the "city" and "it" are considered coreferent. We can also see that human-generated questions are more natural and varied in form with better paraphrasing.

In Table 3, we show the evaluation results for different answer extraction models. First we see that all variants of BiLSTM models outperform the off-the-shelf NER system (that proposes all NEs as answer spans), though the NER system has a higher recall. The BiLSTM-CRF that encodes the character-level and NER features for each token performs best in terms of F-measure.

6.2 Human Study

We hired four native speakers of English to rate the systems' outputs. Detailed guidelines for the raters are listed in the supplementary materials.

	Grammaticality	Making Sense	Answerability	Avg. rank
ContextNQG	3.793	3.836	3.892	1.768
CorefNQG	3.804*	3.847**	3.895*	1.762
Human	**3.807**	**3.850**	**3.902**	**1.758**

Table 5: Human evaluation results for question generation. "Grammaticality", "Making Sense" and "Answerability" are rated on a 1–5 scale (5 for the best, see the supplementary materials for a detailed rating scheme), "Average rank" is rated on a 1–3 scale (1 for the most preferred, ties are allowed.) Two-tailed t-test results are shown for our method compared to ContextNQG (stat. significance is indicated with $^{*}(p < 0.05)$, $^{**}(p < 0.01)$.)

The evaluation can also be seen as a measure of the quality of the generated dataset (Section 6.3). We randomly sampled 11 passages/paragraphs from the test set; there are in total around 70 question-answer pairs for evaluation.

We consider three metrics — "grammaticality", "making sense" and "answerability". The evaluators are asked to first rate the grammatical correctness of the generated question (before being shown the associated input sentence or any other textual context). Next, we ask them to rate the degree to which the question "makes sense" given the input sentence (i.e., without considering the correctness of the answer span). Finally, evaluators rate the "answerability" of the question given the full context.

Table 5 shows the results of the human evaluation. Bold indicates top scores. We see that the original human questions are preferred over the two NQG systems' outputs, which is understandable given the examples in Figure 3. The human-generated questions make more sense and correspond better with the provided answers, particularly when they require information in the preceding context. How exactly to capture the preceding context so as to *ask* better and more diverse questions is an interesting future direction for research. In terms of grammaticality, however, the neural models do quite well, achieving very close to human performance. In addition, we see that our method (CorefNQG) performs statistically significantly better across all metrics in comparison to the baseline model (ContextNQG), which has access to the entire preceding context in the passage.

6.3 The Generated Corpus

Our system generates in total 1,259,691 question-answer pairs, nearly 126 questions per article. Figure 5 shows the distribution of different types of

	Exact Match		F-1	
	Dev	Test	Dev	Test
DocReader (Chen et al., 2017)	82.33	81.65	88.20	87.79

Table 6: Performance of the neural machine reading comprehension model (no initialization with pretrained embeddings) on our generated corpus.

The United States of America (USA), commonly referred to as the United States (U.S.) or America, is a federal republic composed of states, a federal district, five major self-governing territories, and various possessions. ... . The territories are scattered about the Pacific Ocean and the Caribbean Sea. Nine time zones are covered. The geography, climate and wildlife of the country are extremely diverse.
Q1: What is another name for the united states of america ?
Q2: How many major territories are in the united states?
Q3: What are the territories scattered about ?

Figure 4: Example question-answer pairs from our generated corpus.

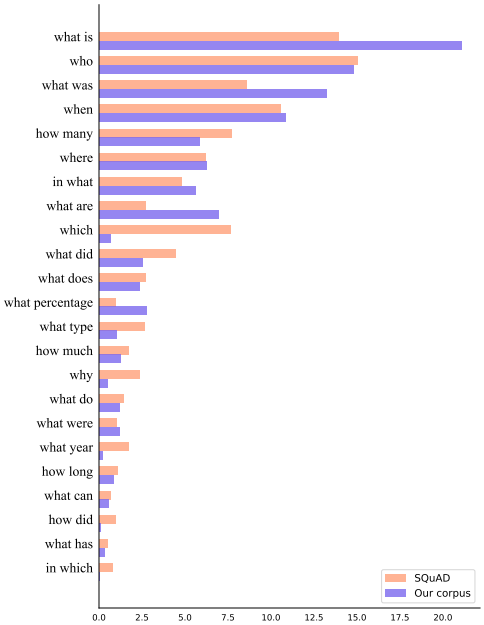

Figure 5: Distribution of question types of our corpus and SQuAD training set. The categories are the ones used in Wang et al. (2016), we add one more category: "what percentage".

questions in our dataset vs. the SQuAD training set. We see that the distribution for "In what", "When", "How long", "Who", "Where", "What does" and "What do" questions in the two datasets is similar. Our system generates more "What is", "What was" and "What percentage" questions, while the proportions of "What did", "Why" and "Which" questions in SQuAD are larger than ours. One possible reason is that the "Why", "What did" questions are more *complicated* to ask (sometimes involving world knowledge) and the answer spans are longer phrases of various types that are harder to identify. "What is" and "What was" questions, on the other hand, are often *safer* for the neural networks systems to ask.

In Figure 4, we show some examples of the generated question-answer pairs. The answer extractor identifies the answer span boundary well and all three questions correspond to their answers. Q2 is valid but not entirely accurate. For more examples, please refer to our supplementary materials.

Table 6 shows the performance of a top-performing system for the SQuAD dataset (Document Reader (Chen et al., 2017)) when applied to the development and test set portions of our generated dataset. The system was trained on the training set portion of our dataset. We use the SQuAD evaluation scripts, which calculate exact match (EM) and F-1 scores.[2] Performance of the neural machine reading model is reasonable. We also train the DocReader on our training set and test the models' performance on the *original* dev set of SQuAD; for this, the performance is around 45.2% on EM and 56.7% on F-1 metric. DocReader trained on the *original* SQuAD training set achieves 69.5% EM, 78.8% F-1 indicating that our dataset is more difficult and/or less natural than the crowd-sourced QA pairs of SQuAD.

[2]F-1 measures the average overlap between the predicted answer span and ground truth answer (Rajpurkar et al., 2016).

7 Conclusion

We propose a new neural network model for better encoding coreference knowledge for paragraph-level question generation. Evaluations with different metrics on the SQuAD machine reading dataset show that our model outperforms state-of-the-art baselines. The ablation study shows the effectiveness of different components in our model. Finally, we apply our question generation framework to produce a corpus of 1.26 million question-answer pairs, which we hope will benefit the QA research community. It would also be interesting to apply our approach to incorporating coreference knowledge to other text generation tasks.

Acknowledgments

We thank the anonymous reviewers and members of Cornell NLP group for helpful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations Workshop (ICLR)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1533–1544. http://www.aclweb.org/anthology/D13-1160.

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

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2358–2367. http://www.aclweb.org/anthology/P16-1223.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1870–1879. https://doi.org/10.18653/v1/P17-1171.

Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 643–653. https://doi.org/10.18653/v1/P16-1061.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*. pages 933–941.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA, pages 376–380. http://www.aclweb.org/anthology/W14-3348.

Xinya Du and Claire Cardie. 2017. Identifying where to focus in reading comprehension for neural question generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2067–2073. http://aclweb.org/anthology/D17-1219.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1342–1352. https://doi.org/10.18653/v1/P17-1123.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 866–874. http://aclweb.org/anthology/D17-1090.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, pages 609–617. http://www.aclweb.org/anthology/N10-1086.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301* .

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 720–728. https://doi.org/10.3115/v1/D14-1080.

Richard Johansson and Alessandro Moschitti. 2010. Syntactic and semantic structure for opinion expression detection. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 67–76.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1601–1611. https://doi.org/10.18653/v1/P17-1147.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. 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)*. volume 1, pages 889–898.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data .

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 260–270. https://doi.org/10.18653/v1/N16-1030.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *Aaai spring symposium: Logical formalizations of commonsense reasoning*. volume 46, page 47.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1400–1409. https://doi.org/10.18653/v1/D16-1147.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing-Volume 2*. Association for Computational Linguistics, pages 17–22.

Andrew M Olney, Arthur C Graesser, and Natalie K Person. 2012. Question generation from concept maps. *Dialogue & Discourse* 3(2):75–99.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Austin, Texas, pages 2383–2392. https://aclweb.org/anthology/D16-1264.

Sathish Reddy, Dinesh Raghu, Mitesh M. Khapra, and Sachindra Joshi. 2017. Generating natural language question-answer pairs from a knowledge graph using a rnn based question generation model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 376–385. http://aclweb.org/anthology/E17-1036.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*. Association for Computational Linguistics, pages 251–257.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 588–598. http://www.aclweb.org/anthology/P16-1056.

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

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. R3: Reinforced ranker-reader for open-domain question answering .

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

Terry Winograd. 1972. Understanding natural language. *Cognitive psychology* 3(1):1–191.

Xuchen Yao, Gosse Bouma, and Yi Zhang. 2012. Semantics-based question generation and implementation. *Dialogue & Discourse* 3(2):11–42.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792* .