CS 6390: Phase 1 Project Report

Team Name: "Persuasion Extraction" – Jared Amen (u1228357)

March 27, 2021

1 IE Task

My IE task is a multi-class sequence tagging task. The input is a sentence and a list of fragments from the sentence, for which each is assigned one of the following labels: Appeal to authority: ATA, Appeal to fear/prejudice: ATF, Black-and-white Fallacy/Dictatorship: BF, Causal Oversimplification: CO, Doubt: D, Exaggeration/Minimisation: E, Flag-waving: F, Glittering generalities (Virtue): GGV, Loaded Language: LL, Misrepresentation of Someone's Position (Straw Man): MOSPSM, Name calling/Labeling: NC, Obfuscation, Intentional vagueness, confusion: OIVC, Presenting Irrelevant Data (Red Herring): PIDRH, Reductio ad hitlerum: RAH, Repitition: R, Slogans: Sl, Smears: Sm, Thought-terminating cliche: TC, Whataboutism: W, and Bandwagon: B. The output is (currently) a list of predictions for each sentence from a test dataset, where each entry in the list corresponds to the predicted label for the token.

This task is based off of *subtask 2* of SEMEVAL-2021 Task 6.

2 Resources List

For this task, I used the following libraries:

- Keras for implementation of a model that utilizes a bidirectional LSTM with a dense mapping from hidden layers to the prediction layer
- NLTK for lemmatization and stopword filtering
- SpaCy for tokenization and document creation
- The repository for this specific task, which contains all the necessary corpora as well as the scoring script for my task

3 Technical Description

My IE system implements a RNN using a bidirectional LSTM that implements a dense layer for predictions. For training, tokens are given a BILOU-labeling scheme according to the IE task description above (for instance: B-ATA, I-ATA, L-ATA, O, U-LL, O, etc.). Before training, stopwords and nonalphanumeric tokens are given a special 'SW' label, which is filtered out before generating the tag tensors to pass to the neural network. The vocabulary provided to the system is currently limited to the training and testing datasets provided with the project. The project is set to have a constant embedding size of the maximum possible length of an input sequence, a hidden layer size of 64, and to run for 35 epochs over the training data. It uses the Adam function to update values in the model and categorical cross-entropy as a loss function (ideal for multi-class sequence tagging problems such as this). The model itself can be seen below:

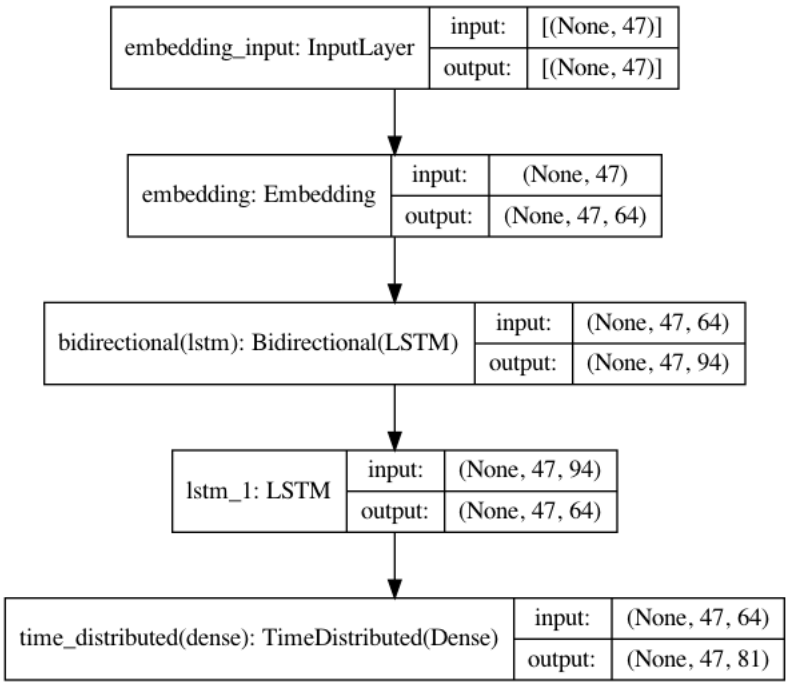

Figure 1: The model implemented by my IE system, visualized

4 Phase 2 vs. Phase 1

The following improvements have been made in phase 2:

- The neural network I have created now implements a bidirectional LSTM with a dense layer instead of a forward LSTM with a linear layer.
- Tokens are assigned a more discrete BILOU label which helps the model to identify them more concretely.

- The script is able to pad all sequences so that they are not of variable length, which allows the model to make proper predictions.
- Stopwords and nonalphanumeric tokens are removed from consideration when the model is trained, which seems to help accuracy.
- Evaluation of accuracy and loss of the model is fully implemented, and the accuracy of my model's predictions on the provided testing data is reportable.

5 Evaluation

While the script itself cannot output exact F1-scores as per the scorer script provided with the task, the model's accuracy on both validation and testing data is impressive. In order to process the testing dataset, the same process is used as was for the training dataset (stopwords and nonalphanumeric tokens are removed). Following this, my model makes predictions and logs the accuracy of those predictions. This is shown below. For the training data:

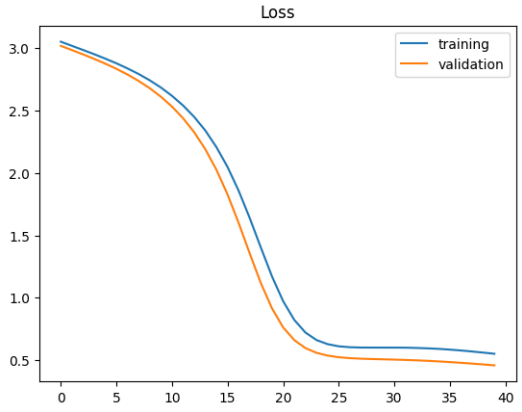

Figure 2: Loss on Validation Data

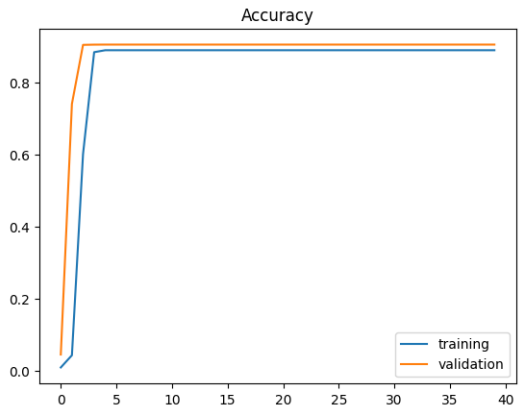

Figure 3: Accuracy for training data

When run on the testing data:

```
test loss: 1.166434645652771
test accuracy: 0.828529417514801
```

As can be observed in the preceding graphs, validation and training loss are relatively comparable, which suggests proper fitting of the data (though this is still to be fully reviewed, as I'm in the process of understanding all the facets that might constitute misrepresentation of data). Additionally, training loss sits at approximately 0.7949 while validation loss sits at approximately 0.6697, with training accuracy at 0.8883 and validation accuracy at 0.9039.

Perhaps the greatest concern about my model is its inability to predict infrequent labels. Categorical cross-entropy accuracy alone clearly doesn't paint a perfect picture of my model, as it cannot be translated directly over to correct predictions. Going forward, I will likely not be considering labels that don't meet a certain frequency threshhold.

Another particular flaw of my model is still that the underlying network does not implement a CRF. Through `Tensorflow-Addons`, I have found a CRF layer that can be added to a Keras model. In the final phase of this project, I will implement this. Additionally, Viterbi decoding has not been employed here, which I feel will help to increase the accuracy of predictions. Finally, I would like to implement some character-level analysis, as well as to continue adjusting my model, adding/removing/adjusting layers and/or adjusting hyperparameters to analyze performance.

Although this system has much to do in order to suffice for the final demonstration, the core model is clearly working, and is able to make accurate probabilistic predictions.

6 Contributions

I am a one-person team, so I have done all the work for this project.