BiDAF with Character and Subword Embeddings for SQuAD

Stanford CS224N Default Project

Yining Zhu
Department of Computer Science
Stanford University
`zhuy0713@stanford.edu`

Abstract

Question Answering (QA) is a type of NLP task that assesses the system's ability to answer questions posed by humans in a natural language. Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset designed to challenge systems in such tasks. Given a question and a context, the system needs to automatically predict the span of text within the context that answers the question, or decide that the answer does not lie within the context. In this paper, we implement a bi-directional attention flow (BiDAF) model, with character-level embeddings and subword-level embeddings. We also experimented with using different learning rates and dropout probabilities. The best performance we got on SQuAD test set is F1: 65.049, EM: 61.385.

1 Introduction

Question Answering (QA) tasks provide an effective way to measure the system's ability of 'understanding' text. A reliable QA system would demonstrate strong ability of understanding the natural language, and has wide applications in our life. The SQuAD dataset provides high quality training data and test set for such tasks, and is frequently used as a benchmark for different QA systems. BiDAF model provides a framework that uses several layers and a bi-directional attention flow to learn the query-aware context representation[1]. Based on the BiDAF model, we implemented character-level embeddings and subword-level embeddings on top of the word-level embeddings in the starter code. The best results on the SQuAD dev set and test set were achieved by concatenating character-level embeddings and the word-level embeddings. In this paper, we would outline the model architecture of BiDAF model, with a focus on the character-level embeddings and subword embeddings that we implemented. Then, we will present our results using different embedddings and hyperparameters. At last, we will discuss the factors that affected the results and future work.

2 Related Work

There are many study on QA that uses SQuAD as the training data and test set. The BiDAF model used for this project is one of them[1]. The hierarchical multu-stage architecture achieved great success on this task. This paper will be focusing on improving the embedding layer of this model to achieve better results. Over the year, there has been major development on the text embeddings. The BiDAF model used character-level embeddings trained with a convolution network and word-embeddings obtained using the GloVe algorithm[1][2]. The GloVe representation can capture some syntactic and semantic information and can be used to predict lexical similarity and analogy[2]. BERT has also been used to achieve good results, even above the human performance. It utilizes transformer to learn the contextual relations between words[3]. Another study has adopted the subword-augmented emdedding to train the reading comprehension model, which helps to mitigate the OOV problems[4]. In this paper, we are also going to implement the Byte Pair Encoding to obtain the subword-level embeddings, as well as the character-level embeddings outlined in the BiDAF paper[1].

3 Approach

- **Baseline:** For our baseline model, we used a bi-directional attention flow model based on [1], provided in the starter code. The baseline used only word-level embeddings, not character-level embeddings.
- **Embedding Layer**:

  we implemented a character-level embedding in addition to the word-level embedding in the starter code. First, we encoded each character as an integer. Each word of length l is then represented as a vector of integers $x \in \mathbb{Z}^l$. Second, we padded every word so that they have the same length m, we have predefined it to be 20 but will see if adjustion needs to be made. Then, we looked up the character embedding for each of the character, which had the shape e. After reshape, we got an embedding of the shape $\mathbb{R}^{m \times e}$. We then combined the character embeddings using a one-dimensional convolution network. We set the kernel size as 5 and the number of output features to that of the word-level embeddings. We then applied the ReLU function and MaxPool to produce the output.

  We also implemented subword embeddings. The segmentation and initial embedding vectors are from BPEmb, the pretrained subword embeddings[5]. We used the python module BPEmb to segment each words into segments and then find the corresponding subword vectors. The python module is based on the Byte Pair Encoding algorithm. It segments the words by clustering the most frequent character sequences. It then applies segmentation to Wikipedia data and generates the subword embeddings using the GloVe algorithm. For every word in the contexts and the questions, we segmented it into subwords, and then lookup the corresponding embeddings for each subword. We applied linear transformation to the embeddings to match the number of hidden layers. We then applied the ReLu function and MaxPool to produce the embeddings.

  We compared two combinations of embeddings. The first one is a concatenation of character-level embeddings and the second one is a concatenation of character-level, subword, and word embeddings.
- **Encoder Layer**: The encoder layer uses a bidirectional LSTM to allow the model to incorporate temporal dependencies between timesteps of the embedding layer's output. layer's output[6].
- **Attention Layer**: The attention flow layer uses the vector representation from the previous layer and computes similarity matrices and then the the query-aware vector representation **G** [6].
- **Modeling Layer**: This layer uses two layers of bi-directional LSTM to obtain a matrix that captures the interaction among the context words conditioned on the query[6].
- **Output Layer**: this layer adapts to specific applications. In this project, the output layer predicts the answer by giving the start index and the end index of the context[6].

4 Experiments

- **Data**: To train the model, we use the SQuAD 2.0 machine comprehension training dataset, which contains 129,941 examples[6].
- **Evaluation method**: We measure it against the test and dev datasets included in SQuAD. We use two metrics to quantify the results:
  - Exact Match: a binary measure which indicates whether the machine prediction matches the human answer exactly.
  - F1: the harmonic mean of precision and recall.

$$F1 = \frac{2 \times prediction \times recall}{precision + recall}$$

- **Experimental details**: For all experiments, we set the number of epochs to 30, and the batch size to 64. We implemented the character-level embeddings on top of the word-level embedding in the baseline model. We set the dropout to 0.2 and the learning rate to 0.5. We also experimented with adjusting the learning rate to 0.7, 0.3, and 0.4, and adjusting the dropout probabilities to 0.3 and 0.15. We also implemented the subword embeddings on top of the word embeddings and character-level embeddings. We set the learning rate to 0.7, and dropout to 0.2 as the default. Due to time constraints, we did not tune hyper-parameters on this model.

- **Results**: Overall, the best performance that we have achieved is F1=65.05, and EM=61.39. The corresponding model used character-level embeddings, a learning rate of 0.7, and a dropout probability of 0.2.
  From the results, adding character-level embeddings improved the model significantly (F1 62.03 -> 65.05). The character-level embeddings increased the granularity of the embeddings, which might have led to better performance. We can also see that increasing the learning rate to 0.7 resulted in better performance while decreasing the learning rate worsened it. Increasing the learning rate might have prevented the model from overfitting. Although it is interesting to see that the model performs better with a learning rate of 0.3 than 0.4. The differences are not significant enough, so more experiments might be done in respect of the learning rate. We also tried to tweak the dropout probabilities. Neither the model with a dropout probability of 0.15 or 0.3 performed better than with the default dropout probability. It could be that 0.2 is close to the optimal dropout for this model, or only the locally optimal dropout. What we did not expect is that adding the subword embeddings worsened the performance by a lot. One of the reasons could be that the subword segmentation is not always morphologically accurate. For example, the subwords generated for 'babysit' is 'bab', 'ys' and 'it'. This might have led to innacurate representation of the texts. Due to time constraint, we did not tune the hyperparameters for the models with subword embeddings, and we might get better results with more tuning.

Model(embeddings)	lr	dropout	Dev NLL	F1	EM	AvNA
Baseline	0.5	0.2	3.18	60.32	56.97	67.57
Word + char-level	0.5	0.2	03.18	62.03	58.98	68.66
Word + char-level	0.7	0.2	02.89	**65.05**	**61.39**	71.18
Word + char-level	0.7	0.3	02.90	63.14	59.99	69.75
Word + char-level	0.7	0.15	03.45	62.93	59.38	70.11
Word + char-level	0.3	0.2	03.22	61.62	58.36	68.44
Word + char-level	0.4	0.2	03.18	61.40	57.99	68.19
Word + subword + char-level	0.7	0.2	03.59	60.96	57.54	67.72

Figure 1: Results from the baseline model and implementation of char-level and subword embeddings on SQuAD 2.0 dev dataset.

5 Analysis

We have a few examples from the test results.

- **Question:** Who went to Fort Dusquesne in June 1755?
- **Context**: Braddock (with George Washington as one of his aides) led about 1,500 army troops and provincial militia on an expedition in June 1755 to take Fort Duquesne. The expedition was a disaster. It was attacked by French and Indian soldiers ambushing them from up in trees and behind logs. Braddock called for a retreat. He was killed. Approximately 1,000 British soldiers were killed or injured. The remaining 500 British troops, led by George Washington, retreated to Virginia. Two future opponents in the American Revolutionary War, Washington and Thomas Gage, played key roles in organizing the retreat.
- **Answer**: Braddock (with George Washington as one of his aides) led about 1,500 army troops
- **Prediction**: N/A

All of our models predicted N/A for this question. It is not surprising since the actual word for the answer is far from the words that appeared in the question, despite being in the same sentence. The model that we created did not change the attention layers and RNN layers, thus was not able to have improvements on retaining of information over a long distance and attending the important information in the sentence. Implementing the self-attention mechanism might be able to solve that.

- **Question:** How far is New Rochelle from New Paltz?
- **Context**: Huguenot immigrants did not disperse or settle in different parts of the country, but rather, formed three societies or congregations; one in the city of New York, another 21 miles north of New York in a town which they named New Rochelle, and a third further upstate in New Paltz. The "Huguenot Street Historic District" in New Paltz has been designated a National Historic Landmark site and contains the oldest street in the United States of America. A small group of Huguenots also settled on the south shore of Staten Island along the New York Harbor, for which the current neighborhood of Huguenot was named.
- **Answer**: N/A
- **Prediction**: 21 miles

Both the baseline model and the model with char-level embeddings predicted 'N/A' while the model with subword embedding predicted 21 miles. One possibility might be that the incorrect subword segmentation confused the model. The segmentation of 'rochelle' is 'ro', 'chel', and 'le'. It might have harmed the representation of the question and the context. We also need to look at the pre-trained word embeddings and compare them with the word embeddings from GLoVe and see how they are contributing to the understanding of semantics and syntax.

6 Future work

There are many experiments that we did not do because of time constraint and the computation cost. We only ran one experiment using the subword embeddings. In the future, we want to run more experiments where we concatenate the subword embeddings with the word embeddings, as well as building the embeddings with only subword embeddings. These experiments will help us understand the effectiveness of subword embeddings better. At the same time, we also need to tune the hyperparameters for the models with subword embeddings for better results. Moreover, since we only focused on the embedding layer for this experiment, we hope to implement other functions like the coattention networks.

References

[1] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[2] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Zhuosheng Zhang, Yafang Huang, and Hai Zhao. Subword-augmented embedding for cloze reading comprehension. *arXiv preprint arXiv:1806.09103*, 2018.

[5] Benjamin Heinzerling and Michael Strube. Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages. *arXiv preprint arXiv:1710.02187*, 2017.

[6] Cs 224n default final project:building a qa system (iid squad track).