Philip Blumin

NLP Spring 2021

Carl Sable

**For this assignment I have done everything in a colab notebook which is attached in this folder. I have also saved my final model and weights in this same folder.**

**What dataset did you use?**

* For this project I chose a dataset from kaggle called “Emotions dataset for NLP”
* <https://www.kaggle.com/praveengovi/emotions-dataset-for-nlp>
* This dataset is a collection of first person sentences that are classified into one of 6 categories: sadness, anger, love, surprise, fear, joy

**How was the dataset split into training, tuning, and testing portions?**

* Kaggle had provided me with 3 files: train, test, validation, so it was not necessary to split for this task

**What task did you work on?**

* The goal of using this dataset was to experiment with different tensorflow models and preprocessing methods to see which one would work best for the dataset

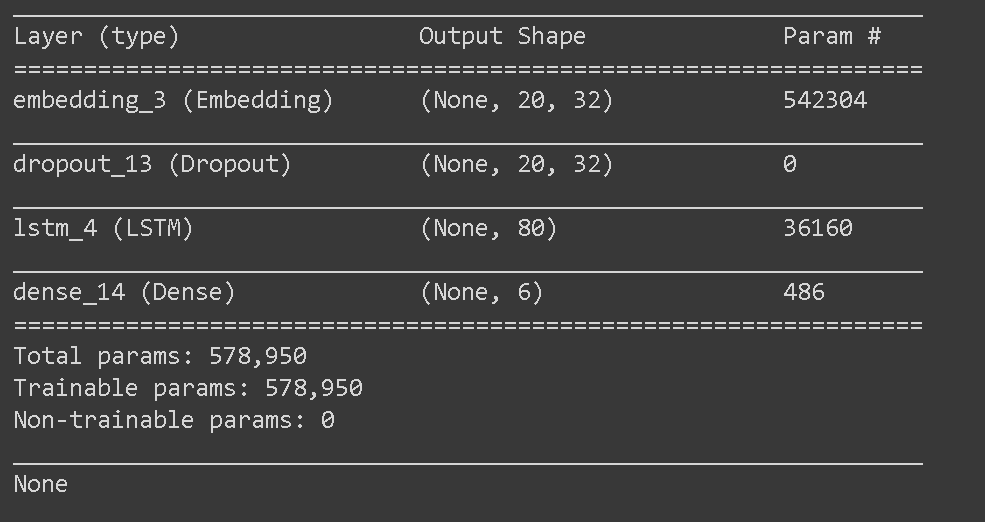
**Which library (TensorFlow or PyTorch) did you use?**

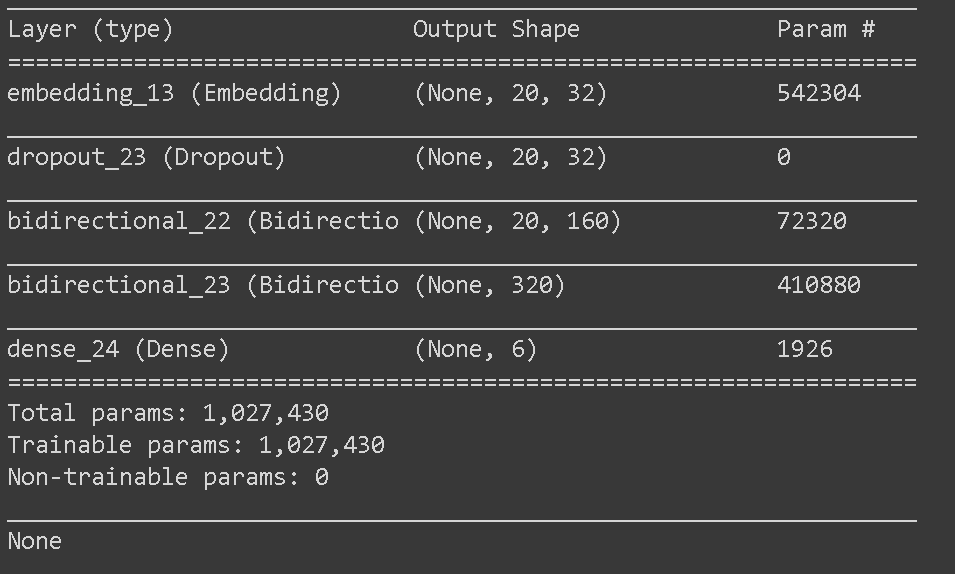
* TensorFlow
  + Dense
  + Dropout
  + Embedding
  + LSTM
  + Bidirectional
  + Tokensizer
  + Pad\_sequences

**Preprocessing:**

* Before training on the data I needed to do a bit of preprocessing to get
* The first thing I did was map each of the emotion categories to a number 0-5
* Next I split up all of the labels and the sentences
* Afterward I counted the total number of unique words in the dataset across the train, test, and validation files.
* After this I applied a tokenizer from the tensorflow preprocessing library. To set up the tokenizer I needed to provide the number of unique words, which I had counted right before. The tokenizer provided each word with a unique numerical index and I mapped each of the words in every sentence to the indexes.
* Finally the last thing that I did before trying my first attempt was to pad all of the sentences to be the same length using pad\_sequences from the tensorflow preprocessing library.
* Aftering trying to train with the first model (I will talk about it in greater detail in the next section), I did not get very good results so I tried to see if there was something I could improve on in the preprocessing stage. Aftering doing some research I saw that a common approach was to remove stop words from all sentences. So using the nltk english corpus I removed all of the stopwords from the sentences and this led to slight improvement

**What architectures and other variables did you experiment with? (Include types of networks, types and sizes of layers, hyperparameters, etc.)**

* First attempt
  + For my first attempt I had not really used tensorflow before so I just followed the tutorial for a sequential on their website. This included alternating between dense and dropout layers. Originally, I tried the relu activation function and I got terrible results, which were around 20% accuracy. Then I switched to softmax and got 35% accuracy which was a huge improvement, so for the rest of the models I built I only used softmax as the activation function. Additionally, I noticed that the model ran pretty fast and ended up leveling out fairly quickly so I chose to run 20 epochs. Finally, I used the “**adam”** optimizer and “**binary\_crossentropy”** for the loss function.
* Second method
  + For this model I tried experimenting with an Embedding layer and a LSTM layer and I got much improved results. I followed the tensorflow tutorials on how to use these. In my first go with this method I used 8 dimensions for the embedding layer and 10 units for the LSTM layer. With these parameters I got around 78% accuracy. I was really pleased to see this huge improvement from the first method, but I still wanted to experiment to see if I could get a higher accuracy. I got slightly better accuracy when I increased the number of dimensions for the embedding layer. I tried 16, 32, and 64 and ended up getting the best results with 32 dimensions. I also tried to increase the LSTM units and got slightly better results. For this method my final model had **32 dimensions for the embedding layer and 80 units for the LSTM layer**. I ended up getting an accuracy of about **89%**. Below is a picture of my model summary from my colab notebook
* Third method
  + This method is very similar to the one I used before except a 2 Bidirectional layer, each with an LSTM layer. From this I got my best results of around **91.5%** accuracy. I experimented with different LSTM units and I got pretty good results when I used 80 and 160.



**What were the choices for the final network?**

* The final network has a 32 dimension embedding layer, a dropout layer, 2 bidirectional layers which each contain an LSTM layer, and a dense layer with a softmax activation function.
* Adam was used for the optimizer and the loss function I choose was binary cross entropy

**What were the final results for the test set?**

