Automated System for Multiple Choice Question Generation from Text

Stav Cohen and **Avitay Geltman** and **Doron Battino**
Technion – Israel Institute of Technology
{cohenstav1, avitaygl, doronbatt}@gmail.com

Abstract

Automatic multiple choice question (MCQ) generation is a useful yet challenging task in Natural Language Processing. It is the task of automatic generation of context–dependent questions and answers, where for a given context, a corresponding question is provided along with a set of answers of which solely one is correct. Automation is of great interest due to the tedious process of manual constructing tests of this type. Therefore, researchers have been putting effort into this automation task since the late 1990's. In our study, we present an innovative Transformer based system automatically generating MCQs for a given text. Our system is constructed of three separate units that fulfill different tasks. The goal of the first is to provide answers from a given context to the second. Consequently, the latter generates questions based on the provided answers and corresponding contexts. The third unit receives all the above and generates relevant false answers. A transformer–based model was trained to distinguish between generated and non–generated answers, resulting in an F1 score of 43–59%, meaning our generated answers were essentially indistinguishable. Additional metrics further demonstrated the prosperity of the answer generation unit. For the question generation task, the commonly used BLEU–4 metric was utilized, obtaining a decent score of 14.76 upon integration of a smoothing function. Finally, in terms of the generated false answers' quality, plausible and diverse candidates were successfully obtained. Many of which presented a proper grammatical structure, consistency with the context, and reasonable logic.

1 Introduction

Multiple choice questions are a form of assessment in which the respondents select the best possible answer out of a set of choices. The answer's alternatives can be either fictitious and incorrect or solely partially correct, making them false answers, from now on called – distractors. Conducting MCQ tests is known to be one of the most popular and widespread methods for assessing examinees' understanding abilities, knowledge, etc. Tests of this type are proven to be highly efficient and benefit from a short evaluation process, as well as a shorter testing time.

To make an MCQ test plausible, finding reasonable distractors is crucial, ensuring the correct answer is not to be too obvious. The objective is to provide distractors that are grammatically correct, semantically consistent with both context and question, and logically might serve as a question's answer, to some degree.

A major issue with MCQ tests is their lengthy and tedious manual construction process, which also requires great experience and satisfactory wording skills of the constructor. To ease the process and make it less costly, automation is of great necessity. In this study, an automatic system for MCQs generation based on given text is developed, utilizing modern robust Transformer architectures. The proposed system is divided into three separate units. The first unit is an answer generation (AG) unit, where based on an inputted context, suitable phrases are identified as potential answers. Consecutively, a corresponding answer is then extracted to serve as the output of the unit. The

second unit is a question generation (QG) unit, which receives as an input the extracted answer accompanied by the context it originated from. Following that, a suitable question is then generated and attached to the context–answer pair, serving as the unit's output. Finally, the trio of context–answer–question is inputted to the third, and last, unit which then generates three distractors. Thus concluding the task of MCQ generation. For all three units, we utilize the power of separate fine–tuned T5 Transformer models.

2 Related Work

The research on automatic MCQ generation started at least 25 years ago (Coniam, 1997). Since then, many MCQ generation systems have been developed in various languages and for various applications. In recent years, focus has shifted towards the utilization of artificial intelligence, specifically deep learning methods.

Regarding AG, past studies attempted to tackle the task without human guidance by training an Inside–Outside–Beginning (IOB) tagger to predict whether each word in the paragraph is a part of an answer or not, consequently selecting all consecutive spans that construct an answer. Additionally utilizing a Name Entity Recognition (NER) system to extract entities and select them as answers as well (David Golub et al., 2017). Furthermore, researchers constructed an Encoder-Decoder Neural Network that encoded each context followed by Part Of Speech (POS) and NER tags per token in order to extract key phrases that will be used as answers (Angelica Willis et al., 2019). We propose a novel approach using state–of–the–art Transformers to generate answers solely based on context.

For the task of QG, in a recent known study (Duan et al., 2017), prominent seq2seq models were proposed, however, target answer was not taken into consideration. A later study (Zhou et al., 2017) suggested that the input for the model would include the position of the answer along with the context. An additional study (Tang et al., 2017) proposed to unite QG with question answering (QA) and treat it as one task. In our work, we propose treating QG as a separate task in our system with a focus of its own.

As for the task of DG, most existing methods are based on semantic similarities between the correct answer and the candidate distractor (Agarwal and Mannem, 2011; Guo et al., 2016; Kumar et al., 2015). Various similarities have been utilized in different works such as WordNet similarity (Miller, 1995), word2vec similarity (Mikolov et al., 2013), n-gram co-occurrence likelihood, etc. These DG methods focused on the relationship of the distractors with the correct answer, however, they have not fully explored how to utilize the information in the context and the question. Thus, we tackle this by introducing and inputting all available information, i.e., context–question–answer, to the third DG unit. Moreover, we integrate similarity boundaries into our algorithm, forcing satisfactory similarities between the generated distractors and the correct answer.

3 Data

Implementation of our system relies on two prolific datasets, SQuAD v1.1 dataset which serves us in both AG and QG units, and RACE dataset in the DG unit. Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset (Rajpurkar et al., 2016), 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. The dataset contains 100,000+ question-answer pairs on 500+ articles.

The ReAding Comprehension Dataset from Examinations (RACE) dataset (Lai et al., 2017) is a machine reading comprehension dataset consisting of 27,933 passages and 97,867 questions from English exams, targeting Chinese students aged 12–18. Each question is associated with 4 candidate answers, one of which is correct. Questions in RACE are specifically designed for testing human reading skills and are created by domain experts.

4 Model

4.1 Answer Generation

To begin with, SQuAD dataset that offers a range of 1–25 answers per context was processed into a dataset containing unique rows of different contexts and 10 corresponding columns, containing up to a maximum of 10 answers. Leading to the creation of 3 datasets, train, validation, and test. Consequently, we utilize a T5 Transformer and its corresponding tokenizer to tokenize the source input as a concatenation of 1–10 different answers, separated by a specific token, followed by a SEP token and the context that the

answers were taken from. The average number of answers per context in the SQuAD dataset is 5 answers, hence we mask each of the 5 first answers with a probability of 40%, the rest are masked with a reduced probability of 20%. Each string was encoded by 500 tokens, strings encoded to less than that were padded to the max length. Strings encoded to more than 500 tokens were truncated. As for the target encoding, we tokenize the corresponding answers stringed together. Thus, for the fine– tuning process of our model, it receives up to 10 answers and the corresponding context, and outputs a set of answers. On the contrary, for the later generation task, the model receives as an input 10 MASK tokens and a context, forcing the model into delivering its outputs based solely on a fed context.

The small version of T5 Transformer was trained for 10 epochs using the AdamW as the optimizer, a learning rate of 0.0001, and a default weight decay of 0.01. As for the generation of the answers, the model is versatile and customizable, enabling us to generate a larger number of unique answers per context by increasing the minimum length of the generated output. By doing so, the model is forced to output a longer string that inherently contains more answers. We utilize a greedy search generation with 10 beams, repetition and length penalties of 5, and a hard restriction on n-gram making each 3-gram in the concatenated answer string unique. This leads us to the generation of longer answers with profound meanings and suitability to the context.

4.2 Question Generation

The first step was loading and preprocessing the SQuAD dataset, after which the outcome was the dataset solely with the relevant features, i.e., context, answer, and question. Consequently, we loaded a T5 tokenizer from a pretrained small version of T5 Transformer. Three classes were then constructed for a later T5 Transformer fine–tuning. First, a class whose responsibility was source and target encoding of the datasets by tokenization was constructed. Source encoding was the tokenization of the context and answer with a SEP token between them. Target encoding was the tokenization of the question. Source and target maximum token lengths were set to 500 and 30, respectively. Next, a second class was constructed to define a module to utilize in the training process of our model, defining loaders for the train, validation, and test sets, with a 16–batch size for the train loader. The third class is the construction of the model, using AdamW optimizer with a learning rate of 0.0001 and a default weight decay of 0.01. The model was trained for 5 epochs and fine-tuned.

Consequently, we defined a function for the generation of questions from pairs of context and answer, using the fine-tuned model. One beam search was used, with length and repetition penalties of 3.5 and 0.95, respectively. Lastly, a function responsible for the connection between the AG and DG units was built.

4.3 Distractors Generation

At first, the RACE dataset was loaded and preprocessed, followed by extraction of the desired features, i.e., context, question, answer and 3 distractors. Afterward, a T5 tokenizer from a pretrained small version of T5 Transformer was loaded. In a similar manner to the QG unit, the three classes required for the fine–tuning of the transformer were constructed. For the class responsible for the datasets' encoding, source and target maximum token lengths were set to 496 and 54, respectively, after examination of our datasets. Source encoding was tokenization of the trio answer–question–context, while target encoding was tokenization of the 3 distractors. For the module class, loaders for the various datasets were defined. A batch size of 16 was chosen for the train loader. The third class was the construction of the model class. AdamW was chosen as the optimizer, with a learning rate and weight decay of 0.001, both. For the decay, a cosine with hard restarts scheduler with a 0.2 warmup was set. The model was then trained and fine–tuned over 87,866 samples for 15 epochs.

Thereafter, a function was constructed to generate the distractors. Number of generations per each function call, i.e., the number of times that 3 different distractors are generated, was set to 20, to enable the selection of the top 3 distractors. Sampling was introduced and defined by a combination of two methods; Top–K sampling which samples the most likely K words, together with Top–P sampling which samples from the smallest possible set of words whose cumulative probability exceeds the probability P. Values were set to K=50 and P=0.95. Furthermore, we instructed the method to avoid repeating 3–gram phrases and also set a length penalty of 0.65, which

forced the method to generate distractors slightly shorter than it normally would. From this point, the output of the generative function is passed through a cleaning function. At last, to seal the DG task, we constructed a function that carries out a punctilious selection of the top 3 distractors. For each row in the inputted dataset containing context–question–answer, by utilizing the previous generative function, 60 distractors are generated for each row. The quality of the distractors is then measured by means of similarity. For a distractor to be selected, we assure that its similarity to the real answer is in the range between two limits we define. On the one hand, a distractor has to resemble the correct answer by some degree, but on the other hand, if the similarity is too high, a distractor might be identical to the correct answer or differ merely by a character. Simultaneously, we make sure that the similarities between all chosen distractors are less than an upper limit, by doing so we avoid the selection of distractors that are too similar to each other. Moreover, different sets of similarity boundaries were given when the correct answer was either short or long.

Having said the above, a loop is defined to run 3 times in order to find the most suitable distractors, each run iterates through all 60 generated distractors. The first run is the strictest, and the last one is the most lenient, meaning it allows the selection of the distractors that did not pass the filtration of the previous 2 runs. The difference between the runs is reflected in the values of the allowed similarities. When the correct answer contains 3 words or less, the similarity between a distractor and the correct answer is set to be between 0.6 – 0.85, 0.4 – 0.6, 0 – 0.4 for the 1st, 2nd, and 3rd runs, respectively. The upper limit for the similarity between distractors is set to be 0.8. When the correct answer is longer than 3 words, the similarity between a distractor and the correct answer is set to be between 0.5 – 0.65, 0.35 – 0.5, 0 – 0.35 for the 1st, 2nd, and 3rd runs, respectively. The upper limit for the similarity between distractors is set to be 0.65. When the currently iterated distractor meets the criteria, it is selected. The process continues until 3 distractors are chosen, and then the algorithm moves on to the next row in the dataset. The ideal scenario is for the distractors to be all selected on the first and strictest run, and the worst-case scenario is for them to be all chosen on the last run. The chosen similarity metric was the cosine similarity. One-word answers were chosen to be embedded by GloVe (Pennington et al., 2014), while all other answers by Google's Universal Sentence Encoder (USE) (Cer et al., 2018).

5 Results

5.1 Answer Generation

Since the number of potential answers in a text can be theoretically limitless, there is no clear benchmark to test the quality of the answers that the model generates. Thus, we did not compare our answers to SQuAD's answers directly. Thereby, we propose several methods to examine the quality of our generated answers (GAs). The first is introducing a DistilBERT model (Sanh et al., 2020) which receives a dataset of GAs made by the T5 model and non-generated answers (NGAs) directly from SQuAD, and then attempts to predict whether the answer is generated or not. We split a dataset of 58K sampled answers to a train and validation sets with a ratio of 70:30. We train the model for 5 epochs, yielding a 57–60% accuracy and 43–59% F1–score on the validation set. Hence, we conclude that the model cannot successfully distinguish between GAs and NGAs. In another approach, we extracted entities from a given context and inspected how many answers included an entity, discovering that 63% of the GAs included an entity, compared to 59% of the NGAs. Furthermore, the average number of words in our model's GAs was calculated to be 4.9, compared to 2.2 of the NGAs. Thus, the model successfully generates longer answers which in turn lead to longer distractors, leading to an increase in the overall complexity of the MCQs generated.

5.2 Question Generation

At first, we tested to see if the questions generated by our fine-tuned T5 model and the SQuAD's questions are distinguishable. We constructed a balanced 21,140 sample dataset. Generated questions were labeled as 1, while non-generated labeled as 0. The dataset was split into 85% train set and 15% validation set. Next, we trained a DistilBERT base model for 5 epochs. The performance on the validation set was valued at 73.53% accuracy and 77.18% F1 score, meaning generated questions are quite noticeably distinguishable when compared to SQuAD's. We then turned to the most used metrics in the QG task, the BLEU-4. The obtained score was 9.81 which is

low in comparison to other studies, demonstrating scores in the range of 13 – 25. After conducting an analysis, we observed that the average length of a generated question was relatively short in comparison with SQuAD's, a phenomenon that distorts the BLEU-4 evaluation. Thus, we evaluated using the smoothed4 BLEU-4 metric (Chen and Cherry, 2014) and obtained a decent score of 14.76.
Observed drawbacks of the model include ever so often the generation of vaguely formulated questions for one–word answers, especially if the provided answer is not of key significance, or when the given context is very short.

5.3 Distractors Generation

After training our fine–tuned T5 Transformer for 15 epochs, a cross–entropy validation loss of 2.13 was obtained over a validation set of 4,887 samples. The value of the loss is not informative when trying to estimate the quality of our model, however, it was observed that models trained to higher validation losses generated noticeable inferior distractors. The inadequacy of these distractors was reflected in their poor consistency with the given context as well as the overall unsatisfying logical structure and grammatical deficiency. For our chosen model, a successful and diverse generation of plausible distractors was very often observed. Due to a severe lack of evaluation techniques, evaluation metrics, or benchmark data for MCQ system evaluation (Rao CH and Saha, 2018), no numerical scores were calculated. Accordingly, most generated MCQs are manually evaluated by human resources.

5.4 Overall Results

We tested our complete system on articles found on Wikipedia. Hereby, we present two examples, the first corresponding to the entry of "Bear" in Wikipedia, and the second corresponding to the entry of "Human".

1)

Context: "Bears are carnivoran……they are widespread, appearing in a wide variety of habitats throughout the Northern Hemisphere and partially in the Southern Hemisphere. **Bears are found on the continents of North America, South America, Europe, and Asia**……"
Question: "Where are bears found?"
Answer: "the continents of North America, South America, Europe, and Asia"
Distractors: 1. "The continents mainly in the Northern Hemisphere, but partly in the Southern Hemisphere". 2. "the Southern Hemisphere and North America". 3. "Australia, Europe, and Asia"

2)

Context: "Humans (Homo sapiens) are the most abundant and widespread species of primate……**Social interactions between humans have established a wide variety of values, social norms, and rituals,** which bolster human society……"
Question: "What have social interactions between humans established?"
Answer: "a wide variety of values, social norms, and rituals"
Distractors: 1. "a wide variety of moral, culture, and language". 2. "a broad variety of traits, social care and ritual protection". 3. "a vast number of social structures".

In conclusion, it is therefore observed that our system has the ability to generate accurately formulated MCQs with reasonable and logical distractors.

6 Experiments

6.1 Answer Generation

At first, we experimented with a T5 model that was fine–tuned directly on an unprocessed SQuAD dataset. The model was trained by receiving an answer or a masked token 40% of the time, and a context as the input. The output was the corresponding answer. Unfortunately, this model fell short, generating shorter answers, and lacking in terms of grammar. Therefore, we decided to tackle this issue with a new approach of ours. Instead of training the model to generate a single answer per context, we trained it to generate multiple answers. The new approach came with a handful of challenges to overcome. The model used to generate less than 10 answers, however, we managed to generate more answers by increasing the number of beams that were used, as well as the minimum length of target encoding, so we could assure an increase in the number of answers extracted from each context. The model used to output similar answers frequently, by introducing repetition penalty and limitations on n-grams,

while also combining text cleaning and processing, we reduced the number of similar answers.

6.2 Question Generation

When first training a base DistilBERT model on the dataset of generated and non-generated questions, in order to test its ability to distinguish between them, we noticed that it easily managed to do so. While trying to avoid this undesirable outcome, alternating the number of beams in the generation function resulted in the insight that reducing it from 5 to 1 was surprisingly optimal, making it a lot harder for the model to distinguish between generated and non–generated questions.

6.3 Distractors Generation

For the fine–tuning stage of the T5 Transformer, different configurations of parameters were experimented with. The cosine with hard restarts scheduler demonstrated its superiority over both standard cosine scheduler and constant scheduler, enabling a further decrease in validation loss. For the DG function, at first, the number of generations per each function call was set to 5, however, low diversity of distractors was observed. Thus it was increased to 20. Furthermore, different decoding methods for language generation were experimented with, including basic sampling and beam search with a ranging number of beams. Eventually, the combination of Top-K sampling and Top-P sampling was proven to be the most successful, generating the highest quality distractors. In terms of distractor length, it was initially observed that quite often a distractor would suffer a trade-off between its length and its grammatical structure and logic, making it less reasonable with an increase in length. Thus, we introduced the length penalty which sorted out the issue, while maintaining satisfactory distractor lengths.

For the function responsible for the selection of the top 3 distractors, it was observed that similarity boundaries were ought to be different for answers that were lengthwise distant. For instance, a similarity of 0.65 between 12–word answers could signify an extremely high resemblance between a distractor and an answer, while the same value would signify quite a far-off relation between one–word answers. Thus, higher boundaries were defined for short answers. Moreover, the answers' embeddings were initially chosen to be done by the USE, however, it was then observed that by embedding one–word answers in this manner, similarity calculations were quite unreasonable, leading to the selection of incompatible distractors. To overcome this hindrance, one–word answers were chosen to be embedded using GloVe, which made a noticeable impact, resulting in successful one–word distractor generations due to more accurate similarity calculations.

References

D. Coniam. 1997. A preliminary inquiry into using corpus word frequency data in the automatic generation of english language cloze tests. CALICO J., vol. 14, no. 2–4, pp. 15–33, 1997.

Manish Agarwal and Prashanth Mannem. 2011. Automatic gap-fill question generation from text books. In BEA@ACL. ACL, 56–64.

Qi Guo, Chinmay Kulkarni, Aniket Kittur, Jeffrey P. Bigham, and Emma Brunskill. 2016. Questimator: Generating Knowledge Assessments for Arbitrary Topics. In IJCAI. 3726–3732.

Girish Kumar, Rafael E Banchs, and Luis Fernando D'Haro Enriquez. 2015. Revup: Automatic gap-fill question generation from educational texts. In BEA. ACL.

George A Miller. 1995. WordNet: a lexical database for English. CACM 38, 11 (1995), 39–41.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In NIPS. 3111–3119.

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

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. arXiv preprint arXiv:1704.04683.

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

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant et al. 2018. Universal Sentence Encoder. arXiv preprint arXiv:1803.11175.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level BLEU. In Proceedings of the ninth workshop on statistical machine translation, pp. 362-367.

Dhawaleswar Rao CH and Sujan Kumar Saha. 2018. Automatic multiple choice question generation from text: A survey. IEEE Transactions on Learning Technologies 13, no. 1: 14-25.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In Proceedings of the 2017 conference on empirical methods in natural language processing, pp. 866-874.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-Stage Synthesis Networks for Transfer Learning in Machine Comprehension. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 835–844, Copenhagen, Denmark. Association for Computational Linguistics.

Angelica Willis, Glenn M. Davis, Sherry Shanshan Ruan, Lakshmi Manoharan, James A. Landay and Emma Brunskill. 2019. Key Phrase Extraction for Generating Educational Question-Answer Pairs. Proceedings of the Sixth (2019) ACM Conference on Learning @ Scale: n. pag.

Victor Sanh, Lysandre Debut, Julien Chaumond and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. ArXiv abs/1910.01108: n. pag.

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

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

A GitHub Repository

Link