Appendix: Improving Answer Selection and Answer Triggering using Hard Negatives

Sawan Kumar [*]
Indian Institute of Science, Bangalore
sawankumar@iisc.ac.in

Shweta Garg [†]
Search Customer Experience, Amazon
shwegarg@amazon.com

Kartik Mehta
India Machine Learning, Amazon
kartim@amazon.com

Nikhil Rasiwasia
India Machine Learning, Amazon
rasiwasi@amazon.com

A Training Details

The models (except for the Transformer encoder model, details below) in this work are trained using Keras (V2.2.2). We use Adam optimizer (Kingma and Ba, 2014) for all the experiments, with a fixed learning rate of 0.001. Batch size for training was varied between 250, 500 and 1000. For all answer selection tasks, we used triplet loss with margin of 0.2 for InsuranceQA and LargeQA and 0.05 for SelQA. For the answer triggering task, we used quadruplet loss, with the margins m_1 and m_2 kept fixed to 0.1 and 0.05 respectively. In each case, we select best epoch and best hyper-parameter using performance on the development set. Each variant was run for a total of 200 epochs.

For Max-Pooling and Max-Min-Pooling models, the word embeddings are randomly initialized with size 500/1000/2000, and learned as part of the training.

For LSTM-CNN models, the input embedding size is kept fixed to 100. The embeddings are initialized with word2vec (Mikolov et al., 2013) embeddings trained on the respective dataset, and are optimized as well during the training. The dimension of LSTM output is set to be 141 and 500 CNN filters of sizes 1,2,3 and 5 are used. The outputs from CNN is then concatenated after max pooling to produce the final representation.

For the Transformer encoder, we use pytorch (V1.2.0)[1]. We employ the architecture used in BERT (Devlin et al., 2019) (BERT$_{\text{BASE}}$) as our base model, which has 12 layers, hidden size of 768 and 12 attention heads. We use the embedding of the first token from the final layer to get the representation of a sentence. The encoder is initialized using a pre-trained uncased BERT model.

[*] Work done as an intern at Amazon
[†] Work done at India Machine Learning, Amazon
[1] We used the library available at https://github.com/huggingface/pytorch-transformers

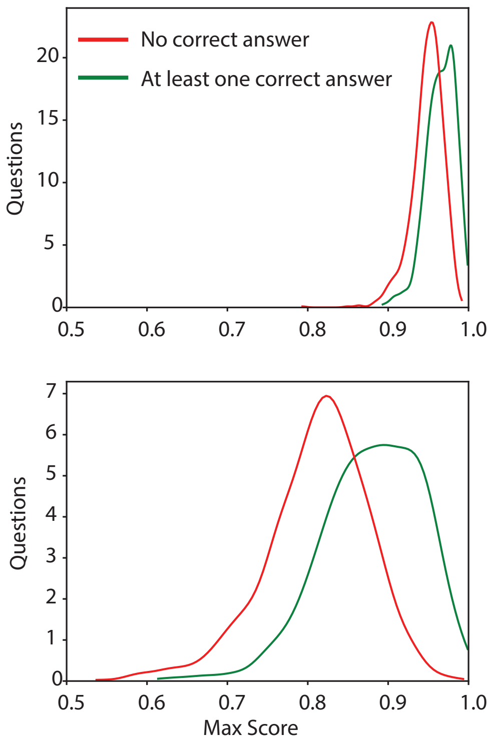

Figure 1: Distribution of maximum score for a test question with the candidate answers (SelQA Answer triggering dataset) **Top**: Random sampling of negatives, **Bottom**: Using hard negatives. See Appendix B for details.

Only the top 4 layers were trained for answer selection, while keeping the other layers frozen. The training batch size was fixed to 64 with hard negatives, and 32 with random negatives, to accommodate the larger size of the parameter set for this model. Each variant was run for a total of 200 epochs.

Data preprocessing is common across all the datasets and models, which includes tokenizing, converting to lowercase and removing words with frequency less than 2.

B Answer Triggering: Distributions of Similarities

To understand the performance gains from hard negatives, we analyze the distributions of question-answer similarity scores. For the task of answer triggering on SelQA dataset, Figure 1 shows the distributions of maximum similarity score of a test question with the answer candidates, separately for questions with and without a correct answer in the candidates. We note that employing hard negatives provides a significant boost with respect to the goal of obtaining consistent similarity scores (incorrect pairs being assigned a smaller score than correct pairs), as necessary for answer triggering.

C The Effect of Batch Size

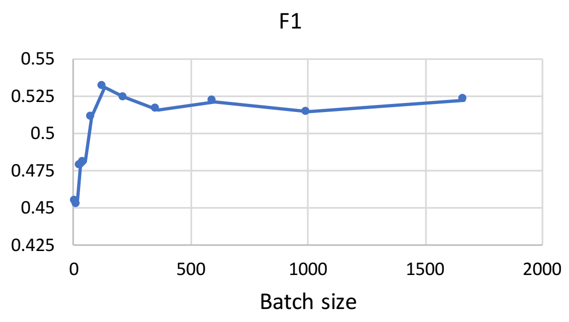

Figure 2: Performance of the proposed approach (F1) on the answer triggering task of SelQA dataset with varying batch sizes.

The proposed approach relies on online selection of hard negatives from within a batch. As noted in the main text, this provides several advantages (Section 4). In this section, we investigate the impact of the choice of batch size on the performance of the proposed approach for the task of answer triggering (Figure 2). Consistent with our hypothesis, we see an increase in performance when we increase the batch size from an initial value of 10. This can be understood in terms of an increase in the hardness of the negatives selected, thus providing more training signal. Further, we note that while the performance initially increases with the batch size, it plateaus out for larger batch sizes. This can be understood in terms of an increased probability of false negatives for large batch sizes which offsets the gain from valid hard negatives. Batch size thus becomes an important hyper-parameter of the approach.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.