QG Augmentation: Generating Novel Question/Answer Pairs for Few-Shot Learning

Stanford CS224N Default Project (RobustQA)
Mentor: Kathy Yu. No external collaborators. Not sharing project.

Ben Alexander
Department of Computer Science
Stanford University
balex@stanford.edu

Gordon Downs
Department of Computer Science
Stanford University
gwdowns@stanford.edu

Abstract

In many real-world settings, only a small volume of data is available for training. In such settings, data augmentation is a key method that improves task performance by artificially increasing the amount of training data. Most data augmentation techniques for Question Answering (QA) datasets focus on creating extra question-answer pairs that are rephrased versions of existing pairs in the training dataset (e.g., through back-translation and synonym replacement). In this project, we explore "QG Augmentation," a data augmentation technique that uses a question generation (QG) pipeline to generate *novel* QA pairs from the training passages. We further improve the basic technique by: 1. Creating a separate filtering module that discards low-quality generated QA pairs, 2. Modifying our QG Augmentation pipeline to better handle long context passages, and 3. Tuning two key hyperparameters in the generation pipeline. We provide experiments and ablations to evaluate the effectiveness of each approach. Our results show that QG Augmentation is effective in improving model performance in the few-shot setting (+2.82 F1, +2.88 EM vs. vanilla finetuning).

1 Introduction

This project focuses on the task of extractive question answering, which involves extracting the correct answer from a context passage, given a question about that passage. Specifically, we focus on the few-shot setting, where we only have a small number of examples to train on. In our case, we have three extractive QA training datasets for finetuning, each with just 127 training samples.

These few-shot settings are quite challenging, because the the lack of training data becomes a major bottleneck. Better model architectures and training strategies can help improve performance, but only to an extent; even the best models will struggle when there is so little data. Therefore, data augmentation can be an especially valuable tool in these few-shot settings, since it helps alleviate the data scarcity problem by synthetically increasing the dataset size.

However, typical data augmentation techniques (such as backtranslation and synonym replacement) usually perform small, local perturbations of existing QA pairs. Synonym replacement involves swapping a random subset of words in the text with their synonyms, and backtranslation involves translating the text to another language and then back to English. These types of methods produce augmentations that are essentially rephrased versions of the original text. These traditional augmentation methods are still very valuable for improving model performance and robustness, but they typically do not add much truly "new" information to the training set.

In contrast, our strategy, which we call "QG Augmentation" or "QGA," involves automatically extracting *novel* QA pairs from the training passages, many of which are quite distinct from those in the original dataset. Some examples are shown in Table 1 below.

Context passage: *ConAgra Foods, Inc. is an American packaged foods company headquartered in Omaha, Nebraska.*

	Question	Answer
Original	What city is ConAgra Foods located in?	Omaha
Synonym replacement	What *town* is ConAgra Foods *based* in?	Omaha
Backtranslation	In which city is ConAgra Foods located?	Omaha
QG Augmentation	What kind of foods does the company sell?	packaged
QG Augmentation	Conagra foods is based in which us state?	Nebraska
QG Augmentation	What is the name of the packaged foods company?	ConAgra Foods

Table 1: Examples of traditional augmentations vs. QGA. The traditional augmentations produce slight perturbations of the original, whereas QG Augmentation produces diverse and novel QA pairs.

We implement QG Augmentation using part of the question generation pipeline from the "Probably Asked Questions" (PAQ) project from Facebook AI Research [1]. We borrow two models from the PAQ project to construct our QG augmentation pipeline: 1. An answer extractor, and 2. A question generator. The PAQ project also includes a third model that they use to filter out low-quality generated questions, but it is not applicable to our use case.

The main contributions of this paper are as follows:

1. We introduce the basic QG Augmentation approach, which involves generating novel QA pairs from the training passages.
2. We implement our own filtering module to filter out low-quality QA pairs generated by the QG Augmentation pipeline.
3. We modify the QG Augmentation pipeline to better handle long context passages.
4. We tune two key hyperparameters in the generation pipeline (chunk size, and the number of QA pairs to generate per sentence) and provide ablations.

2 Related Work

In addition to PAQ, the idea of using models to generate novel training data has been explored before. For example, [1], [2], and [3] study this problem for the purposes of data augmentation. [4] uses this method to improve information retrieval systems. [5] and [6] study question generation as its own task. [7] uses question generation to improve real-time question answering. [8] uses synthetic data generation specifically to make QA models more robust to human adversaries. We did not find much prior work about this type of augmentation specifically applied to the few-shot setting, so we hope that our results will contribute to the literature.

3 Approach

3.1 Extractive QA model

For the actual extractive QA model itself, we use DistilBERT [9] with a question answering head. This is a copy of DistilBERT with linear layers added on top, which output the *span start* and *span end* logits. We use the DistilBertForQuestionAnswering implementation from HuggingFace [10].

3.2 Traditional augmentation methods

In addition to QGA, we also implement two "traditional" data augmentation methods, just to serve as a comparison to QGA.

First, we perform backtranslation on the provided questions, which involves translating the question to French and then back to English. We use the MarianMT model from HuggingFace to perform backtranslation [11].

Second, we perform synonym replacement on the provided questions using the Easy Data Augmentation [12] tool, replacing 10% of words in the text with their synonyms. Note that for simplicity, we only perform synonym replacement on the question, not the context.

3.3 QG Augmentation

QGA is the main component of our project. Here we provide more details about the QGA pipeline.

3.3.1 QGA: Basic pipeline

Our core QG Augmentation pipeline consists of two models borrowed from the PAQ project:

1. An answer extraction model $p_a(a|c)$. This is a BERT-based model that has been trained to predict a probability $p_a(a|c)$ for each candidate answer span a, which helps us identify the spans that are most likely to be answers in the given context c.
   - For example, given the passage, "Christopher Manning is a professor of computer science at Stanford University. He received his PhD from Stanford in 1994." It would likely identify spans such as "Christopher Manning," "Stanford University," and "1994" as some of the highest-probability answer spans.
2. A question generator $p_q(q|a, c)$ that generates a question q, given context passage c and answer a in that passage. This model is a copy of BART-base [13] fine-tuned for text generation.
   - For example, given the above passage and the answer "1994," it would ideally produce a question such as, "When did Christopher Manning receive his PhD?"

We use the pre-trained versions of these models from PAQ, which were trained on Natural Questions [14].

3.3.2 QGA: Filtering module

Of course, the basic QGA pipeline will not produce 100% high-quality QA pairs; some will be poorly written and/or completely wrong. Therefore, we implement a filtering module that identifies and discards lower-quality QA pairs.

To do so, we begin with a vanilla fine-tuned extractive QA model, which has been fine-tuned only on the original data from RACE, DuoRC, and RelationExtraction (i.e., without any QGA data). We use this model as our filter. First, we take each question generated by QGA, and use the vanilla fine-tuned model to predict its answer. Then, we calculate the F1 score between this predicted answer and the answer from QGA. If this F1 score exceeds a certain threshold, we consider that this is a "high-quality" question, and keep it; otherwise, we discard it.

The line between "high-quality" vs. "low-quality" QA pairs can be controlled by setting the F1 threshold at which we filter. A low F1 threshold (close to 0) will keep more of the generated QA pairs; a high F1 threshold (close to 1) is more strict and will keep fewer generated QA pairs. We experiment with different F1 thresholds and provide ablations in the Experiments section.

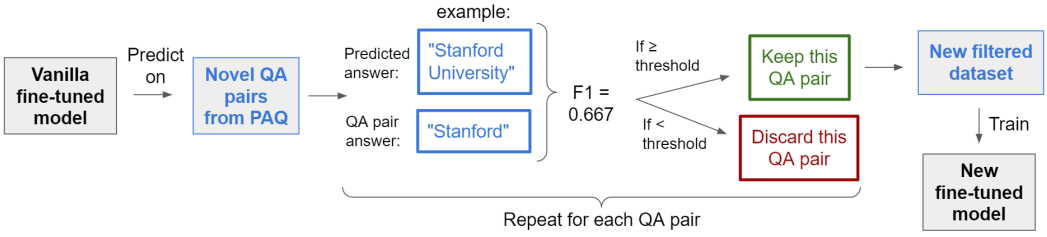


Figure 1: QGA pipeline with filtering module.

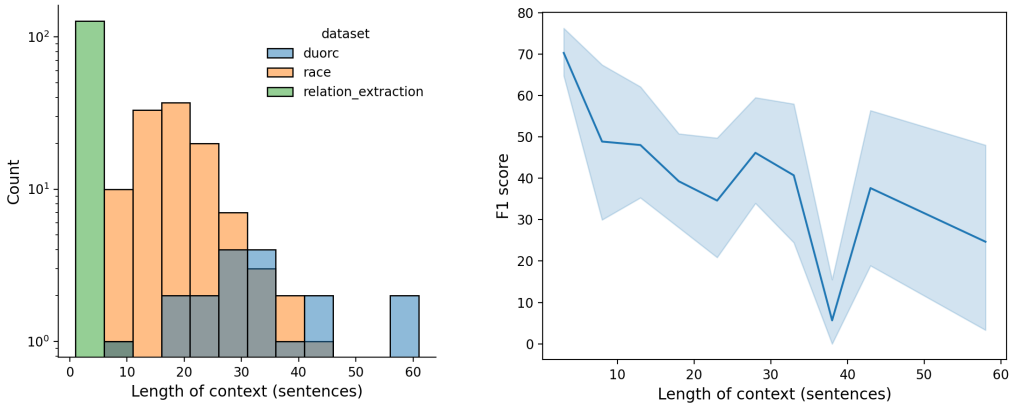
3.3.3 QGA: Improving performance on long contexts with sentence chunking

We noticed that our models struggle with long context passages in two different ways.

1. On the QGA side, from our manual inspection, the model appears to generate noticeably lower-quality QA pairs when run on long context passages (i.e., longer than just a few

sentences). In the Analysis section, we will see a case where the model starts hallucinating and asks questions about Harry Potter, even though the passage is completely unrelated.

2. On the extractive QA side, even before introducing any QGA data, we notice that the vanilla fine-tuned QA model performs worse on long context passages than on short context passages (see plot (b) in Figure 2). This is especialy problematic for DuoRC and RACE, which have longer passage lengths than RelationExtraction (see plot (a) in Figure 2).



(a) (b)

Figure 2: (a) Distribution of context passage lengths across datasets. RelationExtraction has the shortest contexts, while the other two datasets can have significantly longer contexts. (b) Performance of the vanilla fine-tuned model (i.e., not trained on any QGA data) vs. the length of the context passage. Clearly, the model performs better on short contexts than on long contexts.

The second problem is out of scope for this project; extractive QA is simply more difficult on longer contexts. However, we would like to address the first issue, so that we do not further exacerbate the poor performance on long contexts by generating low-quality questions for the RACE and DuoRC datasets.

Therefore, we add an additional modification to our QGA pipeline. Instead of running the QGA generation models on the entire passage, we use the spaCy Sentencizer tool [15] to break the passage into chunks of n sentences before passing them through QGA. Since QGA generates higher-quality QA pairs on shorter passages, sentence chunking yields better results.

4 Experiments/Results

4.1 Data

For the RobustQA Track of the default project, we have two main sets of extractive QA datasets. The "in-domain" datasets are SQuAD [16], NewsQA [17], and Natural Questions [14]. We have 50,000 training samples from each of these. The "out-of-domain" datasets are DuoRC [18], RACE [19], and RelationExtraction [20]. We have just 127 samples from each of these. Our validation and test sets consist only of questions from the out-of-domain datasets.

Of course, we also augment these out-of-domain training datasets with our QGA approach. We do not use QGA on the in-domain datasets, since they already contain a lot of data and we are not directly evaluating on them, but theoretically that is also possible.

4.2 Evaluation metrics

We compare the predicted vs. ground truth answers using the Exact Match (EM) and F1 scores. EM is a harsh metric for whether the prediction is an exact match with the ground-truth. F1 score is the harmonic mean between precision and recall, and is more forgiving than EM; it produces a score between 0 and 1 depending on the degree of overlap between the predicted and ground-truth answers.

4.3 Experiments

For all training runs, we hold the training hyperparameters constant: we train for 3 epochs with a learning rate of `3e-5`, using a random seed of 42. This applies to both the baseline pre-training step, as well as the additional finetuning step (so a fine-tuned model will have been trained for a total of 6 epochs: 3 long epochs for the baseline and 3 shorter epochs for finetuning).

4.3.1 Baseline, vanilla finetuning, and traditional augmentations

To obtain our baseline model, we train the DistilBERT model on the 150,000 samples from the in-domain datasets (SQuAD, NewsQA, and Natural Questions).

For our "vanilla fine-tuned model," we start with the baseline model from above, and fine-tune it on the 381 samples from the out-of-domain datasets (DuoRC, RACE, and RelationExtraction). We refer to this as "vanilla" finetuning because it does not include any augmented data.

For traditional data augmentation (backtranslation and synonym replacement), we train on both the 381 original out-of-domain samples, as well as their augmented versions.

Performance of these models is shown in Table 3 in section 4.4 below.

4.3.2 QGA: Basic version

Next, we finetune a model using the data generated from basic QGA. Here, "basic" refers to the fact that we just run the full training passages through the original QGA pipeline, without any modifications (e.g., sentence chunking or the filtering module). Performance of this model is shown in Table 3 in section 4.4 below.

4.3.3 QGA: Filtering module

Next, we perform experiments with the filtering module. We take one of our generated QGA datasets (chunk size = 3, number of QA pairs generated per sentence = 3) and pass all of the generated QA pairs through the filtering module to get our reduced dataset.

As mentioned earlier, we can choose the threshold between high- vs. low-quality questions, which controls the tradeoff between generating less (but higher-quality) vs. more (but lower-quality) data.

In Table 2, "% kept" indicates the percent of generated QA pairs that make it past the filtering step. We try a range of different filtering thresholds, from F1 = 0.0 (no filtering) up to 1.0 (exact match).

		All Validation		RACE		RE		DuoRC	
Filter threshold	% kept	F1	EM	F1	EM	F1	EM	F1	EM
F1 = 0.0 (no filtering)	100	51.81	34.82	38.87	19.53	74.14	55.47	42.27	29.37
F1 = 0.2	66.2	52.52	35.86	**39.18**	**21.88**	75.48	55.47	42.75	30.16
F1 = 0.4	61.7	52.76	36.39	37.77	**21.88**	75.22	53.91	45.19	33.33
F1 = 0.6	52.6	51.62	35.34	31.14	16.41	77.16	56.25	**46.47**	33.33
F1 = 0.8	41.0	52.55	36.91	36.30	20.31	76.99	**58.59**	44.24	31.75
F1 = 1.0 (exact match)	35.7	**52.98**	**37.17**	35.96	19.53	**77.19**	57.81	45.66	**34.13**

Table 2: Performance of QGA models with varying filtering module thresholds.

We find that the most stringent filtering (F1 = 1.0, which keeps only the highest-quality QA pairs) performs the best overall. This is an interesting result, because it suggests that it is better to generate higher-quality data, even if that means we generate less of it.

4.3.4 QGA: Improving performance on long contexts with sentence chunking

As we mentioned earlier, QGA can struggle with long context passages, so we use sentence chunking to mitigate this issue. Here we evaluate chunk sizes from 1-10 sentences to see if we can find any trends. In Figure 3, we break down the results by dataset.

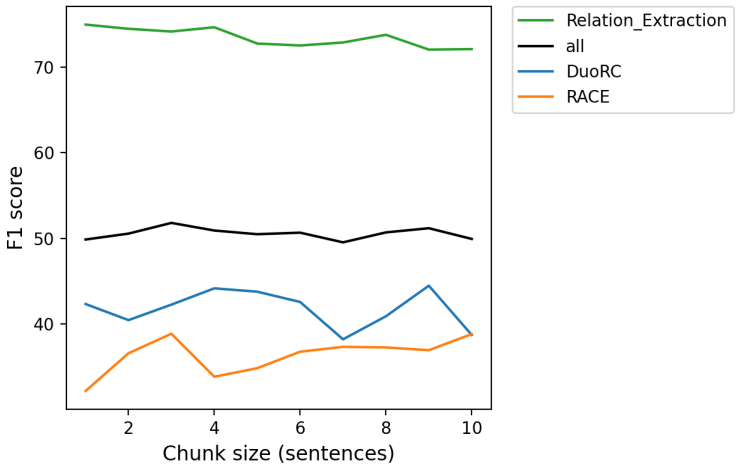


Figure 3: Performance after training on data generated via QGA with different chunk sizes.

The trends are not incredibly dramatic, but we can make some interesting observations.

For RelationExtraction, there is a slight downward trend as chunk size increases. RelationExtraction contains mostly short (1-sentence) contexts, so it makes sense that it would perform best when the generated questions are also generated on relatively small chunk sizes. For DuoRC, it is hard to identify much of a trend. For RACE, which has longer context passages than RelationExtraction, we can see a slight upward trend as chunk sizes increases. This makes sense as well; since RACE passages are longer, some of the questions require synthesizing knowledge across multiple sentences, so generating questions on longer chunks could be beneficial.

4.3.5 QGA: Tuning generation hyperparameters

Next, we perform a grid search over two key QGA hyperparameters. The first is chunk size, which is the same one we examined in the previous section. The second is the number of QA pairs to generate per sentence. If we generate more QA pairs per sentence, we will have more data; however, it may be lower-quality and/or more repetitive. Results are shown in the heatmap in Figure 4 below.

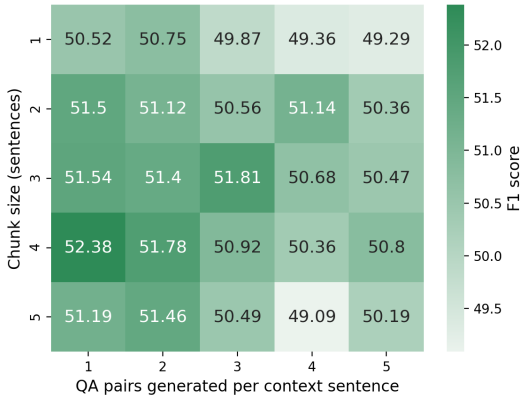


Figure 4: QGA hyperparameter search: validation dataset F1 scores, across various chunk sizes and numbers of QA pairs generated per sentence.

The results are quite informative. Clearly, generating fewer QA pairs per sentence is better, since the left side of the plot has noticeably better results than the right side. This suggests we should generate fewer but higher-quality sentences–which, interestingly, also matches the conclusion we reached in our filtering module experiments.

For the chunk size, it seems that using a moderate value (2-4 sentences) is best, since there is a dark green horizontal stripe in that part of the heatmap. This also makes some intuitive sense; by using moderate chunk sizes, we might get more complex questions than if we only used single-sentence chunks, but we also don't increase the chunk size so much that the QGA models output nonsense.

4.4 Summary of main results

In Table 3 below, we summarize the main results from all of our experiments. For the QGA experiments above (i.e., the filtering module and the hyperparameter grid), we list only the best model from that experiment.

	All Validation		RACE		RelationExtraction		DuoRC	
Model	F1	EM	F1	EM	F1	EM	F1	EM
Baseline	50.06	34.03	**37.51**	**22.66**	69.67	46.88	42.89	32.54
Vanilla finetuning	50.16	34.29	36.98	21.88	70.49	48.44	42.89	32.54
Synonym replacement	50.15	34.03	36.88	21.09	70.95	49.22	42.50	31.75
Back-translation	50.04	34.29	36.63	21.88	70.88	49.22	42.51	31.75
QGA (basic)	50.45	35.60	36.57	20.31	74.41	54.69	40.22	31.75
QGA: best from hyperparameter grid	52.38	36.13	35.27	19.53	76.39	55.47	45.37	33.33
QGA: best filtering module (F1 = 1.0)	**52.98**	**37.17**	35.96	19.53	**77.19**	**57.81**	**45.66**	**34.13**

Table 3: Summary of main results.

In general, all of our QGA approaches outperform the baseline models. Overall, the best model is the QGA model with our filtering module (at F1 threshold = 1.0).

QGA improves performance on RelationExtraction most dramatically (+6.70 F1 over vanilla finetuning). It also improves performance on DuoRC noticeably (+2.77 F1). It doesn't seem to help on RACE (-1.02 F1). RelationExtraction performance may improve so dramatically because the QGA models are better at generating questions for short contexts, which caters more towards RelationExtraction than towards the other two datasets (although we still get a sizable boost on DuoRC).

On the RobustQA hidden test set, we evaluate our two best models (the bottom two rows of the table). Our best QGA hyperparameter grid model achieves 60.37 F1 and 43.03 EM, and our best filtering module model achieves 60.52 F1 and 42.82 EM. Overall, we are pleased with our results, since QGA improved our results almost across the board.

5 Analysis

5.1 Manual analysis of generated QA pairs

Some example QA pairs generated by the PAQ pipeline are found in the table below.

Context: *Ray Eberle died of a heart attack in Douglasville, Georgia on August 25, 1979, aged 60.*

	Question	Answer
Original	Why did Ray Eberle die?	heart attack
Generated #1	Where did ray eberle die of a heart attack?	Douglasville, Georgia
Generated #2	How old was ray eberle when he died?	60
Generated #3	Who died of a heart attack in georgia?	Ray Eberle

The generated questions are very high-quality. All three generated QA pairs are correct and grammatically sound, and they are also diverse in that they ask about different parts of the sentence.

However, as we've discussed, QGA can struggle with long context passages. Here is an example that we generate on a much longer context.

Context: *[long passage about The Sword in the Stone (36 sentences)]*

	Question	Answer
Generated #1	What is the name of the owl in harry potter?	Archimedes
Generated #2	What is the name of arthur's brother in harry potter?	Sir Kaye

In this case, the question generation model gets confused and asks about Harry Potter, which is completely unrelated to the original passage. The model seems to recognize that the passage is about England and is from the fantasy genre, and wrongly assumes that it is from Harry Potter. This is why we use sentence chunking, since it allows the model to focus more closely on a limited number of sentences without having to fully internalize the entire passage.

5.2 Filtering module

We saw above that, empirically, our filtering module achieves strong results. However, we want to point out one important aspect of the filtering procedure: it is somewhat self-reinforcing, since with a high F1 threshold, it only keeps QA pairs that the vanilla finetuned model already performs well on. So, we likely discard many difficult examples that the model would actually benefit from training on. However, risking discarding good QA pairs is somewhat necessary; to perfectly filter the low- vs. high-quality questions automatically, we would need to have already solved the extractive QA task. So, even though we may be throwing out some good, difficult examples, our results suggest that filtering is beneficial because we are also throwing out most of the worst QA pairs. And of course, by tuning the F1 threshold, we can control this tradeoff between keeping more difficult examples (at the expense of also keeping more low-quality examples) vs. keeping more high-quality examples (at the expense of discarding more difficult examples).

5.3 Other comments

When examining our training data, we noticed that in terms of total sentences in the original context passages, RACE is by far the largest (almost 80% of total sentences). Therefore, RACE has a disproportionately large representation in the data generated by QGA with our sentence-chunking pipeline. For this project, we wanted to maximize the amount of data we could augment from, so we left it as is, but moving forward it would be interesting to examine whether this balance affects the final results.

6 Conclusion

In this paper, we explore QG Augmentation, a synthetic data augmentation technique that generates novel QA pairs from training passages. We find that QGA significantly improves performance in the few-shot setting (+2.82 F1, +2.88 EM over the vanilla fine-tuned model). We additionally introduce a filtering module to filter out low-quality generated questions, which proves to be effective and produces our most accurate model overall. We also show that sentence chunking can help improve QGA performance on long context passages, and we tune two key hyperparameters in the generation pipeline.

One limitation of QGA is that, while it dramatically improves performance on short-context datasets like RelationExtraction (+6.70 F1), it improves performance less, or even worsens performance, on long-context datasets like DuoRC (+2.77 F1) and RACE (-1.02 F1). We think that this discrepancy in performance across datasets stems from our generated QA pairs being worse for longer contexts; even with sentence chunking, we likely cause the QA pairs to be more local/less complex. In the future, we would like to improve the question generation model's ability to process long contexts so that we can generate higher-quality QA pairs on them, potentially even without sentence chunking.

References

[1] Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. Synthetic QA corpora generation with roundtrip consistency. *arXiv preprint arXiv:1906.05416*, 2019.

[2] Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. Unsupervised question answering by cloze translation. *arXiv preprint arXiv:1906.04980*, 2019.

[3] Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. Generative data augmentation for commonsense reasoning. *arXiv preprint arXiv:2004.11546*, 2020.

[4] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*, 2019.

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

[6] Tom Hosking and Sebastian Riedel. Evaluating rewards for question generation models. *arXiv preprint arXiv:1902.11049*, 2019.

[7] Yuwei Fang, Shuohang Wang, Zhe Gan, Siqi Sun, Jingjing Liu, and Chenguang Zhu. Accelerating real-time question answering via question generation. *arXiv preprint arXiv:2009.05167*, 2020.

[8] Max Bartolo, Tristan Thrush, Robin Jia, Sebastian Riedel, Pontus Stenetorp, and Douwe Kiela. Improving question answering model robustness with synthetic adversarial data generation. *arXiv preprint arXiv:2104.08678*, 2021.

[9] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[10] Distilbert. `https://huggingface.co/docs/transformers/model_doc/distilbert`. Accessed: 2022-03-02.

[11] MarianMT. `https://huggingface.co/docs/transformers/model_doc/marian`. Accessed: 2022-02-22.

[12] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

[13] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[14] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[15] Sentencizer - spaCy API Documentation. `https://spacy.io/api/sentencizer`. Accessed: 2022-03-02.

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

[17] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*, 2016.

[18] Amrita Saha, Rahul Aralikatte, Mitesh M Khapra, and Karthik Sankaranarayanan. Duorc: Towards complex language understanding with paraphrased reading comprehension. *arXiv preprint arXiv:1804.07927*, 2018.

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

[20] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension, 2017.