Improving Numerical Reasoning in Question Answering CSE 481 N

Tyler Ohlsen, Ravi Patel, Blarry Wang
{tlohlsen,patelr3,zwan48}@cs.washington.edu

June 2019

Abstract

In this paper we introduce a new question-answering (QA) model for solving the Discrete Reasoning Over Paragraphs (DROP) benchmark dataset [Dua et al., 2019]. The DROP dataset challenges models to not only extract relevant information from a passage and question, but also perform discrete operations over them (counting, addition/subtraction, etc.). We introduce the **C**ontextual and **N**umerically **A**ugmented **QANet** (**CNAQANet**), which through contextualized word embeddings and a modified number extraction method, improves upon the performance of the current state-of-the-art model, NAQANet, established by Dua et al. [2019]. CNAQANet achieved a performance of 48.86% EM and 52.09% F1 on DROP, a significant increase compared to NAQANet's EM and F1 scores of 44.24% and 47.24%, respectively.

1 Introduction

Question answering (QA) tasks continue to receive focus in the NLP community. The Stanford Question Answering Dataset (SQuAD) has recently been a very popular corpus related to this task [Rajpurkar et al., 2016]. Current state-of-the-art models on it are reaching near-human performance [Rajpurkar et al., 2018]. As a result, more complex QA datasets have been created, requiring models to perform deeper and more complex reasoning than what is necessary for SQuAD. DROP is one of those datasets, and we aim to improve upon the existing baseline for DROP established by Dua et al. [2019].

Specifically, this dataset requires a system to be able to handle arithmetic reasoning such as addition, subtraction, and counting, as well as the passage span extraction that is necessary in datasets like SQuAD. The best performing model on DROP is the Numerically Augmented QANet (NAQANet) [Dua et al., 2019].

NAQANet inherits from QANet [Yu et al., 2018] the ability to answer questions using passage spans. Additionally, it includes abilities to answer questions using question spans, perform counting operations, and perform addition/subtraction operations. For the addition/subtraction operations, NAQANet extracts numbers from the passages and questions, then uses a neural model to predict how these numbers can be combined to evaluate to a numerical answer.

Our contributions include several modifications to the NAQANet design. Specifically, we modified number extraction, incorporated BERT, and designed a new counting method using a bidirectional LSTM.

2 Technical Design

NAQANet has clear weaknesses related to arithmetic reasoning. Although the model achieves significantly higher scores than the existing QA baselines on the DROP dataset, it still performs poorly on addition/subtraction and counting questions. 51% of NAQANet's errors on 100 randomly sampled erroneous predictions are related to arithmetic operations and 30% were related to counting [Dua et al., 2019]. To improve upon these weaknesses, we identified three potential modifications to the existing design: a more powerful and effective number extraction method, the introduction of contextual word embeddings, and a redesigned system for predicting answers to counting questions.

2.1 Number Extraction

To extract numbers from passages or questions, NAQANet first splits each piece of text into different tokens. It further splits each token again on hyphens, so a token like “13-yard” gets split into the tokens “13” and “yard”. The model then takes any token that it can successfully convert into a number and passes it through the addition/subtraction feedforward layer. We seek to improve upon this number extraction process.

NAQANet tries to convert a token to a number in two ways. First, it attempts to directly cast token strings into integers, so “13” converts to the integer 13. If that fails, it attempts to convert word-number tokens like “six” to numbers using a very small, rudimentary mapping of words to numbers. This results in several examples that NAQANet fails to convert, such as “twenty-one”, “thirty”, and “5.1”.

We modify number extraction in two ways. First, we use the Python library word2number to extract more numbers. Using word2number, our model extracts numbers that were previously being missed, such as “thirty” and “5.1”, as well as hyphenated tokens, such as “twenty-one”. Our second modification was to not split tokens on hyphens before attempting to convert to a number. NAQANet’s implementation of hyphen splitting before number extraction introduced additional irrevelant numbers. For example, “twenty-one” would result in the numbers 20 and 1 being extracted, when the single number 21 is the intended/desired result. Instead, word2number handles such hyphenated cases. Overall, our modified number extraction method extracted 19,003 more numbers on the training set, an increase of 19.8% compared to NAQANet.

2.2 BERT

The introduction of contextualized word embeddings has shown to largely improve benchmark scores for a variety of NLP tasks, including QA. One popular model used to produce such embeddings is Google’s Bidirectional Encoder Representations from Transformers (BERT) model [Devlin et al., 2019]. For our implementations, we incorporate Hugging Face’s pretrained BERT model ‘BERT-Base Uncased’. Using this model, we obtain BERT contextualized embeddings for the entire dataset. In the original NAQANet model, character-level embeddings as well as 300-dimensional pretrained GloVe embeddings are used [Pennington et al., 2014]. In our new model, we concatenate our new BERT embeddings onto these original embeddings. Our pretrained BERT model can only produce embeddings based on input passages that are 512 ‘wordpieces’ or shorter, so we are forced to truncate passage lengths down to this limit (which is approximately 330 tokens) [Devlin et al., 2019]. For passages longer than this limit, our model truncates them to only contain the first 330 tokens. BERT can be adjusted to higher wordpiece limits, but we did not have the necessary resources to create and train such a model. Increasing the wordpiece limits would allow higher passage limits, and therefore add more context and words for a model to train and predict on.

2.3 Counting - BiLSTM

NAQANet models the counting capability as a multi-class classification problem. Specifically, it uses the calculated passage vector $\mathbf{h}^P$, passes that through a feed-forward network (FFN), and produces a set of probabilities over the integers 0 - 9 [Dua et al., 2019]:

$$\mathbf{p}^{\text{count}} = \text{softmax}(\text{FFN}(\mathbf{h}^P))$$

There are many limitations of this method. It does not generalize well, and is only be able to correctly answer counting questions with answers in the range [0,9]. Additionally, by only including inputs related to the passage (and not the question), it may be difficult for the model to know what is relevant in the passage.

Model	Dev (arithmetic)		Dev (span)		Dev (full)	
	EM	F1	EM	F1	EM	F1
Simple Baseline	*27.78%*	*27.96%*	*43.95%*	*51.21%*	*33.69%*	*36.48%*
Simple Baseline + Counting-BiLSTM	7.32%	7.44%	16.10%	19.09%	9.81%	10.81%
Simple Baseline + Modified NE*	34.01%	34.20%	43.55%	50.75%	37.50%	40.30%
Simple Baseline + BERT	30.63%	30.75%	**46.95%**	**55.03%**	36.69%	39.77%
Simple Baseline + Modified NE* + BERT	**34.73%**	**34.78%**	46.76%	53.74%	**39.16%**	**41.82%**

Table 1: Comparison of design improvements on Simple Baseline. All models were trained for 10 epochs. *NE stands for number extractor

We introduce a more generalizable method to produce counts from passages. Using a neural model, we produce class probabilities for each word in a passage, labeling them as 0 (shouldn't be counted), or 1 (should be counted). We sum over these values to produce a final count prediction. For example, given the question "How many interceptions did Tom Brady throw?", a model would assign high 1-probability values for words in the passage like "interception" or "INT", and low 1-probability values for irrelevant words, like "the" or "Sunday".

Specifically for our model, we first encode the concatenation of the passage word embeddings $\mathbf{e}^{|\mathbf{P}|}$ and the question vector $\mathbf{Q}$ using a BiLSTM. We then pass these encodings through a feedforward neural network:

$$\mathbf{p}^{\text{counts}} = \text{softmax}(\text{FFN}(\text{BiLSTM}([\mathbf{Q}; \mathbf{e}^{|\mathbf{P}|}])))$$

The output from the feedforward layer is passed into a softmax activation that produces the probabilities for 0 and 1. The final count is obtained by summing over the 1-probabilities of each word and rounding to the nearest integer. We use mean squared error (MSE) loss for the counting sub-task. This loss is incorporated into the model's total loss [Dua et al., 2019].

3 Results

3.1 Performance

Due to limited computing resources and time constraints, we introduced a simpler version of the NAQANet baseline in order to speed up training times. We call this model *Simple Baseline*, which contains smaller encoding blocks, truncates passages that are read in to 200 tokens, and is only trained for 10 epochs. Using this new baseline, we created and tested several augmented models, each containing different individual modifications. All models were ran for 10 epochs on the training set. To determine how well these individual models were performing on different types of questions, we split up the development set into an arithmetic-only portion (containing only addition/subtraction and counting questions), and a span-only portion (containing only passage-span and question-span questions). Our results are shown in Table 1. We chose the optimal model from Table 1 and fully trained it for 50 epochs, using the original number of encoding blocks used in NAQANet, and increasing the passage limit to 330. We call this fully trained model CNAQANet, and its performance comparison to NAQANet is shown in Table 2.

3.2 Analysis

As shown in Table 1, for arithmetic-based questions, our modified number extraction model increased EM and F1 scores by over 6% after the first 10 epochs. Additionally, for span-based questions, our BERT-combined model increased EM and F1 scores by over 3% after the first 10 epochs. Our combined model, using both the modified number extraction method and BERT contextualized embeddings, achieves higher

Model	Dev (arithmetic)		Dev (span)		Dev (full)		Hidden Test Set	
	EM	F1	EM	F1	EM	F1	EM	F1
NAQANet	44.55%	44.79%	50.69%	60.17%	46.75%	49.87%	44.24%	47.24%
CNAQANet	**49.43%**	**49.64%**	**53.07%**	**60.73%**	**50.46%**	**53.87%**	**48.86%**	**52.09%**

Table 2: Comparison of CNAQANet and NAQANet. Both models were trained for 50-60 epochs.

scores than the simple baseline on the full development set after 10 epochs. As seen in Table 2, our fully trained CNAQANet clearly outperforms NAQANet. Our model achieves an increase of over 3% on EM and F1 scores for the development set, and an increase of over 4.5% on EM and F1 scores for the hidden test set.

Our modified counting method did not improve performance (see Table 1), therefore we omitted this modification from CNAQANet. Correctly predicting counts for each word in several-hundred word passages is a very complex task for a system to successfully be able to do. Although our initial approach did not yield superior results, we think it does show future promise (see Future Work).

4 Related Work

4.1 QANet

The QANet model produces a span from a given passage to answer a given question. It uses convolutions and self-attention as encoder blocks in place of traditional RNNs. This allows the model to speed up its performance, as well as achieve state-of-the-art performance on SQuAD [Yu et al., 2018].

4.2 NAQANet

NAQANet uses an additional encoder block compared to QANet to obtain a numerical passage representation [Dua et al., 2019]. By introducing this fourth encoding block, and by introducing new model abilities including question span, addition/subtraction, and counting, NAQANet was able to achieve state-of-the-art performance on DROP, outperforming BiDAF, QANet (+ELMo), and BERT models [Dua et al., 2019].

5 Future Work

While our model does improve upon the original NAQANet model, we recognize that there are many areas in which further improvements can be made. These include:

- Using different granularity levels for our introduced counting method, such as sentence counts rather than word or passage counts. Parsing and predicting counts over verb phrases or noun phrases may also perform better related to counting-based questions.
- Use other pretrained models for contextualized embeddings. Using a BERT model that can handle higher passage length limits, or combining our model with ELMo, may yield better results.

6 Conclusion

CNAQANet supports the idea that improved number extraction and the use of contextualized word embeddings results in a system better understanding questions and passages related to the discrete operations described in DROP. Although we made improvements on NAQANet, advanced models still perform poorly; human performance on DROP is 96.42% while CNAQANet only performs at 52.09% [Dua et al., 2019].

Acknowledgments

Thanks to Noah Smith, Elizabeth Clark, Lucy Lin, and Yizhong Wang for helpful discussions and supporting us throughout this experience. Also, thanks to Doruk Arisoy, Divye Jain, Dan Tran, Yuchong Xiang, Kevin Zhao, Jeffrey Da, Vardhman Mehta, Garrett Zhang, and Chia-ko Wu for insighful feedback on this paper.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding, May 2019. URL `https://arxiv.org/pdf/1810.04805.pdf`.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matthew Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *CoRR*, abs/1903.00161, April 2019. URL `https://arxiv.org/pdf/1903.00161.pdf`.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL `https://www.aclweb.org/anthology/D14-1162`.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text, October 2016. URL `https://arxiv.org/pdf/1606.05250.pdf`.

Pranav Rajpurkar, Robin Jia, and Percy S. Liang. Know what you don't know: Unanswerable questions for squad. In *ACL*, June 2018. URL `https://arxiv.org/pdf/1806.03822.pdf`.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. QANet: Combining local convolution with global self-attention for reading comprehension, April 2018. URL `https://arxiv.org/pdf/1804.09541.pdf`.