Answer-focused and Position-aware Neural Question Generation

Xingwu Sun[1], Jing Liu[1], Yajuan Lyu[1], Wei He[1], Yanjun Ma[1], Shi Wang[2]
[1]Baidu Inc., Beijing, China
[2]Institute of Computing Technology Chinese Academy of Sciences, Beijing, China
{sunxingwu,liujing46,lvyajuan,hewei06,mayanjun02}@baidu.com
wangshi@ict.ac.cn

Abstract

In this paper, we focus on the problem of question generation (QG). Recent neural network-based approaches employ the sequence-to-sequence model which takes an answer and its context as input and generates a relevant question as output. However, we observe two major issues with these approaches: (1) The generated interrogative words (or question words) do not match the answer type. (2) The model copies the context words that are far from and irrelevant to the answer, instead of the words that are close and relevant to the answer. To address these two issues, we propose an answer-focused and position-aware neural question generation model. (1) By answer-focused, we mean that we explicitly model question word generation by incorporating the answer embedding, which can help generate an interrogative word matching the answer type. (2) By position-aware, we mean that we model the relative distance between the context words and the answer. Hence the model can be aware of the position of the context words when copying them to generate a question. We conduct extensive experiments to examine the effectiveness of our model. The experimental results show that our model significantly improves the baseline and outperforms the state-of-the-art system.

1 Introduction

The task of question generation (QG) aims to generate questions for a given text, and it can benefit several real applications: (1) In the area of education, QG can help generate questions for reading comprehension materials (Du et al., 2017). (2) QG can enable the machine to actively ask questions in a dialogue system. (3) QG can also aid in the development of question answering datasets (Duan et al., 2017). Typically, QG includes two sub-tasks: (1) what to say, i.e. determining the targets (e.g. sentences, phrases or words) that should be asked; (2) how to say, i.e. producing the surface-form of the question. In this paper, we focus on the sub-task of surface-form realization of questions by assuming the targets are given.

Previous work of QG can be classified into two categories: rule-based and neural network-based. Compared to the rule-based approach, the neural network-based approach does not rely on hand-crafted rules, and it is instead data-driven and trainable in an end-to-end fashion. The recent release of large-scale machine reading comprehension datasets, e.g. SQuAD (Rajpurkar et al., 2016) and MARCO (Nguyen et al., 2016), also drives the development of neural question generation.

Recent neural question generation approaches (Du et al., 2017; Zhou et al., 2017) employ sequence-to-sequence model that takes an answer and its context as input and outputs a relevant question. Zhou et al. (2017) further enrich the sequence-to-sequence model with rich features (e.g. answer position and lexical features) to generate answer focused questions, and incorporate copy mechanism that allows to copy words from the context when generating questions. To the best of our knowledge, it achieve the best results on SQuAD dataset so far (Zhou et al., 2017). In this paper, we implement this approach and carefully study its generation results. Specifically, we randomly sample 130 questions generated by the approach, and manually judge their quality by comparing them with the references. We find 54 out of 130 questions are ill generated, and we observe two major issues with the 54 questions: (1) 20 (37.04% errors) questions contain the question words that do not match the answer type, though the answer position feature has been incorporated. Because the model does not pay much attention to the answer that is a key to question word generation. Table 1 gives an example. A when-question

Context:	The tax collector who arrested him rose to higher political office , and Thoreau 's essay was not published until after **the end of the Mexican War**.
Answer:	**the end of the Mexican War**
Question generated by the baseline:	**Why** was Thoreau's essay published ?
Reference:	**When** was Thoreau's essay published ?

Table 1: A bad case where the generated question word does not match the answer type. A when-question should be triggered for answer "the end of the Mexican War", while a why-question is generated by the baseline.

Context:	This mechanism is still the **leading theory** today; however, a **second theory** suggests that most cpdna is actually linear and replicates through **homologous recombination**.
Answer:	**homologous recombination**
Question generated by the baseline:	What is the **leading theory** today ?
Reference:	How does the **second theory** say most cpdna replicates ?

Table 2: A bad case where the model copies the context words far away from and irrelevant to the answer. The baseline copies "leading theory" that is far away from and unrelated to the answer "homologous recombination", but neglects the phrase "second theory" that is close and relevant to the answer.

should be triggered for answer "the end of the Mexican War" while a why-question is generated by the model. (2) 11 (20.37% errors) questions copies the context words that are far from and irrelevant to the answer, instead of the words that are close and relevant to the answer. Because the model is not aware of the positions of context words. Table 2 gives an example. The baseline model copies "leading theory" that is far away from and unrelated to the answer "homologous recombination", but neglects the phrase "second theory" that is close and relevant to the answer.

To address these two issues, we propose an answer-focused and position-aware neural question generation model. (1) By answer-focused, we mean that we explicitly model question word generation by incorporating the answer embedding, which can help generate a question word matching the answer type. (2) By position-aware, we mean that we model the relative distance between the context words and the answer. The relative distance is encoded as position embedding, on which a position-aware attention is generated. The position-aware attention help the model copy the context words that are relatively close and relevant to the answer. We further conduct extensive experiments on SQuAD and MARCO dataset to examine the effectiveness of the answer-focused model and position-aware model, respectively. The experimental results show that the combination of our proposed answer-focused model and position-aware model significantly improves the baseline and outperforms the state-of-the-art system.

The contributions of this paper can be summarized as follows:

- We analyze the generation results by the state-of-the-art neural model, and find two major issues with the model: (a) the generated question words do not match the answer type; (b) the model copies the context words that are far from and irrelevant to the answer.
- To deal with these two issues, we propose an answer-focused and position-aware neural question generation model.
- We conduct extensive experiments to examine the effectiveness of our proposed model.

2 Related Work

Question Generation Previous work of QG can be classified into two categories: rule-based and neural network-based. Regardless of the approach taken, QG usually includes two sub-tasks: (1) what to say, i.e. selecting the targets that should be asked. (2) how to say, i.e. formulating the structure of the question and producing the surface realization. This is similar to other natural language generation tasks. In this paper, we focus on the second sub-task, i.e. surface-form realization of questions by assuming the targets are given.

The rule-based approaches usually include the following steps: (1) Preprocess the given text by applying natural language processing techniques, including syntactic parsing, sentence simplification and semantic role labeling. (2) Identify the targets that should be asked by using rules or semantic roles. (3) Generate questions using transformation rules or templates. (4) Rank the over generated questions by well-designed features (Heilman and Smith, 2009, 2010; Chali and Hasan, 2015). The major drawbacks of rule-based approaches include: (1) they rely on rules or templates that are expensive to manually create; (2) the rules or templates lack diversity; (3) the targets that they can deal with are limited.

To tackle the issues of rule-based approaches, the neural network-based approaches are applied to the task of QG. The neural network-based approaches do not rely on hand-crafted rules, and they are instead data driven and trainable in an end-to-end fashion. Serban et al. (2016) firstly introduce an encoder-decoder framework with attention mechanism to generate factoid questions for the facts (i.e. each fact is a triple composed of a subject, a predicate and an object) from FreeBase. Du et al. (2017) introduce sequence-to-sequence model with attention mechanism to generate questions for the text from SQuAD dataset, which contains large-scale manually annotated triples composed of question, answer and the context (i.e. the passage). Zhou et al. (2017) enrich the sequence-to-sequence model with rich features, e.g. answer position and lexical features, and incorporate copy mechanism that allows it to copy words from the context when generating a question. Their experiments show the effectiveness of the rich features and the copy mechanism. Duan et al. (2017) propose to combine templates and sequence-to-sequence model. Specifically, they mine question patterns from a question answering community and apply sequence-to-sequence to generate question patterns for a given text. Tang et al. (2017) model question answering and question generation as dual tasks. It helps generate better questions when training these two tasks together.

In this paper, we observe two major issues with the exiting neural models: (1) The generated question words do not match the answer type, since the models do not pay much attention to the answers that are critical to generate question words. (2) The model copies the context words that are far from and irrelevant to the answer, instead of the words that are close and relevant to the answer, since the models are not aware the positions of the context words. To address these two issues, we propose an answer-focused and position-aware neural question generation model. As to position-aware models, Zeng et al. (2014); Zhang et al. (2017) introduce position feature in the task of relation extraction. They apply this feature to encode the relative distance to the target noun pairs. In the task of QG, Zhou et al. (2017) apply BIO scheme to label answer position, which is a weak representation of relative distance between answer and its context words.

Sequence-to-sequence In recent years, the sequence-to-sequence model has been widely used in the area of natural language generation, including the tasks of abstractive text summarization, response generation in dialogue, poetry generation, etc. Sutskever et al. (2014) propose a sequence-to-sequence model and apply it to the task of machine translation. Bahdanau et al. (2014) introduce attention mechanism to the sequence-to-sequence model and it greatly improves the model performance on the task of machine translation. To deal with the out of vocabulary issue, several variants of the sequence-to-sequence model have been proposed to copy words from source text (Gu et al., 2016; Gulcehre et al., 2016; Cao et al., 2017; See et al., 2017).

3 Our Models

In this section, we describe the details of our models. We first describe the baseline model, a feature-enriched pointer-generator model. Then, we elaborate the proposed answer-focused model and position-aware model to deal with the two issues discussed in previous section. Finally, a hybrid model is introduced to combine these two models.

3.1 Baseline (Feature-enriched Pointer-generator Model)

Our baseline model is an attention-based pointer-generator model (See et al., 2017) enhanced with various rich features proposed by (Zhou et al., 2017). These features include: named entity (NE), part-of-speech (POS) and answer position in the embedding layer of the encoder.

The encoder in our baseline is a bidirectional LSTM, which takes the joint embedding of word, answer position and lexical features

(NE, POS) as input $(w_1, w_2, ..., w_{T_x})\ with\ w_i \in \mathbb{R}^{d_w+d_a+d_n+d_p}$, where T_x is the input length and d_w, d_a, d_n, d_p is the dimensionality of word embedding, answer position embedding, NE embedding and POS embedding respectively. It produces a sequence of hidden states $(h_1, h_2, ..., h_{T_x})$ to represent its input, each of which is a concatenation of a forward and a backward LSTM representation:

$$\begin{aligned} h_i &= [\overleftarrow{h}_i; \overrightarrow{h}_i], \\ \overleftarrow{h}_i &= \text{LSTM}(w_i, \overleftarrow{h}_{i+1}), \\ \overrightarrow{h}_i &= \text{LSTM}(w_i, \overrightarrow{h}_{i-1}) \end{aligned} \tag{1}$$

where $\overleftarrow{h}_i$, $\overrightarrow{h}_i$ are all d_h-dimensional vectors.

The decoder is a unidirectional LSTM conditioned on all encoded hidden states. At decoding step t, the decoder reads an input word embedding w_t, previous attentional context vector c_{t-1} and its previous hidden state s_{t-1} to update its current hidden state $s_t \in \mathbb{R}^{d_h}$:

$$s_t = \text{LSTM}([w_t; c_{t-1}], s_{t-1}) \tag{2}$$

The context vector c_t together with an attention distribution α_t are generated via attention mechanism (Bahdanau et al., 2014). Attention can be regarded as a semantic match between encoder hidden states and the decoder hidden state. It describes how the model spread out the amount it cares about different encoder hidden states during decoding. At step t, the context vector c_t and the attention distribution α_t are calculated as follows:

$$c_t = \sum_{i=1}^{T_x} \alpha_{ti} h_i \tag{3}$$

$$\alpha_{ti} = \text{softmax}(e_{ti}) \tag{4}$$

$$e_{ti} = v^T \tanh(W_h^T h_i + W_s^T s_t + b) \tag{5}$$

where $W_h, W_s, b\ and\ v$ are all trainable parameters.

Our baseline is based on a pointer-generator framework, which includes two complementary modes: a generation mode and a copy mode. The former mode generates words from a given vocabulary as the vanilla sequence-to-sequence model:

$$P_{vocab} = \text{softmax}\left(g(s_t, c_t)\right) \tag{6}$$

where $g(\cdot)$ is a two-layer feed-forward network with a maxout internal activation. $P_{vocab} \in \mathbb{R}^{|V|}$ denotes the vocabulary distribution with a vocabulary size of $|V|$.

The latter mode copies words directly from the source sequence. As the attention weights already measure the relevance of each input word to the partial decoding state, we treat α_t as the copy probability i.e. $P_{copy}(w) = \alpha_t$. Both modes are switched via a generation probability p_{gen} as follows:

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) P_{copy}(w) \tag{7}$$

where p_{gen} is computed from the context vector c_t, decoder hidden state s_t and the decoder input w_t:

$$p_{gen} = \sigma(f(c_t, s_t, w_t)) \tag{8}$$

$f(\cdot)$ indicates a simple feed-forward neural network that emits a single scalar value. The whole network is thus trained end-to-end according to the negative log likelihood loss of target word probability $P(w^*)$. In the baseline, we apply BIO scheme to label answer position, where B,I,O denote the begin of an answer, the non-begin of the answer and the non-answer context words respectively. Besides, we introduce dropout (Srivastava et al., 2014) with maxout (Goodfellow et al., 2013) to tackle over-fitting problem and pre-trained global vectors (Glove) for word representation (Pennington et al., 2014). All these techniques have been verified in their effectiveness in our model.

3.2 Answer-focused Model

The mismatch between generated question words and answer type is a major issue in neural question generation (NQG). Even though answer is a key to question word generation, most NQG models do not focus on answer or weakly emphasis on it when generating question words. As Table 1 shows, a when-question should be triggered for the answer "the end of the Mexican War", but an answer-irrelevant why-question is generated by the baseline. According to our analysis in Section 1, we discover that nearly 37% bad cases from our baseline fall into this category. To deal with this issue, we develop an answer-focused model.

We observe that the generation of question words is mainly related to the answer and its surrounding words. For example in Table 1, the answer and its context "until after the end of the Mexican War" already involve the essential information to generate a question word "when". This

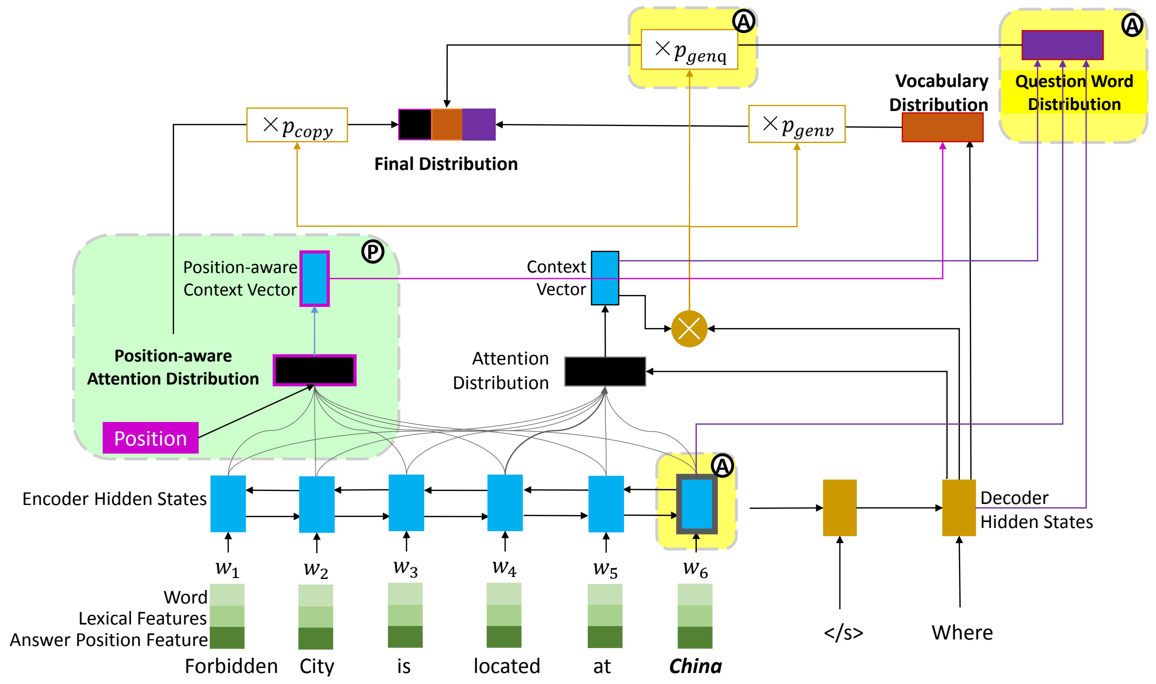

Figure 1: The modules marked with Ⓐ and colored yellow describe the answer-focused model, while the ones marked with Ⓟ and colored green describe the position-aware model. In answer-focused model, comparing to pointer-generator model, we introduce a question word generation mode and generate question words in a restricted vocabulary of question words. In position-aware model, we incorporate word position embeddings to gain a position-aware attention for further generation. The hybrid model has two types of attention distribution: position-aware and non-position-aware attention distribution and two types of context vector accordingly. Question word distribution is generated from non-position-aware context vector while vocabulary distribution is calculated from position-aware one.

suggests that the answer and its context can benefit the generation of question words. We also observe that the number of question words is limited. Therefore, we introduce a specific vocabulary of question words to directly and explicitly model the generation of question words.

As depicted in Figure 1, comparing to pointer-generator model, we introduce an additional question word generation mode to generate question words based on a restricted vocabulary of question words. This mode produces a question word distribution based on an answer embedding v_{ans}, the decoder state s_t and the context vector c_t:

$$P_{question\_word} = \text{softmax}\left(g(v_{ans}, s_t, c_t)\right) \quad (9)$$

where $P_{question\_word}$ is a $|V_{qw}|$-dimensional probability distribution, and $|V_{qw}|$ is the size of vocabulary of question words. We employ the encoded hidden state at the answer start position as the answer embedding, i.e. $v_{ans} = h_{answer\_start}$. We argue that under bidirectional encoding, this answer embedding has already memorized both the left and the right contexts around the answer region, making it a desired choice. To control the balance among different modes, we introduce a 3-dimensional switch probability:

$$p_{genv}, p_{genq}, p_{copy} = \text{softmax}(f(c_t, s_t, w_t)) \quad (10)$$

The final probability distribution is calculated via a weighted sum of the three mode probability distributions:

$$\begin{aligned} P(w) = & \; p_{genv} P_{vocab}(w) + p_{copy} P_{copy}(w) \\ & + p_{genq} P_{question\_word}(w) \end{aligned} \quad (11)$$

3.3 Position-aware Model

Another main issue of NQG is that the generated question copies the context words that are distant from and irrelevant to the answer, instead of the words that are close and relevant to the answer. In other words, attention is distracted by irrelevant words far away from the answer. As shown in Table 2, the baseline model copies "leading theory" that are far away from and unrelated to the answer "homologous recombination", but neglects the words "second theory" and "linear and replicates through" that are closer to the answer. In

Section 1, we analyze the bad cases of our baseline, and find that about 20% suffer from this phenomenon. Further analysis on these cases indicates that the closer a word is to the answer, the more likely it should be copied.

Based on these evidences, we believe that an important reason for the above phenomenon lies in the lack of word position information inside the NQG model. Following this direction, we propose a position-aware model. The model aims at enforcing local attention, and adapting the attention weights so as to put more emphasis on answer-surrounded context words by incorporating word position embeddings.

A straightforward solution to inject position information is to directly incorporate relative word position embeddings. This can inform the model where the answer is and what context words are close to it. As depicted in Figure 1, we achieve this by feeding position embeddings $(d_{p_1}, d_{p_2}, ..., d_{p_{T_x}})$ into the computation of attention distribution through a single layer network:

$$e_{ti} = v^T \tanh(W_d d_{p_i} + W_h h_i + W_s s_t + b) \quad (12)$$

$$\alpha_{ti} = \text{softmax}(e_{ti}) \quad (13)$$

where p_i is the relative distance between the i-th word and the answer, d_{p_i} is the embedding of p_i, which we call word position embedding. By optimizing the attention parameters, our model is expected to discover the correlation between target words and their relative distance from the answer. As a result, distracted attention on irrelevant input words can be avoided.

3.4 A Hybrid Model (Answer-focused and Position-aware)

In Section 3.2 and 3.3, we describe the answer-focused model and position-aware model respectively. In this section, we combine these two models to get a both answer-focused and position-aware hybrid model. As depicted in Figure 1, the hybrid model generates two types of attention distribution: position-aware and non-position-aware distribution. Accordingly, the model has two types of context vector: position-aware and non-position-aware context vector. At time step t, we calculate position-aware attention distribution α'_t as equation (12), (13), and non-position-aware one α_t as equation (4), (5). And we use c_t, c'_t to represent non-position-aware and position-aware context vector calculated as equation (3) respectively.

The hybrid model has three modes as in the answer-focused model. The question word distribution is computed from non-position-aware attention distribution as equation (9), while vocabulary distribution is calculated from position-aware one. The final distribution is weighted sum of the three mode probability distributions:

$$P_{vocab} = \text{softmax}\left(g(s_t, c'_t))\right) \quad (14)$$

$$\begin{aligned} P(w) = {} & p_{genv} P_{vocab}(w) + p_{copy} P_{copy}(w) \\ & + p_{genq} P_{question\_word}(w) \end{aligned} \quad (15)$$

where $P_{copy}(w)$ is the position-aware attention distribution α'_t.

4 Experiments

4.1 Experiment Settings

Dataset In this paper, we conduct the experiments on SQuAD and MARCO. Since the test sets of both data sets are not publicly available, we follow Zhou et al. (2017) to randomly split the development set into two parts and use them as the development set and test set for the task of question generation. In SQuAD, there are $86,635$, $8,965$ and $8,964$ question-answer pairs in our training set, development set and test set, respectively. We directly use the extracted features [1] shared by Zhou et al. (2017). In MARCO, there are $74,097$, $4,539$ and $4,539$ question-answer pairs in our training set, development set and test set, respectively. We use Stanford CoreNLP [2] to extract lexical features. We attach all the processed data sets in the supplemental materials.

Implementation Details In this paper, we set the cutoff length of the input sequence as 100 words. The vocabulary contains the most frequent $20,000$ words in each training set. The vocabulary of question words contains 20 words. We use the pre-trained Glove word vectors [3] with 300 dimensions to initialize the word embeddings that will be further fine-tuned in the training stage. The representations of answer position feature and lexical features at the embedding layer of the encoder

[1] `https://res.qyzhou.me/redistribute.zip`

[2] `https://nlp.stanford.edu/software/`

[3] `http://nlp.stanford.edu/data/glove.6B.zip`

DataSet	SQuAD				MARCO			
Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	BLEU-1	BLEU-2	BLEU-3	BLEU-4
NQG++ (Zhou et al., 2017)	-	-	-	13.29	-	-	-	-
Pointer-generator model See et al. (2017)	32.32	18.04	12.06	8.60	42.40	29.37	20.71	15.16
Feature-enriched pointer-generator model	40.49	26.11	18.94	14.34	44.45	31.85	23.32	17.90
Answer-focused model	42.10	27.52	20.14	15.36	46.59	33.46	24.57	18.73
Position-aware model	42.16	27.37	20.00	15.23	47.16	34.20	24.40	18.19
Hybrid model	**43.02**	**28.14**	**20.51**	**15.64**	**48.24**	**35.95**	**25.79**	**19.45**

Table 3: The main experimental results of baselines, answer-focused model, position-aware model and a hybrid model on SQuAD and MARCO.

are randomly initialized as 32 dimensional vectors that are trainable during training stage. The hidden size of both the encoder and decoder is 512. We use dropout only in the encoder with a dropout rate 0.5. The size of answer embedding in answer-focused model is 512. The position, that indicates the relative distance between the context words and the answer, ranges from 0 to 80 and its embedding size in position-aware model is 16. We use the optimization algorithm Adagrad (Duchi et al., 2011) with learning rate 0.15, an initial accumulator value of 0.1 and batch size as 64. We use gradient clipping with a maximum gradient norm of 2. During training, we select the best model on development set.

Evaluation Metrics We report BLEU (Papineni et al., 2002) as the main evaluation metric of the question generation systems.

Baselines In the experiments, we have three baselines for comparisons:

- **NQG++ (Zhou et al., 2017)** It is the state-of-the-art neural question generation system on SQuAD that incorporates rich features to the embedding layer of a sequence-to-sequence model and introduces copy mechanism proposed by Gulcehre et al. (2016).
- **Pointer-generator model (See et al., 2017)** It is a sequence-to-sequence model with copy mechanism that has different architecture from the one proposed by Gulcehre et al. (2016). We choose this model since its copy mechanism shows better performance (See et al., 2017). Note that we do not enable the coverage mechanism in this model to have a fair comparison.
- **Feature-enriched pointer-generator model** We add the features to the embedding layer of the pointer-generator model as described in Section 3.1.

4.2 Main Results

Table 3 shows the main results, and we have the following observations:

- The feature-enriched pointer-generator model outperforms NQG++. Both of the two models employ the sequence-to-sequence model with copy mechanism and the same features. The major difference between them is that their copy mechanism has different architecture, and pointer-generator shows better performance.
- The pointer-generator model without the features does not perform well. This verifies the effectiveness of the features extracted by Zhou et al. (2017).
- Both answer-focused model and position-aware model outperform the feature-enriched pointer-generator model and NQG++. We will analyze the effectiveness of these two models in the following two sections, respectively.
- The hybrid model shows the best performance and it outperforms the two single models, answer-focused model and position-aware model.

4.3 Answer-focused Model Analysis

As we discussed in Section 1, one major issue with the neural question generation model is that the generated question word does not match the answer type. The design of answer-focused model is explicitly modeling question word generation by incorporating answer embedding. It is expected that the answer-focused model can reduce such errors. Recall the example shown in Table 1 (Section 1). The answer-focused model correctly predicts the question word, though it copies wrong context words that can be further corrected by the hybrid model. The outputs of the answer-focused model and the hybrid model for the case in Table 1 are as follows:

- **Answer-focused model:** ***When*** did Thoreau's essay come to higher political office ?
- **Answer-focused + Position-aware model:** ***When*** was Thoreau's essay published ?

We further evaluate different systems in terms of the same question word ratio (SQWR). This metric measures the ratio of the generated questions that have the same question words as the reference. Table 4 shows the SQWR of different systems on SQuAD. We can see that answer-focused model outperforms the strong baseline, feature-enriched pointer-generator model.

Model	SQWR
Pointer-generator model (See et al., 2017)	53.17%
Feature-enriched pointer-generator model	71.58%
Answer-focused model	**73.91%**

Table 4: The answer-focused model has the highest same question word ratio.

We re-analyze the 20 (37% of errors) questions (discussed in Section 1) that have the answer type mismatching problem. Our answer-focused model can correct 7 out of the 20 bad cases. All the resolved cases have a commonality that we can easily figure out the answer type from the answer itself, as the case shown in Table 1. We also analyze the remaining 13 of the 20 unresolved cases. (1) 4 cases show that the answer type is closely related to the context words other than the answer. But these context words are far from the answer, and the encoding of the answer by LSTM has little memory of them. As shown in Table 5, the answer "an attempt to avoid responsibility for her actions" is useless to generate a question word "why", while the useful context word "because" is far from the answer. (2) 3 cases show that the answer itself has ambiguity to generate the right question words. (3) 3 cases contain the answers with the wrong named entity labels. (4) 2 cases contain answers which are out of vocabulary. (5) 1 case is hard even for human.

4.4 Position-aware Model Analysis

As discussed in Section 1, another major issue with the neural question generation model is that the model copies the context words that are far from and irrelevant to the answer. The design of position-aware model is tackling this issue by modeling the relative distance between context words and the answer, so that the attention distribution for copy mechanism will be biased towards the context words that are close and relevant to the answer. Recall the example in Table 2 (Section 1). The question generated by the baseline copies the wrong context words "leading theory" instead of "second theory" and "linear and replicates through". The outputs of our position-aware and hybrid model for this case are as follows.

Context: This action was upheld because , according to the U.S. court of appeals for the first circuit , her statement suggested a lack of remorse , **an attempt to avoid responsibility for her actions** , and even a likelihood of repeating her illegal actions .

Answer: an attempt to avoid responsibility for her actions

Reference: Why is giving a defiant speech sometimes more harmful for the individual ?

Question generated by the answer-focused model: What did the court of appeals reject ?

Table 5: A bad case of baseline remains unresolved by applying answer-focused model because answer type is closely related to the context word "because" instead of the answer itself, but "because" is far from the answer. Thus, the encoding of answer has little memory of "because".

- **Position-aware model:** The ***second theory*** suggests that most cpdna is actually ***linear and replicates through*** what ?
- **Position-aware + Answer-focused model:** Most cpdna is actually ***linear and replicates through*** what ?

We can observe that the model can copy the correct context words after introducing the position embedding to the attention distribution. Figure 2 illustrates the attention distributions for copy mechanism before and after introducing position embedding of context words. We can see that "second" has much higher probability in the position-aware attention distribution.

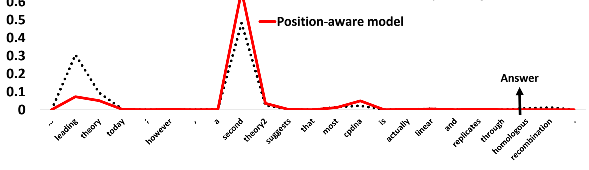

Figure 2: The two attention distributions for copy mechanism when generating questions for the case in Table 2. The position-aware model emphasizes more on the "second" that is close to the answer.

The design of position-aware model aims to help copy the context words close and relevant to the answer. We analyze the effects of position-aware model on copying out of vocabulary (OOV) words from the source sequence. We measure the effects from two perspectives: average precision and average recall. For one generated sequence, the precision is defined as the ratio of the number of OOV words appearing in both the generated question and the reference, and the number of OOV words in the generated question. The recall is defined as the ratio of the number of OOV words appearing in both the generated question and the reference, and the number of OOV words in the reference. The average precision (AP) is the mean of precision of all generated questions, and the average recall (AR) is the mean of recall of all generated questions. As Table 6 shows, our position-aware model can significantly improve the AP and AR, which indicates our position-aware model can help on copying OOV.

Model	AP	AR
Feature-enriched pointer-generator model	21.22%	18.87%
Position-aware model	**22.79%**	**20.62%**

Table 6: Our position-aware model can significantly improve the average precision and recall of copied OOV.

5 Conclusion

In this paper, we find two major issues with the existing neural question generation model. To tackle the two issues, we propose an answer-focused and position-aware model. We further conduct extensive experiments on SQuAD and MARCO dataset. The experimental results show that the combination of our proposed answer-focused model and position-aware model significantly improves the baseline and outperforms the state-of-the-art system.

6 Acknowledgments

This work is supported by the National Basic Research Program of China (973 program, No. 2014CB340505). We thank the anonymous reviewers for their constructive criticism of the manuscript.

References

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

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017. Joint copying and restricted generation for paraphrase. In *AAAI*, pages 3152–3158.

Yllias Chali and Sadid A Hasan. 2015. Towards topic-to-question generation. *Computational Linguistics*, 41(1):1–20.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

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*, pages 866–874.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Michael Heilman and Noah A Smith. 2009. Question generation via overgenerating transformations and ranking. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA LANGUAGE TECHNOLOGIES INST.

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*, pages 609–617. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

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. *arXiv preprint arXiv:1603.06807*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

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.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671. Springer.