International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 11 | Nov 2023 www.irjet.net p-ISSN: 2395-0072

AUTOMATIC QUESTION GENERATION USING NATURAL LANGUAGE PROCESSING

Mohmmad Yasir Khan, Lincy Rebello, Trishali Rao, Nitika Rai

[123]*UG student, Dept. of Information Technology, St. Francis institute of Technology*
[4] *Associate Professor, Dept. of Information Technology, St. Francis institute of Technology*

Abstract **- Natural language processing (NLP) is an upcoming field that encompasses computer science alone with artificial intelligence (AI) along with spoken language and is used to carry out a diverse range of applications such as sentimental analysis, speech recognition, content categorization, and analysis, to name a few. One of the domains where NLP can aid the educational system is a comprehensive question generator that can form unbiased questions from a given subject matter, which is otherwise a tedious and time-consuming task for instructors. Our proposed work focuses on developing an automatic question generator (AQG) using NLP that can generate a set of questions from a given paragraph of text. The approach used is test-to-text transformer-based transfer (T5) learning models, which can emphasize the critical points of a sentence and the language structure to generate questions without the creator inputting profound grammar rules. The proposed system can also assist students in learning and then test their understanding.**

Keywords - ***T5 Text-to-Text Transfer Transformer, Question Generation, Natural Language Processing, UniLM-Unified Pre-training for Language Understanding and Generation, SQuAD-Stanford Question Answering Dataset***

1.INTRODUCTION

The process of producing syntactically sound, semantically valid, and relevant questions from various input forms such as a block of text, a structured database, or a knowledge base is known as automatic question generation (AQG). The purpose of question generation is to produce legitimate, meaningful and fluent questions based on a supplied paragraph that has a discrete target answer.

Automatic Question Generation can be applied to a variety of domains and for a plethora of use cases, including autonomous tutoring systems, increasing the effectiveness of question-answering (QA) models, allowing chatbots to lead a conversation, and many more. Regardless of its versatility, manually framing and writing meaningful and relevant questions is a tedious, time-consuming and difficult process. For instance, while testing students *learning* through reading comprehension, the evaluator must manually frame suitable questions, prepare response sheet and then evaluate the responses accordingly. This exercise can at times be biased and not thorough if the instructor does not pay the needed attention to this otherwise critical aspect in the teaching learning process.

This paper focuses on test-to-text transformer-based transfer (T5) learning models, which can emphasize the critical points of a sentence and the language structure to generate questions without the creator inputting profound grammar rules. The T5 AQG model is trained on the Stanford Question Answering Dataset (SQuAD) version 2.0 [1]

2.LITERATURE REVIEW

Various approaches to implement automatic question generation have been reported in the literature. This section presents a few of the significant ones.

One of the approaches reported is based on the concept of answer recognition question generation. In answer-aware question generation, the model is presented with an answer and a passage of script and is asked to generate a question for that answer given the context of the passage. There is enough literature available on AQG approaches, but they are not as widely used as QA. One of the reasons is that most of the available literature employs complex models/processing pipelines and no pre-trained models are available. Some of the recent works, notably UniLM and ProphetNet, have reported the use of SOTA pre-trained weights that can be used in QG, but using them seems overly complicated [1].

Another approach reported focuses on transformer-based finetuning techniques that can be used to create robust question generation systems using only a single pre-trained language model, without the use of additional mechanisms, or answer metadata, which has extensive features [2]. This work also analyses the effect of various factors the affect the performance of the model, such as input data formatting, the length of the context paragraphs, and the use of answer awareness. Additionally, the failure modes of the model are explored along with identification of possible reasons of failure of the model. The question generation model is trained on version 1.1 of the Stanford Question Answering Dataset (SQuAD). The SQuAD contains context passages, each with corresponding sets of questions and responses related to the content of the passages. SQuAD contains more than 100,000 community-provided questions.

The model reported is smaller, less complex, and faster to operate, making it a possible alternative for a variety of use cases related to question generation. The paper demonstrates that a simple single transformer-based question generation model is able to outperform more complex Seq2Seq methods without the need for additional features, techniques, and training steps. They have evaluated the model on BLEU 1, BLEU 2, BLEU 3, BLEU 4, ROUGE L, and METEOR. The evaluation package is used to quantify the performance of the model. The flaw identified in the model is that the values selected are not good in terms of BLEU 4 and ROUGE L, and slightly worse in terms of METEOR.

In [3], a fully data-driven neural network approach to automatic question generation for reading comprehension is presented. They used an attention-based neural networks approach for the task and investigated the effect of encoding sentence- vs. paragraph-level information. The paper reports both automatic evaluations and human evaluations. It incorporates an into-operated SQuAD dataset wherein the questions are posed by crowd workers and are of relatively high quality. A sequence-to-sequence approach is used which is a basic encoder-decoder sequence learning system for machine translation which is implemented in TensorFlow. The models using RNN encoder-decoder architecture with two variations are investigated where the first one only encodes the sentence and the other encodes both sentence and paragraph-level information. For training and inferencing/predicting the output, beam search is used. For implementation, the models are blended in Torch7 with OpenNMT. A comparison of different baselines like IR, Seq2seq, H&S, etc. with packages/systems viz. Bleu1-4, Meteor, and Rogue is also carried out. Moreover, performance studies were done where the sample output questions were generated by humans, by their system, and by the H&S system (rule-based over-generate and rank system). It was reported that the resulting model could generate better-quality questions than the H&S system. The flaw identified in the model proposed is that it encodes only sentence-level information and achieves the best performance across all the metrics but not so in paragraph-level information.

In [4], the authors propose a novel neural network model with better-encoding co-reference knowledge for paragraph-level question generation, and research studies methods that incorporate correlation information into the training of a question-generation system. Specifically, the paper proposes a closed reference knowledge for neural question generation (CorefNNG), a neural chain model with a novel gate generation mechanism that exploits continuous representations of reference clusters - a set of mentions used to refer to each entity - for better language coding. the knowledge introduced by the reference, to generate passage-level questions. Evaluations with various metrics on the SQuAD autoplay dataset show that its model outperforms leading baselines. The resection study showed the efficacy of different components in our model. Finally, the proposed question generation framework is applied to produce a corpus of 1.26 million question-answer pairs.

In [5], the authors propose a system that automatically generates a quiz on the basis of the learning material provided. This can be used by both instructors and learners to be able to recall and apply major concepts from the study material. The proposed system focuses on generating educationally relevant gap-fill multiple choice questions from educational texts in three ways, viz. (a) Sentence Selection: Selecting coherent and important sentences from the text to ask about (b) Gap Selection: Identity which part of the resulting sentence to choose as the gap. The gap essentially represents the concept being tested (c) Distractor Selection: Crafting effective distractors to be part of the set of options to confuse the learner to ensure that he has a good grasp of the concept being tested. Preliminary testing was also done with students. First, the quality of the questions generated was assessed. It was reported that 94% of the question sentences, 87% of the gaps, and 60% of the distractors were considered to be educationally relevant. Secondly, they found that 14 out of the 15 students surveyed found the system to be effective in helping them to reinforce concepts.

3. PROPOSED METHODOLOGY

3.1 T5 - Text-To-Text Transfer Transformer

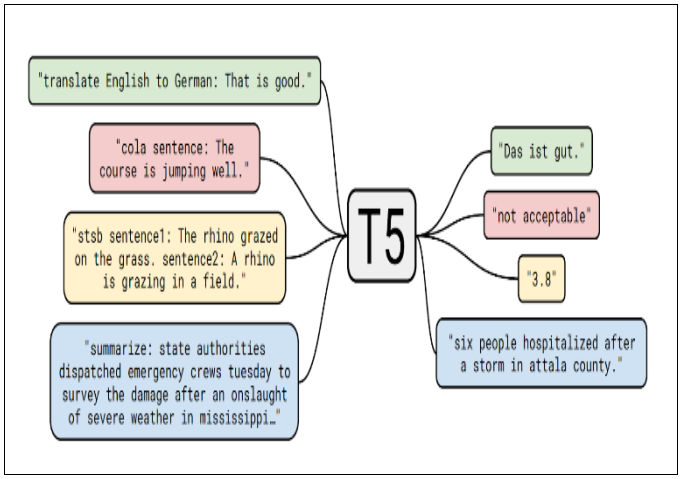

Fig. 1. T5: Text-to-Text Framework [6]

Our proposed work employs T5: Text-to-Text Framework (Fig. 1), where a model is first pre-trained on a data-rich task before being tuned for downstream tasks, T5 is an encoder/decoder that transforms an NLP based problem into text-to-text form, ie. there is always a need for an input sequence and a corresponding target sequence for training. T5 works well with different tasks by adding different prefixes to the inputs corresponding to each task, e.g., for

translation: translate English to German; for summarization: summarize. T5 introduces "Colossal Cleaned Crawled Corpus" (C4), a dataset consisting of hundreds of gigabytes of clean English text scraped from the Internet. Transformers are new models that overpowered recurrent neural networks (RNNs). The primary building block of the transformers is self-attention which processes a sequence with a weighted average of the rest of the sequence. The architecture is not novel and resembles the original transformer's encoder-decoder architecture. Encoder models are designed to produce a single prediction per input token or a single prediction for an entire input sequence. This makes them applicable for classification tasks but not for generative summarization. The decoder-only model can take an input sequence and generate the rest of the text based on the input sequence. So, it is well-suited to fill up an input prompt but not well-suited for rewriting the input sequence into something else. T5 takes the best of words and uses the encoder-decoder architecture to perform any sequence-to-sequence tasks effectively [6].

Fig. 2 shows an example of T5 model functioning. Here, the words "organized", and" theme" (marked with an ×) are randomly chosen for corruption. Each consecutive span of corrupted tokens is replaced by a sentinel token (shown as <X> and <Y>) that is unique over the sentence. The goal is to conceal consecutive token spans and estimate only tokens that were overlooked during the pretraining process.

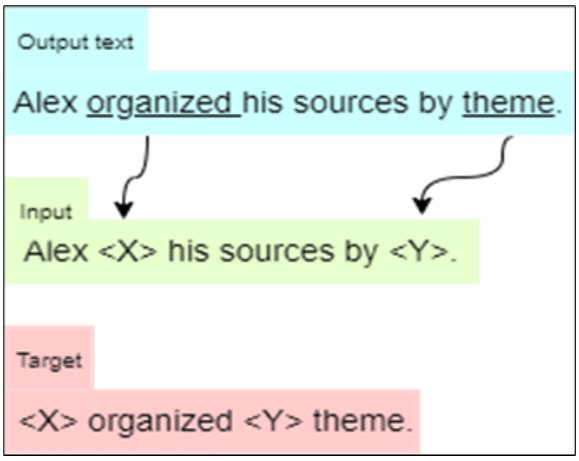

Fig. 2. Example of T5 Model Functioning

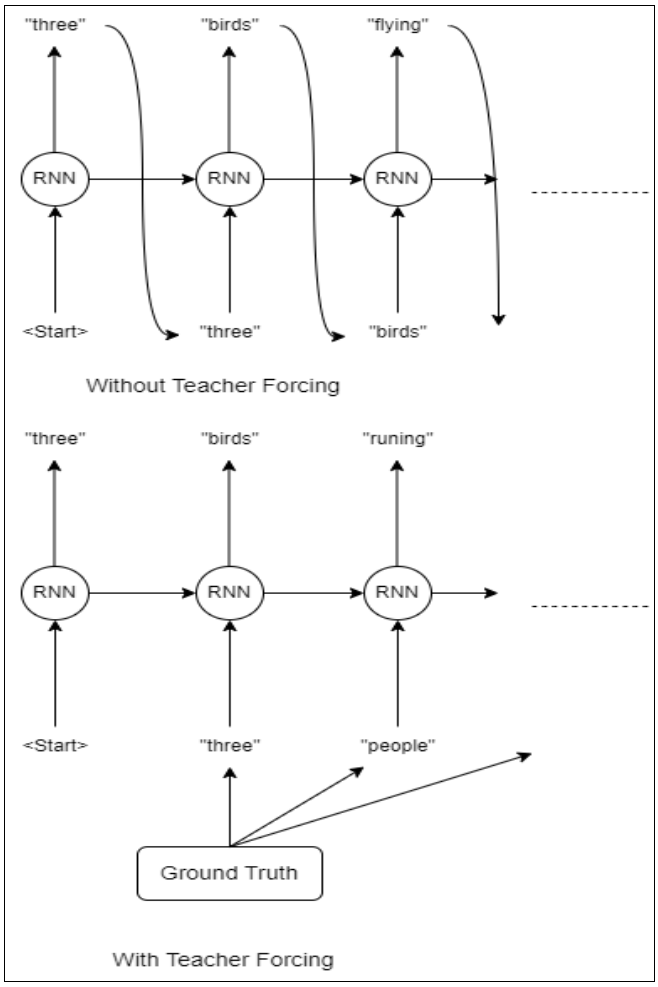

Fig. 3. Teacher Enforcing

Further, the primary building block of the transformer is self-attention, which processes a sequence by replacing each element with a weighted average of the rest of the sequence. T5 is trained using Teacher Enforcing and cross-entropy loss (Fig 3). A denoising (masked language modelling) objective is used. During a denoising operation, the model is trained to predict missing or corrupted tokens in the input. At test time, T5 uses beam decoding (i.e., choosing the highest probability word at every timestamp). Teacher enforcing is a technique where a target word is passed as the next input to the decoder. Teacher enforcing works by using the actual or expected output from the training dataset at the current time step y(t) as input in the next time step X(t+1), rather than the output generated by the network. For example, John is going to… (college), even if the generated output is different eg, college we feed the expected target word(school) back to the decoder [7]. Beginning with "The," the algorithm chooses succeeding words in a greedy manner based on their highest probabilities. As a result, the resulting word sequence is ("The," "nice," "guy"), with an overall probability of 0.5 * 0.4 = 0.2. as shown in Fig. 4 [7]

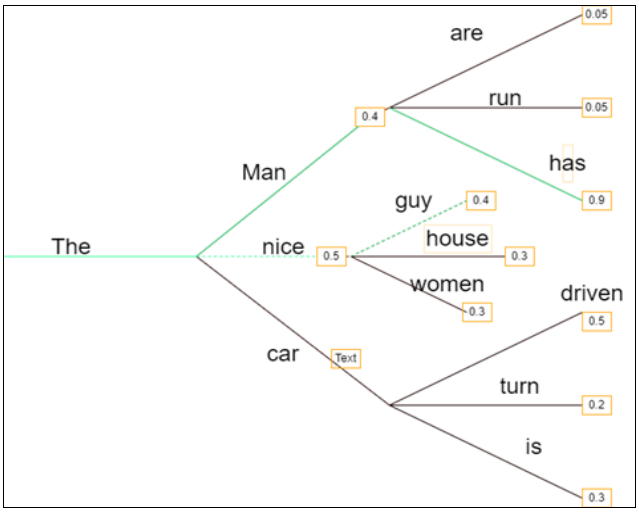

Fig. 4. Traditional beam search

3.2Data Setup

The T5 model is trained on the Stanford Question Answering Dataset (SQuAD). SQuAD is a reading comprehension dataset built by crowd workers using questions based on a collection of Wikipedia articles. Each question requires a response in the form of a text segment, or span, retrieved from the linked reading passage, though some may not have a distinct answer. Squad 1.1 dataset, contains 100,000+ question-answer pairs on 500+ articles.

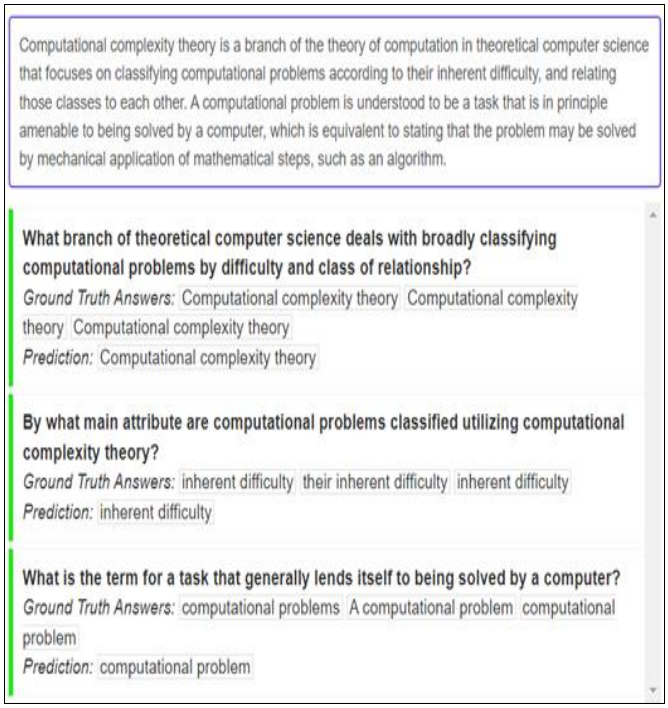

Fig. 5. Squad1.1 Dataset Visualization [9]

However, it selects the first question for each paragraph for training. The context and answer are parsed into model inputs as a continuous body of texts, separated by a special token. Meanwhile, the question is parsed as the targeted value or "labels". The train-test split of 80-20 is used, ie. 80% of the data were used for training, and the remaining 20% were used for validation (Fig. 5).

3.3Understanding T5 Tokenizer

The tokenizer is responsible for preparing the inputs for the model. Tokenize (split a string into sub-word token strings), convert a token string to an ID and back, encode/decode (i.e. tokenize and convert to an integer). Managing special tokens (like a mask, beginning-of-sentence, etc.): Add, assign, and prevent attributes from being split during tokenization in the tokenizer for easy access.

Encoding: Translating text to numbers is known as encoding. Encoding is done in a two-step process: tokenization, followed by the conversion to input IDs. As we have seen, the first step is to split the text into words (or parts of words, punctuation marks, etc.), usually called tokens. The second step is to convert these tokens to numbers so that we can build tensors from them and feed them into our model. Tokenizers have a vocabulary for this.

Decoding: Decoding is going the other way around: from vocabulary indices, we want to get a string. It not only converts the indices back to tokens but also groups the tokens that were part of the same words to produce a readable sentence. The decoding method used is beam search. Beam search maintains the number of beam's most probable hypotheses at each time step and finally selects the hypothesis with the highest overall probability, reducing the risk of missing high-probability hidden word sequences [7].

3.5 Training

Fig. 6 shows the training pipeline of the model. In the training loop, the context, for example, a SQuAD paragraph about the first software the answer ("1948") is concatenated into a single text.

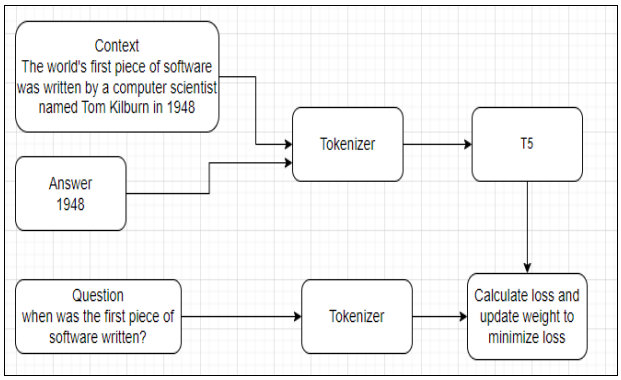

Fig. 6. Question generation by summarizing context the using that text to frame the question

Both context and answer are then passed through a T5 tokenizer, which converts the texts into numerical vectors that the T5 model can understand. Then, T5 will predict the output vectors, which also can be de-tokenized back into text questions. Meanwhile, the given question, "When was the first piece of software written?" is also passed through a tokenizer. The vector representation of the ground-truth question is compared with T5 output vectors using a loss function. Then, T5 will iterate training to update its weights to minimize loss. PyTorch's cross-entropy loss function is used to calculate the loss.

3.5Training Optimization

During training, the number of epochs, batch size, and learning rate are adjusted to ensure the model is well-fitted. The best model has a maximum number of epochs at 100, batch size 8, learning rate 10-6, with AdamW learning rate optimizer and additional L2 regularization at 0.1 to prevent overfitting at large numbers of epochs. The best validation loss for this model is 1.62 with batch size 4, maximum epochs one on a single GPU.

4. RESULTS AND DISCUSSION

Test cases were executed and a comparison was carried out between the questions generated by the model and the questions originally reported in the data set. Two such comparative study is presented below. However, the similarity shall hold good for any given context

4.1 Test Case I:

Context 1: The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries gave their name to Normandy, a region in France. Their ancestors were Norse ("Norman" comes from "Norseman") raiders and pirates from Denmark, Iceland, and Norway. They swore loyalty to King Charles III of West Francia under Rollo's leadership. Over time, their children easily blended with the Carolingian-based civilizations of West Francia through a process of assimilation and intermingling with indigenous Frankish and Roman-Gaulish groups. The Normans' distinct cultural and ethnic identity began to take shape in the early decades of the 10th century, evolving and shifting over the following centuries.

Table 1 presents the comparison between questions generated from the model and the actual questions from the SQuAD dataset for context 1.

Table 1: Comparison between questions generated from the model and the actual questions from the SQuAD dataset for context 1

S. No.	Answer	Questions from the SQuAD Dataset	Question generated by the model
1	France	In what country is Normandy located?	Where is Normandy located?
2	10th and 11th centuries	When were the Normans in Normandy?	When did the Normans give their name to Normandy?
3	Denmark, Iceland and Norway	From which countries did the Norse originate?	Where did the Norse come from?
4	Rollo	Who was the Norse leader?	Who was the leader of the Norse?
5	10th century	What century did the Normans first gain their separate identity?	When did the distinct cultural and ethnic identity of the Normans emerge?

4.2 Test Case 2:

Context 2: He came to power by uniting many of the nomadic tribes of Northeast Asia. After establishing the Mongol Empire and assuming the title "Genghis Khan," he began the Mongol invasions, which resulted in the successful conquest of a large portion of Eurasia. These included raids or invasions of the Qara Khitai, Caucasus, Khwarezmid Empire, and Western Xia and Jin dynasties. These military expeditions frequently coincided with widespread massacres of civilian populations, particularly in Khwarezmian and Xia-controlled areas. By the end of his life, the Mongol Empire had successfully conquered a large portion of both Central Asia and China.

Table 2 presents the comparison between questions generated from the model and the actual questions from the squad dataset for context 2.

Table 2: Comparison between questions generated from the model and the actual questions from the squad dataset for context 2

S. No.	Answer	Questions from the SQuAD Dataset	Question generated by the model
1	Mongol Empire	What do we call the empire that Genghis Khan founded?	What empire did Genghis Khan founded?
2	Nomadic tribes of Northeast Asia	Who did Genghis Khan unite before he began conquering the rest of Eurasia?	What tribes did Genghis Khan unite?
3	Khwarezmian and Xia	In which regions, in particular, did Genghis Khan's armies massacre civilians?	Which two dynasties were particularly targeted by massacres?
4	Central Asia and China	What areas did Genghis Khan control at the end of his life?	What areas did the Mongol Empire occupy by the end of Genghis Khan's life?
5	Qara Khitai, Caucasus, Khwarezmid Empire, Western Xia and Jin	Which other empires or dynasties did Genghis Khan conquer?	What dynasties did Genghis Khan conquer?

A comparative analysis is carried out between the Question-answer pair from the SQuAD dataset with the questions generated from our given model. For each paragraph provided with a single answer, one question is generated by the model. The context above about Normans and Genghis Khan is taken from the SQuAD dataset. The paraphrasing of the questions is almost similar to the questions from the SQuAD dataset. From Table 1, the best matching question was "Who was the leader of the Norse?" and from Table 2, "What dynasties did Genghis Khan conquer?" The T5 model generated cutting-edge results, generating articulate questions based on the provided answers.

5. CONCLUSIONS

Question generation is an essential aspect for various use-cases centred around evaluation methodologies, like testing of student learning in conventional teaching and learning system, autonomous tutoring systems, etc. Manual generation is a tedious and time-consuming process for instructors/ evaluators. It is important that questions framed are unbiased, not human-dependent and comprehensive to cover the entire scope of the context being tested. Hence, the need for an automated system for generation of multiple questions from a given text was felt.

Our work aims to automatically generate multiple questions from a given paragraph of text using NLP. The approach used is test-to-text transformer-based transfer (T5) learning models, which can emphasize the critical points of a sentence and the language structure to generate questions without the creator inputting profound grammar rules. The T5 AQG model is trained on the Stanford Question Answering Dataset (SQuAD) version 2.0.

REFERENCES

[1] Dong, Li, et al. "Unified language model pre-training for natural language understanding and generation." Advances in neural information processing systems 32 (2019).

[2] Lopez, Luis Enrico, et al. "Simplifying paragraph-level question generation via transformer language models." PRICAI 2021: Trends in Artificial Intelligence: 18th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2021, Hanoi, Vietnam, November 8–12, 2021, Proceedings, Part II 18. Springer International Publishing, 2021.

[3] Du, Xinya, Junru Shao, and Claire Cardie. "Learning to ask: Neural question generation for reading comprehension." arXiv preprint arXiv:1705.00106 (2017).

[4] Du, Xinya, and Claire Cardie. "Harvesting paragraph-level question-answer pairs from wikipedia." arXiv preprint arXiv:1805.05942 (2018).

[5] Kumar, Girish, Rafael E. Banchs, and Luis Fernando D'Haro. "Automatic fill-the-blank question generator for student self-assessment." 2015 IEEE Frontiers in Education Conference (FIE). IEEE, 2015.

[6] "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu, 2020

[7] "Guiding Text Generation with Constrained Beam Search in Transformers", constrained-beam-search, [Online]. Available: https://huggingface.co/blog/constrained-beam-search

[8] "SQuAD-explorer", [Online]. Available: https://rajpurkar.github.io/SQuAD-explorer/

[9] Rajpurkar, Pranav, et al. "Squad: 100,000+ questions for machine comprehension of text." arXiv preprint arXiv:1606.05250 (2016).