Q&A on the SQuAD dataset

Elizabeth Yao
Department of Symbolic Systems
Stanford University
Stanford, CA 94305
lizyao@stanford.edu

Matej Kosec
Aeronautics and Astronautics
Stanford University
Stanford, CA 94305
mkosec@stanford.edu

Abstract

This project report outlines several attempts to improve the performance of a question and answering model on on the Stanford Question Answering Database (SQuAD). Primarily, the work focuses on improved span selection using better span search, using L2 regularization to reduce over-fitting, and implementing bidirectional attention flow (BiDAF) to make the context query-aware. Each one of these experiments yielded a noticeable improvement in the model performance. In total the F1 and EM performance were improved by 15.4 and 15.1%, respectively.

1 Introduction

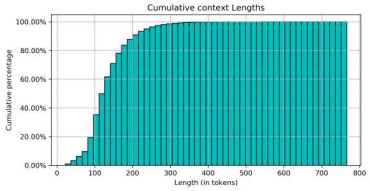
The Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset comprised of context paragraphs, questions, and correct answers. The context paragraphs were pulled from Wikipedia and the questions and answers associated with these paragraphs were crowdsourced from Amazon Mechanical Turk. An important feature of the SQuAD dataset is that the answers to the 100K or so questions come directly from associated context paragraphs. This means that SQuAD systems do not need to generate answer text – they simply select the "span" of text in the paragraph that answers the question.

A SQuAD system is given a context paragraph and a question about that paragraph as input with the goal of answering the question correctly, providing a proxy for how well systems can "understand" text. Practically, a question answering system could be a very efficient way to extract information from large blocks of text.

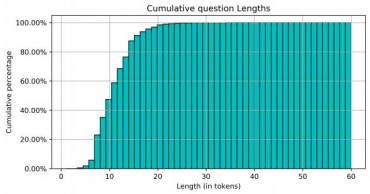
This project is based on the baseline model architecture provided for the default CS224n project, winter 2018. For details see the project guide `http://web.stanford.edu/class/cs224n/project.html`.

2 Analyzing the SQuAD dataset

Before rushing into development of the Q&A model, it is imperative to gain a better understanding of the dataset.
In particular, one of the first steps is figuring out for how many steps to unroll the RNN model such that most of the contexts and answers in the dataset can be captured without truncation. Figures 1a and 1b show what percentage of context and question lengths falls bellow a certain length, respectively. This is of particular interest as the more complicated attention models in this project have very high memory requirements. For instance in an attention mechanism with the context hidden states attending to the question hidden states, there are `context_len` $\times$ `question_len` terms to be computed.



(a) Cumulative (CDF) distribution of context lengths



(b) Log scale frequencies

Figure 1: Cumulative (CDF) distribution of question lengths

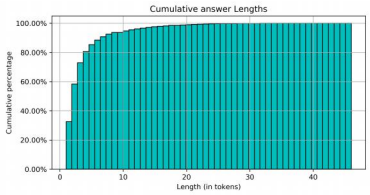
The baseline model uses a default value of 500 for the context length and 40 for the question length. However, from figures 1a, 1b it is apparent that a context length of 400 and a question length of 30 is perfectly sufficient to capture the training set. In fact table 1 shows that to capture $\approx 99.6\%$ of the training set we would only need a context length of 364 and a question length of 24. These facts can be leveraged to fit bigger models onto the GPU without considerable loss in training set size. Furthermore, some of the experimental improvements rely on using a heuristic for the answer length

Table 1: Statistics of context and question lengths

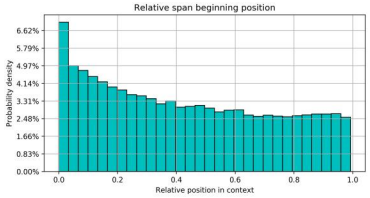
	Context length	Question length	Answer length
Min	22	1	1
Mean	137.9	11.3	3.38
Median	127	11	2
Max	766	60	46
99.6%	364	25	25

to limit what spans can be produced as the possible answers. This searching or the optimal span is necessary as the model is optimized for the cross-entropy loss against the true labels rather than the F1 and EM scores on which it is evaluated. Figure 2a shows the cumulative histogram of answer lengths (difference between $start\_pos$ and end_pos integers). The distribution of answer lengths is further summarized in 1.

In addition to restricting answer lengths, it may be also be interesting to use the statistic of the start



(a) The cumulative distribution of answer lengths in the dataset



(b) The PDF of the relative position of the start token in the context

of the span as a prior distribution when performing the search. Figure 2a shows that the answer is much more likely to occur in the first few tokens of the context (relative position 0). The probability of the start token occurring at a given relative position then generally decreases towards the end of the context (relative position 1). Biasing the search for the optimal span toward the beginning of the context can also be helpful when the answer token occurs multiple times in the context and the predicted end and start positions may latch on to different occurrences.

3 Quantitative analysis of baseline performance

The baseline model was run for 49k iterations over a span of over 19hrs at approximately 1.4 seconds per iteration. In figure 3(left) the baseline exact match performance is shown to be 29.5% on the dev set and 65.0% on the train set. This discrepancy in performance is indicative of a large amount

of over-fitting being present in the model. In spite of the relatively large amount of over-fitting the baseline model does not achieve a very high score (80 or more) on the train set indicating that a more complex model is still necessary. The F1 scores for the baseline shown in figure 3 similarly show a high degree of over-fitting with the F1 score being 74.8 on the train set and 40.3 on the dev set. This high degree of over-fitting suggest that the model should be further regularized. This, and further improvements is discussed in section 5. Figure 3 also shows the model which achieved the highest dev EM score to be 37.5k iterations. This demonstrates that it may be beneficial to run the training process for many iterations as random oscillations of the dev score may result in improvements of almost a percentage point.

In addition to the F1 and EM performance it is also relevant to observe the reduction in the cross-

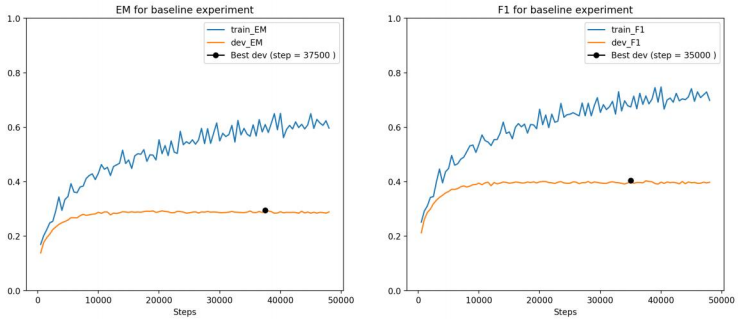


Figure 3: The exact match (EM) and F1 scores for the baseline model

entropy loss achieved during training shown in Figure 4. It can be observed that the loss of predicting the start of the span is similar to the loss on predicting the end of the span. This is a result of the fact that the start and end predictions are computed independently of one-another. In reality, a human answering a particular question would most likely spend most of their time determining the start of the question and then, with much less effort, keep highlighting words until the end of the answer is reached. That is, the prediction of the end token should be a fundamentally less lossy endeavor (not least because the end must occur after the start and is thus more constrained).

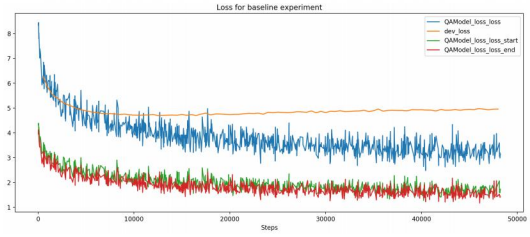


Figure 4: The loss on the training and dev datasets during the training procedure

4 Qualitative analysis of baseline performance

In order to select architectural enhancements to implement in the limited time-frame it is crucial to analyze the types of errors that the baseline model is most susceptible to.

The Q&A examples of errors presented below the following color convention is used: *italicized text* represents the true answer, magenta text denotes the predicted start of the answer sequence, and red text is the predicted end. An exact match is denoted by *italicized magenta text*. Underscores represent unknown tokens.

Issue 1: predicted answer start comes after predicted end

The most readily observable error that the baseline model makes is predicting the end position of the answer before the start position. This is to be expected as the end and start positions are predicted independently in the baseline model.

The example below illustrates this issue which occurs 1727 times when predicting on the dev set.

This corresponds to a total of **16.6%** of the 10391 dev set examples.

the moon landing data was recorded by a special apollo tv camera which recorded in a format incompatible with broadcast tv . this resulted in lunar footage that had to be converted for the live television broadcast and stored on magnetic telemetry tapes. during the following years , a magnetic tape shortage prompted nasa to remove massive numbers of magnetic tapes from the national archives and records administration to be recorded over with *newer satellite data* . stan lebar , who led the team that designed and built the lunar television camera at westinghouse electric corporation , also worked with nafzger to try to locate the missing tapes .
QUESTION: what did nasa record over the older archived tapes ?
TRUE ANSWER: newer satellite data
PREDICTED ANSWER:
F1 SCORE ANSWER: 0.000
EM SCORE: False

To resolve this issue one could condition the value of the end token on the value of the start token. One way of achieving is presented in [2]. Alternatively, a better span search algorithm could be developed which ensures the end is always predicted after the start.

Issue 2: length of predicted answer is statistically improbable

The next example illustrates a prediction where the start of the answer and the end of the answer are separated by such a large margin that essentially most of the context ends up being selected. Additionally, there are 2095 examples in the dev set where the predicted answer is at least twice as long as the true answer. This corresponds to **20.1%** of the dev set.

the most useful instrument for analyzing the performance of steam engines is the steam engine indicator . early versions were in use by 1851 , but the most successful indicator was developed for the high speed engine inventor and manufacturer charles porter by charles richard and exhibited at london exhibition in 1862 . the *steam engine indicator* traces on paper the pressure in the cylinder throughout the cycle , which can be used to spot various problems and calculate developed horsepower . it was routinely used by engineers , mechanics and insurance inspectors . the engine indicator can also be used on internal combustion engines . see image of indicator diagram below (in types of motor units section) .
QUESTION: what instrument is used to examine steam engine performance ?
TRUE ANSWER: steam engine indicator
PREDICTED ANSWER: high speed engine inventor and manufacturer charles porter by charles richard and exhibited at london exhibition in 1862 . the steam engine indicator traces on paper the pressure in the cylinder throughout the cycle , which can be used to spot various problems and calculate developed horsepower . it was routinely used by engineers , mechanics and insurance inspectors . the engine indicator
F1 SCORE ANSWER: 0.107
EM SCORE: False

Issue 3: Insufficient attention flow

The following example demonstrates the need for attention to flow in both directions. One would like for our model to hone in on the words in the context that appear on either side of any question keywords (in this case "strike"). It is hard to count how many times such errors occur in the dev set.

the *writers guild of america* strike that halted production of network programs for much of the _2007-08_ season affected the network in _2007-08_ and _2008-09_ , as various abc shows that premiered in 2007 , such as dirty sexy money , pushing daisies , eli stone and samantha who ? , did not live to see a third season ; other series such as boston legal and the u.s. version of life on mars suffered from low viewership , despite the former , a spin off of the practice , being a _once-highlighted_ breakout series when it debuted in 2005 . one of the network 's _strike-replacement_ programs during that time was the game show duel , which premiered in december 2007 . the program would become a minor success for the network during its initial six-episode run , which led abc to renew duel as a regular series starting in april 2008 . however , duel suffered from low viewership during its run as a regular series , and abc canceled the program after sixteen episodes . on august 15 , 2008 , disney denied rumors started by caris & co. that it would be selling the ten abc owned-and-operated stations .
QUESTION: a strike by what entity resulted in a halt to production for network programs in the 2007-2008 season ?
TRUE ANSWER: writers guild of america
PREDICTED ANSWER: caris & co.
F1 SCORE ANSWER: 0.000
EM SCORE: False

5 Overview of selected experiments

Based on the qualitative analysis of the baseline model performed in section 4 the following two major improvements have been prioritized:

1. Perform a better search for the answer span, such that the start token alway occurs before end token and that the span length is not excessive (section 5.2)
2. Implement bi-directional attention flow as presented in [1]. (see section 5.3).

The first of these improvements is intended to address the 16.6% of baseline dev answers for which the end token occurred before the start token (resulting in 0 EM and F1). The second improvement is intended to allow the model to better understand complex interactions between the question and the context, and to better model the way a human would approach answering a question and context pair.
In addition to these two major experiments it is also crucial to search for the appropriate hyper-parameters which optimize a given model. Among such smaller experiments were:

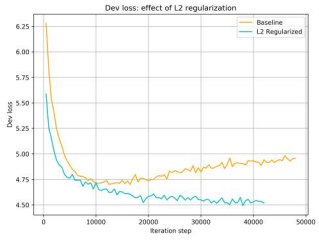
- Increase the pre-trained **embedding size** from 100 to 300. This was found to provide smaller models with a better ability to generalize to the dev set. However, for larger an more powerful models, the larger word vectors resulted in excessive memory use.

- Increasing the **hidden layer size** from 200 to 300. While the baseline model was able to over-fit to the training set and achieve an F1 score of 74.8, this score is still much lower than the test performance of the top models on the SQUAD leader-board. As such, it is desirable to increase the complexity of the model such that it can fit the training set even better. Increasing the hidden layer size is a simple and fast way to achieve this. However, for more advanced this again lead to exceeding GPU space limitations.
- In addition to increasing the hidden layer size, it is also possible to increase the model complexity by changing the **type of encoder RNN** cell from GRU to LSTM. The LSTM cell is even better suited to capturing long-term relationships in the context/question. In practice it was found that the LSTM-based models would train faster (have a higher learning slope) than the GRU-based models. This is to be expected at the LSTM is able to handle the vanishing gradients problem better than the GRU.
- A final option for easily increasing the model complexity is to **increase the depth** of the network operating on the encoder output (blended_reps_final). The baseline model uses only a single fully connected layer with ReLU activation. Adding an additional two fully connected layers to the baseline model was found to increase the dev F1 and EM by $\approx 1.5\%$.
- **Conditioning end on start prediction** while the original goal was to implement the answer pointer decoder from [2] implementing this in tensorflow using the GreedyEmbeddingHelper proved to be too challenging and has been left as future work. As a much simpler means of conditioning the end on the start, the softmax probability distribution and logits of the start token were tiled 30 times and added to the input of the softmax of the end token. This significantly increased the degree of over-fitting to the training set but left the dev performance relatively unchanged. However, with additional regularization the dev performance was also increased.
- **Regularization** is discussed in more detail in section 5.1.
- Increasing the **batch size** from 100 to 400 provided a much better gradient estimate and allowed the **learning rate** to be increased from 0.001 to 0.003. The increased batch size came at only a $\approx 15\%$ increase in computation time per step, while training the baseline model decreased from ≈ 4.5hrs to ≈ 1.5hrs (till onset of over-fitting). Unfortunately, on larger models (BiDAF) the batch size had to be decreased to as low as 15, leading to a much noisier gradient estimate.

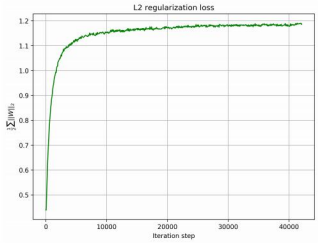
5.1 Experiment 1: L2 regularization

The initial baseline model used a dropout regularization with a keep probability of 0.85. However as can be seen in figure 5a the loss on the dev set starts to increase after about 13k iterations, while the loss on the train set continues to decrease until the training is terminated. The baseline achieve a train F1 of 74.8 but a dev F1 of only 40.3. The difference is due to the model over-fitting to the training set.

The initial instinct is to increase the dropout regularization to further decrease level of over-fitting



(a) Comparison between the loss on the dev set without and with L2 regularization



(b) The L2 regularization loss plateaus as the training progresses

Figure 5: Effectiveness of L2 regularization in reducing over-fitting

experienced by the model. However, decreasing the keep probability also decreases the level of

complexity of the neural network. For instance, using a keep probability of 0.5 caused F1 to drop by almost 10%.
L2 regularization is a standard technique for reducing over-fitting by penalizing large parameter weights (note that biases are commonly excluded). The L2 loss is scaled by a hyper-parameter λ before being added to the cross-entropy loss. As a general guideline, λ was tuned to make the regularization loss the same order of magnitude (but smaller) than the cross-entropy loss components.
Specifically, for a choice of $\lambda = 0.0005$ (on the baseline model) figure 5b shows that the L2 loss converges to a value of about 1.2, while the start and end cross-entropy losses are 2.4 and 2.3, respectively. Additionally, figure 5a shows that for the regularized model the dev loss continues to have a decreasing trend through the whole training process (note that the L2 term has been subtracted so that the losses are comparable). This reduction in dev loss directly translated to about 5% increase in F1 and was therefore used also in the final submission model.

5.2 Experiment 2: Better search for start and end token

Cross entropy loss is a lot less stringent than EM or F1 in penalizing negative span lengths (answer is predicted to end before it starts). Thus, it cannot truly be hoped that the model will effectively learn how to ensure that the end prediction should occur after the start. While models such as the answer pointer decoder [2] can certainly help by conditioning the end prediction on the start prediction, it is should ultimately be up to the span selection algorithm during runtime to ensure that negative lengths of answers do not occur.
The baseline uses a simple algorithm which chooses the start prediction to be the argmax of the softmax $p_{\texttt{start}}(i)$ probabilities of the start position (i) and the end to be the argmax of the softmax probabilities $p_{\texttt{end}}(j)$ of the end position (j) prediction. As such the algorithm performs no explicit checks that the span length is positive. Indeed in 16.6% of the dev set the span length was found to be negative. Additionally, in 20.1% of the dev set the predicted span length was over two times the length of the correct span lengths.
To help reduce such errors, the search algorithm has been adjusted to choose the start index i and end index j such that $p_{\texttt{start}}(i)p_{\texttt{end}}(j)$ is maximized under the constrains that then distance $j - i \leq 15$. The choice of 15 as the cut-off length is supported both by figure 2a, which shows that over 95% of the answers are shorter than 15 tokens.
In practice, this algorithm resulted in a boost in F1 and EM performance of around 8%. The number of answers with negative length is reduced to 0, and only 12.9% of answers are more than two times as long as truth. Because of these encouraging results, the improved span search was included in the final submission.

5.3 Experiment 3: Bidirectional Attention Flow

The initial baseline model uses standard attention, in which only the context states attend to the question states. To accurately answer questions based on a certain context in practice, it is often helpful to know what the query is while processing the context. Thus, it is also important to have the question states attend to the context states, as in the Bidirectional Attention Flow layer used in BiDAF[1].

The first step is to compute a similarity matrix $\boldsymbol{S} \in \mathbb{R}^{N \times M}$, containing a similarity score $\boldsymbol{S}_{ij} = \boldsymbol{w}_{sim}^T[\boldsymbol{c}_i; \boldsymbol{q}_j; \boldsymbol{c}_i \circ \boldsymbol{q}_j]$ for each pair of context and question hidden states $(\boldsymbol{c}_i, \boldsymbol{q}_j)$, where $\boldsymbol{c}_i \circ \boldsymbol{q}_j$ represents an element wise product and $\boldsymbol{w}_{sim}^T \in \mathbb{R}^{6h}$ is a weight vector. The second step is to perform Context-to-Question (C2Q) Attention, which can be described in equations as follows:

$$\alpha^i = \text{softmax}(\boldsymbol{S}_{i,:}) \in \mathbb{R}^M \quad \forall i \in \{1, ..., N\}$$
$$\boldsymbol{a}^i = \sum_{j=i}^{M} \alpha_j^i \boldsymbol{q}_j \in \mathbb{R}^{2h} \quad \forall i \in \{1, ..., N\}$$

The third step is to perform Question-to-Context (Q2C) Attention, which can be described in equations as follows:

$$\boldsymbol{m}_i = \max_j \boldsymbol{S}_{ij} \in \mathbb{R} \quad \forall i \in \{1, ..., N\}$$
$$\beta = \text{softmax}(\boldsymbol{m}) \in \mathbb{R}^N$$
$$\boldsymbol{c}' = \sum_{i=1}^{N} \beta_i \boldsymbol{c}_i \in \mathbb{R}^{2h}$$

The bidirectional attention flow layer can quite memory intensive due to the large matrices that are multiplied together in the process of creating the similarity matrix. Thus, we initially lowered our batch size from 100 down to 15. This bidirectional attention flow model achieved a train F1 of 69.1, a dev F1 of 46.0, a train EM of 58.5, and a dev EM of 33.5.

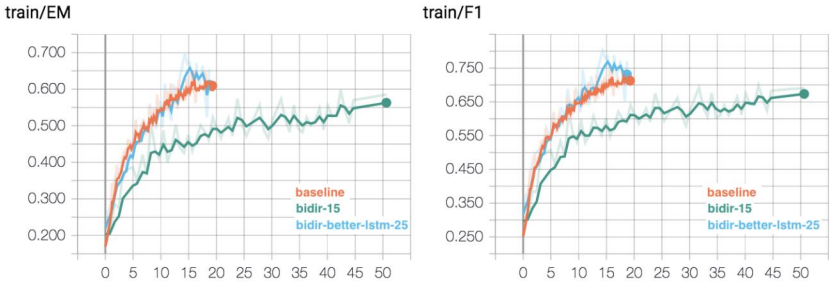


Figure 6: The train EM and F1 plotted against training time in hours.

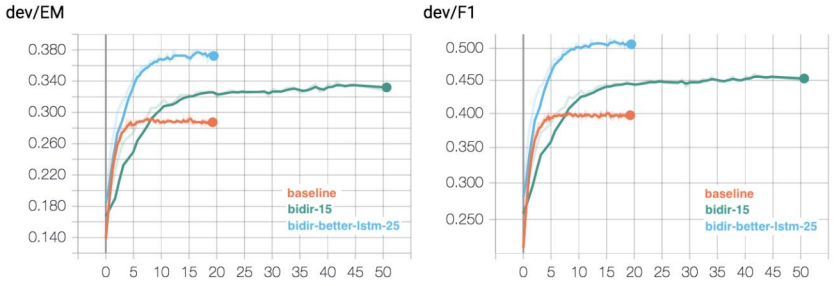


Figure 7: The dev EM and F1 plotted against training time in hours.

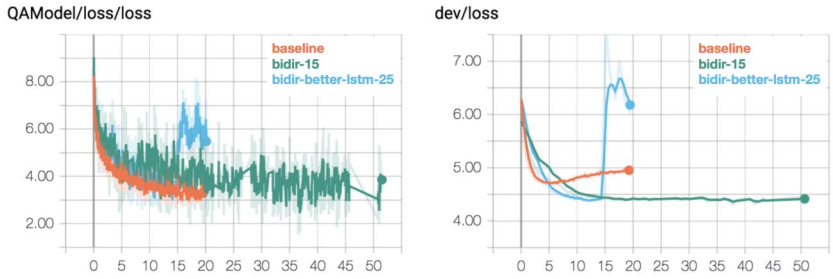


Figure 8: The train and dev loss plotted against training time in hours.

Additionally, the GRU cells were changed to LSTM cells even though GRUs train faster and often perform better on small training datasets. Overall LSTMs are able to remember longer sequences and it seemed that GRUs did not provide enough attention flow. Much of the computational overhead of LSTM was counteracted by the realization that array broadcasting could be leveraged instead of tiling matrix concatenation. This enabled the training batch sized to be increased from 15 to 25.
After making these changes, the model still over-fit to the training set. The impact of this was reduced using L2 regularization starting at $\lambda = 0.0001$, raising λ by 0.0001 until we hit our final λ at 0.0003.

6 Final submission model description

The final submission model had a bidirectional attention layer and used better search for start and end token, LSTM cells instead of GRU cells, and L2 regularization. Codalab reported test values of: **F1 56.281** and **EM 45.085**.

Table 2: Parameters for the final submission model

Learning Rate	Dropout	Batch Size	Hidden Size	Context Len	Question Len	Embedding Size	λ
0.001	0.15	25	200	500	40	100	0.0003

7 Conclusions and future work

Table 3 summarizes the major experiments that were undertaken during the course of the project. The final row of the matrix represent the final submission model.
The baseline model performed well with an F1 score of 40.3 and and EM of 29.5 but suffered from a high degree of over-fitting, insufficiently complex attention flow, and an overly-simplistic way of choosing the start and end of the span. The experiments in table 3 represent attempt to ameliorate these issues. Future work should aim to directly optimize the F1 and EM scores instead of

Table 3: Summary of the performance of different experiments

	Train [%]				Dev [%]			
Experiment:	F1	Δ F1	EM	Δ EM	F1	Δ F1	EM	Δ EM
Baseline	74.8	+0.0	65.0	+0.0	40.3	+0.0	29.5	+0.0
Bidirectional attention flow (BiDAF)	69.1	-5.7	58.5	-6.5	46.0	+5.7	33.5	+4.0
BiDir + better search + LSTM + L2 reg	80.2	+5.4	69.7	+4.7	**55.7**	+15.4	**44.6**	+15.1
Additional layer on blended reps	73.9	-0.9	63.9	-1.1	41.3	+1.0	30.1	+0.6
Cond. end on start	77.8	+3.0	68.6	+3.6	39.7	-0.6	29.0	-0.5
Con. end on start + better search + L2 reg	67.0	-7.8	54.6	-10.4	**54.0**	+13.3	**43.0**	+13.5

the cross-entropy proxy-loss. This can be achieved by using a policy gradient RL method (such as REINFORCE) to learn a locally optimal policy for choosing the start and end of the span. This is particularly promising as REFINFORCE can work on non-differentiable functions (which F1 and EM are) and are therefore suitable for maximizing the F1 or EM scores of a model while gradient descent is not.
Additionally, it was originally envisioned that the Answer Pointer method from [2] would be implemented to help address the issue of improbable question lengths and negative span lengths (end before start). An implementation of answer pointer which provided significant gains over the baseline was, however, not developed in time for the submission and is left as future work.
Finally, while the better search algorithm presented in 5.2 represents a significant improvement over the baseline argmax search model, future work should explore further increasing the sophistication of the search algorithm. For instance instead of using a fixed maximum for the span length of 15 tokens, one could use the answer length probability distribution (based on training data) as a prior to weight the different span lengths. An example CDF of such a prior is shown in 2a. This can be considered as a prior on the predicted end of the span. Additionally, a prior could be used for the beginning of the span. This can be achieved by taking the PDF from 2b which shows the probability of the start of the span occurring at a certain relative position in the context.

References

[1] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[2] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905, 2016.

‘’