# Improving BiDAF on SQuAD 2.0

Stanford CS224N Default Project

**Yufei Liu**

## Abstract

Question Answering is an important field in natural language processing, where models are designed to answer questions based on given context. In this project, we have enhanced the baseline model, BiDAF, by adding char embeddings, introducing additional training tasks, and implementing model ensembling. Our chosen dataset is the famous SQuAD 2.0, which was published by Stanford University.

## 1 Introduction and Related Work

**SQuAD 2.0:** It contains over 100,000 question-answer pairs, sourced from Wikipedia articles. Unlike the previous version, SQuAD 2.0 includes unanswerable questions, where the answer is not present in the provided context. This requires models to not only provide accurate answers but also identify when a question cannot be answered.

**BiDAF:** The baseline model in this paper heavily relies on the BiDAF model, which was originally developed by Seo et al. in 2018. The provided code for BiDAF in this project contains several key components. It includes a word embedding layer, an encoder layer that utilizes a single bidirectional LSTM layer, a bidirectional attention flow layer, a modeling layer that consists of two bidirectional LSTM layers, and an output layer. Noting that the code lacks a char-level embedding layer, which is present in the original model described in the paper.

## 2 Approach

### 2.1 Char embedding

It is widely admitted that character embedding plays an important role in enhancing the model's understanding of the internal structure of words and its ability to handle out-of-vocabulary words. Therefore, we initially implemented character embeddings in our approach.

To achieve this, I utilized 1-dimensional CNN. Specifically, we have the representation of each character in a word, with each representation being of $\boldsymbol{h}$ dimensions. Then I apply $\boldsymbol{h}$ convolutional kernels on each character to obtain the new representation. Subsequently, I extracted the maximum value within every character of a word for each dimension. This process yielded a character-level representation of the word, denoted as $\boldsymbol{h_c}$. Additionally, we obtained a word-level representation, denoted as $\boldsymbol{h_w}$. These two representations were concatenated as $[\boldsymbol{h_c}; \boldsymbol{h_w}]$. Finally, we passed this concatenation through a linear layer ($\boldsymbol{R^{2h \times h}}$) to obtain the input for the highway network.

### 2.2 mask

To answer a question accurately based on context, it is crucial to comprehend the meaning of a passage. This is achieved through the encoder layer in the BiDAF model. In order to enhance this understanding, I employed only the embedding layer and encoder layer to predict the masked context before engaging in the full QA task.

Specifically, I masked 8% of the words in the context randomly and tasked the model with predicting all of the masked words using the output of the encoder layer during the initial 5 epochs. Subsequently, the model will do the complete QA task, same to the baseline approach with character embedding.

### 2.3 Ensembling

I selected three models for ensemble: BiDAF with char-emb and mask, BiDAF with char-emb and Adadelta optimizer, BiDAF with char-emb and Adam optimizer.

After obtaining the output probability distribution from each model, I got the maximum value for each answer position within the three and it produce a new probability distribution. Utilizing this new probability distribution, I predicted the answer position. The performance of the ensemble model significantly improved.

## 3 experiments

### 3.1 Data

We use SQUAD 2.0 dataset in the experiment. The data is structured as over 150k context, question, answer combinations on more than 500 articles. Additionally, the dataset contains more than 50000 unanswerable questions. If a question is answerable, the answer will be taken directly from a chunk of the corresponding paragraph, so, you just need to predict the start and end position. And if the question has no answer, model need to output “No Answer”.

### 3.2 Evaluation method

We use exact match (EM) and F1 to evaluate our models. EM measures if the output exactly matches the human answer while the F1 score is the harmonic mean of precision and recall.

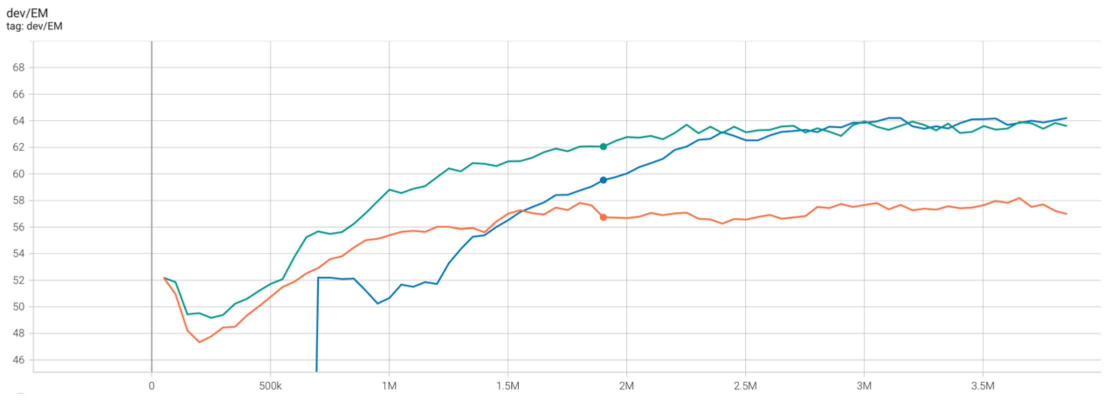
### 3.3 Experimental details

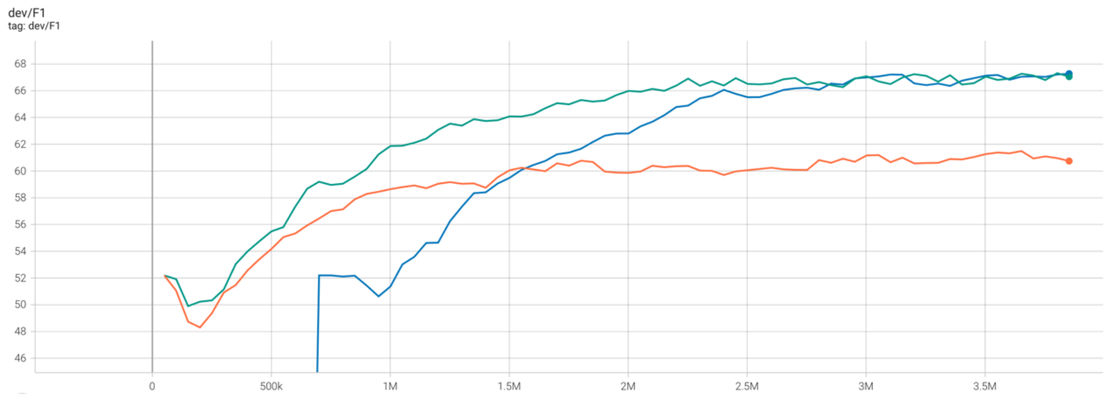
I preserved the default settings for all models, with the exception of the model using Adam optimizer. For this particular model, I opted for a learning rate of 1e-3.

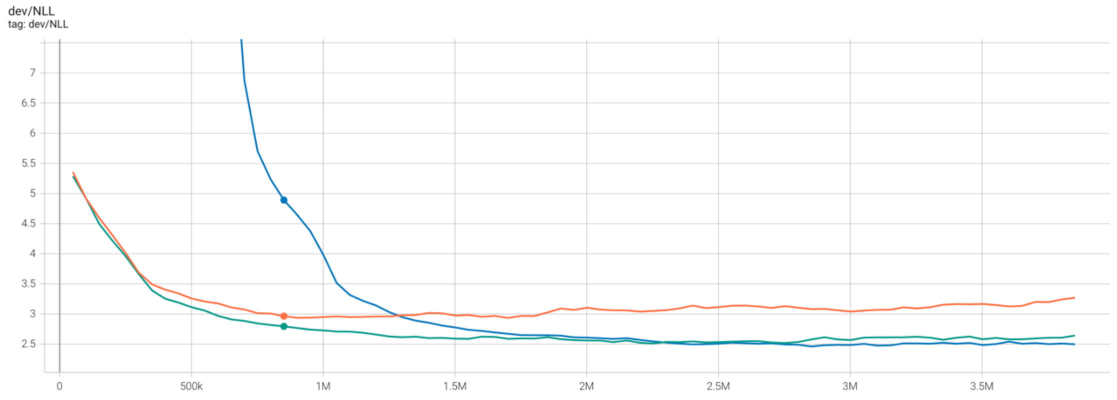
## 3.4 Results

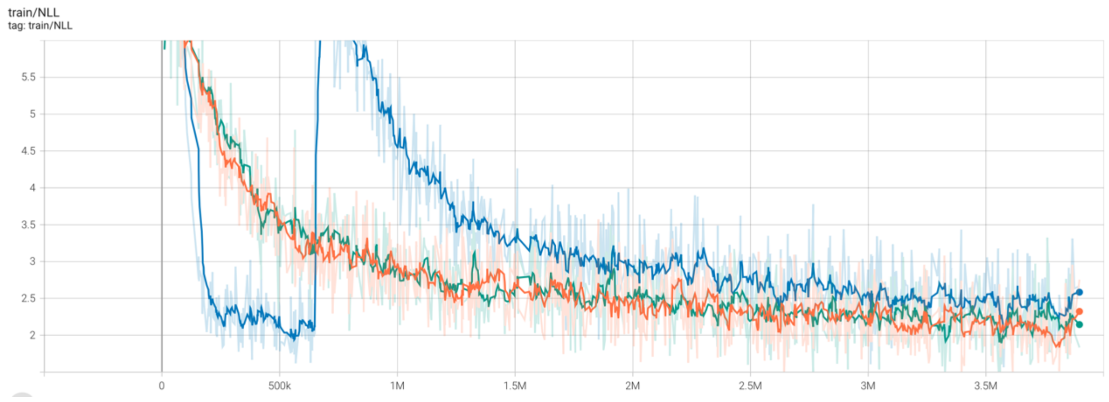
| *Method* | *EM* | *F1* |
|---|---|---|
| ***Baseline*** | 57.31 | 60.94 |
| ***Character embedding*** | 63.60 | 67.06 |
| ***Char-emb and mask*** | 64.06 | 67.28 |
| ***Ensembling*** | **66.64** | **69.58** |

Below is the performance of three models on the dev dataset. (baseline is orange, char-emb is green, char-emb and mask is blue)









## 4 Analysis

### 4.1 Character embedding

Based on the curves, baseline and char-emb models perform similarly on the train dataset. However, when it comes to the dev dataset, the char-emb model outperforms the baseline model significantly. I guess this difference in performance could be attributed to the presence of uncommon or out-of-vocabulary words in the dev dataset that are rarely seen in the train dataset. Unlike the baseline model, the char-emb model utilizes every character in a word to capture its meaning. Consequently, the char-emb model demonstrates superior performance on the dev dataset, despite the similar performance on the train dataset.

### 4.2 Mask

Analyzing the training curves, we observe that the models with char-emb and mask have the best performance on dev datasets, although it performs worst on the train dataset. Because the mask is random, I believe that the initial 5 epochs of training contribute to better generalization and a stronger ability to comprehend the context. By guiding the model to understand the context before engaging in the QA task, it not only save training time (approximately half the time in the first 5 epochs) but also enhance the overall performance

### 4.2 Ensembling

Ensembling improved the performance a lot according to the data above.

## 5 Conclusion

In this project, I investigated various techniques to enhance the performance of the baseline model on the SQuAD 2.0 dataset.

Among the these approaches, our best-performing model was the Ensembling model, which achieved an impressive F1 score of 69.58% and an EM score of 66.64%. In comparison, the baseline model achieved an F1 score of 60.94% and an EM score of 57.31%.

Additionally, we experimented with several other techniques, such as self-attention and conditioning the end prediction based on the start prediction. However, it was observed that Ensembling is still the best.