SQuAD-ing Up as a Winning Team: Character Embeddings and Dynamic Coattention Networks

Stanford CS224N Default Project - IID SQuAD Track | Mentor: Ethan Chi

Kevin Guo
Department of Computer Science
Stanford University
kyguo@stanford.edu

Brandon Kang
Department of Computer Science
Stanford University
bmkang@stanford.edu

Abstract

Recent developments in the Question Answering space, a large interest in NLP, have been facilitated by both the advancement of deep learning models as well as more robust datasets, such as the Stanford Question Answering Dataset (SQuAD) [1]. In this paper, we seek to improve upon the provided baseline Bi-directional Attention Flow (BiDAF) model that only includes word-level embeddings. We first add character-level embeddings in addition to the provided word-level embeddings and implement the coattention mechanism described by Xiong et al. to replace the attention layer in BIDAF [2] [3]. We found that adding character-level embeddings as an input exceeds baseline performance. However, the combination of the Dynamic Coattention implementation and character embeddings only improved on the F1 scoring while scoring lower on the EM score when using the default hyperparameters. After model testing and a hyperparameter search, the Coattention-Layer BIDAF model with both char-level embeddings and word-level embeddings had the best performance with an EM score of 59.138 and an F1 score of 62.704 on the test set when using a learning rate of 0.6 and a dropout probability of 0.15.

1 Introduction

Question Answering is a task in the Natural Language Processing community that has received a lot of attention due to its broad range of implications for issues ranging from web search to data analytics. The task involves a model answering a question about a given paragraph and explores a machine's ability to accurately understand a context-query pair. Question Answering has further increase in popularity due in part to the robustness and reasoning-based nature of the SQuAD dataset [1].

In this paper, we implement multiple features within the Bi-directional Attention Flow (BiDAF) model. We first build upon the given baseline BiDAF model that contains only word-level embeddings by incorporating the character-level embeddings as described in Seo et al [2]. We hypothesize that this improves the performance of the baseline since the character-level embedding mechanism allows for the handling of out-of-vocabulary words - words that are not present in the GloVe vocabulary. We additionally implement Dynamic Coattention and create a model combining character-level embeddings with the Dynamic Coattention Network described in Xiong et al [3]. We do this by substituting coattention into the baseline's Attention Layer which further adds robustness to the model.

We measure the success of our implementations using EM and F1 scores and further experiment upon our models with hyperparameter tuning experiments. We find our model performs best with the Coattention + Character-Level Embeddings configuration with a learning rate of 0.6 and a dropout probability of 0.15 and associated EM and F1 scores of 59.138 and 62.704 on the test set, respectively.

2 Related Work

2.1 BiDAF

Our work and baseline draw upon the Bi-Directional Attention Flow (BiDAF) network introduced by Seo et al [2]. The model employs character-level, word-level, and contextual embeddings alongside a multi-stage architecture for modeling the representations of context paragraphs. The model uses the bi-directional attention flow mechanism which improves upon information loss and memory metrics. This is done through attention being computed for every time stop and allowing the attended vector to flow through to the modeling layer, reducing information loss from early summarization. The memory-less structure also gives the mechanism an advantage as attention at each time step is not dependent on that of the previous time step. The baseline employed in our work is a simplified version with only the word-level embeddings provided and we build upon the code provided for the construction of this baseline.

2.2 Dynamic Coattention Networks

Dynamic Coattention Networks are end-to-end neural network architectures that have also been a relevant development in the Question Answering space and was first proposed by Xiong et al [3]. The novelty of this model is derived from its usage of co-dependent representations of the question and the document captured by a coattentive encoder. The model also includes a dynamic pointing decoder that alternates between the start and end of the answer span. The proposed model leads to improved EM and F1 scores which is based upon the model's ability to recover from local maxima that may correspond to incorrect answers. We draw upon this work to implement a Coattention Layer that we substitute in place of the baseline's basic Attention Layer.

3 Approach

We were provided a default baseline BiDAF model that was implemented with word-level embeddings. For our approach, we first built upon this baseline by making adjustments in the embedding layer to account for character-level embeddings. We then went on to focus on the attention layer and implemented a co-attention layer that could be substituted in place of the provided Attention Layer. Finally, our approach included tuning hyperparameters such as learning rate and dropout probabilities to improve our model's performance.

3.1 Character-Level Embeddings

Our model extends upon the baseline provided with our implementation of character-level embeddings as described in Seo et al [2]. In our implementation, we pass preprocessed context and query words through a convolutional neural network (CNN) to produce character-level embeddings. We then take the CNN output and apply a max pooling step to obtain the maximum vector values. We then concatenate the max pooling output to our word-level embeddings and then passed the concatenated output through a Highway Network.

3.2 Coattention Layer

We extend upon the character-level embedding implementation described above by implementing a coattention layer as described by Xiong et al [3]. The coattention mechanism proposed attends to the question and document at the same time and the coattention encoder proposed learns question-document co-dependent representations. For the coattention encoder, we calculate the affinity matrices, L which are the affinity scores between pairs of of document and question words. L is calculated from appending sentinel vectors to the tanh function of the question hidden states. Then, we use masked softmax on L and use the weighted sum of each question's hidden state to obtain the Context-to-Question attention output to calculate what the paper calls matrices A^Q and A^D, row-wise and column-wise, respectively. We then calculate the summaries of the document in light of each word of the question and of the question in light of each word of the document. Lastly, we concatenate the second level attention output, the weighted sum or the summaries, with the attention summaries and feed the sum through a bidirectional LSTM layer.

3.3 Hyperparameter Search

With regards to hyperparameter finetuning, we chose to focus on the learning rate and dropout probability. The default values for the learning rate was set to 0.5 while the default value for the dropout probability was set to 0.2. We performed a hyperparameter search on the BiDAF model that employs both coattention and character-level embeddings as well on the model with just character-level embeddings.

4 Experiments

4.1 Data

The dataset we are using in this project is the Stanford Question Answering dataset (SQuAD) 2.0 dataset that has been extended from the original SQuAD dataset to include unanswerable questions. SQuAD 2.0 builds upon the original dataset with an additional 50,000 unanswerable questions. The dataset is split into 3 parts: the training set with 129,941 examples taken from the official SQuAD 2.0 training set, the dev set with 6078 randomly selected examples (roughly half of the official dev set), and the test set with 5915 examples (remaining examples from the official dev set, plus hand-labeled examples).

4.2 Evaluation method

We calculated the F1 scores and Exact Match (EM) scores and juxtaposed each pair of scores across all models, including the baseline, to determine the best performance. We also consider AvNA and NLL to evaluate our model's training. From the comparisons, we determine which features have major contributions to or effects on performance.

4.3 Experimental details

Training was performed on Microsoft Azure NCv3-series VMs with NVIDIA Tesla V100 GPUs. We trained 4 different models and performed a hyperparameter search for the coattention and character-level embedding BiDAF. Each model took around 3-4 hours to train with 30 epochs, a default learning rate of 0.5, and a default dropout rate of 0.2. Additionally, we utilized TensorBoard to provide our visualizations for NLL, EM, F1, and AvNA.

4.4 Results

Figure 1 (shown below) provides a table that shows the variations in performance that we saw on the dev set as we finetuned hyperparameters on both our Character-level Embedding only model as well as our Coattention + Character-level Embedding.

The following figure shows our leaderboard results for our Coattention + Character-level Embeddings model with a learning rate of 0.6 and dropout probability of 0.15. The results show how the model performed relative to the baseline's performance on the dev set while also showing the performance of our model on the test set.

Below that is a figure that shows the performance of our tuned coattention + character-level embeddings model on Tensorboard.

We summarize some of our main points below.

Model	lr	dropout	F1	EM
Baseline	0.5	0.2	60.74	57.39
Char-level Embs	0.5	0.2	62.14	58.86
Char-level Embs	0.4	0.2	61.12	58.23
Char-level Embs	0.6	0.2	60.86	58.11
Char-level Embs	0.5	0.25	61.92	58.47
Char-level Embs	0.6	0.15	62.27	59.32
Char-level Embs + Coattention	0.5	0.2	62.22	58.80
Char-level Embs + Coattention	0.4	0.2	60.03	57.19
Char-level Embs + Coattention	0.6	0.2	61.78	59.06
Char-level Embs + Coattention	0.2	0.3	57.31	54.17
Char-level Embs + Coattention	0.6	0.15	63.91	60.43

Figure 1: Hyperparameter Tuning vs Performance (EM and F1 scores) on Dev Set

Model	F1	EM
Baseline on Dev	60.737	57.385
Char-Level Embs + Coattention + Tuned Params - Dev	63.914	60.427
Char-Level Embs + Coattention + Tuned Params - Test	62.787	59.189

Figure 2: Model Performance (EM and F1 scores)

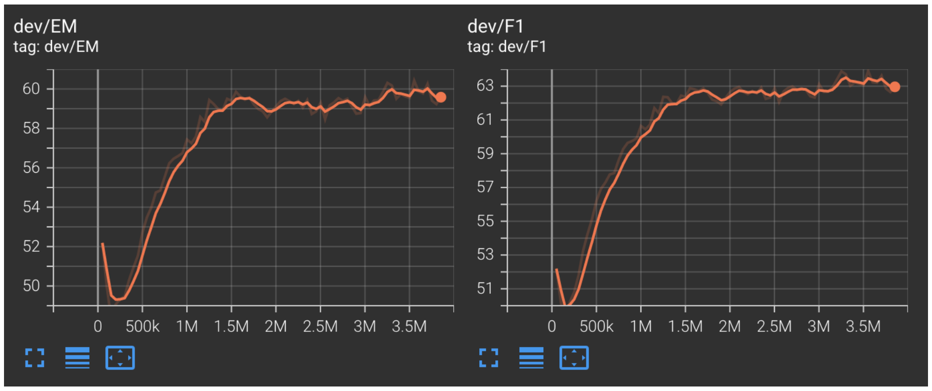


Figure 3: Best Model (Coattention + Char-level Embeddings + Hyperparameter Search) Performance (EM and F1 scores)

- As expected, we found that adding Character-Level Embeddings improved the overall performance of the baseline model. We believe this improved the performance of the model as the robustness of the model was enhanced by allowing for conditioning on the internal structure of words which also helped with handling words not present in the GloVe vocabulary.

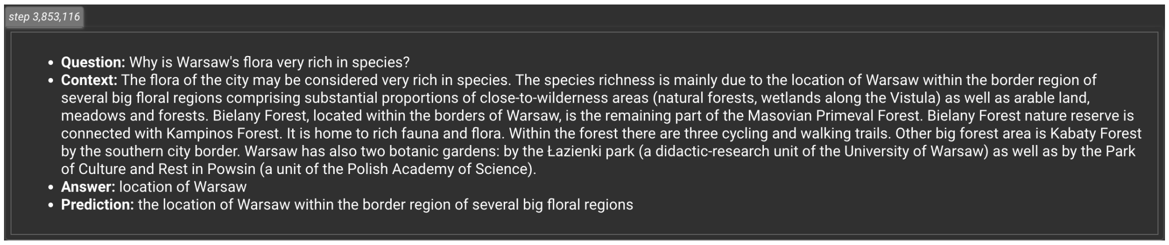
- Adding a Coattention Layer proved to be less effective overall than we expected. Though the Coattention + Character-Level Embeddings performed well on the dev set, the performance on the test set is less impressive. This difference may be explained by overfitting to our dev set. Additionally, in many cases in the dev set, we found that adding the Coattention Layer reduced performance relative to only including the character-level embeddings. We suspect this to be a product of the fact that substituing the Coattention layer does not really add any more trainable parameters. Furthermore, we hypothesize that adding a Coattention layer in addition to the Character-level Embeddings may have placed too large of an importance to the character-level which may have distracted the model from focusing on words important to answering the question.

- With respect to our hyperparameter finetuning, though the initial parameters proved to be fairly effective, we find that increasing learning rate and dropping the dropout probability generally increases the model's performance. We believe that increasing the learning rate ended up helping the performance of the model as the default learning rate may have been too low, causing training to finish before an optimum was arrived at. With respect to the dropout probability, we suspect that lowering the dropout probability generally improved our model's performance as initially, there may not have been enough information from which the model could learn from.

5 Analysis

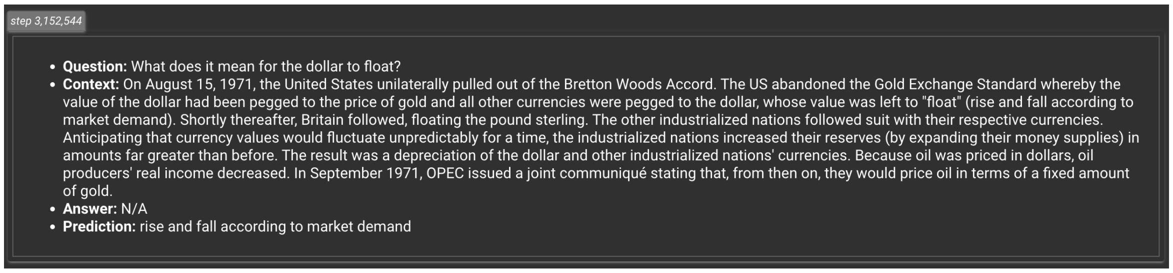
We chose to analyze the results from out best model (BiDAF with character-embeddings and coattention after hyperparameter tuning) and pulled examples from the final steps of the development set.

A common error that we saw was that the model pulled more data than necessary from the context; however, we note that this is not necessarily incorrect since the extra information was also relevant to the ground truth answer.


step 3,853,116

- **Question:** Why is Warsaw's flora very rich in species?
- **Context:** The flora of the city may be considered very rich in species. The species richness is mainly due to the location of Warsaw within the border region of several big floral regions comprising substantial proportions of close-to-wilderness areas (natural forests, wetlands along the Vistula) as well as arable land, meadows and forests. Bielany Forest, located within the borders of Warsaw, is the remaining part of the Masovian Primeval Forest. Bielany Forest nature reserve is connected with Kampinos Forest. It is home to rich fauna and flora. Within the forest there are three cycling and walking trails. Other big forest area is Kabaty Forest by the southern city border. Warsaw has also two botanic gardens: by the Łazienki park (a didactic-research unit of the University of Warsaw) as well as by the Park of Culture and Rest in Powsin (a unit of the Polish Academy of Science).
- **Answer:** location of Warsaw
- **Prediction:** the location of Warsaw within the border region of several big floral regions

Another common error that we saw was that the model was able to pull answers from the context, when the answer was labeled N/A. Although the prediction did not match the answer, many examples of this error can be attributed to human labeling error within the dataset. In the following example, the definition for a dollar to float is included within the context, but the answer is labeled N/A. The model was still able to pull the definition out of the context but was unfortunately counted as incorrect.


step 3,152,544

- **Question:** What does it mean for the dollar to float?
- **Context:** On August 15, 1971, the United States unilaterally pulled out of the Bretton Woods Accord. The US abandoned the Gold Exchange Standard whereby the value of the dollar had been pegged to the price of gold and all other currencies were pegged to the dollar, whose value was left to "float" (rise and fall according to market demand). Shortly thereafter, Britain followed, floating the pound sterling. The other industrialized nations followed suit with their respective currencies. Anticipating that currency values would fluctuate unpredictably for a time, the industrialized nations increased their reserves (by expanding their money supplies) in amounts far greater than before. The result was a depreciation of the dollar and other industrialized nations' currencies. Because oil was priced in dollars, oil producers' real income decreased. In September 1971, OPEC issued a joint communiqué stating that, from then on, they would price oil in terms of a fixed amount of gold.
- **Answer:** N/A
- **Prediction:** rise and fall according to market demand

6 Conclusion

In this project, our work was based upon augmenting the embedding layer of the provided baseline model with character-level embeddings and also included implementing a Coattention Layer that we used as a substitute for the provided basic Attention Layer. We were able to produce conclusive results that improved over the baseline with our Coattention + Character-Level Embeddings model while using a learning rate of 0.6 and a dropout probability of 0.15. This model led to an EM score of 60.43 and a F1 score of 63.91 on the dev set, higher performance compared to the baseline model's EM score of 57.385 and F1 score of 60.737. On our test set, the same model also performed the best with an EM score of 59.189 and an F1 score 62.787. We also found it notable that simply including character-level embeddings to the baseline model improved the model's performance with EM/F1 scores of 58.864 and 62.179 respectively, demonstrating the significance of having the robustness to capture OOV words.

Though we wished to implement additional and explore additional attention mechanisms as well as additional hyperparameter tuning, we were limited by both the time constraints that come with the extensive testing required to tune hyperparameters as well the time constraints that inherently come with the long training times of these models. Going forward, we would be interested in exploring the impact of attention mechanisms like self-attention on performance and would also be interested in implementing either additional preprocessing or a feed-forward model to identify and separate different question types and have separate models train on each question type. We believe this could potentially solve the errors made by the current implementation as the separate models can specialize and train on a singular question type and reduce the number of predicted dimensions within the output.

References

[1] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[2] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension, 2016.

[3] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. In *arXiv*, 2018.