NLP Final Project
NLP with Deep Learning
Theo Song

For the final project of the NLP course, TensorFlow(Keras) was used as the framework. The project was aimed to perform the task of Text Classification(Categorization) that was done during the first project of this course.

For the data set, only the data set of corpus 1 was given. The reason that corpus 1 was selected was because, in most of the tutorials of the tensor flow, a binary classification was done, which showed amazing results. Thus, wanting to test and see how it would work for multi-class classification, corpus 1 was selected.

The training set of corpus 1 contained total of 885 documents (Cri: 100, Str: 282, Dis: 207, Pol: 243, Oth: 53). The training set was divided into two sets, according to the portion parameter. The separation was done so that the ratio between the categories would remain the same. The bigger set was used as the training set and the smaller set was used as the validation set during the fitting of the model. And for the test set (443), everything remained as provided.

The architecture of the project started out as: Embedding -> LSTM -> Dense -> Dense
(as LSTM is known to be good for removing vanishing gradients and handling long term dependencies)

However, the results turned out to be very disappointing, showing an accuracy around 0.7. A change to the architecture was made by replacing the LSTM with a bi-directional layer using the LSTM. Using this, the accuracy improved, with a result around 0.8. Then stacking the bi-directional layer was attempted, which only slowed down the process without making any improvements.
However, during these tests, an overfitting was done through out the fitting. In order to fix the overfitting, a dropout layer was added to the structure. Initially two were used, but was realized that one would suffice.

Thus, the architecture was formed as: Embedding -> Bi-directional (LSTM) -> Dense -> Dropout -> Dense

Other hyperparameters that were examined were the dictionary size for the embedding layer, maximum length of the input data (for each document), the dimensions of the layers, the number of epochs, and the activation functions.

Intuitively, the first guess was that the larger the dictionary size is, better results would be obtained. However, that wasn't necessarily true. As the size of the dictionary got larger than 10000, the accuracy rather seemed to get worse.
Also, initially it was considered that the maximum length of the input for each document should be large enough to allow all the documents to maintain their initial state. As some documents showed a length around the 400s, the initial value was set as 500. However, as experiments were done, it turned out that the accuracy improved as the maximum length decreased, peaking at 210.
For the dimensions of the layers, 64 turned out to be a good value.
The number of epochs were limited to 10 initially, believing that having more would rather overfit the network and give worse results. However, it turned out that wasn't true. Before the number of epoch was increased, the highest accuracy turned out to be 0.83. But, as the number of epoch was increased to 40, the accuracy showed a value of 0.87.
As for the activation functions, rely (for the first Dense layer) and softmax (for the last layer) seemed to work best for this network.

The final choices are seen like below:

```
tf.keras.layers.Embedding(dict_size, embed_D),
tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(embed_D)),
tf.keras.layers.Dense(embed_D, activation='relu'),
tf.keras.layers.Dropout(0.5),
tf.keras.layers.Dense(6, activation='softmax')

training_portion = 0.9
dict_size = 5000
max_len = 210
embed_D = 64
epochs_num = 40
```

results:

The highest accuracy obtained was 0.8736, which turned out to be lower than the results obtained, when using Naive Bayes. Compared to other tensorflow projects of classification, the size of the data set was definitely small. Due to the lack of training set, the result showed no consistency for every iteration, even if given the same parameters. Although the obtained result was 0.8736, it will turn out to be different for different trials.