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

Stanford University

Ben Alexander[1], Gordon Downs[1]

CS224N: Natural Language Processing with Deep Learning

[1]Department of Computer Science, Stanford University

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. 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).

Background

In our few-shot setting (Robust QA project track), we are provided with three extractive QA training datasets, each with 127 samples. The datasets are:

- **RACE**, from reading comprehension exams for middle and high school students
- **RelationExtraction** (RE), with questions about relationships between entities
- **DuoRC**, from movie plot summaries

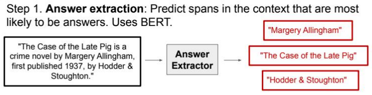
Typical data augmentation techniques, such as backtranslation and synonym replacement, perform small, local perturbations of existing QA pairs. In contrast, our strategy, which we call "QG Augmentation" or "QGA," involves automatically extracting novel QA pairs from the training passages.

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: an answer extractor and a question generator (more on this below). The PAQ project also includes a third model, for open-domain question answering, that they use for filtering out low-quality generated questions. Their filtering model is not applicable for our use case, so we develop our own filtering module instead.

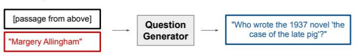
Methods

Basic QG Augmentation

Step 1. **Answer extraction**: Predict spans in the context that are most likely to be answers. Uses BERT.



Step 2. **Question generation**: Given the context passage and an answer, generate a question. Uses BART-base.

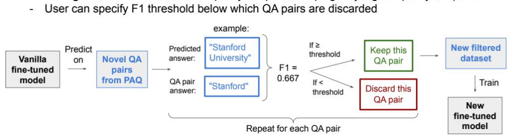


"Traditional" Augmentations

1. **Backtranslation** - Translate question to French, then back to English.
2. **Synonym replacement** - Randomly replace words in questions with synonyms.

QG Augmentation (with improvements)

1. **Filtering module** - Discards lower-quality QA pairs, keeping only higher-quality QA pairs.
   - User can specify F1 threshold below which QA pairs are discarded

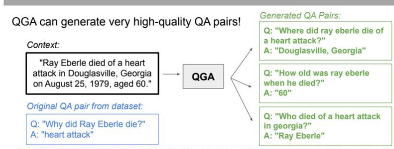


2. **QGA pipeline optimizations** - We add two modifications to our QGA pipeline:
   - We split long context passages into shorter chunks before passing them through PAQ, since PAQ appears to produce higher-quality QA pairs on shorter contexts.
   - We vary the number of QA pairs generated per sentence (fewer high-quality QA pairs, vs. more but potentially lower-quality QA pairs).

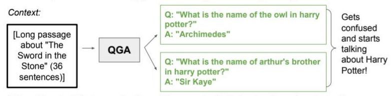
Experiments + Analysis

Example Generated QA Pairs

QGA can generate very high-quality QA pairs!



However, QGA often struggles with longer context passages:



We mitigate this issue by breaking contexts into shorter chunks before generating QA pairs. See our experiments (right).

Filtering Module

We vary the threshold for our filtering module. We find that the most stringent filtering (F1 = 1.0, which keeps only the highest-quality QA pairs) performs best.

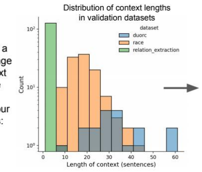
Filter threshold	% kept	All	RACE	RE	DuoRC
F1 = 0.0 (no filtering)	100	51.81	38.87	74.14	42.27
F1 = 0.2	66.2	52.52	**39.18**	75.48	42.75
F1 = 0.4	61.7	52.76	37.77	75.22	45.19
F1 = 0.6	52.6	51.62	31.14	77.16	**46.47**
F1 = 0.8	41.0	52.55	36.30	76.99	44.24
F1 = 1.0 (exact match)	35.7	**52.98**	35.96	**77.19**	45.66

The right 4 columns contain F1 scores for each validation set.

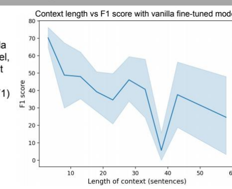
% kept indicates the percent of generated QA pairs that make it past the filtering step.

PAQ Generation Optimizations

There is a wide range of context passage lengths across our datasets:



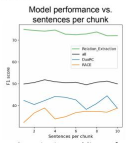
Using our vanilla fine-tuned model, we observe that validation performance (F1) decreases for longer context passages.



Long contexts pose two distinct challenges:
1. QGA struggles to generate QA pairs for long passages (e.g. Harry Potter examples)
2. Our model already performs worse on long passages *even before* QGA (they are more difficult).

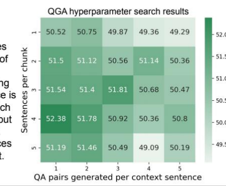
#2 is out of scope, but we attempt to mitigate #1.

To improve QGA performance on long contexts, we break up passages into chunks before generating QA pairs on them. Here we evaluate chunk sizes from 1-10 sentences.



RE contains mostly short (1-sentence) contexts, and its performance improves with smaller chunk sizes. RACE has larger contexts, and its performance improves with larger chunk sizes. DuoRC does not show a clear trend.

Finally, we perform a grid search over two QGA hyperparameters: sentences per chunk and the number of QA pairs to generate per sentence. Clearly, generating fewer QA pairs per sentence is better (left side of plot), which means we generate fewer but higher-quality sentences. A moderate value for sentences per chunk (2-4) seems best.



Summary of Results

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

QGA improves performance on RE most dramatically (+6.70 F1). Performance on DuoRC also improves noticeably (+2.77 F1). It doesn't seem to help on RACE (-1.02 F1).

Validation set performance by model and dataset

	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 hyper-parameter 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**

Conclusions

- Generating novel QA pairs with QGA significantly improves performance in the few-shot setting (+2.82 F1, +2.88 EM).
- QGA improves over basic "traditional" augmentation methods like backtranslation and synonym replacement, perhaps because it generates novel QA pairs rather than just perturbing already-existing QA pairs.
- Using a filtering module to filter out low-quality generated questions is quite effective.
- Tuning the chunk size and number of sentences generated per sentence is also beneficial.
- QGA improves performance on RelationExtraction the most. This may be because QGA is better at producing "local" QA pairs, as compared to synthesizing long-term information from across long passages. This caters to RE since it mainly consists of short 1-sentence contexts, in contrast to DuoRC and RACE, which have much longer contexts (see histogram above).

References

[1] Lewis, Patrick, et al. "Paq: 65 million probably-asked questions and what you can do with them." *Transactions of the Association for Computational Linguistics* 9 (2021): 1098-1115.