Improving a QA System for SQuAD using Attention- and Augmentation-based Methods

Amelia Woodward [1] Angela Zhao [1] Stone Yang [1]

[1]Computer Science, Stanford

Problem

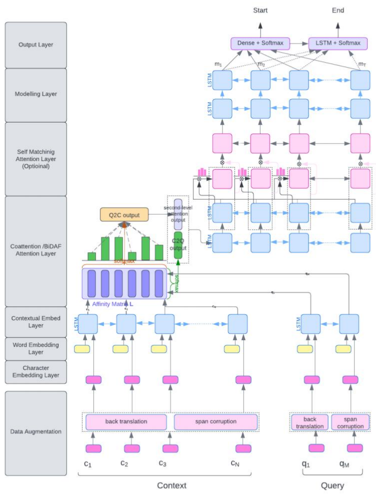
The problem we are trying to solve is to improve the F1 and EM scores for a QA system for SQuAD. Challenges with the baseline BiDAF model include lack of data augmentation, as well as simplicity of embedding layer and coattention layer. Our work is meaningful in that it explores these three overlooked areas and improves the baseline by [TODO: fill in] points in accuracy.

Background

QA systems have two levels of significance. From a research perspective, it serves as a measure for how well systems can 'understand' text. From a practical perspective, they are useful for better understanding any piece of text. To improve the QA system for SQuAD, we start from a baseline Bidirectional Attention Flow (BiDAF) model [4]. Since the baseline model only includes word level embedding, the first approach we used is adding character level embedding. Then, we explored coattention [7], self-attention [6], span corruption [3] and back-translation [2].

Methods

We explored three main areas of improvements: data augmentation, embedding layer, and attention layer. This is an overview of our structure.



Data Augmentation

We use two techniques for data augmentation.

- **Span Corruption**: Each example has a 30% chance of having part of the context replaced by a span of OOV tokens [3]. If part of the answer was corrupted, then the example is marked as one with no answer. Otherwise, the original answer is kept.
- **Back Translation**: For each example, we first attempted translating both the query and context into German and then back into English. Then we updated the answer [2]. Due to compute limitations, we backtranslated only the query on 2.5% of the data.

Embedding Layer

We added a character-level embedding layer to the BiDAF model. This layer maps each word to a vector space using character-level Convolutional Neural Networks (CNNs). The vectors are 1D inputs to the CNN. The CNN outputs are max-pooled over the entire width to obtain a fixed-size vector for each word [1].

Attention Layer

We implemented both co-attention and self-attention.

- **Coattention** refers to attending over representations that are themselves attention outputs. We implement the Coattention Layer following Xiong (2016)'s approach for Dynamic Coattention Networks [7].
- **Self-attention** means the hidden state $\mathbf{h_t}$ attends to all the previous hidden states so far $\mathbf{h_1}, \ldots, \mathbf{h_{t1}}$. We implement the Self-Matching Attention Layer from R-Net [6].

Experiments

- **Data**: We use the official SQuAD 2.0 dataset with three splits: train, dev and test as described in the default SQuAD handout. The three splits in detail:
  - train (129,941 examples): All taken from the official SQuAD 2.0 training set.
  - dev (6078 examples): Roughly half of the official dev set, randomly selected.
  - test (5915 examples): Remaining examples from official dev set, plus hand-labeled examples.
- **Evaluation method**: We will evaluate based on two key metrics.
  - Exact Match: a binary measure. It is equal to 1 if exactly matches the ground truth.
  - F1: calculates the harmonic mean of precision and recall. Harmonic mean: the reciprocal of the arithmetic mean of the reciprocals of the given set of observations.

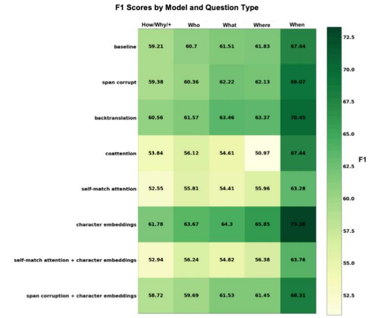
  When evaluating on the validation or test sets, we take the **maximum** F1 and EM scores across the three human-provided answers for that question. This makes the evaluation more forgiving. Additionally, we look at the negative log-likelihood (NLL) and AvNA. AvNA measures the classification accuracy of answer vs no answer predictions.
- **Experimental details**: For the baseline model, we follow the provided configurations, where:
  - Learning rate: 0.5
  - Training time: 30 epochs, or about 3 hrs
- **Results**:

Model	Dev NLL	F1	EM	AvNA
Baseline	3.10	61.47	58.04	68.26
Span Corrupt Only	3.185	60.64	57.42	67.95
Back-translation	3.106	61.74	59.21	71.22
Coattention	3.42	55.28	51.60	63.65
Self-matching attention	3.08	62.55	59.11	69.04
Character Embedding	**2.98**	**64.10**	**60.80**	**70.78**
self-matching attention + character embedding	02.75	62.55	59.23	68.54
Span corruption + character embedding	3.178	59.97	56.49	67.11

Table 1. Results table

Analysis

We categorized the questions to analyze the model's performance in different cate-



gories.

- Models performed best on When questions, which is intuitive based on the simplicity of date finding.
- Attention-based models exhibited more variable performance across question categories.
- Coattention was poor at where questions and great at when questions, while embeddings-based models gave a boost across the board.

Conclusions

- **Character embeddings**, implemented alone, produce the greatest improvement in baseline.
- Improvement methods, when combined, did not always lead to an improvement.
  - Character embedding did not train well with data augmentation or our two implementation designs of attention, coattention or self-matching attention. We did, however, see a 1 point lift in F1 using self-matching attention alone.
  - Different methods require different hyperparameters to perform well.
- **Model performance varies** across different question categories.
  - All models performed the best on "When" questions
  - Coattention is particularly bad at where and what questions but stands out at the when questions. Since the dataset has more what questions, this may help it do better than the where questions. This feeds into the hypothesis that coattention is better handling shorter phrases and worse at longer phrases.

References (Part)

[1] Yahui Chen.
Convolutional neural network for sentence classification.
Master's thesis, University of Waterloo, 2015.

[2] Shayne Longpre, Yi Lu, Zhucheng Tu, and Chris DuBois.
An exploration of data augmentation and sampling techniques for domain-agnostic question answering.
arXiv preprint arXiv:1912.02145, 2019.

[3] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and