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 repurposed 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 (GQ) 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 ($\tau$ 2.62 F1, $\tau$ 2.88 EM vs. vanilla fine-tuning).

Background
In our few-shot setting (Robust QA project track), we are provided with three extractive QA training datasets, each with 127 examples. 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 back-translation and synonym replacement, perform small, local perturbations of existing QA pairs. In contrast, our strategy, which we call QG Augmentation or QGQA, involves automatically extracting novel QA pairs from the training passages.

We implement QG Augmentation using part of the question generation pipeline from the “Probabilistic Questioning” (PQG) project from Facebook AI Research [1]. We borrow two modules from the PQG project to construct our QG augmentation pipeline: an answer extraction and a question generation module, more on this below. The PQG project also includes a third module, for open-domain question answering, that we 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
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. QG pipeline optimizations - We add two modifications to our QA pipeline:
   - We filter long context passages into shorter chunks before passing them through PAQ2, since PAQ2 appears to produce higher-quality QA pairs on shorter contexts.
   - We vary the number of QA pairs generated per sentence (lower/high-quality QA pairs, vs. more but potentially lower-quality QA pairs).

Experiments + Analysis
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.

Summary of Results

Conclusions

References