CS224N Default Final Project Report: Building a QA System Using BiDAF and Subword Modeling Techniques

Stanford CS224N Default Project

Yanichka Ariunbold
Department of Computer Science
Stanford University
yanichka@stanford.edu

Nina Prabhu
Department of Computer Science
Stanford University
nsp@stanford.edu

Abstract

Recently, the task of question answering (QA) has gained significant popularity in the field of natural language processing. For this project, we explore deep learning architectures, in particular the Bi-Directional Attention Flow (BiDAF) network, to construct a better-performing model than the baseline we were provided with. Our baseline model achieved 57.54 EM and 60.90 F1 in the dev set. Based on this, we experimented with concatenating character embeddings with word embeddings and other forms of subword modeling. We implemented a subword vocabulary with Byte-Pair Encoding and made a corresponding embedding layer, but found that it had suboptimal performence, likely due to its potential confusion with out-of-vocabulary words in the dev and test datasets. Our final system and best-performing model is the BiDAF network with the character embedding layer, where character and word embeddings are concatenated in equal part (50/50). Our best results achieved 60.595 EM and 63.587 F1 on the dev set and 59.222 EM and 62.662 F1 on the test set.

1 Introduction

Applying machine learning to the challenging task of reading comprehension can demonstrate the extent to which systems can truly understand text. Neural network architectures in particular have exhibited the ability to learn reasoning over various textual contexts. Recently, the task of question answering (QA) has gained significant popularity in the field of natural language processing. This is in part due to the variety of important and widely used applications that benefit from QA technology, such as search engines. For this project, we will be exploring deep learning architectures to build a question answering system that works well on the SQuAD 2.0 dataset.

2 Related Work

Seo et al. [1] proposed a novel attention mechanism that uses bidirectional attention flow (BiDAF). The hierarchical end-to-end network, which uses a context-query attention layer in addition to embedding and encoder layers, is composed of 6 total layers. The following stood out out to us as potential layers to implement, due to their promising results in the paper:

- **Character Embedding Layer**. Maps each word to a high-dimensional vector space and uses Convolutional Neural Networks (CNN) to get character-level embeddings for each word.

Sennrich et al. proposed using Byte-Pair Encoding to build up a subword vocabulary to use for a corresponding embedding layer. This would function as a hybrid between character and word

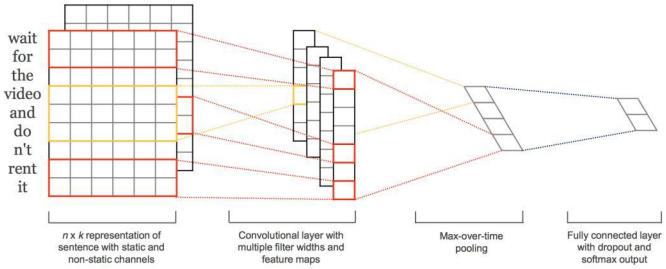
embeddings, providing just enough flexibility to interpret unknown words retain meanings of our units (subwords). [2] Gan et al. further showed that using subword embeddings in a question-answering model made it more robust with question paraphasing on the SQuAD dataset. Without using subword embeddings, rephrased questions had a far lower correct answer rate then with subword embeddings, where accuracy rates were very similar. [3] For this project, we explore deep learning architectures to build a question answering system that works well on the SQuAD 2.0 dataset, using these papers as guidance to implement multiple forms of subword modeling.

3 Approach

The baseline model in the starter code is based on Seo et al.'s BiDAF model but without the character embedding layer. Thus, the starter model is made up of five layers: Word Embedding Layer, Contextual Embedding Layer, Attention Flow Layer, Modeling Layer, and Output Layer. We extend this starter model by adding character embedding and Byte-Pair encoding to improve its performance.

1. **Character Embedding Layer.** As mentioned in the paper we reviewed, character level embeddings often improve the performance of NLP models because they make up for many of word embedding's shortcomings, particularly the ability to provide context with unfamiliar or rarely-seen words. In order to implement character embeddings, we used two separate embedding modules, one for word and the other for character, and concatenated the two outputs together to input into a Highway Network. The word embedding layer was already provided to us in the starter code, and we used a similar framework to build the character embedding module. Using Kim et al.'s work [4] as guidance, we implemented a 1-dimensional convolutional network (CNN) in which to input the provided pre-trained character vectors, whose size we made the input channel size of the CNN. We then max-pooled the outputs to get the most useful vector for each word. A demonstration of the process, pulled from Kim's paper, is shown below.

Figure 1: Character CNN layer

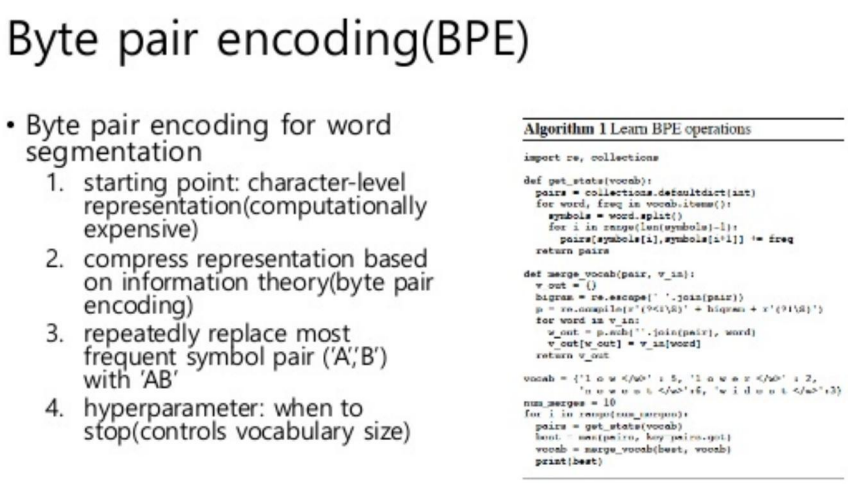


2. **Subword Embedding Layer.** To give our model a level of further context between the specificity of character embedding and word embedding, we decided to create subword embeddings and a corresponding layer. We believed this would help increase our F1 score, given that this approach in [3] made the model more robust to paraphrased questions. Since we discussed Byte-Pair Encoding in lecture and it was strongly supported in [2], we decided to use BPE to create our subword vocabulary. We implemented this in two different ways:
   (a) **Using pre-trained embeddings.** Using a Python module that contains a collection of pre-trained subword embeddings based on BPE and trained on Wikipedia (https://nlp.h-its.org/bpemb/). We instantiated an instance of the module in English with vocabulary size 100,000 and wrote code in setup.py to create the subword embedding and dictionary json files.
   (b) **Manually implementing Byte-Pair Encoding.** Using the BPE algorithm (shown below), we wrote code to build a vocabulary of size 10,000 based on the training data. (Note: 10,000 was chosen due to time constraints, as it would have taken a significant amount of time to build up our vocabulary to 100,000 or a larger size.) This code was adapted from the approach used in https://leimao.github.io/blog/Byte-Pair-Encoding/ and modified for our vocabulary size, dataset formatting, and other differences to best

suit the task at hand. We also wrote code to split a word into the longest possible subwords, based on our vocabulary. We then used the vocabulary and word-splitting function to create the subword embedding and dictionary json files.

Both of these approaches involved modifying setup.py and in turn, the .npz files that held the train, dev, and test data. To accommodate this change in data format, we modified utils.py and args.py to read in the proper data files in the right way and use subword embeddings as well as just character and word embeddings. Once the data was read in properly, we implemented a subword embedding layer in layers.py, similar to the character and word embedding layers. Since we now had an additional layer, we increased the hidden size to 150 to keep the hidden sizes for the word and character embedding layers at 50, the same as the subword embedding layer. This change meant that all three layers could be sufficiently used by the model, while adding up correctly when concatenating the three embeddings in the forward function.

Figure 2: BPE Algorithm



For the loss function for all attempted models, per the handout we used the sum of the negative log-likelihood loss for the start and end locations. For a single example, the function is as follows:

Figure 3: Loss Function

$$\text{loss} = -\log \boldsymbol{p}_{\text{start}}(i) - \log \boldsymbol{p}_{\text{end}}(j)$$

4 Experiments

4.1 Data

We use the SQuAD 2.0 dataset as mandated, which contains almost 130,000 (context, question, answer) triples for training, along with around 6000 triples in each of the dev and test sets. As specified in the project description, code for preprocessing this data was provided to us.

4.2 Evaluation method

We use EM and F1 scores as evaluation metrics. Our rank on the leaderboard with respect to the rest of the class helps us judge these scores.

4.3 Experimental details

All models were run using a virtual machine on Azure. For standard training, we set the number of epochs to 30, a dropout probability of 0.2, a batch size of 64, and the optimizer as Adadelta.. With all models, we also experimented with various hyperparameters, discussed below.

When running setup.py to create the manual subword BPE embeddings, the process took roughly 12 hours to complete even after using memoization, mainly due to the splitting-word function. When using the pretrained subword embedding module, setup.py only took roughly 30 minutes to complete. All model versions below include a word embedding layer unless stated otherwise.

- **Baseline.** The baseline model took around 11 hours to train. We used a learning rate of 0.5.
- **Character Embedding.** The character embedding model took around 11.5 hours to train. We experimented with varying learning rate of 0.4, 0.5, 0.7, and 0.9. We used kernel (or window) sizes of 4 and 5, which were both within the range of kernel sizes mentioned in the Kim paper. We primarily used a dropout probability of 0.2, but we also experimented with 0.1. We further primarily used the Bidirectional LSTM as a RNN, but we tried using a GRU for a trial to see if it would cut down the trining time. We also experimented with the Adam optimizer, although we ended up doing an early stop with that trial due to unpromising results.
- **Manual Subword BPE Embedding.** The manual subword embedding model took around 12 hours to train. We used a learning rate of 0.5.
- **Character Embedding + Manual Subword BPE Embedding.** The combined embedding model took around 12 hours to train. We used a learning rate of 0.5.
- **Pretrained Subword BPE Embedding.** When trying to train from the pretrained subword embeddings, we received dimension mismatch errors likely due to the difference in formatting between the module and the starter code. In the interest of time and effort, we decided to focus on the manual subword embeddings instead, as we felt they would be more representative of the specific texts we were training on and prevent the model from knowing information and subwords it shouldn't.

4.4 Results

Table 1: BiDAF-based Model Results on Dev Dataset

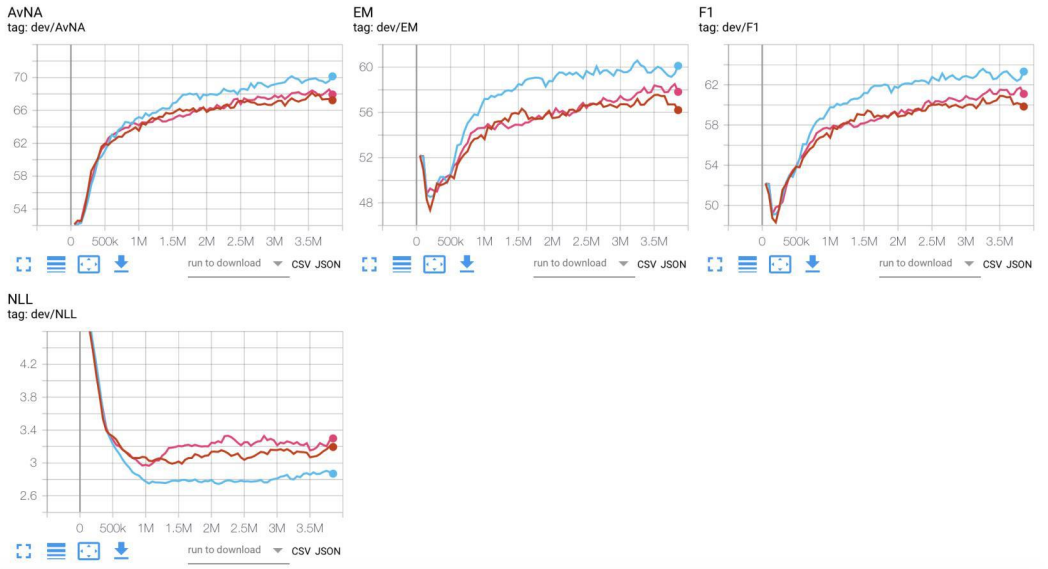
Case	Char	BPE	LR	Concat-Ratio	RNN	Opt.	Dropout	Kernel	EM	F1
1	×	×	0.5	–	LSTM	Adadelta	0.2	–	57.54	60.90
2	✓	×	0.5	50/50	LSTM	Adadelta	0.2	4	60.60	63.59
3	✓	×	0.5	50/50	GRU	Adadelta	0.2	4	57.92	61.16
4	✓	×	0.5	50/50	LSTM	Adadelta	0.2	5	58.91	62.48
5	✓	×	0.5	50/50	LSTM	Adadelta	0.1	4	59.10	62.81
6	✓	×	0.5	50/50	LSTM	Adam	0.2	4	early stop	early stop
7	✓	×	0.5	60-c/40-w	LSTM	Adadelta	0.2	4	early stop	early stop
8	✓	×	0.5	70-c/30-w	LSTM	Adadelta	0.2	4	60.26	63.54
9	✓	×	0.5	30-c/70-w	LSTM	Adadelta	0.2	4	early stop	early stop
10	✓	×	0.7	50/50	LSTM	Adadelta	0.2	4	59.94	63.19
11	✓	×	0.9	50/50	LSTM	Adadelta	0.2	4	early stop	early stop
12	✓	×	0.4	50/50	LSTM	Adadelta	0.2	4	early stop	early stop
13	×	✓	0.5	50/50	LSTM	Adadelta	0.2	–	58.51	61.73
14	✓	✓	0.5	50/50/50	LSTM	Adadelta	0.2	4	58.18	61.76
15	✓	✓	0.5	60-c/30-b/60-w	LSTM	Adadelta	0.2	4	58.90	62.48

The "Concat-Ratio" column is the ratio between each type of embedding layer's dimensions before they are concatenated together. If there are two numbers, it is the ratio of character or subword to word embedding dimensions, and if there are three numbers, it is the ratio of character to subword to word embedding dimensions. Above, for increased transparency, "c" stands for character, "b" stands for BPE, and "w" stands for word. Furthermore, trials labeled with "early stop" do not necessarily

mean that they were extremely unpromising during training; due to time constraints, we stopped some trials early if it was clear they were performing below the best-performing model thus far.

Our best-performing model with character embeddings achieved 60.60 EM and 63.59 F1 in the dev set, showing a marked improvement over the baseline model (+3.06 EM and +2.69 F1). They then achieved a very comparable 59.22 EM and 62.66 F1 on the test set, showing that the model is fortified against overfitting. As of submission time for this report, our scores are 51/76 models on the test leaderboard.

Figure 4: Training Metrics for Baseline, Char+Word, and BPE+Word models



The Tensorboard plots above show the baseline model (dark red), the best-performing char embedding model (blue), and the BPE model (pink). For our baseline model, we obtained 57.54 EM and 60.90 F1 in the dev set, and we see that the dev EM and F1 scores initially worsen before increasing due to the model predicting no-answer to optimize NLL loss early on. We also see that the dev loss begins to rise between 1 and 2 million iterations, which we can attribute to overfitting.

For the char-word embed model, we see that the dev NLL stabilizes around 2.8, with less sharp declines and increases than the baseline model, whose NLL fluctuated between 3 and 3.2 for most of training. (No smoothing was applied to any plot.) Our dev NLL still increases a bit around 3 million iterations due to overfitting. We also see a slightly more stable curve in the EM and F1 plots.

The BPE(subword)-word model has crossover with the baseline in multiple timesteps but ends up with a dev set result of 58.51 EM and 61.73 F1, still showing an improvement over the baseline (+0.97 EM and +0.83 F1). However, the model clearly performs worse than our best-performing char embed model, showing a decrease of 2.09 EM and 1.86 F1. Interestingly, the BPE model has a higher NLL curve than the baseline. It also has a comparably stable curve in the F1 as the char embed model, and has a more stable curve in the EM despite being consistently below the char embed model in both measurements.

With our char embed model, we experimented with a wide variety of concatenation ratios and found the best-performing one to be the 50/50 split between character and word embeddings. We also ran a trial with a GRU instead of a LSTM early on, hoping that training time would be cut, but we instead got a slightly worse performing model and no definite decrease in training time, so we chose to use LSTMs the remaining trials. We further found the best (out of our trials) optimizer was Adadelta, the best learning rate was 0.5, and the best dropout rate was 0.2, though we did not experiment with a higher dropout rate than 0.2, only lower.

We also experimented with combining character embeddings, BPE embeddings, and word embeddings. Doing a 50/50/50 split between the three led to a EM of 58.18 and F1 of 61.76. Interestingly, our BPE-word model performed better EM-wise with a EM of 58.51. However, when we increased

the ratio of character embeddings and word embeddings to 60 in the three-embedding model so that the overall ratio was 60-character/30-BPE/60-word, we did see a uniform increase in EM and F1 (59.90 EM and 62.48 F1) of around 0.7 from the plain BPE-word model. Needless to say, the BPE-word model's better performance in EM than the 50/50/50 three-embed model with character embeddings doesn't necessarily show that BPE embeddings are additive with respect to exact matches since it still performs worse than the char-word model.

5 Analysis

The greater increase in the EM and F1 scores for the character embedding model in particular is attributable to how character embeddings allow for every single word's vector to be formed even if it is out-of-vocabulary. Surprisingly, adding a subword embedding layer to our model (in both the BPE-word and three-embedding model) decreased its performance relative to the model with only character and word embeddings. However, we believe this can be attributed to higher confusion concerning out-of-vocabulary words, as we built up our vocabulary through iterating over the training dataset.

Additionally, it is worth noting that our F1 scores for our subword models (cases 13-15 in the table) were relatively high. This leads us to believe that the subword model failed to match the exact wording due to confusion with out-of-vocabulary words, but was able to utilize its subword knowledge to understand some meaning in the questions.

Figure 5: Incorrect baseline and BPE-word embedding model example

- **Question:** Economy, Energy and Tourism is one of the what?
- **Context:** Subject Committees are established at the beginning of each parliamentary session, and again the members on each committee reflect the balance of parties across Parliament. Typically each committee corresponds with one (or more) of the departments (or ministries) of the Scottish Government. The current Subject Committees in the fourth Session are: Economy, Energy and Tourism; Education and Culture; Health and Sport; Justice; Local Government and Regeneration; Rural Affairs, Climate Change and Environment; Welfare Reform; and Infrastructure and Capital Investment.
- **Answer:** current Subject Committees
- **Prediction:** N/A

Figure 6: Slightly correct char-word embedding model example

- **Question:** Economy, Energy and Tourism is one of the what?
- **Context:** Subject Committees are established at the beginning of each parliamentary session, and again the members on each committee reflect the balance of parties across Parliament. Typically each committee corresponds with one (or more) of the departments (or ministries) of the Scottish Government. The current Subject Committees in the fourth Session are: Economy, Energy and Tourism; Education and Culture; Health and Sport; Justice; Local Government and Regeneration; Rural Affairs, Climate Change and Environment; Welfare Reform; and Infrastructure and Capital Investment.
- **Answer:** current Subject Committees
- **Prediction:** Subject Committees

Figure 7: More correct three-embed model example

- **Question:** Economy, Energy and Tourism is one of the what?
- **Context:** Subject Committees are established at the beginning of each parliamentary session, and again the members on each committee reflect the balance of parties across Parliament. Typically each committee corresponds with one (or more) of the departments (or ministries) of the Scottish Government. The current Subject Committees in the fourth Session are: Economy, Energy and Tourism; Education and Culture; Health and Sport; Justice; Local Government and Regeneration; Rural Affairs, Climate Change and Environment; Welfare Reform; and Infrastructure and Capital Investment.
- **Answer:** current Subject Committees
- **Prediction:** The current Subject Committees

In the question above, the three-embed model (that includes character, BPE, and word embeddings) got closest to the correct answer with "The current Subject committees". The char-embed, baseline, and BPE-embed models had less informative answers, showing in some cases that the BPE embedding layer with the character embedding layer can together result in a higher F1 score.

In other examples, the character embedding layer works best alone, shown below. We see that the three-embed model results in the same incorrect answer as the baseline, despite containing the character embedding layer. This supports how our char-embed model can perform substantially better than the other combinations.

Figure 8: Incorrect baseline and three-embed model example

- **Question:** Who designed the garden for the University Library?
- **Context:** Another important library – the University Library, founded in 1816, is home to over two million items. The building was designed by architects Marek Budzyński and Zbigniew Badowski and opened on 15 December 1999. It is surrounded by green. The University Library garden, designed by Irena Bajerska, was opened on 12 June 2002. It is one of the largest and most beautiful roof gardens in Europe with an area of more than 10,000 m2 (107,639.10 sq ft), and plants covering 5,111 m2 (55,014.35 sq ft). As the university garden it is open to the public every day.
- **Answer:** Irena Bajerska
- **Prediction:** Marek Budzyński and Zbigniew Badowski

Figure 9: Incorrect BPE-word embedding model example

- **Question:** Who designed the garden for the University Library?
- **Context:** Another important library – the University Library, founded in 1816, is home to over two million items. The building was designed by architects Marek Budzyński and Zbigniew Badowski and opened on 15 December 1999. It is surrounded by green. The University Library garden, designed by Irena Bajerska, was opened on 12 June 2002. It is one of the largest and most beautiful roof gardens in Europe with an area of more than 10,000 m2 (107,639.10 sq ft), and plants covering 5,111 m2 (55,014.35 sq ft). As the university garden it is open to the public every day.
- **Answer:** Irena Bajerska
- **Prediction:** architects Marek Budzyński and Zbigniew Badowski

Figure 10: Correct char-word embedding model example

- **Question:** Who designed the garden for the University Library?
- **Context:** Another important library – the University Library, founded in 1816, is home to over two million items. The building was designed by architects Marek Budzyński and Zbigniew Badowski and opened on 15 December 1999. It is surrounded by green. The University Library garden, designed by Irena Bajerska, was opened on 12 June 2002. It is one of the largest and most beautiful roof gardens in Europe with an area of more than 10,000 m2 (107,639.10 sq ft), and plants covering 5,111 m2 (55,014.35 sq ft). As the university garden it is open to the public every day.
- **Answer:** Irena Bajerska
- **Prediction:** Irena Bajerska

However, in some cases, the BPE-word model is capable of outperforming the char-word and three-embed models, showing that there is promise using BPE subword modeling. Further investigation is warranted on how to best combine the character and subword layers to result in an answers that combine the best of the two layers. We see an example case in Figures 10-12.

6 Conclusion

In this project, we implemented different embedding methods and encoder architectures based on BiDAF to solve question answering problems. Using various papers and algorithms mentioned in lecture, we decided to focus on character embeddings and BPE subword embeddings. The former proved more successful than the latter, giving us our best model with scores of 60.595 EM and 63.587 F1 on the dev set and 59.222 EM and 62.662 F1 on the test set in the non-PCE division. Though our

Figure 11: Incorrect baseline and char-word embedding model example

- **Question:** When had the Brotherhood renounced violence as a means of achieving its goals?
- **Context:** While Qutb's ideas became increasingly radical during his imprisonment prior to his execution in 1966, the leadership of the Brotherhood, led by Hasan al-Hudaybi, remained moderate and interested in political negotiation and activism. Fringe or splinter movements inspired by the final writings of Qutb in the mid-1960s (particularly the manifesto Milestones, a.k.a. Ma'alim fi-l-Tariq) did, however, develop and they pursued a more radical direction. By the 1970s, the Brotherhood had renounced violence as a means of achieving its goals.
- **Answer:** By the 1970s
- **Prediction:** 1970s

Figure 12: Incorrect three-embed embedding model example

- **Question:** When had the Brotherhood renounced violence as a means of achieving its goals?
- **Context:** While Qutb's ideas became increasingly radical during his imprisonment prior to his execution in 1966, the leadership of the Brotherhood, led by Hasan al-Hudaybi, remained moderate and interested in political negotiation and activism. Fringe or splinter movements inspired by the final writings of Qutb in the mid-1960s (particularly the manifesto Milestones, a.k.a. Ma'alim fi-l-Tariq) did, however, develop and they pursued a more radical direction. By the 1970s, the Brotherhood had renounced violence as a means of achieving its goals.
- **Answer:** By the 1970s
- **Prediction:** the 1970s

Figure 13: Correct BPE-word embedding model example

- **Question:** When had the Brotherhood renounced violence as a means of achieving its goals?
- **Context:** While Qutb's ideas became increasingly radical during his imprisonment prior to his execution in 1966, the leadership of the Brotherhood, led by Hasan al-Hudaybi, remained moderate and interested in political negotiation and activism. Fringe or splinter movements inspired by the final writings of Qutb in the mid-1960s (particularly the manifesto Milestones, a.k.a. Ma'alim fi-l-Tariq) did, however, develop and they pursued a more radical direction. By the 1970s, the Brotherhood had renounced violence as a means of achieving its goals.
- **Answer:** By the 1970s
- **Prediction:** By the 1970s

models with a BPE subword embedding layer performed suboptimally, they received relatively high F1 scores, leading us to believe that these models were able to understand the meaning of words but struggled with out-of-vocabulary words and was misled in exact wording in these cases. For future work, we would like to look into improving the performance of our subword embedding layer by experimenting with parameters, such as the maximum number of subwords a word can be split into or the vocabulary size, that we had to reduce due to time constraints. We would also like to add a self-attention layer to our model, as we experimented with implementing one and then decided to focus on subword modeling due to time constraints and our desire to modify hyperparameters in our character and BPE models.

References

[1] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension, 2018.

[2] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units, 2016.

[3] Wee Chung Gan and Hwee Tou Ng. Improving the robustness of question answering systems to question paraphrasing, 2019.

[4] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models, 2015.