$\mathbb{E}$[score]-aware Question Answering

Stanford CS224N IID Default Project

Soon Kyu (Matthew) Lee
Department of Computer Science
Stanford University
mattskl@stanford.edu

Abstract

Use of F1 score for evaluating question answering systems place a higher penalty on bad answer prediction than on bad 'no answer' prediction. This is because a predicted answer for an answerable question may not match the reference answer, and can often result in an F1 score below 100; on the other hand, predicted 'no answer' correctly always results in an F1 score of 100. This paper aims to exploit this imbalance by biasing a model to prefer 'no answer' in three ways: biased prediction layer; biased probability distribution discretization; and augmented training sets with over-representation of unanswerable examples. The biased model successfully achieves this goal and improves the F1 score by 4.13 compared to an unbiased model; it obtains F1 and EM scores of **68.28** and **65.47** on the development set and **65.25** and **62.42** on the test set.

1 Introduction

Question answering (QA) is important for demonstrating computer systems' ability to understand human language. In the words of Wendy Lehnert [1977], "since questions can be devised to query any aspect of text comprehension, the ability to answer questions is the strongest possible demonstration of understanding". With the rapid advancement in the research of QA systems, QA tasks have increased in difficulty as well; tasks range from answering multiple-choice questions given short contexts [Lai et al., 2017] to open-domain questions where the answers must be found from entire books [Kočiský et al., 2018] or databases [Yang et al., 2015].

Contemporary state-of-the-art QA models, like BERT [Devlin et al., 2018] and ELECTRA [Clark et al., 2020], achieve very high accuracy on QA tasks by making heavy use of transformers and pre-trained contextual embeddings (PCE) from very large datasets. Prior to the advent of transformers, models relied on various RNN techniques to comprehend questions and contexts without much use of pre-trained information [Seo et al., 2016, Wang and Jiang, 2015, Wang et al., 2017].

This paper uses SQuAD 2.0 dataset by Rajpurkar et al. [2018], where the task is to take context and question as the input and, if an answer exists, extract a span of the context as the answer. Furthermore, the paper focuses on models that do not use PCE, making improvements to the BiDAF model [Seo et al., 2016] by incorporating R-NET-style pointer network for prediction [Wang et al., 2017]. By focusing on models without PCE, the paper explores ways in which the model can be modified to understand the trade-offs between selecting the wrong answer span and selecting no answer.

2 Related Work

BiDAF by Seo et al. [2016] is a multi-layer model which uses bidirectional attention flow to focus on parts of the context relevant to the question. A slight modification of this model is used as the baseline in this paper. As outlined in the project handout [staff, 2021], BiDAF consists of 5 layers: 1. character/word embedding; 2. contextual embedding; 3. attention flow; 4. modeling; and 5. output. In particular, the output layer computes the probability distribution of the start and end positions

of the answer span based on the context words most relevant to the question from layer 3 and the question-aware context encoding from layer 4.

Rather than using a simple output layer, Wang and Jiang [2015] introduce the use of a zero-initialized pointer network in predicting the answer span. The pointer network is an RNN that runs two steps, and it generates attention across the question-aware context and conditions the answer span's end position on its start position. Wang et al. [2017] take this idea further in their work on R-NET, by initializing the RNN with the encoded question from layer 2.

SQuAD 2.0 dataset [Rajpurkar et al., 2018] builds on SQuAD 1.1 dataset [Rajpurkar et al., 2016] by adding adversarial unanswerable questions which appear answerable. In particular, for such unanswerable questions, a correct prediction (i.e. model predicts that there is no answer) receives an F1 score of 100.0 while incorrect prediction receives a score of 0.0. On the other hand, for an answerable question, correctly predicting that the question is answerable can still result in an F1 score in the range of $[0.0, 100.0]$, since the selected answer span may be incorrect. If a model outputs the same probability for 'no answer' and an answer span, the discrete F1 score for unanswerable questions mean the expected F1 score is higher when selecting 'no answer'. This paper explores several ways of increasing the overall F1 score by exploiting this difference in expected F1 scores. To the author's knowledge, no paper attempts to explicitly model this discrepancy in expected F1 scores.

3 Approach

BiDAF model without character embedding, as presented in staff [2021], is used as the baseline. Improvements to the baseline were made in the same sequence as presented, and the best-performing model for each modification is used for the subsequent step. All code is written from scratch.

3.1 Character embedding

The baseline is restored to the original BiDAF [Seo et al., 2016] by adding the character embedding layer described in the paper. The output of the character embedding layer is concatenated with the word embedding layer, and then fed into the contextual embedding layer. Note that the dimensions of the model's highway networks are increased to match the concatenated input, rather than sub-sampling or using a plain layer to change the input dimension as suggested in the highway network paper [Srivastava et al., 2015]. This choice was made to minimize premature loss of information. Different size of character vectors and number of characters are explored as this information is omitted in the paper. The character embedding of the best-performing model is frozen and used in all subsequent experiments.

3.2 Pointer network

The baseline's output layer is replaced with a pointer network. Three pointer networks are investigated: 1. zero-initialized pointer network; 2. R-NET pointer network; and 3. shared parameter pointer network.

The zero-initialized pointer network is the network described by Wang and Jiang [2015]. It computes an answer span's start position, p^1, and its end position, p^2, given a question-aware context encoding $\{h_t^P\}_{t=1}^n$:

$$s_j^t = \mathrm{v}^T \tanh\left(W_h^P h_j^P + W_h^a h_{t-1}^a\right) \tag{1}$$

$$a_i^t = \exp(s_i^t)/\Sigma_{j=1}^n \exp(s_j^t) \tag{2}$$

$$p^t = \arg\max(\{a_i^t\}_{t=1}^n) \tag{3}$$

$$h_t^a = \text{BiRNN}\left(h_{t-1}^a, \Sigma_{i=1}^n a_i^t h_i^P\right) \tag{4}$$

v, W_h^P, and W_h^a are learnable parameters, and h_0^a is the zero vector.

The R-NET pointer network calculates h_0^a using layer 2's encoded question output, u^Q:

$$s_j = \mathrm{v}^T \tanh\left(W_u^Q u_j^Q + b_u^Q\right) \tag{5}$$

$$a_i = \exp(s_i)/\Sigma_{j=1}^n \exp(s_j) \tag{6}$$

$$h_0^a = \Sigma_{i=1}^n a_i u_i^Q \tag{7}$$

W_u^Q and b_u^Q are learnable matrix and bias respectively.

The shared parameter pointer network uses the same parameter for W_h^a and W_u^Q in equations 1 and 5. This was inspired by the sharing of context and question encoders in many models like the baseline model and QANet [Yu et al., 2018], where shared versions achieve similar or better results. Since W_h^a and W_u^Q serve a similar role in the calculation of attention scores, sharing them seemed promising.

3.3 Biased pointer network

This modification attempts to exploit the higher expected F1 score of predicting 'no answer' given similar predicted probability between 'no answer' and an answer span. This is achieved by modifying equation 1 for the pointer network — s_0^t is increased by some proportion of the max score prior to predicting the answer's start position, since 0th position corresponds to 'no answer':

$$s_j^t = \mathrm{v}^T \tanh\left(W_h^P h_j^P + W_h^a h_{t-1}^a\right) \tag{8}$$

$$s_0^t = s_0^t + \alpha p^t \tag{9}$$

Here, $\alpha \in [0.0, 1.0]$ is a configurable input argument such that a value of 0.0 restores the original pointer network, whereas a value of 1.0 would cause s_0^t to always have the highest score. This explicit biasing of the model is done as it was deemed that the pointer network cannot learn to prefer 'no answer' selection automatically. This is due to the lack of F1 information flowing into the model, since the training metric minimizes NLL rather than maximizing F1 score.

An important caveat for biasing towards 'no answer' is that its effectiveness depends greatly on the dataset. Concretely, assume that 50% of questions are unanswerable, and the average F1 score of answered questions is 50. Then, for expected F1 score of 'no answer' and an answer to be equal, the predicted probability of an answer must be 100% higher than the probability of 'no answer'. However, if 0% of questions are unanswerable, then obviously 'no answer' will never have a higher expected F1 score than an answer. Thus, without knowing the proportion of unanswerable questions, biasing towards 'no answer' does not make sense. Fortunately, SQuA D 2.0's development and test datasets are IID, so biasing can be effective.

3.4 Biased discretization

Similar to biased training, the discretization code was modified to favor 'no answer' selection. Define the upper-triangular matrix of joint probabilities, P, such that P_{ij} is the probability that an answer starts at position i and ends at position j. Also, P_{00} represents the probability that there is 'no answer', and $P_{0j} = 0 \; \forall j > 0$. The biased discretization is achieved by using a modified P':

$$P'_{ij} = \begin{cases} \beta P_{ij} & \text{if } i = 0 \text{ and } j = 0 \\ P_{ij} & \text{otherwise} \end{cases} \tag{10}$$

Here, $\beta \in [1.0, \infty)$ is a configurable input argument such that a value of 1.0 restores the original discretization and higher values bias the selection towards 'no answer'.

3.5 Biased training set

Another approach to make the model favor 'no answer' selection is to augment the training set so that it has an over-representation of unanswerable examples. These augmented training sets are used for fine-tuning only since initial testing showed poor performance when the model is full training on the augmented training sets.

3.5.1 Oversampling existing unanswerable examples

An argument, $\rho \in [0.0, \infty)$, is added to sample examples without answers additional times during the pre-processing of the training set. The integer portion of ρ corresponds to the number of additional times every unanswerable example is sampled, while the decimal portion corresponds to the probability that an unanswerable example is sampled once more. For example, with $\rho = 2.5$, all unanswerable examples are sampled 2 additional times (3 times total) and 50% of the unanswerable examples are sampled one more time on top of that; in total, 50% are sampled 3 times total and 50% are sampled 4 times total.

3.5.2 Generating new unanswerable examples

Given an answerable training example, <context, question, answer> (<C, Q, A>), a new unanswerable example is generated by removing the answer from the context. More concretely, the following steps are taken:

1. the training set is filtered for examples where there is a consensus among all three answer candidates
2. the sentence that encompasses the answer is found by searching for sentence delimters to the left of the start of the answer and to the right of the end of the answer; sentence delimiters of '.', '?', and '!' are used for simplicity
3. if the answer-encompassing sentence is not the entire context, then it is removed from the context to create the new example <C', Q, A'>, where A' is simply 'no answer'

Note that these generated unanswerable examples are quite different from the existing unanswerable examples — newly generated ones are not adversarial unlike the existing ones.

4 Experiments

4.1 Data

SQuAD 2.0 dataset [Rajpurkar et al., 2018] is used throughout the experiment, except when fine-tuning using the biased training sets. SQuAD training dataset contains 129941 examples, of which 86580 are answerable and 43361 are unanswerable. Section 3.5.1 oversamples from these $\sim$43k unanswerable examples, while section 3.5.2 generates 84706 new unanswerable training examples from the $\sim$87k answerable examples.

4.2 Evaluation method

The evaluation metrics are F1, EM, and NLL as defined in the project handout [staff, 2021] and SQuAD 2.0 paper [Rajpurkar et al., 2018]. Specifically, NLL is used as the training metric while the model's performance on SQuAD leaderboard is measured by the F1 score.

4.3 Experimental details

For all experiments, the hidden dimension of all layers is set to 100 and batch size is set to 64. A dropout probability of 30% is used for inputs of layers. A constant Learning rate of 0.5 is selected. Finally, exponential moving average of model parameters is used, with decay rate of 0.999.

4.4 Results and Analysis

Table 1 summarizes the performance on the SQuAD 2.0 development set achieved by models with different modifications. All modifications build on top of the previous modifications, and for each modification, the best-performing variant is selected for the summary. The best-performing model overall achieves F1 score of 65.25 and EM score of 62.42 on the test leaderboard. Figure 1 shows the learning curves for the same model modifications, except for biased discretization since it does not involve training.

Model modification	F1	EM	Δ F1	Δ EM
Baseline	60.15	56.97	—	—
Character embedding	63.13	60.44	+2.98	+3.47
Pointer network	64.15	60.66	+1.02	+0.22
Biased pointer network	65.61	62.19	+1.46	+1.53
Biased discretization	66.50	63.50	+0.89	+1.31
Fine-tuned on oversampled unanswerable	66.92	64.02	+0.42	+0.52
Fine-tuned on generated unanswerable	67.15	64.22	+0.23	+0.20
Fine-tune on oversampled then generated training sets	68.28	65.22	+1.13	+1.00

Table 1: Summary of best results with each type of model modification

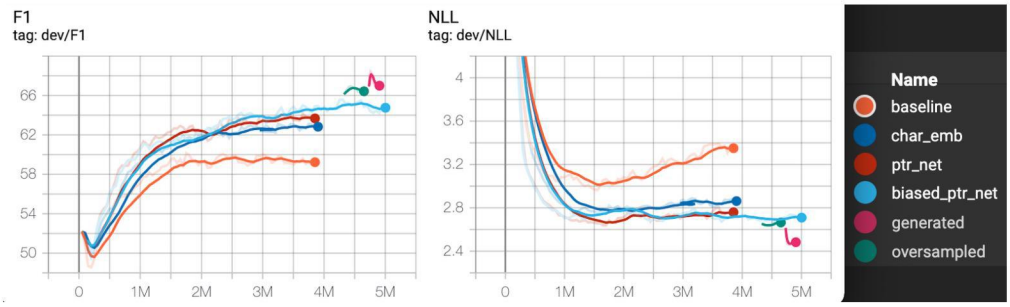

Figure 1: Learning curves of model modifications

4.4.1 Character embedding

A small embedding vector size of 8 was used initially, with the rationale that there is not too much information to be represented. In a similar vein, it seemed unlikely that many of the rare unicode characters that appear in SQuAD 2.0 would add more information than indication of 'out of vocabulary' at the word level. Thus, the character set was culled from more than 10000 to just the 94 characters that appear in glove.840B.300d-char.txt. Another way for culling the character set was to accept characters that appear at least 200 times in the training set.

However, as can be seen in table 2, it turned out that character embeddings express a variety of information that could not be captured with the small vector size of 8, and rare characters were useful in improving performance. Although the performance was similar for embedding vector size of 64 and 200, the larger vector had slower training speed both in wall-clock time and in number of training epochs. Thus, the embeddings trained on all characters with vector size of 64 was frozen and used in subsequent experiments.

Embedding vector size	Accepted characters	F1	EM
8	Any	60.58	57.59
64	Any	**63.13**	**60.44**
200	Any	62.86	60.02
64	Appears in glove.840B.300d-char.txt	61.76	58.28
64	Appears minimum 200 times in training set	62.61	59.70

Table 2: Summary of character embedding parameter exploration

4.4.2 Pointer network

The performance of the different types of pointer networks were surprisingly similar, as shown in table 3. However, the zero-initialized pointer network and the shared-parameter pointer networks were slower to train, as can be seen in figure 2. This suggest all these pointer networks are expressive

enough to model the same thing, but having the explicit initialization and separate learanble parameters guided the R-NET-style pointer network to learn faster. Thus, the quick-training R-NET-style pointer network is used in subsequent experiments.

Pointer network type	F1	EM
Zero-initialized	63.91	60.46
R-NET	**64.15**	**60.66**
R-NET with shared parameter	63.98	60.56

Table 3: Summary of pointer network architecture exploration

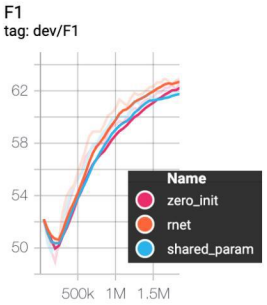

Figure 2: Corresponding learning curves

4.4.3 Biased pointer network

Table 4 shows the result of changing the bias, α, in the pointer network so that the model is encouraged to predict that a question is unanswerable. As expected, some bias towards predicting 'no answer' improves the F1 score; this properly exploits the meager sub-100 ($\sim$84) F1 score both the biased and unbiased models get on answerable examples for which answers were predicted.

Table 5 shows the performance separated by examples where the model predicted an answer and examples where the model predicted 'no answer'. The first two rows show that bias helps improve both the F1 scores of examples where the model predicted an answer and 'no answer'. This makes sense since the model predicts an answer for fewer examples with higher confidence, getting things wrong less often. The F1 score for predicted 'no answer' examples improves too due to the average F1 score for an answer being lower than 100.0. The last two rows are more surprising, as AvNA metric is improved with the bias. This shows that not all F1 score improvements are from exploiting the expected score of 'no answer'; rather, bias also helped correctly identify the answerability of questions.

Bias factor, α	F1	EM
0.0 (no bias)	64.15	60.66
0.1	64.85	61.45
0.2	**65.61**	**62.19**
0.4	62.86	60.02

Table 4: Summary of bias exploration

Bias and metric	Predicted answerable	Predicted unanswerable
No bias, F1 score	54.71	78.04
$\alpha = 0.2$, F1 score	55.94	79.43
No bias, AvNA	64.43	78.59
$\alpha = 0.2$, AvNA	66.90	79.35

Table 5: Bucketed error analysis

4.4.4 Biased discretization

Similar to the results in section 4.4.3, bias towards 'no answer' during answer probability discretization helps performance, as shown in table 6. However, unlike the bias in the pointer network, bias during discretization does not help the model distinguish between answerable and unanswerable questions, as shown in table 7. Specifically, it can be seen that F1 score of answered examples rise at the cost of a fall in the F1 score of unanswered examples. The AvNA metric clearly shows, as expected, that the discretization blindly converts some answers to 'no answers'. The overall F1 score still improves because answers are predicted for more examples than not: 3199 answered vs 2752 unanswered.

Note that the biased discretization is helpful even without the biased pointer network. As an example, the F1 score of the unbiased model from section 4.4.2 improves from 64.15 to 65.23 when $\beta = 2.5$.

Bias factor, β	F1	EM
1.0 (no bias)	65.61	62.19
1.3	66.01	62.73
1.5	66.18	62.98
1.8	**66.50**	**63.42**
2.0	66.41	63.30

Table 6: Summary of bias exploration

Bias and metric	Predicted answerable	Predicted unanswerable
No bias, F1 score	55.94	79.43
$\beta = 1.8$, F1 score	58.12	76.24
No bias, AvNA	66.90	79.35
$\beta = 1.8$, AvNA	68.52	76.16

Table 7: Bucketed error analysis

4.4.5 Biased training set

Table 8 shows the performance of the biased pointer network from section 4.4.3 after it has been fine-tuned with biased training sets. ρ is the oversampling ratio, 'generated' refers to the training set with additional generated unanswerable examples, and $\rightarrow$ indicates that the model was fine-tuned twice with two different biased training sets. Note that no bias in discretization was used as it had negligible or negative impact on the performance. The fine-tuned model that has been bolded was submitted to the test leaderboard, and achieved **F1 score of 65.25 and EM score of 62.42**.

Fine-tuning method	F1	EM
$\rho = 0.5$	66.16	63.03
$\rho = 1.0$	66.92	64.02
$\rho = 2.0$	66.62	63.69
generated	67.15	64.22
$\rho = 1.0 \rightarrow$ gen unans	**68.42**	**65.47**
generated $\rightarrow \rho = 1.0$	67.53	64.63

Table 8: Summary of biased training set fine-tuning

Table 9 shows that this biased fine-tuning improves the F1 score in similar ways to the biased discretitzation in section 4.4.5; F1 score of predicted answers improve at the cost of lowered predicted 'no answer' scores.

Fine-tuning and metric	Predicted answerable	Predicted unanswerable
No fine-tuning, F1 score	55.94	79.43
$\rho = 1.0 \rightarrow$ gen unans, F1 score	62.59	73.57
No fine-tuning, AvNA	66.90	79.35
$\rho = 1.0 \rightarrow$ gen unans, AvNA	71.99	73.57

Table 9: Bucketed error analysis

5 Conclusion

This paper improves the baseline BiDAF model [Seo et al., 2016] by adding back character embedding layer and replacing the output layer with a pointer network. These large architecture changes improve the model's F1 score from 60.15 to 64.15. Then, the paper investigates various ways to exploit the discrete nature of F1 score on unanswerable questions to improve the model's F1 score from 64.15 to 68.28. A bias is introduced to the pointer network so that the score for 'no answer' is increased; a different bias is introduced in the probability distribution discretization such that more weight is given to 'no answer'; finally, the model is fine-tuned on augmented training sets which have duplicated unanswerable questions or generated unanswerable questions. Each of these methods of biasing the model towards 'no answer' proved useful in increasing the model's performance. In particular, the biased pointer network improved the model's ability to determine a question's answerability as well. While the biased model has the clear limitation that it cannot work without prior knowledge of the distribution of the dataset, it produces clear performance wins when the training and test sets are IID.

References

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/2003.10555.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://www.aclweb.org/anthology/Q18-1023.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. RACE: large-scale reading comprehension dataset from examinations. *CoRR*, abs/1704.04683, 2017. URL http://arxiv.org/abs/1704.04683.

Wendy Lehnert. Human and computational question answering. *Cognitive Science*, 1(1):47–73, 1977. ISSN 0364-0213. doi: https://doi.org/10.1016/S0364-0213(77)80004-9. URL https://www.sciencedirect.com/science/article/pii/S0364021377800049.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016. URL http://arxiv.org/abs/1606.05250.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124. URL https://www.aclweb.org/anthology/P18-2124.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016. URL http://arxiv.org/abs/1611.01603.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015. URL http://arxiv.org/abs/1505.00387.

CS224N staff. *CS 224N Default Final Project: Building a QA system (IID SQuAD track)*. Stanford University, 2021. URL http://web.stanford.edu/class/cs224n/project/default-final-project-handout-squad-track.pdf.

Shuohang Wang and Jing Jiang. Learning natural language inference with LSTM. *CoRR*, abs/1512.08849, 2015. URL http://arxiv.org/abs/1512.08849.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1018. URL https://www.aclweb.org/anthology/P17-1018.

Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1237. URL https://www.aclweb.org/anthology/D15-1237.

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. *CoRR*, abs/1804.09541, 2018. URL http://arxiv.org/abs/1804.09541.