Natural Language Processing

Assignment 1: Part-of Speech (POS) tagging as Sequence Labelling using Recurrent Neural Architectures

Rasha Jabbour, Ramez Nafeh, Jasmine El Afyouni

20/12/2021

Abstract

Part-of-speech (POS) tagging is a process in which a tag is assigned to each word in a text to indicate its category such as noun, verb, adjective, etc. depending on context and meaning. Our aim is to experiment with several architectures including LSTM and GRU layers. The Keras API was used for pre-processing as well as the building and training of the models. In this report we will discuss our findings as well as what we considered to be our best models, after a number of combinations.

1 Task Description

The objective of this assignment was to implement and experiment with several variations of recurrent neural architectures to predict POS tags for each word in a corpus of documents. The first step was to pre-process the data given to obtain the inputs and outputs of the models. After splitting the data into train, validation and test splits, we built the vocabulary of the words and labels using the Keras built-in tokenizer. Words were embedded using pre-trained GloVe embeddings of dimension 200, and OOV words (6%) were given random vectors. Four models (which are discussed more in detail in the next section) were defined, trained and then validated and among them, the two best performing ones on the validation set were chosen to be evaluated on the test data.

2 Models

All models have the same input layer which is a non-trainable Embedding layer and the same output layer which is a TimeDistributed Dense layer with a softmax activation. A mask was also added after the embedding layer to mask the 0 values which are padding. The models were also all compiled with an 'adam' optimizer and a 'sparse categorical crossentropy' loss.

Baseline: The baseline model consists of a Bidirectional LSTM layer and a Dense/Fully-Connected layer on top; we tried multiple hyper-parameters for the LSTM layer but finally decided on using 64 units and a dropout of rate 0.1.

GRU: The GRU model consists of a GRU layer instead of the Bidirectional LSTM layer used in the baseline, while keeping the Dense layer as is. We also saw that 64 units was the most optimal hyper-parameter for the GRU layer.

Two LSTM: For this model, we added an extra Bidirectional LSTM layer to the Baseline model as the second layer. We kept the first Bidirectional LSTM layer as it is but observed that having the second Bidirectional LSTM layer use 32 neurons elicited the best results.

Two Dense: Here we added an extra TimeDistributed Dense layer to the Baseline model above the original Dense layer. We set this Layer's parameter to 128 and used 'relu' activation. We also noticed that lowering the Bidirectional LSTM layer parameter to 32 results in better performance.

3 Results

Models were trained with a batch size of 32 and up to 1000 epochs, as well as Early stopping with a patience of 15 epochs.

The following table shows the Training and Validation accuracy of each model (see Table 1).

Table 1: Train and Validation Accuracy of each model.

Model	Train Accuracy	Validation Accuracy	Execution Time
Baseline	0.9266	0.8006	6 minutes
GRU	0.8769	0.7798	3 minutes
2 LSTM	0.9263	0.8125	10 minutes
2 Dense	0.8989	0.8012	4 minutes

We chose our two best models according to the best validation accuracy, and we can see in Table 1 that the two best models are the models using 2 LSTM layers and 2 Dense layers. However, we also noticed that the Validation Accuracy of the Baseline Model was incredibly close to that of the model using 2 Dense layers, with an execution time much better than the model using 2 LSTM layers. Therefore, even if it was not required, we decided to evaluate the Baseline model as well as the models using 2 LSTM layers and 2 Dense layers.

The following table shows the results of our evaluation (see Table 2).

Table 2: Test Accuracy and F1 Macro score of each of the best models.

Model	Test Accuracy	F1 Macro score
Baseline	0.81	0.73
2 LSTM	0.83	0.69
2 Dense	0.81	0.71

As we can see, the Test Accuracy is always higher than the F1 Macro score. This difference will be explained in the next section.

4 Error Analysis

First recall that Accuracy is measured using the TP, FP, TN, FN of all classes together, while F1 Macro score calculates the F1 score of each class independently then averages them, making it more telling than normal Accuracy. However, if we plot (refer to plots in the last section of the notebook) the frequency of each class, we notice that there is a **huge class imbalance** which is inherent to language (for example, determinants -DT- are more frequent than plural proper nouns -NNPS-). This can negatively affect the learning process of the model since it is not provided the same quantity of each class to learn from. This, along with the proper difficulty of a specific class decide how difficult it is to learn. This might explain the discrepancy between the test accuracy score and macro F1 scores since the individual F1 scores among classes varied hugely (refer to findings in the last section of the notebook). Another more suitable method of evaluation could be to use a weighted F1 score instead of macro, giving a weight to each class proportionally to its frequency of occurrence.