Question Generation on Text Data using NLP

By Nikhil Bagaria

Question Generation on Text Data using NLP

Nikhil Bagaria

To my family and my guide

Abstract

Question Generation is a fast-growing field of Natural language Processing. The question can be generated on various input data formats such as Text, Image, Structure database, and many others. In this thesis, we focused on how to generate questions on text corpus, sentences, and even paragraphs.

Here in the Question Generation system, we generated questions on the SQuAD V1.0 dataset, a Reading Comprehension dataset with approximately 1,00,000 question-answers pairs. We have fine-tuned Sequence-to-Sequence and BERT model on the SQuAD dataset to generate questions on the available text corpus. Later we compared the two results generated by Seq-2-Seq and BERT on various comparison metrics such as BLEU, METEOR, and ROGUE scores.

Contents

Chapter 1

Introduction

In recent years, the field of research in Question generation has become an interesting field for academic and industrial communities. Nowadays chatbots such as Siri, Alexa, and Google assistant, all the chatbots, use question generation techniques to have a continuous interactive conversation with the user. If we consider in the academic area, it is an essential component of the self-learning intelligent systems used for evaluating the knowledge and simulating the student's learning. NewsQA, SQuAD[25], WIKIHOP, CNN/Daily Mail[3], Google's Natural Question and TriviaQA give enough datasets to train the machine comprehension models. The goal is to utilize the knowledge whatever the machines have learned to solve the problems. These can be done in two ways, either by making it answer the questions, or letting it to ask the meaningful questions. So, asking the question is the direct way to access the knowledge which machine had learned. Question generation goals is to generate the human like questions from the given context (paragraphs) and the specific target answer.

Question Generation : To access the knowledge from the machine by asking the questions as a form of output is impressive for language comprehension. Question Generation's main task is to generate the questions from the paragraph automatically. It is the answer-aware question generation. In answer-aware question generation, context and answer are given as an input to the model and ask to generate the question for that particular answer by considering the context. QG (Question Generation) mainly relies on rule-based[20] methods such as temple-based, syntax-based, and semantic-based methods to convert the paragraph to the related question.

1.1 Motivation

The motivation behind doing the Question generation is it's applications. What is the need to generate the questions with the given information? The answer to this question is for the reading comprehension[10] task or for the interactive conversations by chatbots. Suppose to draft questions for reading comprehension question generation can be used by giving the software the context and the answer and it can automatically generate the sensible question for the particular context. It can also be used for the evaluating the knowledge of the students by generating the questions for student in the form of the course quiz. The question generation can be used in the data augmentation techniques that is to modify the current data to increase the number of the training data. If we are training the model for question answering, we can create the relevant questions for the context and answer and that can be a module for the datasets of the question-answer. In this way there are many applications for the question generation technique in different fields.

1.2 Problem Statement and Contribution

Our goal is to generate the question from the given context and answer as an input to the model and get the question as the output of the model. In this work, we have attempted to reproduce the state-of-the-art model. We have used the SQuAD 1.0[25] data set which has approximately 1,00,000 questions and answer pairs. Seq2Seq learning method and BERT model is used for fine tuning using SQuAD data set to generate the question on the text corpus available.

Chapter 2

Related Work

In the early days, question generation methods used rule-based approaches. They transform input into the syntactic form, which is further used to generate the interrogative form of sentences. Research works such as [15] [16] focused on constructing the question templates and applying them for question generation. These methods uses a set of templates and then rank them for other domains.

As these question generation model uses rule-based approaches, these approaches do not use semantic roles of words and instead just use the syntactic roles of words. The authors such as Heilman and Smith [10], Mazidi and Nielsen [19], and Labutov et al, [13] use the heuristic rules to create the question with the constructed templated and rank them. Overgenerate-and-rank approach is introduced in Heilman and Smith [10], the proposed model overgenerates the questions and rank them. The main aim is to convert the declarative sentences into questions using the manually created rules to perform the transformation. These questions are ranked using the logistic regression model.
The work by Xuchen Yao, and G.B., Zhang. [31] proposes converting the sentence into the MRS (Minimal recursion Semantics) using linguistic parsing and constructing grammar rules and semantic structures from MRS to generate the questions. These all approaches require manual work, which mainly depends on human creation.

The approach by Nan Duan and Duyu Tang [9] trains an end-to-end neural network by using the Sequence to Sequence or encoder-decoder framework and it is training the question generation model using the sequence-to-sequence model. This work inspired several follow up works in question generation. This automatic question generation model outperformed the previous proposed rule-based model by Heilman and Smith[10] , which is human dependent.
Du et al.[22], presented the question generation to a context by extracting the information from the context for the model or taking the whole context as the input, Kim et al. [11], Sun et al.[28]
In paper[27], the two-stage neural model is created, which generates the question by detecting the essential phrases from the context. Linfeng Song, Z.W., and Hamza [26], propose encoding both the answer and the context with the multi-perspective matching mechanism to employ both Question-Generation and Question-Answering.

Earlier, rule-based approaches were used, then with time, they switched to the sequence to sequence, RNN based model, BERT based model [30] and reduce the human effort dependency. The model's evaluation still relies on the traditional metric like the BLEU score and METEOR score. In our work, we were using sequence-to-sequence learning and BERT Model for Question Generation Task.

2.1 Background

The question generation uses the concepts of tokenization, language models, and deep learning. The first step in the process is to tokenize the text means to split the text or sequence in the form of tokens for further process. The token cannot be defined as a word; it is the basic unit of the generation task and language model. The direct and most straightforward way to tokenize any text is to split it based on white spaces. For example – " Siri uses the QG method" can be tokenized into five tokens: "Siri","uses", "the", "QG", "method,". Seq2Seq is an end-to-end learning task in which a model converts a sequence to another sequence. It uses the encoder-decoder technique, the fixed-size vector input maps, and converts it to the target. Devlin, Chang, Lee, and Toutanova [8] presented a new language model as BERT . BERT uses a transformer encoder, which prevents the tokens from attending to the future tokens for preserving the transformer as an autoregressive language model.

2.1.1 Sequence-to-Sequence Learning (Seq-2-Seq)

Seq-2-Seq is an end-to-end learning task in which a model converts a sequence to another sequence. It uses the encoder-decoder technique, the fixed-size vector input map, and convert it to the target. In this process the context for each item is the output from the last step. In the encoder-decoder architecture[1], the encoder convert the item to the hidden vector, which contains context and item and the decoder reverses it using the previous output as the input, convert the vector into the output.

Recurrent Neural Network (RNN) and Long Short-term Memory(LSTM) : Let there be a sequence of inputs $(x_1, x_2 \ldots\ldots, x_t)$ the recurrent neural network, it computes the sequence of outputs as $(y_1, y_2, \ldots . y_t)$ by iterating[29] :

$$h_t = \sigma(W^{hx}x_t + W^{hh}h_{t-1})$$
$$y_t = (W^{yh}h_t)$$

Here are the details associated with the parameters:

1. $h_t = \sigma(W^{hx}x_t + W^{hh}h_{t-1})$: The relationship to compute the hidden layer out put features -
   - $x_t \epsilon R^d$ is the input word vector at time t.
   - $W^{hx} \epsilon R^{D_h * d}$ is the matrix(weight) used to condition the input word vector x_t.
   - $W^{hh} \epsilon R^{D_h * D_h}$ is the matrix(weight) used to condition the output of the previous time-step h_{t-1}.

- $h_{t-1} \epsilon R^{D_h}$ is the output of the non-linear function at the preceding time step,$t-1$.
- $\sigma()$ - non-linear function.

2. $y_t = W^{yh} h_t$: The output probability over the vocabulary at every time step. $W^{yh} \epsilon R^{|V|*D_h}$ and $|V|$ is the vocabulary.

The recurrent neural network can map the seq-2-seq whenever the alignment between the outputs and inputs is known. There is a case when the input and output sequences have different lengths with non-monotonic and complicated relationships. In this condition, it is unclear how to use the recurrent neural network in this problem.
In the work by K. Cho, B. Merrienboer[6] the approach for general sequence method is to map the input to the fixed-sized vector using one recurrent neural network and then map the vector to the target vector with another rnn.
As the rnn have all the required information, it will be tough to train the recurrent neural network due to the long-term dependencies. The LSTM can succeed in this; it learn the problems with the long-range temporal dependencies. The aim of the LSTM is to evaluate conditional probability $p(y_1, y_2, \dots .y_{t'} \mid x_1, x_2, \dots .x_t)$ where the output sequence length is t' which might be different from t , $(x_1, x_2, \dots ..x_t)$ is the input sequence and $(y_1, y_2, \dots ..y_t)$ is the output sequence. The LSTM computes the conditional probability:

$$p(y_1, y_2, \dots .y_{T'} \mid x_1, x_2, \dots .x_T) = \Pi_{t=1}^{T} p(y_t \mid v, y_1, y_2, \dots .y_{t-1})$$

The fixed dimension as v of the input sequence $(x_1, x_2, \dots .x_T)$, provided by the last hidden-state of Long Short-term Memory and then calculated the probability of output sequence $(y_1, y_2, \dots .y_{T'})$. Each $p(y_t \mid v, y_1, y_2, \dots .y_{t-1})$ is represented with the softmax of the words in the vocabulary.

2.1.2 BERT

Devlin, Chang, Lee, and Toutanova [8] a new model as BERT (Bidirectional Encoder Representations from Transformers). Bert uses a transformer encoder, which prevented the tokens from attending to the future tokens for preserving the transformer as an autoregressive language model.

Architecture

The BERT is similar and identical to the architecture of the transformer[30] encoder and is created by a stack of multilayer bi-directional transformer encoders.
Transformer: Vaswani et al. [30] proposed the transformer model . It has neural network encoder-decoder architecture. The transformer produces state-of-art results for many different types of translation tasks with the help of attention and dense normalization layers. Its architecture is faster in training, and the scope of parallelization compared to RNN and LSTM is not possible in both. These model properties have decreased the computational cost and become the base for most of the research in the NLP field.
The transformer model [30] relies entirely on the self-attention concept to compute the input and output representations without using sequence-aligned RNNs. Self-attention is the mechanism that allows the inputs to interact with each other with the mathematical operation and decide to whom to pay more attention. Self-attention can be perceived as an embedding vector of one of the words in the input sequence. The embedding vector undergoes calculations with itself and other embeddings of different words from the same sequence and forms interrelations among themselves. As the entire process goes in parallel, the embedding vectors of the words are encoded with positional embeddings, which are calculated to represent the order of appearance of the words in the original sequence.

The transformer's encoder and decoder both produce the required output from the given input sequences as embedding. The encoder encodes each word of the batch of sentences into a 512-dimensional embedding vector.
The decoder uses the inputs from the encoder's outputs and the target phase to train the network. The encoder's result is given to the decoder

at each N level as a top portion of the decoder is stacked N times. Some components of the architecture of the transformer model -

Multi-head attention: Multi-head attention performs various parallel computations to produce different results for the word. The best word is then produced by concatenating these findings with the Softmax, and the softmax determines the word with the highest probability. Taking single attention with the d_{model} - dimensional queries, keys and values h times different, learned linear projections to d_v, d_k and d_k. Then the attention function is performed parallel and yielding d_v - dimension output. The attention allows the model to combined attend to information from various subspaces at various positions. Q, K, V is referred as the feature representation of query, key, and value. Single attention head, averaging –

$$MultiHead(Q, K, V) = Concat(head_1, \ldots\ldots.head_h)W^O$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

The projection are parameter matrices $W_i^Q \epsilon R^{d_{model}*d_k}$, $W_i^K \epsilon R^{d_{model}*d_k}$, $W_i^V \epsilon R^{d_{model}*d_v}$ and $W^O \epsilon R^{d_{model}*hd_v}$.

Feed Forward Networks(FFN) : In the position wise Feed Forward Networks, with attention sub-layers, each layers in the encoder and decoder have a full connected FFN, which is applied to every position. This contain two linear transformations with a ReLU activation in between them –

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

Linear Transformations : The linear transformations are the same at various positions and use other parameters from layer to layer.

The three parameters in the BERT model are – the hidden size, the number of self-attention heads, and the number of layers. The output layer, the activation function, and the input representation of BERT are different from the transformer encoder. Like BERT have one extra embedding module, it uses it to indicate the other parts of tokens in

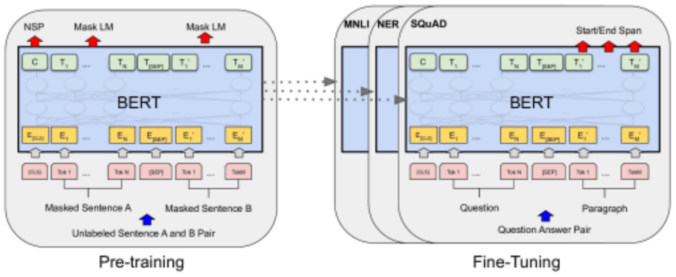

Figure 2.1: BERT Architecture[8]

input; The Gaussian Error Linear Unit is an activation function, and the purpose of the output layer is task-specific means converting the attended sequence to the task-specific output. The input for the model must be aligned means [CLS] the classification token is inserted as the first token in the input sequence, and the end hidden state of [CLS] token is used as the final sequence for classification work. The complete token sequence is the pack of multiple sentences. The special token is used as [SEP] between the tokens of two sentences to separate the sentences, and a learned embedding is added to the tokens that denote which tokens belong to which sentence.

For example let sentence pair be $(s1, s2)$, Number of tokens in s1 as $|s1|$, Number of tokens in s2 as $|s2|$ .Then Input Sequence as $([CLS], ti.......,$ $|s1|, [SEP], tj......, |s2|)$.The input token is the sum total of the three embeddings – the position embedding, the token embedding, and the segmentation embedding. Then it is sent to extra layers for the fine-tuning process.

2.1.3 BERT-QG

The base model of bert-qg which has the architecture from the previous art model and added features as (POS) Part of Speech and (NER) Name Entity. Then applies deep contextualized word vectors and tie the two matrix - the word embedding matrix and the output projection matrix. Suppose a passage as $p = [x_i]_{i=1}^{N}$ and answer as a, the output

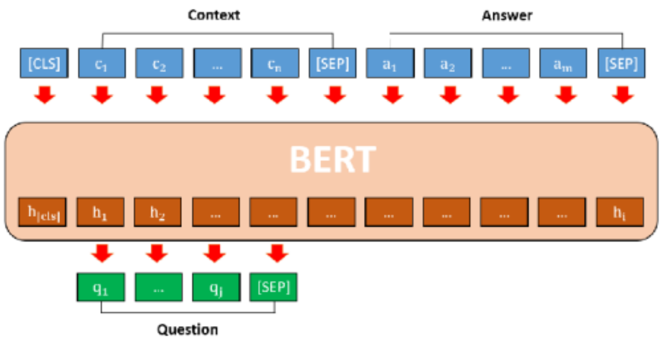

Figure 2.2: BERT-QG[2]

will be a question as $q = [y_i]_{i=1}^{N}$. So, the question q can be answered by a based on the passage p.
The pre-trained BERT model in the question generation task to generate the question from the given context and answer as the input like in the figure. The BERT question generation model encodes the context and outputs the question token one at a time.

$$\text{Let, } X = ([CLS], C, [SEP], A, [MASK]).$$

This input X consists of [CLS] classification token, C as the context token, [SEP] as separator token, and [MASK] as a mask token. The q is the question tokens which are generated and then recurrently fed into the BERT model.

Let the BERT model be as bert(), the hidden representation as $H \epsilon R^{|X|*h}$ is obtained by calculating $H = bert(X)$, X is the input sequence and $|X|$ is the size(length) of the input sequence. h is the hidden dimension. Then the H is passed through the dense layer $W \epsilon R^{h*|V|}$ and followed by the softmax function and get the q_i is the correct question token.

$$P(q \mid X) = argmax(softmax(H.W + B))$$
$$q_i = argmax_w Pr(w \mid x_i)$$

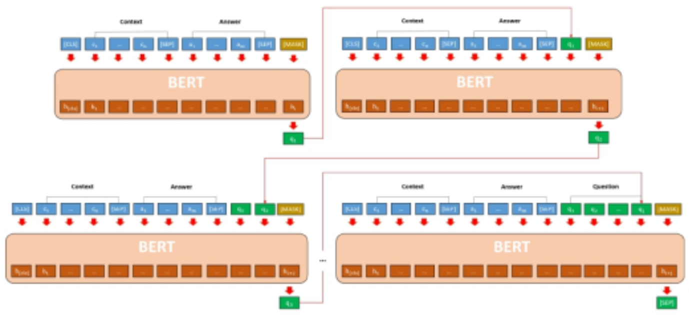

Figure 2.3: BERT-SQG architecture[2]

2.1.4 BERT-SQG

In BERT-QG model, the token generation is performed without previous decoded result due to this BERT-SQG model is introduced in which there is sequential question generation. In the BERT-SQG model the information of the previous decoded result is taken into consideration for encoding the token. In this model let's suppose context paragraph as $C = [c_1, c_2, \ldots..c_n]$ and answer as $A = [a_1, a_2, \ldots..a_m]$ and the question as $Q = [q_1, q_2, \ldots..q_i]$. The input sequence –

$$X_i = ([CLS], C, [SEP], A, [SEP], q_1, ..., q_i, [MASK])$$

the [SEP] is predicted. This input sequence is then travelled forward into the model. Then we take the final hidden state for the end token [MASK] in the input sequence. The final hidden vector of $[MASK]$ as $h_{[MASK]} \epsilon R^h$ and adapt the BERT model by adding an affine layer as $W_{SQG} \epsilon R^{h*|V|}$ to the output of the [MASK]. The label Probabilities as-

$$P_r(w \mid X_i) = softmax(h_{[MASK]}.W_{SQG} + b_{SQG})$$
$$q_i = argmax_w P_r(w \mid X_i)$$

The generated q_i is appended into the input sequence X, and the question generation process is repeated with the new X until [SEP] is predicted.

Chapter 3

Our Work

We approached by setting up and understanding a baseline model and adding the features to the model. Then implementing this model to SQuAD dataset. This helped us in understanding the model and dataset in the natural manner with this we tried to refine the performance metrics of the model. We implemented the BERT-QG model which generate the questions by taking the previous decoded results into consideration for the decoding process, the generated question token is appended in the input sequence for the question generation, and Sequence-to-Sequence Model and fine-tuned with the dataset.

We started training the model for the dataset and run the training process with the different hyper-parameters values. Then trying with different values of the training hyper-parameters and with numerous failed attempts, We got the right values of the hyper-parameters. In the Bert-QG model, the two features added are POS and NER, also tied the word embedding matrix with the output matrix.

The Sequence-to-Sequence model task is to generate human-like question. The people ask question by paying attention to the part of the sentences as well as associating the context information from the passage. We modelled the conditional probability using encoder-decoder architecture and to focus on the elements(tokens) of the input sentences when generating the word during the decoder process, we used the global attention mechanism. In this work we encoded both the paragraph level and sentence level.

3.1 Dataset

3.1.1 The SQuAD Data Set

We used the SQuAD v1. (Standford Question Answering Dataset) for our work. The SQuAD dataset contains a set of question and answers pairs. The dataset mainly focuses on the task related to question answering. We used it for the question generation model.

The SQuAD dataset was created with the help of Wikipedia articles. 23,215 paragraphs were extracted and then split into 0.8 of complete data as a training set and 0.1 of complete data as a validation set, and 0.1 of complete data as a testing set.

The properties of the dataset : The answer is partitioned into the categories such as a person, date, location, clause, numeric, etc. It consists of 19.8% of the answers as dates and numbers, 32.6% of answers as nouns, 31.8%of answers as noun phrases, and 15.8% as other categories. The questions are labeled into different types of reasoning required to answer.

As there are different datasets available, I chose the SQuAD dataset because, it is big. It has 1,00,000+ questions. Secondly, it is challenging as another dataset has the answer to the particular question present in many documents. Thirdly, the SQuAD has more complex answers, so generating questions from them was better for evaluating the model.

3.1.2 Data Pre-processing

We used Standford CoreNLP[18] in the pre-processing for tokenizing and sentence splitting and added two more other features as (POS)

Part-of-speech and (NER) Name-Entity. We then converted the entire dataset into the lower-case. This process was done on the context, question, answer and sentence of the SQuAD dataset.

3.2 Embedding

Vectors that capture the meaning of the word, similar type of words have closer numbers in their vectors.These vectors are called as Embedding. Embedding is used in text analysis for word representation. It encodes the words in such a way that words closer to them in vector space have a similar meaning. In this approach, the model is trained on the SQuAD dataset, and it is a Seq-2-Seq model, taking context as the input to the encoder. It uses the GloVe embeddings[23] to convert the word into the token form and then using into the network. We removed the GloVe vectors and replaced them with the BERT word embeddings. The baseline model, in which the context is passed as the input and used the output embedding as the input to the encoder. We also added POS and NER features in the model.

3.3 Encoder

The paragraph encoder is used only for the paragraph level information and the attention based encoder is used for the model. In attention-based encoder bi-directional long short term memory is used to encode the sentence,

$$\begin{aligned} \overrightarrow{b_t} &= \overrightarrow{LSTM_2}(x_t, \overrightarrow{b_{t-1}}) \\ \overleftarrow{b_t} &= \overleftarrow{LSTM_2}(x_t, \overleftarrow{b_{t+1}}) \end{aligned}$$

$\vec{b_t}$ is defined as the hidden state at the time t for forward and $\overleftarrow{b_t}$ is the hidden state for the backward.
c_t is the attention-based encoding at the decoder at the time t, and b_t is the context dependent represented by $b_t = [\overrightarrow{b_t}; \overleftarrow{b_t}]$

$$c_t = [\sum_{i=1,....|x|} a_{i,t} b_i]$$

$a_{i,t}$ is the attention weight, calculated with the help of Softmax function and the bilinear scoring function

$$a_{i,t} = \frac{exp(h_t^T W_b b_i)}{\sum_j exp(h_t^T W_b b_j)}$$

In the attention weight $a_{i,t}$ equation, h_t is the hidden state at t (time step), b_i and b_j are the context dependent representation at time step i and j respectively and W_b is the parameter to be learned.
We used bidirectional LSTM for paragraph encoder and paragraph as p. Paragragh at time step t is p_t and $\overrightarrow{d_t}$ is defined as the hidden state at the time step t for forward and $\overleftarrow{d_t}$ is the hidden state for the backward.

$$\overrightarrow{d_t} = \overrightarrow{LSTM_2}(p_t, \overrightarrow{d_{t-1}})$$
$$\overleftarrow{d_t} = \overleftarrow{LSTM_2}(p_t, \overleftarrow{d_{t+1}})$$

3.4 Model

As we got the model's input, we would experiment with it, train it with these inputs, and gradually improve the model's performance. Here, we implemented two models – The sequence-to-Sequence model and the Bert-QG model.
In the seq-2-seq model, we tried to implement the model Bahdanau et al.[1] , Cho et al.[5] the conditional probability using the encoder-decoder architecture. In the model, to make it concentrate on the elements(tokens) of input when generating the word at the time of decoding, we adopted the global attention Luong et al. [17] So, here question generation used the model, and the end-to-end architecture converted the input into the vector, and then the decoder used the vectors to make the prediction.
In the BERT model, we implemented the BERT from the official git repository, downloaded it, and fine-tuned it with our own set of hyper parameters. We had used the BERT base model (uncased), naming it "bert-base-uncased," which was the pre-trained model. The repository

had 24 models with different layers and hidden state sizes. We used the BERT model, which had 12-layers, 768-hidden state sizes, 12 attention heads, and 110M parameters. These parameters with each hidden state of maximum sequence length 768 and output of 768 length. We used this BERT model by adding features during the pre-processing and embedding layer to achieve better results.

3.4.1 BERT-QG

We used the (pre-trained) BERT model in the question generation work to generate the question from the given context and answer as the input. The BERT question generation model encoded the context and outputs the question token one at a time.

$$\text{Let, } X = CLS, c_1, c_2...c_n, SEP, q_1, q_2, \ldots .q_n, MASK.$$

This input X consisted of [CLS] classification token, C as the context token, [SEP] as separator token, and [MASK] as a mask token. The q was the question tokens which generated and then recurrently fed into the BERT model.
This input sequence is then travelled forward into the model. Then we take the end hidden state for the end token [MASK] in the input sequence. The final hidden vector of $[MASK]$ as $h_{[MASK]} \epsilon R^h$ and adapt the BERT model by adding an affine layer as $W_{SQG} \epsilon R^{h*|V|}$ to the output of the [MASK]. The new generated token q_i token is appended to X the input sequence and the question generation process is repeated with the new X till the [SEP] is predicted.The label Probabilities as-

$$P_r(w \mid X_i) = softmax(h_{[MASK]}.W_{SQG} + b_{SQG})$$
$$q_i = argmax_w P_r(w \mid X_i)$$

The BERT model uses two training strategies: The following sentence prediction and the masked LM strategy. In the masked LM strategy, some words in the sequence are replaced by the [MASK] token before giving input. Then the model predicts the values of the words which are masked based on the given context. The classification layer is addedon the top of the encoder to predict the output. Then multiplying the embedding matrix and the output vectors, with the help of softmax, calculating the probability of every word, and the loss function only considers the predicted mask values.

In the next sentence prediction, the model received various sentences as an input; it tried to enticipate whether the following sentence was the subsequent sentence of the original text. At the time of the training process, most of the inputs were given as a pair in which the second sentence was the following sentence in the original text, and the remaining pairs were given in the random form. Some pairs were given in the combination of random sentences from the context, so to help the model, we used the [CLS] token at the starting of the first sentence. To separate the first sentence from the following sentence, we used the [SEP] token at the last of every sentence. The position embedding was added to each token in the sentence for the position. The token embedding and sentence embedding had a similar concept. We were training the BERT model using both strategies, which could minimize the loss function.

3.4.2 Sequence-to-Sequence(Seq-2-Seq)

We trained the seq-2-seq model[22], which took the context(paragraph) sentence as the input sequence to the encoder and then used the end hidden state of the encoder for initializing the decoder. We used the (Pennington et al.,) GloVe embeddings (300 dimension – glove.840B.300d [24]pre-trained) to translate the words into the tokens and give them to the network. We used approximately 45k frequent tokens on the source side and approximately 28k frequent tokens on the target side vocabulary. Then other tokens which are not in the vocabulary are replaced by the [UNK]. Then we set the values of the parameters as the long short term memory hidden unit size is 600, and the number of layers is 2 in the encoder-decoder architecture. We optimized suing the SGD(Stochastic Gradient Descent), learning rate as 1. We started doing half of the learning rate at epoch 8. The dropout probability is 0.3, and the batch size was 64. We ran the model for the training for 15 epochs. We did beam search with different beam sizes but then took beam size as three during the decoding process. We had directly used the PyTorch; if we had used OpenNMT[12] on top of PyTorch, we could get better performance.

3.5 Fine-Tuning

During the fine-tuning time, it do not require extra pre-training, and no extra parameters must be trained from scratch. This BERT-QG model was fine-tuned by fine-tuning the (pre-trained) BERT model on the SQuAD v1 dataset. In this model, the inputs were context as C and answer as A, and the output was generated question as Q from the question answering SQuAD dataset. We used the [CLS], [SEP], and [MASK] tokens with the other two feature tokens of [POS] and [NER].

3.6 Evaluation Method

We used BLEU[21], METEOR[7] and ROUGE-L[14] as performance metrics to evaluate the model. These meterics was used to calculated the correspondence between the reference question in the text means the human question and the model's generated question means the machine's output.
BLEU performance metrics computes the modified precision metric using n-grams and compared the resemblance between the reference question of text and the machine generated question. The $\hat{S}$ is the candiate corpus and S is the reference candidate corpus.

$$BLEU = BP.\exp\left\{\sum_{n=1}^{N} w_n \log p_n\right\}$$

w_n = weight of the n-gram similarity, p_n is defined as n-gram similarity and the BP is defined as the penalty for question that are too short in length. BP is the brevity penalty and r is the effective length corpus length and c is the length of candidate corpus. When $r \leqslant c$ the $BP = 1$

that is the long candidates are not given any penalty and only given to short candidates.

$$BP((\hat{S}, S)) = \exp\{-(\tfrac{r}{c} - 1)\}$$
$$(\tfrac{r}{c} - 1) = max(0, \tfrac{r}{c} - 1)$$

METEOR is defined as the harmonic mean of the recall of the generated question by machine and the unigram precision. P denotes to Unigram Precision, R denotes to Unigram Recall anD F_{mean} is the harmonic mean.

$$F_{mean} = \frac{P.R}{(\alpha.P+(1-\alpha).R)}$$

To get the final METEOR score multiply it by the penalty of question(Question that are short).

ROUGE-L is defined as it is used for the longest common sub-sequence problem, it take the similarity between the longest occurring in the reference question in text and the generated question of machine.

Chapter 4

Experimental Evaluation

4.1 Sequence-to-Sequence Model

We had the neural question generation model on the (processed) SQuADv1.0 dataset. We extracted the sentences from the SQuAD dataset, paired them with the questions, and trained the model with the sentence-question pairs. Previously in, the chapter described that the dataset has 536 articles with over 100000 questions present about the articles. There is a hidden section of the original SQuAD dataset that we don't have access to, so we took the 905 datasets as the whole data for our model.

Then we used the Standford CoreNLP for the pre-processing stage, which helped with tokenizing and sentence splitting, as the dataset contains some uppercase, so we converted the entire dataset in the lower case. There is the answer to each question in the dataset, so we located the sentence which have the answer and used it as the input sentence. Less than the 0.16% in the training set, the answer spanned more than two or two sentences, and then we used the concatenation of the sentences as the input. We added to the beginning of the sentences and the end of the sentences.

We divided the dataset for the article-level into three sets training set (0.8 of the complete dataset), the valid set(0.1 of the complete dataset), and the test set (0.1 of the complete dataset). There are approximately 70000 training samples; valid samples are approx. 10000 and test samples are approximately 11000. These statistics are of the processed data.
We implemented the model in Torch11, and we tried to install it on top of the released onmt (OpenNMT[12]) system but didn't get complete success in it, as mentioned in the paper. In the source side vocabulary, we only kept 45000 most frequent tokens and 28000 most frequent tokens on the target side vocabulary. Other than the most frequent tokens in the vocabulary list, all the tokens were replaced by the [UNK] token symbol.
We chose the 300 dimension embedding of the glove.840B.300d pre-trained embeddings for the initialization. The training parameters were set as – number of layers to 2 in both encoder and decoder and hidden unit size to 600. Firstly we tried to optimize using the stochastic gradient descent(SGD) with an starting learning rate of 1.0, batch size of 64, and epoch 8. But we didn't get the required result for the model. So trying different values, we set the learning rate to 0.00005, batch-size to 32, and epoch 15. The dropout probability of 0.3 was the same while trying to get better results by changing the parameter's value. The model trained on a single GPU took approx. 1.8 hours to get trained. We selected the model that achieved the lowest perplexity.

During the decoding process, we took different beam sizes to see the varying result, but after seeing the results, we found that changing the beam size was not reflecting a major change in the result. After the 7 beam size, there was no change in the model results. We took beam size of 3 for the beam search. The decoding process stopped when every beam in the stack generated the token.
We used the repository to evaluate and calculate the metrics of the model and not the evaluation package released by Chen et al. [4], which was initially used to score the captions of the image. We were planning for the human evaluation after getting the model's best result.

4.2 Bert-QG

In the Bert-QG model, we implemented the base model, which mainly adopts the model architecture from the state-of-the-art Zhao et al. [32]. We used the SQuADv1 dataset for the model experiment. The difference was that we added two language features, POS and NER, applied the deep contextualized word vectors, and tied the output projection matrix with the word embedding matrix. Adding these features improved the results of the model. The model concatenated four-word representation in the embedding process – part-of-speech tag embedding, name entity embedding, answer tag embedding, and the word vector. Then the embedding layer's output is encoded by two-layer bidirectional LSTM-RNN, resulting in the list of hidden representations. The gated self-attention mechanism is applied to hidden representations to aggregate the long-term dependency with the paragraph. The attention mechanism aggregates the hidden vector at each decoding step to the context(paragraph) vector, which is then used to update the decoder state.

We implemented the model by taking the baseline of the BERT model from the official repository and fine-tuning it with our own set of hyper parameters and the preprocessed SQuAD dataset. We used the (pre-trained) BERT model in the question generation work to generate the question from the given paragraph(context) and answer as the input. We had used the BERT base model(uncased), naming it as bert-base-uncased, which is the pre-trained model, for further BERT-QG model. The input sequence, that is, the context and answer token sequence had the POS and NER tags and also consisted of the [CLS] classification token, [SEP] separator token, and the [MASK] mask token. This whole sequence was given as the input to the model to generate the required question tokens sequence. The BERT model has 24 models with different layers and hidden state sizes. We used the BERT model, which has 768 hidden state sizes, 12 layers, 12 attention heads, and 110M parameters.
In this model, we took the hidden state of maximum sequence length 768 and output length of 768 with the best checkpoint of 12000. The

parameters set for the training were the batch size as 32, size of hidden layer 768, not training the word embedding along, with a beam size for beam search as 7, in the training process not doing the multinomial sample, with the period of 100 to save the batch loss and 1000 checkpoints to save and evaluate the model by comparing the best bleu score from the previous step. The dropout probability across the layers was 0.3, and the learning rate was 0.00005. At the less number of checkpoints, we didn't get an improved score for the model. By changing the parameters during the training, we found that in this model also, when we were changing the beam size to more than 7, it was not giving any change in the result, and in the case of below seven, there was little difference in the result. The significant changes in results were due to changing the learning rate parameter and the batch size. We are trying to improve the model score by training the model on more combinations of modified parameters and increasing the checkpoints during the training process. We applied three evaluation metrics to deal with the exposure bias issue and try to improve performance.

4.3 Result

We had used the deep learning models in our experiments. We used two models: the sequence-to-sequence model and the bert-qg model. Before experimenting, we split the dataset to create the training data and test the dataset to evaluate our model. The model took the paragraph(context) and answer as the input and generates the question as to the output. After experimenting with the models, we calculated the question generation performance of our approach. We reported the performance of the model on the basis of the BLEU score, METEOR score, and ROUGE-L score.

Train	70484
Valid	10570
Test	11877

4.3.1 BERT-QG Model result

In the BERT-QG model, we got the BLEU score of 18.768 with the best checkpoint of 12000. The parameters set for the training during this score were with the batch size of 32, size of hidden layer of 600, not training the word embedding along, for beams search having beam size of 7, in the training not doing the multinomial sample, with the period of 100 to save the batch loss and 1000 checkpoints to save and evaluate the model by comparing the best bleu score from the previous step. The dropout probability across the layers was 0.3, and the learning rate was 0.00005.
We also compared testing data between the beam size and bleu score and found that after the beam size was 5, there was no change in the bleu score. We compared and found that till 5, there is either little or more change in the beam size of 1, 3, and 5. With 7 and 11, we got a similar bleu score. It was just a try to increase the bleu score during the testing process. Like with beam size of 1 we got the Scores as – BLEU score = 13.564, METEOR score = 20.606 , ROUGE-L score = 42.622 ; beam size of 3 we got scores as - BLEU score = 14.411, METEOR score = 20.619, ROUGE-L score = 42.584; and beam size of 5 we got scores as - BLEU score = 14.46, METEOR score = 20.60, ROUGE-L score = 42.622. After these, we saw the beam size as 7 and 11, which does not made much change in the scores.

Model	Bleu4
BERT-QG Model (best checkpoint-12000)	18.768

In the model proposed in paper [2] BERT-SQG model have a BLEU score of 20.11 and the result on reproducing the same model we got the score of 18.768.

Seq2Seq Model	Bleu Score
BERT-SQG (Ying, Yao) [2]	20.11
Model (reproduced)	18.768

Model	Beam Size	Bleu4	METEOR	Rougue-L
BERT-QG Model (Test1)	1	13.564	20.606	42.622
BERT-QG Model (Test2)	3	14.411	20.619	42.584
BERT-QG Model (Test2)	5	14.46	20.60	42.622

4.3.2 Sequence-to-sequence(Seq-2-Seq) Model result

In the Seq-2-Seq model, we got the BLEU-1 score of 20.035, but for BLEU-2, BLEU-3, and BLEU-4, we got a very low score with the parameters the batch size of 64 and learning rate of 1.0. As we changed the parameters to batch size of 32 and the learning rate to 0.00005 the BLEU score increased to 22.20985 and we got BLEU-2 is 8.099, BLEU-3 score was 4.125, and BLEU-4 score was 2.328. We were trying to improve that by changing the model's hyper-parameters and improving the scores. We also got the ROUGE-L score as 24.23170 with the parameters as batch size of 32, size of hidden layer of 600, and the number of layers is two. We took the learning rate of 0.00005, the number of epochs to 15, and the drop probability of 0.3. With these parameters, we got the values.

Seq2Seq Model	Batch Size	Learning Rate
Seq2Seq Model 1	64	1.0
Seq2Seq Model 2	32	0.00005

In the model proposed in paper [22] Sequence-to-Sequence model have a BLEU score of 4.26 and the result on reproducing the same model we got the score of 2.328 and 1.163.

Seq2Seq Model	Bleu Score
Seq2Seq Model (Du et al.) [22]	4.26
Seq2Seq Model 1	2.328
Seq2Seq Model 2	1.16330

Seq2Seq Model	Bleu1	Bleu2	Bleu3	Bleu4	Rouge-L
Seq2Seq Model 1	20.035	8.099	4.125	2.328	21.7893
Seq2Seq Model 2	22.20985	6.70345	2.47593	1.16330	24.23170

Chapter 5

Conclusion and Future Work

In this thesis, we implemented two models for the question generation work – The seq-2-seq model and the BERT-QG model. We had made a neural network approach for the question generation task sequence-to-sequence model. We had used an attention-based method for question generation. We had seen the automatic evaluations for sentence-level and paragraph-level both and found out that it did not give the best result at paragraph-level.
We also examined the transformer-based model such as BERT and looked at the generation quality through metrics like BLEU, METEOR, and ROUGE-L scores. We fine-tuned the model on the SQuAD dataset. We made attempts to refine the model by position embedding.
The computational resources for the training of model was limited so we used the (pre-trained) models available on the internet for our approach and work. Due to this limitation we considered the BERT for our work, as BERT has multi size pre-trained models. We were unable to test all the possible training hyper-parameters, as it was taking more training timing.
In future work in the Sequence-to-Sequence model, we can explore in what way to use paragraph-level to improve the system's performance regarding the generation of different types of questions. This model can be more studied for the language generation, like the dialogue gen-

eration, to improve the quality of the output. In the BERT model, different sizes of the word embeddings, pretraining on the big corpus, or using various other features during pre-processing the data. We used the beam search strategy; we can also use the BULB strategy, which might improve the accuracy during the decoded process.

Bibliography

[1] Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)

[2] Chan, Y.H., Fan, Y.C.: A recurrent BERT-based model for question generation. In: Proceedings of the 2nd Workshop on Machine Reading for Question Answering. pp. 154–162. Association for Computational Linguistics, Hong Kong, China (Nov 2019), https://aclanthology.org/D19-5821

[3] Chen, D., Bolton, J., Manning, C.D.: A thorough examination of the cnn/daily mail reading comprehension task. CoRR abs/1606.02858 (2016), http://arxiv.org/abs/1606.02858

[4] Chen, X., Fang, H., Lin, T., Vedantam, R., Gupta, S., Dollár, P., Zitnick, C.L.: Microsoft COCO captions: Data collection and evaluation server. CoRR abs/1504.00325 (2015), http://arxiv.org/abs/1504.00325

[5] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (Oct 2014), https://aclanthology.org/D14-1179

[6] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using

RNN encoder-decoder for statistical machine translation. CoRR abs/1406.1078 (2014), http://arxiv.org/abs/1406.1078

[7] Denkowski, M., Lavie, A.: Meteor universal: Language specific translation evaluation for any target language. In: Proceedings of the Ninth Workshop on Statistical Machine Translation. pp. 376–380. Association for Computational Linguistics, Baltimore, Maryland, USA (Jun 2014), https://aclanthology.org/W14-3348

[8] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), https://aclanthology.org/N19-1423

[9] Duan, N., Tang, D., Chen, P., Zhou, M.: Question generation for question answering. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 866–874. Association for Computational Linguistics, Copenhagen, Denmark (Sep 2017), https://aclanthology.org/D17-1090

[10] Heilman, M., Smith, N.A.: 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. p. 609–617. HLT '10, Association for Computational Linguistics, USA (2010)

[11] Kim, Y., Lee, H., Shin, J., Jung, K.: Improving neural question generation using answer separation. CoRR abs/1809.02393 (2018), http://arxiv.org/abs/1809.02393

[12] Klein, G., Kim, Y., Deng, Y., Senellart, J., Rush, A.: OpenNMT: Open-source toolkit for neural machine translation. In: Proceedings of ACL 2017, System Demonstrations. pp. 67–72. Associa-

tion for Computational Linguistics, Vancouver, Canada (Jul 2017), https://aclanthology.org/P17-4012

[13] Labutov, I., Basu, S., Vanderwende, L.: Deep questions without deep understanding. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 889–898. Association for Computational Linguistics, Beijing, China (Jul 2015), https://aclanthology.org/P15-1086

[14] Lin, C.Y.: ROUGE: A package for automatic evaluation of summaries. In: Text Summarization Branches Out. pp. 74–81. Association for Computational Linguistics, Barcelona, Spain (Jul 2004), https://aclanthology.org/W04-1013

[15] Lindberg, D., Popowich, F., Nesbit, J., Winne, P.: Generating natural language questions to support learning on-line. In: Proceedings of the 14th European Workshop on Natural Language Generation. pp. 105–114. Association for Computational Linguistics, Sofia, Bulgaria (Aug 2013), https://aclanthology.org/W13-2114

[16] Lindberg, D., Popowich, F., Nesbit, J., Winne, P.: Generating natural language questions to support learning on-line. In: Proceedings of the 14th European Workshop on Natural Language Generation. pp. 105–114. Association for Computational Linguistics, Sofia, Bulgaria (Aug 2013), https://aclanthology.org/W13-2114

[17] Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1412–1421. Association for Computational Linguistics, Lisbon, Portugal (Sep 2015), https://aclanthology.org/D15-1166

[18] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. pp. 55–60. Association for Computational Linguistics, Baltimore, Maryland (Jun 2014), `https://aclanthology.org/P14-5010`

[19] Mazidi, K., Nielsen, R.D.: Linguistic considerations in automatic question generation. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 321–326. Association for Computational Linguistics, Baltimore, Maryland (Jun 2014), `https://aclanthology.org/P14-2053`

[20] Mitkov, R., Ha, L.A.: Computer-aided generation of multiple-choice tests. In: Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing. pp. 17–22 (2003), `https://aclanthology.org/W03-0203`

[21] Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: A method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. p. 311–318. ACL '02, Association for Computational Linguistics, USA (2002), `https://doi.org/10.3115/1073083.1073135`

[22] Pennington, J., Socher, R., Manning, C.: GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543. Association for Computational Linguistics, Doha, Qatar (Oct 2014), `https://aclanthology.org/D14-1162`

[23] Pennington, J., Socher, R., Manning, C.: GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543. Association for Computational Linguistics, Doha, Qatar (Oct 2014), `https://aclanthology.org/D14-1162`

[24] Pennington, J., Socher, R., Manning, C.: GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543. Association for Computational Linguistics, Doha, Qatar (Oct 2014), https://aclanthology.org/D14-1162

[25] Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100, 000+ questions for machine comprehension of text. CoRR abs/1606.05250 (2016), http://arxiv.org/abs/1606.05250

[26] Song, L., Wang, Z., Hamza, W.: A unified query-based generative model for question generation and question answering. CoRR abs/1709.01058 (2017), http://arxiv.org/abs/1709.01058

[27] Subramanian, S., Wang, T., Yuan, X., Trischler, A.: Neural models for key phrase detection and question generation. CoRR abs/1706.04560 (2017), http://arxiv.org/abs/1706.04560

[28] Sun, X., Liu, J., Lyu, Y., He, W., Ma, Y., Wang, S.: Answer-focused and position-aware neural question generation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 3930–3939. Association for Computational Linguistics, Brussels, Belgium (Oct-Nov 2018), https://aclanthology.org/D18-1427

[29] Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. CoRR abs/1409.3215 (2014), http://arxiv.org/abs/1409.3215

[30] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. CoRR abs/1706.03762 (2017), http://arxiv.org/abs/1706.03762

[31] Yao, X., Bouma, G., Zhang, Y., Piwek, P., Boyer, K.: Semantics-based question generation and implementation. Dialogue Discourse 3, 11–42 (03 2012)

[32] Zhao, Y., Ni, X., Ding, Y., Ke, Q.: 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. pp. 3901–3910. Association for Computational Linguistics, Brussels, Belgium (Oct-Nov 2018), https://aclanthology.org/D18-1424

Question Generation on Text Data using NLP

ORIGINALITY REPORT

16%
SIMILARITY INDEX

PRIMARY SOURCES

1	aclanthology.org Internet	127 words — 2%
2	www.aclweb.org Internet	126 words — 2%
3	blog.csdn.net Internet	59 words — 1%
4	Nair, Rathin Radhakrishnan. "Metadata Analysis in Unstructured Documents Using Classical and Deep Learning Methods.", State University of New York at Buffalo, 2018 ProQuest	53 words — 1%
5	arxiv.org Internet	52 words — 1%
6	dokumen.pub Internet	49 words — 1%
7	"Complex, Intelligent and Software Intensive Systems", Springer Science and Business Media LLC, 2021 Crossref	48 words — 1%
8	ebin.pub Internet	45 words — 1%

9	hdl.handle.net Internet	43 words — 1%
10	link.springer.com Internet	28 words — < 1%
11	upcommons.upc.edu Internet	28 words — < 1%
12	Aritra Kumar Lahiri, Qinmin Vivian Hu. "Chapter 21 GameOfThronesQA: Answer-Aware Question-Answer Pairs for TV Series", Springer Science and Business Media LLC, 2022 Crossref	27 words — < 1%
13	www.mdpi.com Internet	27 words — < 1%
14	www.statmt.org Internet	26 words — < 1%
15	"Natural Language Processing and Chinese Computing", Springer Science and Business Media LLC, 2019 Crossref	22 words — < 1%
16	"ECAI 2020", IOS Press, 2020 Crossref	18 words — < 1%
17	B. Senthil Kumar, D. Thenmozhi, S. Kayalvizhi. "Chapter 3 Tamil Paraphrase Detection Using Encoder-Decoder Neural Networks", Springer Science and Business Media LLC, 2020 Crossref	18 words — < 1%
18	Lecture Notes in Computer Science, 2015. Crossref	18 words — < 1%

19	www.arxiv-vanity.com Internet	18 words — < 1%
20	"Natural Language Processing and Chinese Computing", Springer Science and Business Media LLC, 2018 Crossref	17 words — < 1%
21	mafiadoc.com Internet	17 words — < 1%
22	"Chinese Computational Linguistics", Springer Science and Business Media LLC, 2019 Crossref	16 words — < 1%
23	"Text, Speech, and Dialogue", Springer Science and Business Media LLC, 2021 Crossref	16 words — < 1%
24	d2l.ai Internet	16 words — < 1%
25	ojs.aaai.org Internet	16 words — < 1%
26	"Intelligent Systems", Springer Science and Business Media LLC, 2020 Crossref	14 words — < 1%
27	Ruan, Sherry Shanshan. "Smart Tutoring Through Conversational Interfaces.", Stanford University, 2021 ProQuest	14 words — < 1%
28	Alibadi, Zaid. "Detecting the Intent of Email Using Embeddings, Deep Learning, and Transfer Learning", University of South Carolina, 2021 ProQuest	13 words — < 1%

29	openaccess.city.ac.uk Internet	13 words — < 1%
30	lxie.nwpu-aslp.org Internet	12 words — < 1%
31	openaccess.thecvf.com Internet	12 words — < 1%
32	spectrum.library.concordia.ca Internet	12 words — < 1%
33	Jiaxin Hu, Zhixu Li, Renshou Wu, Hongling Wang, An Liu, Jiajie Xu, Pengpeng Zhao, Lei Zhao. "Distant Supervised Why-Question Generation with Passage Self-Matching Attention", 2019 International Joint Conference on Neural Networks (IJCNN), 2019 Crossref	11 words — < 1%
34	mdpi-res.com Internet	10 words — < 1%
35	Álvaro Peris, Marc Bolaños, Petia Radeva, Francisco Casacuberta. "Chapter 1 Video Description Using Bidirectional Recurrent Neural Networks", Springer Nature, 2016 Crossref	10 words — < 1%
36	"Advances in Big Data and Cloud Computing", Springer Science and Business Media LLC, 2019 Crossref	9 words — < 1%
37	"Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track", Springer Science and Business Media LLC, 2021 Crossref	9 words — < 1%

38	"PRICAI 2018: Trends in Artificial Intelligence", Springer Science and Business Media LLC, 2018 Crossref	9 words — < 1%
39	babel.hathitrust.org Internet	9 words — < 1%
40	books.openedition.org Internet	9 words — < 1%
41	borea17.github.io Internet	9 words — < 1%
42	docshare.tips Internet	9 words — < 1%
43	icon2021.nits.ac.in Internet	9 words — < 1%
44	poleval.pl Internet	9 words — < 1%
45	www.comsis.org Internet	9 words — < 1%
46	www.ecva.net Internet	9 words — < 1%
47	yorkspace.library.yorku.ca Internet	9 words — < 1%
48	"17th International Conference on Information Technology–New Generations (ITNG 2020)", Springer Science and Business Media LLC, 2020 Crossref	8 words — < 1%

49 "Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data", Springer Science and Business Media LLC, 2017 — 8 words — < 1%
Crossref

50 "Computational Processing of the Portuguese Language", Springer Science and Business Media LLC, 2018 — 8 words — < 1%
Crossref

51 "Information Retrieval", Springer Science and Business Media LLC, 2020 — 8 words — < 1%
Crossref

52 Alana de Santana Correia, Esther Luna Colombini. "Attention, please! A survey of neural attention models in deep learning", Artificial Intelligence Review, 2022 — 8 words — < 1%
Crossref

53 Baghaee, Tina. "Automatic Neural Question Generation Using Community-Based Question Answering Systems.", University of Lethbridge (Canada), 2018 — 8 words — < 1%
ProQuest

54 Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, Yu Xu. "Learning to Generate Questions by LearningWhat not to Generate", The World Wide Web Conference on - WWW '19, 2019 — 8 words — < 1%
Crossref

55 Choudhury, Sourav Roy. "Deep Natural Language Generation Using BERT for Summarization", The University of North Carolina at Charlotte, 2021 — 8 words — < 1%
ProQuest

56	Natasha Gandhi, Wanling Zou, Caroline Meyer, Sudeep Bhatia, Lukasz Walasek. "Computational Methods for Predicting and Understanding Food Judgment", Psychological Science, 2022 Crossref	8 words — < 1%
57	Tseng, Ya-Min, Yi-Ting Huang, Meng Chang Chen, and Yeali S. Sun. "Generating Comprehension Questions Using Paraphrase", Lecture Notes in Computer Science, 2014. Crossref	8 words — < 1%
58	Xu Chen, Jungang Xu. "An Answer Driven Model For Paragraph-level Question Generation", 2021 International Joint Conference on Neural Networks (IJCNN), 2021 Crossref	8 words — < 1%
59	Yanmeng Wang, Wenge Rong, Jianfei Zhang, Shijie Zhou, Zhang Xiong. "Multi-turn dialogue-oriented pretrained question generation model", Complex & Intelligent Systems, 2020 Crossref	8 words — < 1%
60	krex.k-state.edu Internet	8 words — < 1%
61	medinform.jmir.org Internet	8 words — < 1%
62	opus.uleth.ca Internet	8 words — < 1%
63	res.mdpi.com Internet	8 words — < 1%
64	uwspace.uwaterloo.ca	

Internet

8 words — < 1%

65 www.inf.uniri.hr
Internet

8 words — < 1%

66 www.qt21.eu
Internet

8 words — < 1%

67 Bang Liu, Haojie Wei, Di Niu, Haolan Chen, Yancheng He. "Asking Questions the Human Way: Scalable Question-Answer Generation from Text Corpus", Proceedings of The Web Conference 2020, 2020
Crossref

7 words — < 1%

68 Rohit Agrawal, Ajjen Joshi, Margrit Betke. "Enabling Early Gesture Recognition by Motion Augmentation", Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference on - PETRA '18, 2018
Crossref

7 words — < 1%

69 "Advances in Artificial Intelligence", Springer Science and Business Media LLC, 2020
Crossref

6 words — < 1%

70 "Chinese Computational Linguistics", Springer Science and Business Media LLC, 2021
Crossref

6 words — < 1%

71 "Information Retrieval", Springer Science and Business Media LLC, 2018
Crossref

6 words — < 1%

72 "PRICAI 2021: Trends in Artificial Intelligence", Springer Science and Business Media LLC, 2021
Crossref

6 words — < 1%

73	"The Semantic Web – ISWC 2019", Springer Science and Business Media LLC, 2019 Crossref	6 words — < 1%
74	Dongxiao Han, Juan Chen, Jian Sun. "A parallel spatiotemporal deep learning network for highway traffic flow forecasting", International Journal of Distributed Sensor Networks, 2019 Crossref	6 words — < 1%
75	Lamba, Deepti. "Deep Learning with Constraints for Answer-Agnostic Question Generation in Legal Text Understanding", Kansas State University, 2021 ProQuest	6 words — < 1%
76	etheses.whiterose.ac.uk Internet	6 words — < 1%

EXCLUDE QUOTES	ON	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	ON	EXCLUDE MATCHES	OFF