End-to-End Quiz-Style Question Generation For Educational Purposes

Mohammed Ashraf Mohammed
Computer Science Department
Faculty of Computer and Information Science Ain Shams University
Cairo, Egypt
mohammed20191700509@cis.asu.edu.eg

Mostafa Ashraf Borhamy
Computer Science Department
Faculty of Computer and Information Science Ain Shams University
Cairo, Egypt
mostafa20191700639@cis.asu.edu.eg

Mostafa Ayman Awad
Computer Science Department
Faculty of Computer and Information Science Ain Shams University
Cairo, Egypt
mostafa20191700640@cis.asu.edu.eg

Mostafa Labib Mohamed
Computer Science Department
Faculty of Computer and Information Science Ain Shams University
Cairo, Egypt
mostafa20191700646@cis.asu.edu.eg

Mostafa Hesham Abd-Elhassib
Computer Science Department
Faculty of Computer and Information Science Ain Shams University
Cairo, Egypt
mostafa20191700656@cis.asu.edu.eg

Aisha Soliman Fath-Allah
Computer Science Department
Faculty of Computer and Information Science Ain Shams University
Cairo, Egypt
aisha20191700330@cis.asu.edu.eg

Asmaa Bahai
Computer Science Department
Faculty of Computer and Information Science Ain Shams University
Cairo, Egypt
Asmaa.bahai@cis.asu.edu.eg

Sally Saad
Computer Science Department
Faculty of Computer and Information Science Ain Shams University
Cairo, Egypt
sallysaad@cis.asu.edu.eg

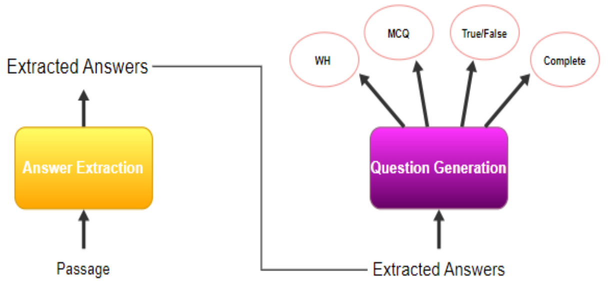
ABSTRACT
Question Generation (QG) is a branch of Natural Language Processing (NLP). It aims to generate natural language questions based on given contents. It's a combination of natural language understanding (NLU) and natural language generation (NLG) tasks. The purpose of this paper is to develop an end-to-end question generation system which can be used in real-world applications for educational purposes. Crafting exam-style questions is an essential component of education, it serves diverse objectives where both the students and educators can use it to facilitate the education process. We propose a pipelined system composed of two modules, an Answer Extraction module, and a Question Generation module. The Answer Extraction module is an encoder-decoder model that extracts question-worthy key-phrases from contexts. The Question Generation module consists of a set of specialized models finetuned on different types of questions that are generated in an end-to-end manner. The types of questions that can be generated by our system are: WH, MCQ, complete, and true/false. Evaluation of the models in our pipeline shows that our proposed system can generate educationally meaningful question-answer pairs with only context paragraphs as input, demonstrating the practicality of using automatic question generation.

I. INTRODUCTION

Recent advances in Natural Language Processing (NLP) can be greatly attributed to the invention of the Transformer architecture [1]. Advancements in deep learning paved the way for the Transformer architecture. The Transformer architecture allowed development of models that have achieved state-of-the-art results on all NLP tasks, including question generation (QG). The original Transformer architecture consists of an encoder and a decoder connected to it. The encoder maps an input sequence of symbol representations $(x1, \dots, xn)$ to a sequence of continuous representations $z = (z1, \dots, zn)$. Given z, the decoder then generates an output sequence $(y1, \dots, ym)$ of symbols one element at a time. At each step the model is auto-regressive, consuming the previously generated symbols as additional input when generating the next. The key innovation in the Transformer architecture is self-attention which allows the model to look at the other positions in the input sequence while processing each word, which will lead to a better encoding. All the models in our system adopt the Transformer architecture for the task of end-to-end question generation. Our system is a pipelined system composed of two modules, an Answer Extraction module, and a QG module (Fig. 1). We claim that our system can be used for educational purposes to create exercises. In turn it can be used to generate exercises on teaching material as they may not always be available, which puts a lot of pressure on both educators and students. For educators, this pressure comes from manually constructing questions which is a complex task that demands expertise, practice, and adequate resources. For students, this pressure comes from summarizing studying material and finding questions on it. These tasks are time-consuming therefore automating them would save time and money.

The goal of the Answer Extraction module is to train a neural model that when given a paragraph as input, outputs phrases in that paragraph that can be answers to questions. The goal of the model is to extract question-worthy key phrases that when given to a QG model will result in generating a question that is not only correct syntactically but also semantically. The Answer Extraction model utilizes the powerful T5 [2] model to extract key phrases. The T5 model is based on the Transformer architecture, which is a neural network

Figure 1 overview of the proposed system.



architecture that is well-suited for NLP tasks. The T5 model utilizes a text-to-text format, meaning that both the input and ouput are text. The goal of the QG module is to finetune a set of QG models for generating different types of questions. These types are as follows: WH, MCQ, complete, and true/false. Each model takes a passage, and an answer as input and generates high-quality, contextually relevant questions. Additionally, The WH-QG model is enhanced by utilizing a BERT-based classifier to predict the interrogative word [3]. Given a passage–answer pair the classifier predicts the interrogative word among seven classes. By incorporating the T5 model in our pipeline, which has shown remarkable performance in NLP tasks, We aim to enhance the QG process and provide accurate, informative, and diverse questions.

II. Related Works

Historically, rule-based systems [4][5] have been used for QG. They are a type of Artificial Intelligence (AI) that uses templates and heuristics to generate questions. The rule-based approaches typically include the following steps: 1) Preprocessing of the input text using NLP techniques. 2) Identifying the words or phrases that will be considered an answer by using semantic roles and rules. 3) Using a set of predefined rules and templates, questions are generated. 4) Lastly, the generated questions are ranked by using a set of features. There are multiple disadvantages to using rule-based systems which are as follows: 1) The cost of creating the templates and rules manually is high. 2) Both the answers, and templates are not diverse as they depend on a set of rules. Most of the QG models use neural networks that rely on a sequence-to-sequence architecture to craft questions about text in a natural language manner. In the Interrogative-word-Aware Question Generation (IWAQG) [3] work, a sequential system consists of two components: the first model predicts the appropriate interrogative word, which is passed to the second model to generate the corresponding questions. This allows the QG model to be able to focus more on the rest of the question rather than the question word.
In [6], the authors propose an answer-focused and position-aware neural QG model. Answer-focused means that question word generation is explicitly modeled by incorporating the answer embedding, which can help generate an interrogative word matching the answer type. Position-aware means that the relative distance between the context words and the answer is modeled. Hence the model can be aware of the position of the context words when copying them to generate a question. In [7], the key idea is that since question answering and generation are naturally related tasks, leveraging their connection should be mutually beneficial in terms of performance as well as reducing the amount of labeled data, e.g., for training a QA system. The paper's contributions can be summarized into two main points: 1) Leveraging question generation by tying together GPT-2 and BERT in an end-to-end trainable fashion facilitating semi-supervised learning. 2) Using QA as a surrogate measure for assessing question generation quality.
In [8], the QG task has been studied from Knowledge Graph (KG) queries. Essentially, the QG task reflects the ability of intelligent systems to translate abstract and schema-specific KG representations into natural language questions. The paper studies complex question generation from KG, with an emphasis on how to leverage existing simple questions.

In this study [9], a clue word predictor is introduced, which predicts potential clue words for target questions based on the context of the input passage. To understand the relationship between answer tokens and other tokens in a sentence, the predictor employs a syntactic dependency tree. For clue word sampling, it employs a GCN-based encoder and a Straight-Through Gumbel-Softmax estimator. The predicted clue word distribution is then fed into the passage encoder, which includes a low-frequency masking strategy to reduce complexity in tuning input word embeddings. Using a multitask learning strategy, the decoder learns word generation probabilities from the vocabulary as well as word copying from the input passage. The copy gate in the model employs explicit binary labeling where the target vocabulary is further shortened based on the frequency distribution of non-overlapping words.

III. The proposed model

A. Problem Statement

Our problem can be divided into two subproblems:
First, given a passage P, we want to extract a set of question-worthy key-phrases K. More formally:

$$\overline{K} = \underset{K}{argmax}\ Prob(K|P)$$

Second, given a passage P, and an answer $A \in \overline{K}$, we want to find a question Q whose answer is A. Additionally, $Q \in G$ where G is the set of types of questions our system can generate. More formally:

$$\overline{Q} = \underset{Q}{argmax}\ Prob(Q|P, A)$$

We assume that P is a paragraph composed of a list of words: $P = \{x_t\}_{t=1}^{M}$, and the answer is a subspan of P. The problem is modeled in a pipelined manner consisting of two modules. The first module is the Answer Extraction module which aims to solve the first subproblem. The second module is the QG module which aims to solve the second subproblem.

B. Answer Extraction

Most literature in the field of QG doesn't tackle the task of extracting key-phrases although it enables the QG models to generate meaningful questions. Often third-party libraries and automatic key-phrase extraction methods will be employed. Automatic key-phrase extraction methods are divided mainly into two categories: 1) statistical-based methods such as TF-IDF, and YAKE. 2) graph-based methods such as TextRank, and RAKE. Although these methods are good at extracting key-phrases that summarize text, they don't always satisfy the requirement of extracting key-phrases which are question-worthy. In some literature in

QG the answer will be assumed to be a given input to the QG model. This has the downside of not being applicable to real-life scenarios. We believe training a dedicated model for this task can yield better results for the quality of the generated questions. We argue a dedicated model can learn the semantics of the passage and extract human-like answers from the text. Additionally, since our model generates a set of answers from a context, it eliminates the need for a heuristic for the number of key-phrases to extract, thus reducing the number of illogical questions.

The answer extraction model utilizes the powerful T5 model to extract key phrases. The T5 model has several unique features that made it a fit for our task. First, the T5 model is pretrained on a massive dataset of text (C4 dataset) [2], which allows it to learn a wider variety of relationships between words. Second, the T5 model can learn the context of a phrase, which helps it to identify phrases that are important even if they do not appear frequently. These features enable it to achieve good performance when it comes to the task of extracting question-worthy key phrases.

C. WH-question Generation

1) Interrogative word Classifier

In our WH-question generation approach, we utilize an interrogative word classifier based on BERT. We provide the passage and the extracted answer from the Answer Extraction module as input. To distinguish the answer span and highlight its importance, we incorporate a special token, [ANS], which informs BERT about its distinct significance compared to the rest of the passage. BERT outputs a special token, [CLS], originally designed for classification tasks. We construct a feed-forward network on the top of BERT that takes the concatenated embeddings of the [CLS] token, and a trainable embedding representing the entity type of the answer as input. We connect the interrogative-word classifier to the second module by using the predicted interrogative word as the first input to the WH-QG model.

2) Question Generation

We utilize a robust T5-based model to generate questions. This model receives as input a passage, an answer, and an interrogative word, and it produces high-quality questions that are relevant to the given context. By incorporating the T5 architecture, which has demonstrated exceptional performance in various NLP tasks, we aim to improve the QG process and deliver precise, informative, and diverse questions. Since the QG module employs a T5 model, both the input and the output are in text format. The following is an example input provided to the model:
Generate question using question word:
<interrogative word> Answer: <answer> Context: <context>

D. Boolean Question Generation

Boolean QG is the task of generating questions for which the answers are either true or false. We argue that this task differs from the task of generating WH-style questions. Our argument is that the structure of the question significantly differs from the typical WH-style question. In WH-style question there exists an answer, based on which, the question is generated. On the other hand, for Boolean QG the answer is either true or false. This makes choosing a suitable sentence for generating a question a complex task. For this reason, we finetune a separate model for this task. We chose T5 for this task.

E. MCQ & Complete

MCQ and complete are related tasks, in complete the task is to extract a key phrase and its sentence omitting the key phrase (answer), for MCQ we add the task of generating distractors related to the answer. MCQ and complete modules share the same steps except that MCQ gives you multiple possible answers in the form of distractors. The distractors in MCQ are generated by the Sense2Vec [10] model. Sense2Vec is an extension of Word2Vec. Unlike Word2Vec, which generates embeddings for individual word tokens, Sense2Vec creates embeddings for "senses" of words. A sense is a combination of a word and a label that represents the specific context in which the word is used. This label can indicate various aspects such as part-of-speech (POS) tags, polarity, entity names, or dependency tags. For example, in Word2Vec you can have the following:

$$v[king] - v[man] + v[woman] \approx v[queen]$$

where v denotes vector embedding of word. In Sense2Vec POS tags as well as entity names are used to generate word embeddings. We take advantage of spaCy [11] being able to support adding Sense2Vec as part of a pipeline and use this pipeline to generate MCQ and complete questions.

IV. Experiments

A. Datasets

We used four reading comprehension datasets in the training our models. The datasets used are as follows:

- Stanford Question Answering Dataset (SQuAD) [12] is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage.
- NewsQA [13] is a challenging machine comprehension dataset of over 100,000 human-generated question-answer pairs. Crowdworkers supply questions and answers based on a set of over 10,000 news articles from CNN, with answers consisting of spans of text from the corresponding articles.
- AdversarialQA [14] is a comprised of three new reading comprehension datasets constructed using an adversarial model-in-the-loop. The authors used three different models in the annotation loop and to construct three datasets. Each dataset has 10,000 training examples, 1,000 validation, and 1,000 test examples.
- BoolQ [15] is a question answering dataset for yes/no questions containing 15,942 examples. These questions are naturally occurring-- as in they are generated in unprompted and unconstrained settings.

As can be seen the datasets are from different sources which introduces variety in the data that should help the models learn and generalize on unseen data examples.

For Answer Extraction SQuAD and NewsQA were used. Additionally, preprocessing was applied to group all answers of a unique context into a single record. After appending all the answers of a context with a special separator token, the text is passed as the output in the training of our Answer Extraction model. For WH question generation all datasets were used except for BoolQ. The distribution of interrogative words within the datasets is not balanced. To train the Interrogative-Word Classifier, the training set is downsampled to create a balanced dataset. This downsampling process ensures an equal representation of different interrogative words in the training data for effective training of the classifier. We sampled 10000 samples for each class except for "why" type which was around 2700 samples. For Boolean QG BoolQ was used.

B. *Implementation*

1) Answer Extraction

Our Answer Extraction model is a T5 model, specifically a t5-small model. The t5-small model is provided by HuggingFace [16]. The model is trained for 8 epochs with batch size 4. The first epoch being a warmup epoch where the learning rate starts increasing gradually until reaching 3e-4. For the remaining 7 epochs a polynomial decay function of power 1 is applied to decrease the learning rate gradually until it reaches 3e-5 at the end of the last epoch. The optimizer used for training is the Adam optimizer and the loss function is the cross-entropy loss function. The size of the encoder was set to 250, and the size of the decoder was set to 70. The task prefix used was "extract answers".

2) WH-question Generation

We developed an interrogative word classifier for our system using BERT-base-uncased, a pre-trained model provided by HuggingFace. The classifier is trained for two epochs using cross-entropy as the objective function. To obtain entity types embedding, we utilize spaCy [11], which provides six dimensions: PERSON, CARDINAL, DATE, ORG, GPE, and NONE (when no entity type is determined). The input dimension of the Feed-Forward Network (FFN) is 773, consisting of 768 dimensions from BERT and 5 dimensions from the entity type embedding. The FFN predicts the interrogative words: what, who, which, where, how, when, why, and others, and its output dimension is 8. The FFN architecture consists of a SoftMax layer stacked over a single linear layer. For optimization, we use the Adam optimizer with weight decay and set the learning rate to 5e-5.

We utilized the t5-base model for our QG model. The task prefix used was "generate question using question word". During training, we set the learning rate to 5e-5 and the encoder size to 250 tokens. For decoding, we used a size of 70 tokens. The QG model was trained for three epochs, allowing it to learn and improve over multiple iterations.

3) Boolean Question Generation

The Boolean question generation model uses the HuggingFace TensorFlow implementation of the t5-base model. The model was trained with Adam optimizer with weight decay using the cross-entropy loss function. The model was trained for 15 epochs with a constant learning rate of 3e-4 and a batch size of 8. The size of the encoder was set to 500, while for the decoder the size was set to 70. The task prefix used was "generate Boolean question".

4) MCQ & Complete

We use Sense2Vec as part of a spaCy pipeline to generate questions. Sense2Vec can extract the sense of the word on its own, but its accuracy wasn't good enough for us. We use the "en_core_web_lg" spaCy model for our pipeline. Sense2Vec is pretrained on two datasets of Reddit comments, the first being in 2015, and the second being in 2019. We use the 2019 dataset as it can capture the most recent meanings of words for example the word ghost can mean a supernatural spirit (2015 dataset) or the action ignoring your spouse (2019 dataset). We use spaCy to extract all named entities in the input text. We apply several filtering techniques to enhance the quality of the distractors. The first technique is that we make sure that the NER tag of the distractor matches that of the correct answer. The second technique is making sure there are no common words between the distractor and the correct answer. This is done as the dataset is collected from comments on the Reddit website which are prone to spelling mistakes. As such the same word with a different spelling can have two different embeddings where similarity between them is high leading to multiple correct answers or duplicate distractors. The third technique is making sure that cardinals in different formats are not chosen as distractors e.g., if "1" is the answer then "one" is filtered from being chosen as a distractor. The complete questions are the same as MCQ questions but without any distractors and the correct answer is omitted from the text.

V. EVALUATION & RESULTS

A. *Answer Extraction*

We used the evaluation in [17] as their task matched ours. The authors proposed an extension of the SQuAD F1 metric (for a single answer span) to multiple spans within a document, which is called the multi-span F1 score. This metric is calculated as follows. Given the predicted phrase $\hat{e}_i$ and a gold phrase e_j, we first construct a pairwise, token-level F1 score matrix of elements $f_{i,j}$ between the two phrases $\hat{e}_i$ and e_j. Max-pooling along the gold-label axis essentially assesses the precision of each prediction, with partial matches accounted for by the pairwise F1 (identical to evaluation of a single answer in SQuAD) in the cells: $p_i = \max_j (f_{i,j})$. Analogously, the recall for label e_j can be computed by max-pooling along the prediction axis: $r_j = \max_i (f_{i,j})$. We define the multi-span F1 score using the mean precision $\bar{p} = avg(p_i)$ and recall $\bar{r} = avg(r_j)$:

$$F1_{MS} = \frac{2\bar{p} \cdot \bar{r}}{\bar{p} + \bar{r}}$$

Our baseline model is PtrNet proposed in [17]. PtrNet is a pointer network which uses RNN encoder and decoder that points to key-phrase start and end boundaries. We use another baseline YAKE which is automatic key-phrase extraction method based on statistical methods. We mark all the key phrases as answers. Additionally, we compare our model with a simple method which marks all named entities as answers using spaCy library. The results can be seen in Table 1.Our model outperforms all the baseline models in the F1 score when it comes to the SQuAD dataset. We observe that for NewsQA dataset our model outperforms both the spaCy and YAKE but is exceeded in evaluation by PtrNet. We believe

Table 1 Comparison between benchmark and the proposed Answer Extraction model denoted by "*"

Model	F1	Precision	Recall
		SQuAD	
spaCy NER	26.4%	30.7%	23.2%
YAKE	23.8%	28.6%	20.3%
PtrNet	40.4%	**44.8 %**	38.7%
T5-small*	**45.4%**	44.3%	**46.6%**
		NewsQA	
spaCy NER	10.1%	12.5%	8.5%
YAKE	24.0%	28.7%	20.7%
PtrNet	**43.5%**	**46.7%**	**42.7%**
T5-small*	39.1%	36.9%	41.7%

that this is due to NewsQA having more answers per unique context than SQuAD. Additionally, the lengths of contexts are significantly bigger in NewsQA which makes only part of the context visible to the encoder of our model. Since SQuAD is collected from Wikipedia articles and NewsQA is collected from CNN articles, we believe that SQuAD performance is more relevant when it comes to educational settings.

This is because text in educational settings is more similar to that of Wikipedia articles than CNN articles. Therefore, we believe the lack of performance when it comes to NewsQA isn't representative of the model's ability to extract question-worthy answers.

B. WH-question Generation

1) Interrogative Word Classifier

We use classification accuracy to evaluate our Interrogative-Word Classifier. We compare our model with the Interrogative-Word Classifier proposed in [3]. The Interrogative-Word Classifier is a BERT based classifier that uses context and answer NER tag embedding to make a classification decision. As can be seen in Table 2 our model achieves the same results as the baseline model.

2) Question Generation

We use the following metrics for evaluating our WH QG model: BLEU1:4, ROGUE-L, and METEOR. We use as a baseline the Interrogative-Word-Aware Question Generation (IWAQG) model [3]. IWAQG is a sequence-to-sequence neural network that uses a gated self-attention in the encoder and an attention mechanism with maxout pointer in the decoder. Additionally, the model utilizes the interrogative word classifier described in Section III to choose the interrogative which is prepended to the input text. We also use the Clue Guided Copy Network for Question Generation (CGC-QG) model [9] as a baseline. The clue word predictor of CGC-QG uses a syntactic dependency tree to predict potential clue words for questions based on passage context. It employs a GCN-based encoder and a Gumbel-SoftMax estimator for clue word sampling. The predicted clue word distribution is used in the passage encoder with a low-frequency masking strategy. The decoder uses multitask learning to learn word generation probabilities and word copying from the passage, employing a copy gate, and shortening the target vocabulary based on non-overlapping word frequencies. As can be seen in Table 3, our model outperforms the baselines in all metrics.

Table 2 Comparison between benchmark and the proposed Interrogative Word Classifier model denoted by "*".

Table 3 Comparison between benchmarks and chosen WH QG model denoted by "*".

Table 4 Comparison between benchmark and the proposed Boolean QG model denoted by "*".

C. Boolean Question Generation

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROGUE-L
			SQuAD			
IWAQG	47.69	32.24	24.01	18.53	22.23	46.94
CGC-QG	40.45	23.52	15.68	11.06	17.11	43.16
T5-base*	**49.59**	**33.79**	**25.27**	**19.59**	**24.51**	**47.36**

For evaluating our model, we use the following metrics BLEU1:4, ROUGE-L, and METEOR. We compare our model to the Finetuned-QG model [18]. Finetuned-QG is a

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROGUE-L
Finetuned-QG	49.49	33.70	24.71	18.51	23.68	**47.04**
T5-base*	**50.05**	**37.86**	**29.14**	**22.60**	**24.80**	46.62

T5 QG model that has been trained on multiple QG datasets with different formats in a transfer learning manner. Finetuned-QG is the authors' best performing model in Boolean QG hence being chosen as a baseline model. As can be seen in Table 4, our model outperforms Finetuned-QG on all metrics except for a 0.4 difference in ROGUE-L metric in favor of Finetuned-QG.

VI. CONCLUSION & FUTURE WORK

In this work, we proposed a pipelined system for end-to-end question generation. This system is composed of two modules, an Answer Extraction module, and a QG module. The first module is used to extract question-worthy key phrases from text. The second module is comprised of a set of finetuned models that generate the following types of questions: WH, true/false, MCQ, and complete. Our proposed system generates high quality, contextually relevant questions that can be used for educational purposes. We prove our hypothesis by conducting quantitative analysis on all our models which show that the models of our system outperform powerful baseline models and are viable for educational settings.
In the future, we would like to improve our Boolean QG model by creating a new dataset. We believe this can be done by applying the methods used in the MCQ and Complete module on the contexts of a reading comprehension dataset and training the model on the new dataset. Furthermore, making our system not only limited to the English language by using multilingual models and datasets should prove beneficial for the domain of education.

REFERENCES

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Model	Accuracy
Interrogative aware classifier	**73.8%**
Bert-base*	72.7%

[2] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, *21*(1), 5485-5551.

[3] Kang, J., Roman, H. P. S., & Myaeng, S. H. (2019). Let me know what to ask: Interrogative-word-aware question generation. *arXiv preprint arXiv:1910.13794*.

[4] Heilman, M., & Smith, N. A. (2010, June). 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 (pp. 609-617).

[5] Chali, Y., & Hasan, S. A. (2015). Towards topic-to-question generation. *Computational Linguistics*, *41*(1), 1-20.

[6] Sun, X., Liu, J., Lyu, Y., He, W., Ma, Y., & Wang, S. (2018). Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 3930-3939).

[7] Klein, T., & Nabi, M. (2019). Learning to answer by learning to ask: Getting the best of gpt-2 and bert worlds. *arXiv preprint arXiv:1911.02365*.

[8] Zhao, J., Deng, X., & Sun, H. (2019). Easy-to-hard: Leveraging simple questions for complex question generation. *arXiv preprint arXiv:1912.02367*.

[9] Liu, B., Zhao, M., Niu, D., Lai, K., He, Y., Wei, H., & Xu, Y. (2019, May). Learning to generate questions by learningwhat not to generate. In The world wide web conference (pp. 1106-1118).

[10] Trask, A., Michalak, P., & Liu, J. (2015). sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings. arXiv preprint arXiv:1511.06388.

[11] Honnibal, M. (2015). Spacy · industrial-strength natural language processing in python. · Industrial-strength Natural Language Processing in Python. https://spacy.io/.

[12] Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

[13] Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., & Suleman, K. (2016). Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

[14] Bartolo, M., Roberts, A., Welbl, J., Riedel, S., & Stenetorp, P. (2020). Beat the AI: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, *8*, 662-678.

[15] Clark, C., Lee, K., Chang, M. W., Kwiatkowski, T., Collins, M., & Toutanova, K. (2019). BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

[16] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020, October). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations* (pp. 38-45).

[17] Subramanian, S., Wang, T., Yuan, X., Zhang, S., Bengio, Y., & Trischler, A. (2017). Neural models for key phrase detection and question generation. arXiv preprint arXiv:1706.04560.

[18] Yuan, W., Yin, H., He, T., Chen, T., Wang, Q., & Cui, L. (2022, April). Unified question generation with continual lifelong learning. In Proceedings of the ACM Web Conference 2022 (pp. 871-881).