Improving Neural Question Generation using Deep Linguistic Representation

Wei Yuan
State Key Laboratory for Novel Software Technology, Nanjing University
China

Tieke He*
State Key Laboratory for Novel Software Technology, Nanjing University
China
hetieke@gmail.com

Xinyu Dai
State Key Laboratory for Novel Software Technology, Nanjing University
China

ABSTRACT

Question Generation (QG) is a challenging Natural Language Processing (NLP) task which aims at generating questions with given answers and context. There are many works incorporating linguistic features to improve the performance of QG. However, similar to traditional word embedding, these works normally embed such features with a set of trainable parameters, which results in the linguistic features not fully exploited. In this work, inspired by the recent achievements of text representation, we propose to utilize linguistic information via large pre-trained neural models. First, these models are trained in several specific NLP tasks in order to better represent linguistic features. Then, such feature representation is fused into a seq2seq based QG model to guide question generation. Extensive experiments were conducted on two benchmark Question Generation datasets to evaluate the effectiveness of our approach. The experimental results demonstrate that our approach outperforms the state-of-the-art QG systems, as a result, it significantly improves the baseline by 17.2% and 6.2% under the BLEU-4 metric on these two datasets, respectively.

CCS CONCEPTS

• **Computing methodologies** → **Natural language generation**; Lexical semantics; • **Information systems** → *Question answering*.

KEYWORDS

Question Generation, Embedding, Pre-trained Model, Linguistic Features

ACM Reference Format:
Wei Yuan, Tieke He, and Xinyu Dai. 2021. Improving Neural Question Generation using Deep Linguistic Representation. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449975

1 INTRODUCTION

Neural Question Generation (NQG), the task of generating a question with given a context and optionally an answer with neural networks, has been attracting more and more attention from the research community in recent years. It can automatically create Question Answering (QA) datasets, which is meaningful for QA systems [10]. It is also useful for actively gathering users' feedback in AI conversational systems [45] and for generating educational materials [16]. Typically, a QG task consists of two subtasks [49]: (1) content selection, i.e. determining which part should be asked; (2) question construction, i.e. generating the surface-form of the question. The first subtask refers to the linguistic information over the input context, such as the entity information. The second one deals with creating grammatically correct and semantically precise natural language questions.

Most of the current approaches address QG by applying the sequence-to-sequence framework, composed of two parts: an encoder that extracts the meaning of input context and converts it into a vector; a decoder generates a question with this given vector [10, 54]. These sequence-to-sequence models usually incorporate attention mechanism and copy network, which are first applied by Du et al. [9] and Zhou et al. [60]. To improve QG performance, Kim et al. [20] proposes to separate answer and applies another encoder to analyze it, Zhao et al. [59] attempts to leverage paragraph-level information and Song et al. [48] utilizes multi-mode selection methods to put more emphasis on keywords in the passage. All these works are trying to mine information from the input context and the relevant answer. There are also studies treating question generation and question answering as dual tasks. Tang et al. [50] trains QA and QG simultaneously. Zhang et al. [57] uses a QA-based evaluation method to guide the QG training.

Generating adequate and fluent natural questions is a challenging task, as it requires a deep understanding of the input text, including the semantic meaning, the syntactic structure, the entity information, and so on. Although previous works achieve significant success in QG area [33], we find they do not investigate the usage of linguistic features (e.g. POS, NER, etc.) and underrate the power of these features.

Linguistic features carry rich information that is useful for generating questions. Harrison et al. [15] proves the advantages of utilizing POS and NER by conducting extensive comparison experiments. Zhang et al. [57] verifies that through simply incorporating linguistic features, the model's performance improves. As far as we know, almost all the existing approaches of utilizing linguistic features in QG are the same. They use a trainable matrix to transform the linguistic feature labels into vectors, which uses the traditional word embeddings. Unfortunately, this means has some obvious flaws resulting in poorly leveraging these linguistic features. The

*Corresponding author: hetieke@gmail.com

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449975

biggest problem of this look-up embedding method is that it overlooks the relationship among labels. For example, the NER labels of 'Los Angeles' is 'LOC LOC' ('LOC' is short for **LOC**ATION). The current approach transforms each 'LOC' label into the same vector separately. Obviously, it can describe the 'Los' and the 'Angeles' are both spatial entities, but it cannot represent that 'Los Angeles' as a whole is also a spatial entity, which is actually we want the model to know. Another problem of this naive linguistic usage is that it identifies each linguistic feature label with a static vector, resulting in poorly capturing the dynamic meaning of labels. For example, the POS tag 'RB' denotes adverb. The look-up embedding gives all 'RB' labels the same vector. But there may be some differences between 'RB' labels since the adverb can modify a verb, an adjective, and another adverb. To the best of our knowledge, current approaches do not consider this meaningful information.

Motivated by this, we propose a new approach for effectively utilizing linguistic features based on the pre-trained models. Through fine-tuning, the model generates different hidden states for different inputs, and the output layer predicts the linguistic label according to these hidden states. If the model performs well in label prediction, it means that the hidden states contain enough information to denote the linguistic labels. In other words, these hidden states can be used for representing linguistic labels. Utilizing these hidden states as representation has obvious advantages: 1) since the representation depends on both entire context and linguistic labels, it can capture the relationship between different labels, 2) the representation of linguistic labels are dynamic, therefore, enriched feature information can be presented.

We employ two commonly used pre-trained models, ULMFit and BERT, as the skeleton of our linguistic representation models. By fine-tuning these models in different NLP tasks, they become different linguistic representation models. Besides, we invent a novel linguistic feature customized for QG, namely QAF, which is short for Question Answering Feature. This novel feature is obtained from fine-tuned BERT in QA tasks. It can represent the relationship among the answer, context and question, considering QA and QG are dual tasks. We conduct extensive experiments on two QG datasets, SQuAD and MARCO. The results show that our approach significantly improves the basic model's performance by 17.2% on SQuAD and 6.2% on MARCO, with the BLEU-4 metric. Specifically, it achieves state-of-the-art results on both datasets with BLEU-4 18.99 on SQuAD and 21.98 on MARCO. Detailed results and comparisons are shown in Section 4.4.

To sum, our work makes the following contributions:

- A novel linguistic representation approach based on pre-trained models is proposed to improve QG performance, by utilizing linguistic features in an effective way. To the best of our knowledge, it is the first attempt to represent linguistic features via pre-trained models in the QG area.
- Instead of considering typical linguistic features such as POS and NER, a new linguistic feature, QAF, is invented for enhancing QG.
- Extensive experiments are conducted on two commonly used QG datasets, SQuAD and MARCO, for evaluating the performance of our proposed NQG approach. The comparison results over metrics: BLEU, ROUGE-L and METEOR all demonstrate the superiority of our approach, achieving 17.2% and 6.2% improvements with BLEU-4 metric on SQuAD and MARCO, respectively.
- Furthermore, a detailed analysis of our approach's effects on question type prediction and gated self-attention alignment is presented.
- Also, a case study about the necessity of utilizing linguistic features with sophisticated representations is conducted.

The remainder of this article is organized as follows. Section 2 presents the related work covering QG, text representation, and large pre-trained neural networks. Section 3 provides the problem definition and presents the details of our proposed NQG approach. Section 4 describes the implementation of our approach, as well as the comparative methods, datasets and metrics adopted, followed by results and discussions. Section 5 concludes our work.

2 RELATED WORK

The problem addressed and the approach proposed in this paper is related to several topics: Neural Question Generation (NQG), text representation and NLP pre-trained models. A short overview of these topics is presented in the following subsections.

2.1 Neural Question Generation

Driven by successes in deep learning, emerging studies on QG has begun to develop 'end-to-end' models with 'neural' techniques to generate question automatically. The creation of large datasets like SQuAD [40], MARCO [32] and WikiQA [55] accelerates the trend.

Du et al. [9] and Zhou et al. [60] first propose a sequence-to-sequence model to address the QG problem, and achieve success compared to the previous state-of-the-art model using automatic metrics and human evaluation. The seq2seq model incorporates attention [1] and copy mechanism [14] for content selection, which are more flexible than traditional rule-based approaches. Following that, almost all the existing works are derived from the seq2seq network.

QG can be divided into two categories: answer-aware QG and answer-unaware QG, according to whether the target answer is used as an input or not. In this paper, we focus on answer-aware QG. For answer-aware QG, the answer is crucial because it serves as the focus of asking. Sun et al. [49] proposes an answer-aware seq2seq model, which incorporates a special decoding mode to generate question words, based on the hidden state of the answer. However, this method doesn't consider lexical features and the answer structure. Kim et al. [20] separates the answer from relevant context and utilizes an extra encoder for better utilizing the information from both sides.

Linguistic information is also critical for generating natural and correct questions. Harrison et al. [15] explores the benefit of employing different linguistic features, including answer signal, word case, NER, and coreference feature. However, they do not further investigate the usage of these linguistic features, as a result, the value of linguistic features is not fully reflected. Liu et al. [25] incorporates syntactic dependency tree via Graph Convolutional Network (GCN). Khullar et al. [19] and Dhole et al. [7] attempt to use some novel linguistic features like wh-pronouns, wh-adverbs, and SRL structures. Unfortunately, their approaches are rule-based,

resulting in a limited surface form of questions due to the number of templates.

Taking wider context into consideration is practical for QG. Du et al. [8] proposes a dataset that contains long passage QA pairs. Song et al. [48] introduces a model that matches the answer with the passage before generating questions, in order to learn the information from the answer and other context in the passage. Zhao et al. [59] employs a gated self-attention network to better utilize long passage when generating questions. In order to better link the answer and broad document context, Tuan et al. [51] represents relevant context via a multi-stage attention network.

In addition to encoding rich information, some researchers pay attention to the decoding side. Interrogative words are vital for generating question, because it guides the process of generating and determines the question type. Sun et al. [49] first notices the mismatch between question type and the answer. Zhou et al. [61] and Kang et al.[18] address question word mismatch issue through incorporating the answer information in the decoding stage. Elgohary et al. [11] improves the quality of questions by rewriting the original generated question based on the context.

Improving the training process by combining supervised and reinforcement learning is another trend in QG community [4]. Yao et al. [56] deploys a model in Generative Adversarial Network (GAN) and modify the discriminator for evaluating question authenticity and predicting question type. Zhang et al.[57] proposes a QA-based reward to guide QG model training, their method outperforms previous works.

In this study, we improve QG on the encoder side. We propose a new approach to better utilize the linguistic features, furthermore, we explore a novel linguistic feature named QAF. Meanwhile, we exploit the whole paragraph as context to help generate high-quality questions.

2.2 Text Representation

Text representation can be categorized into discrete and distributional representations. The judging criterion is their ability to convey information. One-hot encoding is one of the simplest methods of representation. A single word is converted into a N-dimensional vector, filled with only one position with 1. It is straightforward but carries little extra information. Bag of Word (BoW) [58] calculates the count of each word. However, it neglects the order of words in a sentence and doesn't give any information of what the word means. TF-IDF [37, 41] is similar to BoW although it is based on the frequency of each word. Word Embedding [12] is a mapping of words into low dimensional space. There are two methods to get word embedding: Skip-gram and CBOW [29]. The word vector from embedding encodes semantic relationships among words. The intuition behind word embedding is that semantically similar words are close to each other. However, there are some disadvantages of word embedding. The word vector cannot represent different meanings of a word in different contexts. Besides, it cannot represent rare words. To tackle the issues mentioned above, a deep contextualized word representation-ELMo [36] is proposed. ELMo models both complex characteristics of word usages and the distinctions of these usages across linguistic contexts. These word vectors are obtained from the hidden states of a deep bidirectional language model (biLM), pre-trained on a large text corpus. The emerging of ELMo draws researchers' attention to pre-trained models.

2.3 NLP Pretrained Model

The idea of the pre-trained model first arose in the computer vision (CV) area. Training a large model from scratch requires a very large dataset and takes a lot of time. The pre-trained model converges fast because its weights are already optimized in some relevant tasks.

ELMo is the first large pre-trained neural network model in NLP. Universal Language Model Fine-tuning (ULMFit) [17] achieves transfer learning in three stages: general-domain LM pre-training, targeted task LM fine-tuning, and targeted task classifier fine-tuning. BERT [6], short for Bidirectional Encoder Representations, considers the context from both sides of a word. It is the first unsupervised bidirectional pre-trained model in NLP. It produces state-of-the-art performance on 11 NLP tasks. GPT [38] and GPT-2 [39] leverage transformer model to execute both supervised learning and unsupervised learning to learn text representation for downstream tasks. Since BERT is presented, there are lots of efforts on refining it. XLM [22] is an optimized cross-lingual language model based on BERT. RoBERTa [26] optimizes BERT with a larger number of weights and a bigger size of dataset. DistilBERT [42] is a distilled version of BERT, which is smaller, faster, and lighter than the original BERT. With the rapid growth of computing power, the pre-trained neural model becomes larger and larger.

In this study, we get linguistic feature representations via pre-trained models. Benefited from the strong learning ability of large pre-trained models, these representations convey more useful information than previous representations.

3 OUR APPROACH

In this section, we describe the problem statement in the first place. Then, we present the details of our approach including the baseline model and the linguistic representation method.

3.1 Problem Statement

Given a context C, and an answer A, we want a model to generating a question Q whose answer is A. Formally:

$$\hat{Q} = \underset{Q}{argmax}\, P(Q|C, A) \tag{1}$$

The context C is composed of a list of words: $C = \{x_t\}_{t=0}^{m}$, the answer A is a subspan of C. The words generated in Q are from the context C or a vocabulary V. And then we take the linguistic features F into consideration during question generation, the problem becomes:

$$\hat{Q} = \underset{Q}{argmax}\, P(Q|C, A, F) \tag{2}$$

3.2 Base Model Description

Our baseline model is a paragraph-level NQG model that mainly adopts the structure from the previous state-of-the-art model[59]. This model generates questions based on a whole passage and an answer, and uses linguistic features (POS or NER) in a traditional way.

In the later implementation, we will append different linguistic features with our representation approach to improve the performance of NQG.

3.2.1 Paragraph-Level Encoder. The encoder is composed of an embedding layer, a Recurrent Neural Network (RNN), and a gated self-attention network. In the embedding layer, a word is represented by word vector and the answer tag, following the BIO[1] tagging [60] scheme. The POS and NER tags are embedded by a trainable parameter.

The output of embedding layer is then feed into a two-layer bidirectional LSTM, producing a sequence of hidden states $H = (h_1, ..., h_T)$. At each time step i, the hidden state h_i is the concatenated representation of the forward hidden state $\overrightarrow{h}_i$ and the backward hidden state $\overleftarrow{h}_i$.

$$\begin{aligned} \overrightarrow{h}_i &= LSTM^E(e_i, \overrightarrow{h}_{i-1}) \\ \overleftarrow{h}_i &= LSTM^E(e_i, \overleftarrow{h}_{i+1}) \\ h_i &= [\overrightarrow{h}_i, \overleftarrow{h}_i] \end{aligned} \tag{3}$$

The e_i is the concatenation of the word embedding, answer position, POS tag and NER tag, $e_i = [y_i, a_i, p_i, n_i]$.

The gated self-attention mechanism [53] is used on H to aggregate the intra-dependency in the whole passage to refine the passage representation at every time step.

$$\begin{aligned} \alpha_t^e &= softmax(H^T W^s h_t) \\ s_t &= H\alpha_t^e \\ f_t &= tanh(W^f [h_t, s_t]) \\ g_t &= sigmoid(W^g [h_t, s_t]) \\ \hat{h}_t &= g_t \odot f_t + (1 - g_t) \odot h_t \end{aligned} \tag{4}$$

in which, α_t^e is the attention vector between H and h_t, s_t is the self matching representation. The self matching representation s_t then is combined with original representation h_t to calculate the new enhanced self matching representation f_t, and g_t is a learnable gate vector. The $\odot$ is the element-wise multiplication, and $\hat{h} = \{\hat{h}_t\}_{t=1}^T$ is the final gated passage-answer representation. While W^s, W^F and W^g are trainable weight matrices.

3.2.2 Maxout Pointer Decoder. The decoder is another two-layer unidirectional LSTM that generates words sequentially based on the encoder's representation and the previously generated words at each decoding step i.

$$\begin{aligned} d_i &= LSTM^D(d_{i-1}, y_{i-1}) \\ p(w_i|(w_{<i})) &= softmax(W^V y_i) \end{aligned} \tag{5}$$

The d_i denotes the hidden state of the decoder LSTM at decoding step i. d_0 equals the final hidden state h_T of the encoder. y_i is the corresponding word embedding of w_i. w_i is the i'th word in generated question. The d_i is projected to a space with vocabulary size dimensions by a linear layer. Then, a $softmax$ function is applied to calculate the probability distribution of all words in the vocabulary V.

[1] "B", for "Begin", denotes the first token of the answer; "I", for "Inside", denotes the tokens in the answer; "O", for "Others", denotes other tokens in the passage.

In order to better incorporating the encoded input representation, a Luong attention mechanism [27] is applied to dynamically aggregate $\hat{H}$ to context vector c_i at each decoding step.

$$\begin{aligned} \alpha_i^d &= softmax(\hat{H}^T W^a d_i) \\ c_i &= \hat{H} \cdot \alpha_i^d \\ \hat{d}_i &= tanh(W^c [d_i, c_i]) \end{aligned} \tag{6}$$

The W^a and W^c are trainable weights. Then, the context vector c_i is used to update the decoder state. Thus, the equation of LSTM decoding changed to:

$$d_i = LSTM^D(\hat{d}_{i-1}, y_{i-1}) \tag{7}$$

The same as previous NQG models, to alleviate the out-of-vocabulary (OOV) problem, pointer network [13, 43, 52] is introduced to allow both selecting words from input and generating words from a prepared vocabulary during generation. The probability of copying words is computed based on the raw attention score, since the attention weights already indicate the importance of each input words to the decoding word. Different from the typical pointer network, instead of using the sum of raw attention score over input word, we use the max raw attention score to avoid repetition problems.

$$r_i = \hat{H}^T W^a d_i \tag{8}$$

$$score^{cp}(y_i) = \begin{cases} \sum\limits_{x_k = y_i} r_{i,k} & y_i \in V_{in} \\ -\infty & y_i \notin V_{in} \end{cases} \tag{9}$$

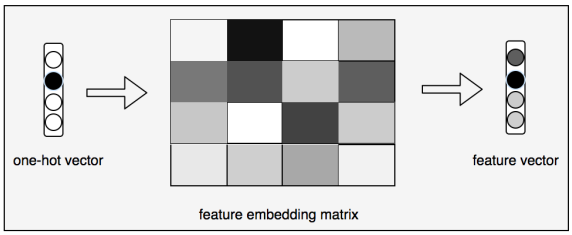
where $r_i = \{r_{i,k}\}_0^T$, x_k is the k'th word in the input passage, V_{in} is the input sequence vocabulary. The copy score $score^{cp}$ is concatenated with the generative score $score^{gen} = W^V y_i$ to compute probabilities over all the words in the fixed vocabulary V and the input sequence vocabulary V_{in}. Finally, the probability of w_i is:

$$p(w_i|(w_{<i})) = softmax([score^{gen}(y_i), score^{cp}(y_i)]) \tag{10}$$

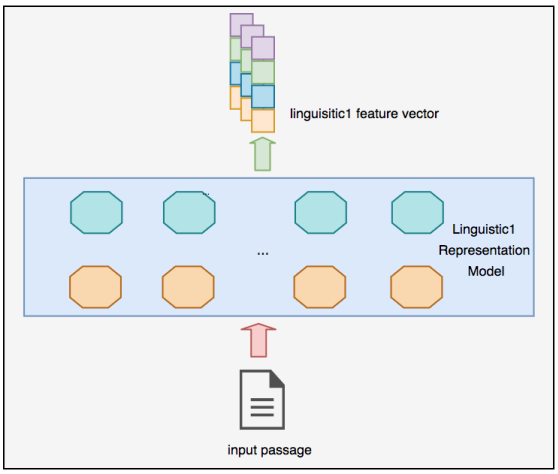
3.3 Linguistic Representation Models

Although we use the long passage as the context to generate a question, it is necessary to utilize the linguistic information as supplement during generation. Previous works simply incorporate the linguistic features by constructing a look-up based embedding for each linguistic label, which is similar to traditional word embeddings. However, this naive usage cannot fully exploit the meaning of each linguistic feature, therefore, the power of these features is limited. To tackle this problem, we design linguistic representation models for each linguistic feature. The representations from these sophisticated models carry more information than the previous method and boost the base model's performance significantly, and finally achieves the state-of-the-art. Fig. 1 illustrates the difference between the traditional linguistic feature representation and our representation. The following parts present the approaches to obtaining different linguistic representations.

3.3.1 Contextualized Word Representation Model. There are at least two defects of applying the look-up word embeddings to represent the words in the input passage: (1) Each word may have more than one meaning but only get a single vector. That means the word vector either inaccurately represents the 'average' meanings of



(a) Traditional Linguistic Feature Representation



(b) Our Linguistic Feature Representation

Figure 1: The difference between traditional linguistic feature representation and our linguistic feature representation

the word or only carries one of the meaning. (2) Word with different context contains different meanings, the look-up embeddings cannot dynamically capture this change. Thus, we apply deep contextualized word vectors in the base model. The contextualized word vector is obtained from the hidden states of ULMFit that is essentially a deep neural language model (LM). At first, we utilize a AWD-LSTM[2] based LM which is pre-trained on wikitext-103[3]. Then, we fine-tune the model on QG datasets in the LM tasks. In order to get QG-specific deep contextualized word vector, we fine-tune the model in a question-type prediction task. In the question-type prediction task, a prediction layer is added to the representation model. The input is the passage followed by an answer, and the output is the question-type such as: what, when, where, how, who, etc.

3.3.2 NER Representation Model. Named Entity Recognition (NER) is a subtask of information extraction that identifies the named entities in a text. In most cases, the question answering is about the specific entity in a passage. Thus, incorporating the NER labels which directly locate the important entities is undoubtedly advantageous for question generation. Table 1 presents an example demonstrating the important role of NER in QG. The standard question is asking about religions, with the identified NER label, the NQG model can easily locate the key position in the passage.

Table 1: An example proves the critical role of NER in question generation (Bold denotes NER label, Underline denotes named entity)

passage	It is used in particular in reference to Christianity[**RELIGION**], Judaism[**RELIGION**] , Islam[**RELIGION**] and Marxism[**RELIGION**] .
question	What religions and idea of thought is heresy cited as being used frequently in ?

Previous works [48, 57] utilize the NER feature in the same way as word embeddings. However, it is inferior to deal with each NER label separately for two reasons: (1) In some cases, named entities are phrases, therefore, only a span of NER labels can precisely represent it. (2) Different entities in the same passage may have some relations. For example, the *PERSON* entity with a *LOCATION* entity may imply that a person is in someplace. Therefore, it is inappropriate to simply adopt the word embedding for NER labels.

Seeking to address the issues mentioned above, we employ a pre-trained transformer to learn the representation of NER linguistic information. The model is trained in the NER task on the QG dataset. We obtain the NER labels in the data preprocessing stage. The model is forced to figure out the associations between the NER labels, as it needs such associations to tag words. After training, the hidden states of the model are used for NER representation. The representation is then incorporated into encoding via concatenation.

3.3.3 POS Representation Model. POS tags are a set of labels that assign part of speech to each word, such as nouns, verbs, adverbs, etc. These tags denote the property of the word in a certain sentence. The instance in Table 2 is from SQuAD. By incorporating the POS tags, the structure of this passage becomes clear, then, the NQG model can generate a question as deep as the standard question with confidence. Compared with NER, POS emphasizes the syntax role of each word. Naturally, there are interactions among POS labels, such as one qualifies another. Using look-up embedding to represent POS labels will lose the interaction information, as the same label owns the same representation no matter what the circumstances are. Therefore, we apply a POS representation model to alleviate this problem. Similar to the NER representation model, we fine-tune a pre-trained model in the POS task and utilize the hidden states of this model as the POS representations. We concatenate the POS representation with other representations and feed them to the base model.

3.3.4 QAF Representation Model. QAF, short for Question Answering Feature, is a novel linguistic feature designed for QG system in our work. As shown in Figure. 2, QA is a dual task of QG, it generates a correct answer according to the question and relevant context. In order to select the correct answer span in the input passage, a QA system has to figure out the latent relationship among the question,

[2]https://s3.amazonaws.com/fast-ai-modelzoo/wt103-fwd.tgz
[3]https://s3.amazonaws.com/research.metamind.io/wikitext

Table 2: An example proves the critical role of POS in question generation (Bold denotes POS label)

passage	According[**VBG**] to[**TO**] Titus[**NNP**] 3:10[**CD**] a[**DT**] divisive[**JJ**] person[**NN**] should[**MD**] be[**VB**] warned[**VBN**] two[**CD**] times[**NNS**] before[**IN**] separating[**VBG**] from[**IN**] him[**PRP**] .
question	How many times is it suggested that you should warn people you are in disagreement with before parting ways ?

the passage, and the answer. This is significant for QG, since QG aims to figure out this relationship to generate proper questions. According to this intention, we invent the QAF linguistic feature to indicate the relationship among passage, answer and question. We employ a previous state-of-the-art QA model and train it on the QG dataset. During QA task, the input of the model is the passage and a specific question, the output is a pair of numbers indicating the start and end position of the answer in the input passage. When using it as a representation model, the input of the model is only the passage, and we select parts of its hidden states as the representation of QAF. The hidden states contain QAF information because the QA model outputs the answer position based on them.

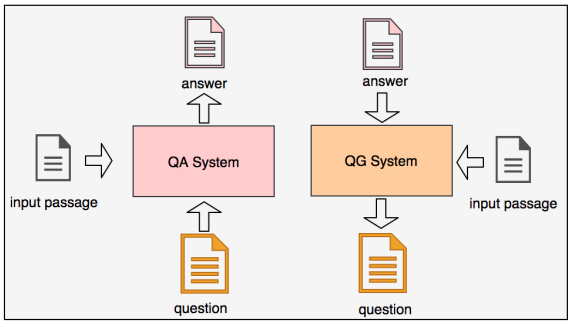


Figure 2: QA as dual task of QG.

3.4 The Framework of QG

Figure. 3 shows the steps of training a QG model in our work. The process can be categorized into three stages: data preprocessing, representation model training, and QG model training. In the data preprocessing stage, by using spaCy, the original QG data is tokenized, then POS and NER are performed. All the features used by our model are calculated in this stage. In the linguistic representation model training stage, we get the representation model by fine-tuning the pre-trained model. Different linguistic features correspond to different representation models fine-tuned in different tasks. These linguistic feature representations are incorporated into the base QG model via concatenation. In the final stage, the QG model takes the passage and the answer with the linguistic feature representation as input, and outputs the proper question. After using the new linguistic feature representation as complements in input, the performance of QG model significantly improves.

4 EXPERIMENTS

In this section, we present the evaluation of our proposed NQG approach. First, we briefly introduce the datasets used in this paper. Second, we describe the implementation details of our experiments. Then, we present the metrics we adopt. Finally, we describe the experimental results with discussions.

4.1 Datasets

4.1.1 SQuAD. The SQuAD[4] is one of the most commonly used QG dataset. It contains 536 Wikipedia articles and more than 100k questions posed about the articles by Amazon Mechanical Turks crowd-workers [40]. The questions are produced by these workers by using their own words without copying any phrase from the passage. Later, the crowd-workers provide answers to the questions which are spans in the passage. We use SQuADv1.1 as our QG dataset, it provides the question, the passage and the answer in a JSON format. Since there are about 10% data that are unavailable in the original dataset, we treat the remaining data as the entire dataset. Then, we randomly split the dataset into training, dev and testing set, each set containing 87, 599, 5, 286 and 5, 285 question-answer-passage triples respectively. After that, we employ spaCy[5] for preprocessing: tokenization, POS, and NER. We also lower-case the entire dataset for reducing the number of different words.

4.1.2 MS-MARCO. In MS-MARCO[6], the 1, 010, 916 questions with 1, 026, 758 unique answers are generated by sampling queries from Bing's search logs[32]. Unlike SQuAD, the answers in MS-MARCO are editorially generated, which means they are not just the span in the passage. We use the data released by Zhou et al. [60]. There are 74, 097, 4, 539 and 4, 539 question-answer-passage triples in the train, dev, test set respectively. The same as the preprocessing of SQuAD, we apply spaCy to tokenize, tag POS and NER for these data, and then lower-case the entire dataset. Table 3 provides some statistics on the processed datasets. The average of passage and question lengths are about 139 and 11, 84 and 7 in SQuAD and MS-MARCO, respectively.

4.2 Model Implementation

All of our models are implemented using PyTorch[7]. For the representation models, there are two types of pre-trained models: ULMFit and BERT. We implement ULMFit by utilizing Fast.ai[8] library which provides a single consistent interface to train and use the ULMFit model. It is trained for two epochs on both SQuAD and MARCO datasets. We apply one-cycle-policy [46] to cycle the learning rate between lower bound and upper bound during a complete run which could keep the model from overfitting. The BERT based representation model is made using the PyTorch implementation of BERT made by HuggingFace[9]. We fine-tune BERT for five epochs in NER, POS, and QA tasks, respectively. For optimization, we use Adam [21] optimizer with weight decay and learning rate of 3e-5. After training, we pick the three model with the highest accuracy

[4]https://rajpurkar.github.io/SQuAD-explorer/
[5]https://spacy.io/
[6]https://microsoft.github.io/msmarco/
[7]https://pytorch.org/
[8]https://www.fast.ai/
[9]https://github.com/huggingface/transformers

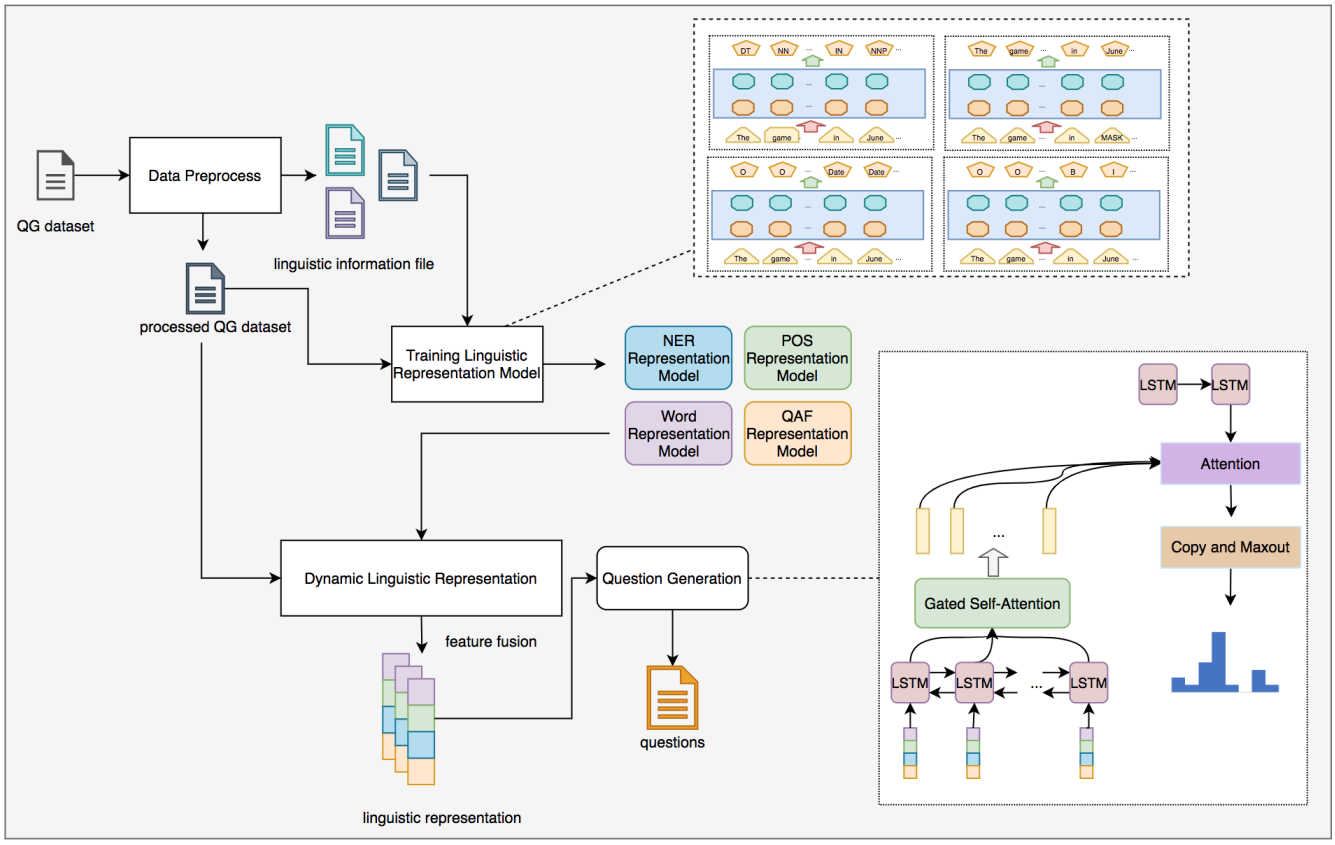


Figure 3: The Steps of Training our NQG Models.

Table 3: Statistics of train/dev/test sets for SQuAD and MARCO

Dataset	total	passage length avg.	question length avg.
SQuAD	87599/5286/5285	139.7/142.5/129.6	11.2/11.4/11.6
MARCO	74097/4539/4539	84.3/84.4/84.3	7.0/7.0/7.0

and select the one which uses the most number of epochs as the final model.

For the QG model, 2 layers of LSTM cells is used for the encoder and decoder. For the encoder, the LSTM is bidirectional and the hidden size is 300. The LSTM cell is with 0.3 dropout rate. Word embeddings are initialized with the pre-trained 300-dimensional Glove[10] vectors [35]. The vocabulary contains the most frequent 45,000 words in the training set. The representation of answer positions is randomly initialized as 3-dimensional vectors. The max length of input is set to 400. The hidden size of the decoder is 600. In testing, we apply beam search with 10 beam size and the probability of output 'unknown' word is set to 0.

4.3 Evaluation Metrics

Following existing works, we apply the traditional n-gram similarity based automatic evaluation metrics: BLEU-1, BLEU-2, BLEU-3, BLEU-4 [34], METEOR [5] and ROUGE-L [23] to evaluate our models. BLEU is one of the earliest and the most popular automatic metrics [3]. It evaluates the quality of text by calculating the co-occurrence n-gram frequency between candidate text and reference text. METEOR is based on the harmonic mean of unigram precision and recall, usually, the recall weighted is higher than precision. Besides the standard exact word matching, it can evaluate stemming and synonymy matching [2]. ROUGE-L essentially is a Longest Common Subsequence (LCS) based metric [24]. It identifies the longest co-occurring in sequence automatically. All these metrics are computed by using the package[11] released by Sharma et al [44]. In addition, we adopt a new metric called 'Q-Metrics' [30],

[10] http://nlp.stanford.edu/data/glove.840B.300d.zip

[11] https://github.com/Maluuba/nlg-eval

Table 4: Experimental results of our model comparing with previous works on SQuAD (The best value is in bold). '*' is the model that uses NER and POS in the traditional way.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	Q-BLEU1
Du et al. [9]	43.09	25.96	17.50	12.28	16.62	39.75	-
Song et al. [47]	-	-	-	13.98	18.77	42.72	-
Sun et al. [49]	43.02	28.14	20.51	15.64	-	-	-
Zhao et al. [59]	45.07	29.58	21.60	16.38	20.25	44.48	-
Nema et al. [31]	46.41	30.66	22.42	16.99	21.10	45.03	-
Zhang et al.* [57]	-	-	-	17.00	21.44	46.76	-
Zhang et al. [57]	-	-	-	18.65	**22.91**	45.89	-
our model							
base model	44.99	29.39	21.39	16.20	20.25	44.38	53.59
+w	45.19	29.81	21.90	16.75	22.16	44.64	54.06
+q	46.95	31.51	23.29	17.82	21.73	46.98	56.29
+p	46.87	31.77	23.62	18.20	21.62	46.86	56.02
+n	47.53	31.98	23.64	18.10	21.94	47.10	56.59
+p+n	47.91	32.69	24.49	**18.99**	22.15	47.53	56.68
+p+n+w	47.82	32.52	24.21	18.66	22.10	47.45	56.61
+p+n+w+q	**48.25**	**32.99**	**24.63**	18.96	22.39	**47.92**	**57.05**

Table 5: Experimental results of our model comparing with previous works on MARCO (The best value is in bold).

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L
Du et al. [9]	-	-	-	10.46	-	-
Duan et al. [10]	-	-	-	11.46	-	-
Sun et al. (answer-focused model) [49]	46.59	33.46	24.57	18.73	-	-
Sun et al. (hybrid model) [49]	48.24	35.95	25.79	19.45	-	-
Zhao et al. (paragraph level) [59]	-	-	-	17.24	-	-
Ma et al. [28]	50.33	37.10	27.23	20.46	24.69	49.89
our model						
base model	54.50	37.24	27.29	20.61	25.44	56.41
+w	55.13	37.73	27.48	20.72	**27.88**	57.72
+q	55.27	37.64	27.30	20.42	26.03	56.79
+p	55.45	38.11	27.92	21.17	26.18	57.10
+n	55.52	38.28	28.02	21.18	26.30	57.36
+p+n	55.83	38.52	28.38	21.55	26.25	57.58
+p+n+w	**56.19**	**39.07**	**28.90**	**21.98**	26.70	**57.96**
+p+n+w+q	55.37	38.10	28.09	21.35	26.28	57.47

which is specifically designed for evaluating QG[12]. We only use 'Q-BLEU1' which has the highest correlation with human evaluation on SQuAD. Following Nema et al., we set w_{ner} to 0.41, w_{imp} to 0.36, w_{sm} to 0.03, w_{qt} to 0.20 and δ to 0.66 to calculate Q-BLEU1.

4.4 Results and Discussions

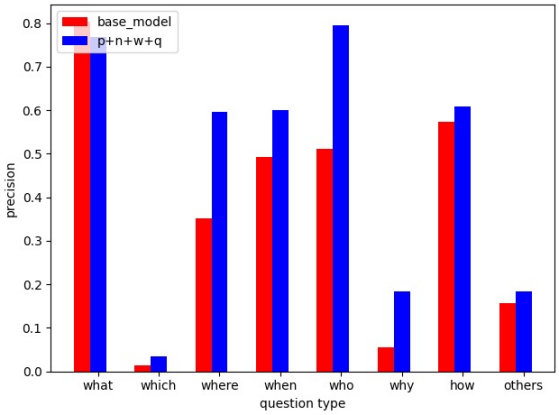
4.4.1 Main Results Analysis. On SQuAD, we compare the BLEU, METEOR, ROUGE-L and Q-BLEU scores of our models with several previous methods in Table 4, including Du et al. [9], Song et al. [47], Sun et al. [49], Zhao et al. [59], Nema et al. [31] and Zhang et al. [57]. Du et al. is one of the first to apply deep learning in QG task. Song et al. is the first to use a query-based generative model for solving both tasks of QG and QA. Sun et al. and Zhao et al. improve the QG performance through the question type and long context views.

[12]https://github.com/PrekshaNema25/Answerability-Metric

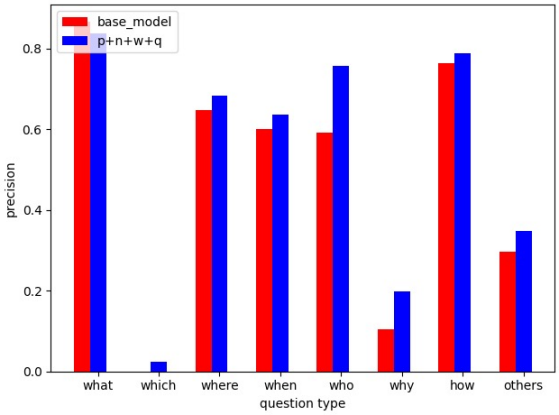
We mainly adopt Zhao et al.'s model architecture as the base model. And we add a linear layer and a dropout layer at the encoder to avoid overfitting and remove $< SOS >$ and $< EOS >$ tokens of the input passage. Zhang et al.'s work is the previous state-of-the-art. The comparative models in this work are listed as follows:

- *base model*: the base model (mentioned in Section 3.2)
- *+ w*: the base model + contextualized **w**ord representation model (Section 3.3.1)
- *+ q*: the base model + **Q**AF representation model (Section 3.3.4)
- *+ p*: the base model + **P**OS representation model (Section 3.3.3)
- *+ n*: the base model + **N**ER representation model (Section 3.3.2)

- *+ p + n*: the base model + **P**OS representation model + **N**ER representation model
- *+ p + n + w*: the base model + **P**OS representation model + **N**ER representation model + contextualized **w**ord representation model
- *+ p + n + w + q*: the base model + **P**OS representation model + **N**ER representation model + contextualized **w**ord representation model + **Q**AF representation model

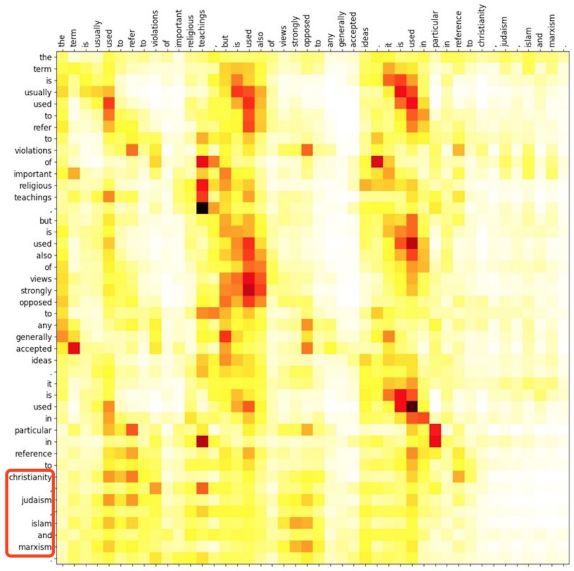


(a) Accuracy of Question Type on SQuAD

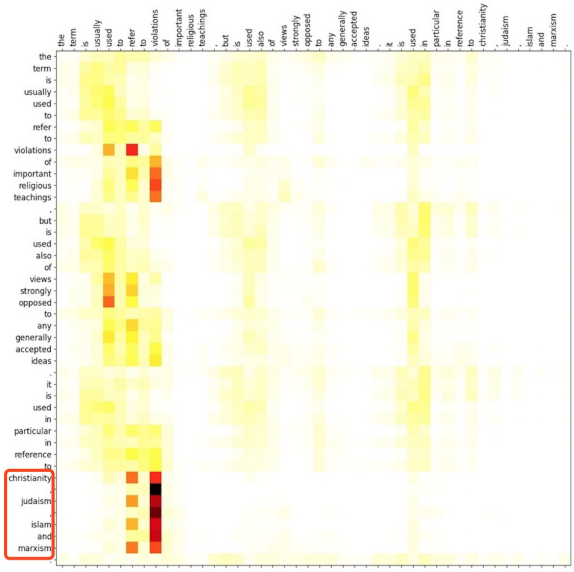


(b) Accuracy of Question Type on MARCO

Figure 4: The improvements on question type accuracy via incorporating new linguistic representations



(a) Gated self-attention map of base model



(b) Gated self-attention map of base model (+ p + n + w + q)

Figure 5: Gated self-attention map (the darker the grid, the higher the attention score). The standard question of this example is 'What religions and idea of thought is heresy cited as being used frequently in ?' The words in the red box are the relevant answer.

As shown in Table 4, overall, by incorporating new feature representations, our model obtains obvious performance gain over the base model, with at most 17.2% (2.79 points) gain under metric BLEU-4, about 10.5% (2.14 points) gain with METEOR, about 7.9% (3.54 points) gain with ROUGE-L and 3.46 points gain with Q-BLEU1 on SQuAD. In addition, compared with the model that uses linguistic features in the traditional way (the '*' in Table 4), our model outperforms it over all the metrics. Which, demonstrates the superiority of our approach. Furthermore, we can tell from Table 4 that the special designed linguistic feature QAF, improves the Q-BLEU1 scores significantly. The Q-BLEU1 metric highly corresponds to the answerability of the generated questions. Finally, we compare it with the previous works. Our model outperforms the existing state-of-the-art method in all the metrics except METEOR.

Table 6: Question type proportions on two datasets

Dataset	What	Which	Where	When	Who	Why	How	Others
SQuAD	43.36%	4.70%	3.79%	6.27%	9.44%	1.39%	9.39%	21.66%
MARCO	44.29%	1.11%	4.16%	1.52%	1.47%	1.88%	16.28%	29.29%

We also conduct comparative experiments on MARCO to further measure the effectiveness of our method. The same as SQuAD, we do paragraph-level experiments on MARCO. It means that the input is a whole paragraph related to the answer and question, rather than a single sentence. Table 5 presents the experimental results on MARCO. First, we compare BLEU-1, BLEU-2, BLEU-3 and BLEU-4 scores reported by Du et al. [9], Duan et al. [10], Sun et al. [49] and Zhao et al. [59]. As can be seen from Table 5, our models obtain dominating performance on MARCO with BLEU-1, BLEU-2, BLEU-3, BLEU-4. Meanwhile, the base model assembled with NER, POS and contextualized word representation models achieves the state-of-the-art results on MARCO, with BLEU-1 56.19, BLEU-2 39.07, BLEU-3 28.90, BLEU-4 21.98, and ROUGE-L 57.96. To mention, the base model with contextualized word representation model outperforms all the other models with METEOR 27.88 on MARCO.

4.4.2 Question Type Analysis. As we know, question type (also called interrogative word) is vital for question generation, because it guides the remain generating process and determines part of question meaning. Only a slight change in the interrogative word will lead to giant changes in the whole question's meaning. For example, the only difference between 'What was a major reason and justification for the European wars of religion ?' and 'Who was a major reason and justification for the European wars of religion ?' is the question type, however, the meaning of these two sentences are totally different. Therefore, if the question type is incorrect, the semantic of generated question will drift far away. We can make a better decision on which question is correct with the context and relevant answer. If the context and the answer emphasize an event, the prior question type is correct. If the context and the answer are about a person, the second question type is correct.

As mentioned above, the context and relevant answers are crucial for generating proper interrogative words. By applying the new linguistic feature representations, our method provides abundant linguistic information for the NQG model. Therefore, our method can enhance the model's ability to generate correct question type.

On the SQuAD and MARCO, 7 most common question words make up most of the questions (78% on SQuAD and 70% on MARCO). Table 6 is the statistic of question types on these two dataset. Therefore, we divide question types into 8 categories, including the most common 7 question types and an additional type 'others'. Then, we calculate the accuracy of question type with the base model and 'base model + p + n + w + q' model on both datasets. Fig. 4 shows the accuracy of different question types on SQuAD and MARCO in detail. Our model outperforms the baseline for most question types on SQuAD and MARCO.

4.4.3 Gated Self-Attention Analysis. As a paragraph contains more context information than a sentence, taking a paragraph as input may better guide NQG models. However, RNNs suffer greatly from vanishing or exploding gradients when handling long sequences, which limits the usage of paragraphs. Therefore, a gated self-attention is introduced to aggregate information from the whole passage. Before gated self-attention applied to QG, the performance of taking paragraphs as input is usually worse than a sentence for NQG tasks [59].

In our work, we do not specifically refine the gated self-attention network, but as we enrich the input feature information, the gated self-attention works more effectively than before, since it has more information to refer to. To demonstrate this finding, we visualize the self-attention alignment vector for one example of two models in Fig. 5. This example corresponds to the example in Table 1. As we can see, in Fig. 5b, the alignment distribution concentrates on the answer tokens and the most relevant context 'violations of important religious teachings'. While the alignment distribution in Fig. 5a concentrates near the answer span but it is a little scattered.

4.4.4 Case Study. In Table 7, we provide two examples for which the linguistic features help the NQG model generate a proper question. These examples demonstrate that when rich feature information is presented, the model will generate question containing relatively accurate information comparing with no feature presented.

In the first example, firstly, the standard question is asked about 'band', without linguistic features, the base model generates question about 'song', which is obviously improper. Meanwhile, from the content of the base model's question, we can see that the model does not know 'paul gascoigne' is a whole entity. So, the question generated by the base model is confusing. In contrast, after incorporating our linguistic feature representation models, the model generates the question correctly both in the theme and the semantic meaning.

The second example is relatively challenging because it contains so many named entities and most of these entities are not a single token. As mentioned in Section 3.3, in this situation, in order to generate correct questions, the linguistic information need to not only identify the linguistic features, but also represent the relationship among these entities. Without powerful linguistic representation, the base model is confused by the vast crowd entities. Thus, the meaning of base model's question is not intact. It contains an unnecessary entity '2007' but misses the important entity 'newcastle'. On the contrary, by utilizing the new linguistic feature representations, the model generates a question whose meaning is the same as the standard question. It generates the whole entity sequence 'newcastle international airport' in the question.

5 CONCLUSION

In this paper, we propose a novel linguistic representation approach that is based on large pre-trained neural networks to tackle the question generation problem. Compared with previous linguistic

Table 7: Examples that comparing the results of the base QG model with our QG model. (Bold denotes NER label, Underline denotes named entity, Boxed denotes the answer, Bold and italic denotes the merits of our model results. Taking readability into consideration, we do not show other linguistic labels in this table.)

paragraph	lindisfarne[**ORG**] are a folk-rock group with a strong tyneside[**GPE**] connection . their most famous song , " fog on the tyne[**GPE**] " (1971[**DATE**]) , was covered by geordie[**NORP**] ex-footballer paul gascoigne[**PERSON**] in 1990[**DATE**]. venom , reckoned by many to be the originators of black metal and extremely influential to the extreme metal scene as a whole , formed in newcastle[**GPE**] in 1979[**DATE**]. folk metal band skyclad[**ORG**] , often regarded as the first folk metal band , also formed in newcastle[**GPE**] after the break-up of martin walkyier thrash metal [**PERSON**] band , sabbat . andy taylor[**PERSON**] , former lead guitarist of duran duran[**PERSON**] was born here in 1961[**DATE**]. brian johnson[**PERSON**] was a member of local rock band geordie[**NORP**] before becoming the lead vocalist of ac/dc[**ORG**] .
standard question	what band is considered by many to be the first black metal group ?
base model QG	what famous song was covered by geordie ex-footballer paul ?
base model (+ p + n + w + q) QG	what was the name of the ***band*** that was formed by ***paul gascoigne*** ?
paragraph	newcastle international airport[**FAC**] is located approximately 6 miles[**QUANTITY**] (9.7 km [**QUANTITY**]) from the city centre on the northern outskirts of the city near ponteland[**GPE**] and is the larger of the two[**CARDINAL**] main airports serving the north east[**LOC**] . it is connected to the city via the metro light rail system and a journey into newcastle city[**GPE**] centre takes approximately 20 minutes[**TIME**] . the airport handles over five million[**CARDINAL**] passengers per year , and is the tenth[**ORDINAL**] largest , and the fastest growing regional airport in the uk[**GPE**] , expecting to reach 10 million[**CARDINAL**] passengers by 2016[**DATE**] , and 15 million[**CARDINAL**] by 2030[**DATE**]. as of 2007[**DATE**] , over90 [**CARDINAL**] destinations are available worldwide .
standard question	how many destinations are available worldwide from newcastle 's airport ?
base model QG	in 2007 , how many destinations are available worldwide ?
base model (+ p + n + w + q) QG	how many destinations are available in ***newcastle international airport*** ?

feature usages, our approach can not only represent more abundant linguistic information, but also identifies the relationship among these linguistic features, which is infeasible before. In addition, a new linguistic feature, QAF, is invented for the QG task. We demonstrate the effectiveness of this approach by conducting extensive experiments on two most commonly used QG dataset, SQuAD and MARCO. The results show that our approach achieves state-of-the-art performance on both datasets. We also prove that our approach is applicable to improving the accuracy of question type prediction and is beneficial to gated self-attention alignment. Finally, two examples are presented to further illustrate the advantages of utilizing our approach.

ACKNOWLEDGMENTS

This work was partially supported by the Fundamental Research Funds for the Central Universities, the Equipment Development Department Pre-Research Foundation of China (31511110310), and the National Natural Science Foundation of China (61936012).

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[2] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 65–72.
[3] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluation the role of bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.
[4] Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2019. Reinforcement learning based graph-to-sequence model for natural question generation. *arXiv preprint arXiv:1908.04942* (2019).
[5] 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*. 376–380.
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[7] Kaustubh D Dhole and Christopher D Manning. 2020. Syn-QG: Syntactic and Shallow Semantic Rules for Question Generation. *arXiv preprint arXiv:2004.08694* (2020).
[8] Xinya Du and Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from wikipedia. *arXiv preprint arXiv:1805.05942* (2018).
[9] Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106* (2017).
[10] 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*. 866–874.
[11] Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019. Can you unpack that? learning to rewrite questions-in-context. *Can You Unpack That? Learning to Rewrite Questions-in-Context* (2019).
[12] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722* (2014).
[13] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393* (2016).
[14] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148* (2016).
[15] Vrindavan Harrison and Marilyn Walker. 2018. Neural generation of diverse questions using answer focus, contextual and linguistic features. *arXiv preprint arXiv:1809.02637* (2018).
[16] 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*. 609–617.
[17] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* (2018).
[18] Junmo Kang, Haritz Puerto San Roman, and Sung-Hyon Myaeng. 2019. Let Me Know What to Ask: Interrogative-Word-Aware Question Generation. *arXiv preprint arXiv:1910.13794* (2019).

[19] Payal Khullar, Konigari Rachna, Mukul Hase, and Manish Shrivastava. 2018. Automatic question generation using relative pronouns and adverbs. In *Proceedings of ACL 2018, Student Research Workshop*. 153–158.

[20] Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6602–6609.

[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[22] Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291* (2019).

[23] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[24] Chin-Yew Lin and FJ Och. 2004. Looking for a few good metrics: ROUGE and its evaluation. In *Ntcir Workshop*.

[25] Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019. Learning to generate questions by learningwhat not to generate. In *The World Wide Web Conference*. 1106–1118.

[26] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[27] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[28] Xiyao Ma, Qile Zhu, Yanlin Zhou, and Xiaolin Li. 2020. Improving Question Generation with Sentence-Level Semantic Matching and Answer Position Inferring. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 8464–8471.

[29] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[30] Preksha Nema and Mitesh M Khapra. 2018. Towards a better metric for evaluating question generation systems. *arXiv preprint arXiv:1808.10192* (2018).

[31] Preksha Nema, Akash Kumar Mohankumar, Mitesh M Khapra, Balaji Vasan Srinivasan, and Balaraman Ravindran. 2019. Let's Ask Again: Refine Network for Automatic Question Generation. *arXiv preprint arXiv:1909.05355* (2019).

[32] 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. (2016).

[33] Liangming Pan, Wenqiang Lei, Tat-Seng Chua, and Min-Yen Kan. 2019. Recent advances in neural question generation. *arXiv preprint arXiv:1905.08949* (2019).

[34] 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 of the Association for Computational Linguistics*. 311–318.

[35] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[36] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[37] Shahzad Qaiser and Ramsha Ali. 2018. Text mining: use of TF-IDF to examine the relevance of words to documents. *International Journal of Computer Applications* 181, 1 (2018), 25–29.

[38] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

[39] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1, 8 (2019), 9.

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

[41] Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, Vol. 242. New Jersey, USA, 133–142.

[42] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

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

[44] Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *arXiv preprint arXiv:1706.09799* (2017).

[45] Heung-Yeung Shum, Xiao-dong He, and Di Li. 2018. From Eliza to XiaoIce: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering* 19, 1 (2018), 10–26.

[46] Leslie N Smith. 2018. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820* (2018).

[47] Linfeng Song, Zhiguo Wang, and Wael Hamza. 2017. A unified query-based generative model for question generation and question answering. *arXiv preprint arXiv:1709.01058* (2017).

[48] Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 569–574.

[49] Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3930–3939.

[50] Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027* (2017).

[51] Luu Anh Tuan, Darsh J Shah, and Regina Barzilay. 2019. Capturing Greater Context for Question Generation. *arXiv preprint arXiv:1910.10274* (2019).

[52] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in neural information processing systems*. 2692–2700.

[53] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 189–198.

[54] Zichao Wang, Andrew S Lan, Weili Nie, Andrew E Waters, Phillip J Grimaldi, and Richard G Baraniuk. 2018. QG-net: a data-driven question generation model for educational content. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*. 1–10.

[55] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2013–2018.

[56] Kaichun Yao, Libo Zhang, Tiejian Luo, Lili Tao, and Yanjun Wu. 2018. Teaching Machines to Ask Questions.. In *IJCAI*. 4546–4552.

[57] Shiyue Zhang and Mohit Bansal. 2019. Addressing semantic drift in question generation for semi-supervised question answering. *arXiv preprint arXiv:1909.06356* (2019).

[58] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics* 1, 1-4 (2010), 43–52.

[59] Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3901–3910.

[60] 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*. Springer, 662–671.

[61] Wenjie Zhou, Minghua Zhang, and Yunfang Wu. 2019. Question-type Driven Question Generation. *arXiv preprint arXiv:1909.00140* (2019).