# Report for Assignment 4: “POS Tagger with RNNs”

Baltzi Sofia (P3351915), Kafritsas Nikos(P3351905), Mouselinos Spyridon (P3351914)

June 27, 2020

## Introduction

The purpose of this document is to report the results for the 4th Assignment of class Text Analytics, April – June 2020. Exercise analyzed is “Exercise 2: Develop a Part of Speech Tagger using RNN”. Code for the assignment can be found [here](https://colab.research.google.com/drive/1-cS4fKlbihavB9ifPsIzdvj2HgzV6Wzz?fbclid=IwAR1LQFYxxD6YElcWju2QAkiTGfM5D5jhtLDSgKxVlGZkOcdRSgROaHp8Lnw).

## Data

Data used in this assignment is the “GUM” English treebank found via Universal Dependencies. Data came in the form of Conllu files, from which only word/ sentence tokenization and Part of Speech labels/ classes where used. Training, Development and Test sets where provided, so no splitting was applied. Training set consisted of 4,094 sentences with average size of 19 words. Development set consisted of 751 sentences with average size of 20 words. Test set consisted of 852 sentences with average size of 18 words. Any word found in the three sets would belong to one of the 18 classes presented below along with their numeric representation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0: ADJ | 4: CCONJ | 8: NUM | 12: PUNCT | 16: X |
| 1: ADP | 5: DET | 9: PART | 13: SCONJ | \*17: PADDING |
| 2: ADV | 6: INTJ | 10: PRON | 14: SYM |  |
| 3: AUX | 7: NOUN | 11: PROPN | 15: VERB |  |

Utilizing training set, a vocabulary of length 8,746 words was created, to be used in the next phases. This vocabulary included the tokens '\_\_PADDING\_\_' and '\_\_UNK\_\_' for padded sentences and unknown words respectively.

Also, “Fasttext” pretrained word embeddings where downloaded and used for the representation of the words. Initial form was an array of size 2,000,000 by 300 but was reduced to an array of size 8,746 (as the length of the training vocabulary) by 300.

Minimal preprocessing was applied to the sets, during which all emails where substituted by the character “@” and all numbers by the number “0”, for them not to be treated as different words.

The longest sentence of the train, development and test dataset consisted of 99 words. For this reason, every sentence of the three sets was padded to reach the length of 100 words. As a result, training set (x\_train) was now an array of 4,094 rows and 100 columns, development set (x\_dev) was an array of 751 rows and 100 columns and test set (x\_test) was an array of 852 rows and 100 columns. Accordingly, all y sets had the size of the respective x sets and contained the labels/classes of the words.

Note:

A new class was created, 17: PADDING, that was added after the correct labels in the rows of y arrays that were padded. This class was created only to make y’s have the same size as x’s and was not taken into consideration for tuning, training, or performance evaluation.

## Baseline Classifier

A custom Baseline Classifier was created which that tags each word with the most frequent tag it had in the training data and for words that were not encountered in the training data it tags them with the most frequent class, being 7: Noun. Below are the results of the baseline classifier:

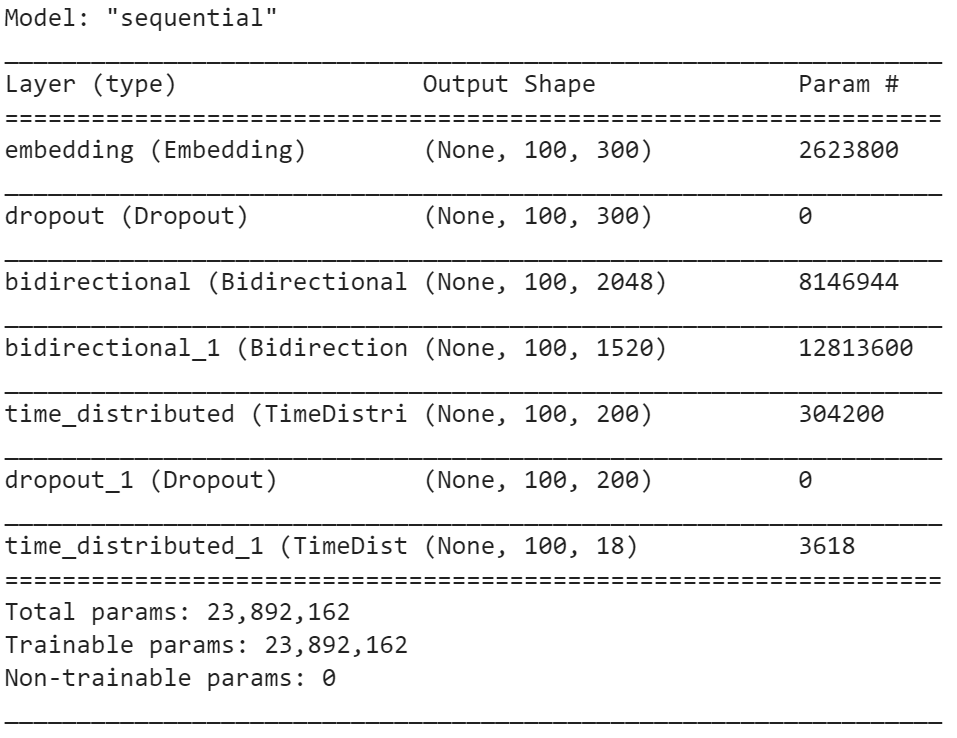
|  |  |  |
| --- | --- | --- |
| Training Accuracy | Testing Accuracy | Macro-average F1 score |
| 91.91% | 83.51% | 75.44% |

In general, the baseline classifier achieves low scores since there are many classes.

## Hyperparameter Tuning, Training

For the purposes of model tuning, the Keras Tuner Package was used. However, since fine tuning was performed based on F1-score that was not supported, a custom Bayesian Tuner optimizing for F1 was created. Hyperparameters tuned were number of bidirectional GRU layers and number of units per GRU layer, number of Dense layers and number of units per Dense layer and dropout rates.

Below are presented the best hyperparameter combinations as found by the tuner:

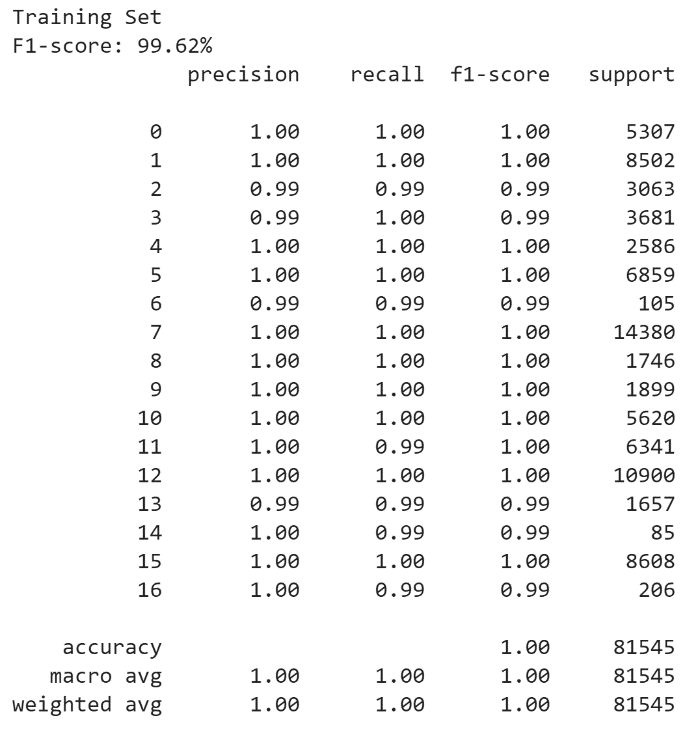
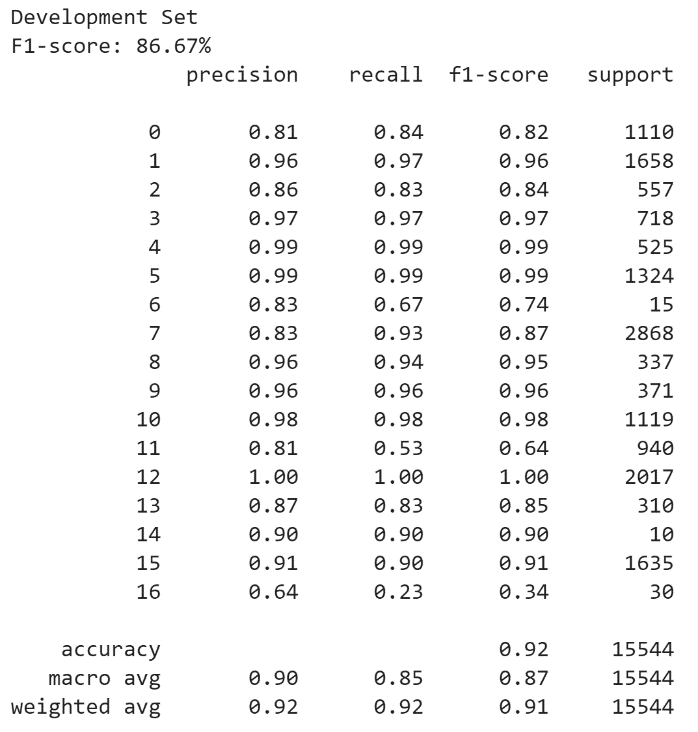
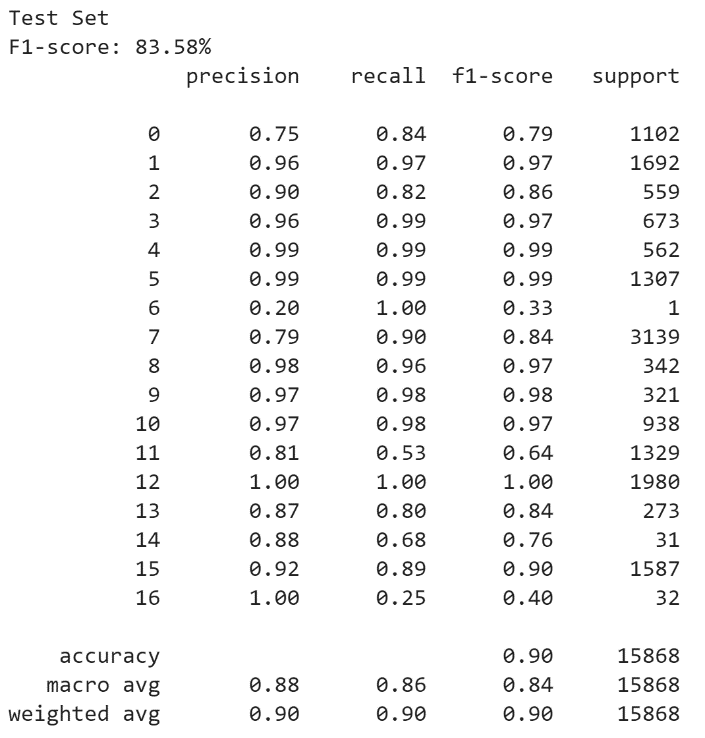


These combinations were then used for training, during which, number of epochs and learning rate were dynamically tuned using Early Stopping and Reduce LR On Plateau callbacks.

## Evaluation

The following table shows the F1 score achieved on Training, Development and Test sets by the RNN classifier, the Baseline classifier of Section 2 and the best MLP classifier of previous Assignment. In terms of macro-averaged F1 score, RNN gives the best results on all sets:

|  |  |  |  |
| --- | --- | --- | --- |
| Window Size | F1 score Training | F1 score Development | F1 score Test |
| Baseline | 89.15% | 81.08% | 75.43% |
| MLP | 96% | 86% | 81% |
| RNN | 100% | 86.67% | 83.58% |

Below are presented classification reports on all sets:

Obviously, there is a fair amount of overfitting since model performs much better on Training set compared to both Development and Test.

The fact that the sets are so imbalanced – can be seen in the support column of the reports – seems to have a significant impact to the learning ability of the algorithm, since classes that appear a few times are proven to be the most difficult to identify (e.g. class 6).

## Bootstrapping

Bootstrap statistical significance test will be used to compare the aforementioned classifier with the baseline classifier and investigate if their difference in terms of macro-averaged F1 score is statistically significant, or due to chance.

The goal of this process is to estimate a p-value (probability of obtaining an equal or better increase of the macro-averaged F1 score on the test set than the observed one, given that the 2 classifiers being compared have equal potential). The testing part involved creating 50 versions of the test set by sampling with replacement. More specifically, the process assumes 2 hypotheses:

* H0: The RNN classifier is not better than the baseline classifier
* H1: The RNN classifier is better than the baseline classifier.

The results are shown below:

|  |  |
| --- | --- |
| Classifier | P-value |
| Baseline vs RNN | 0 |

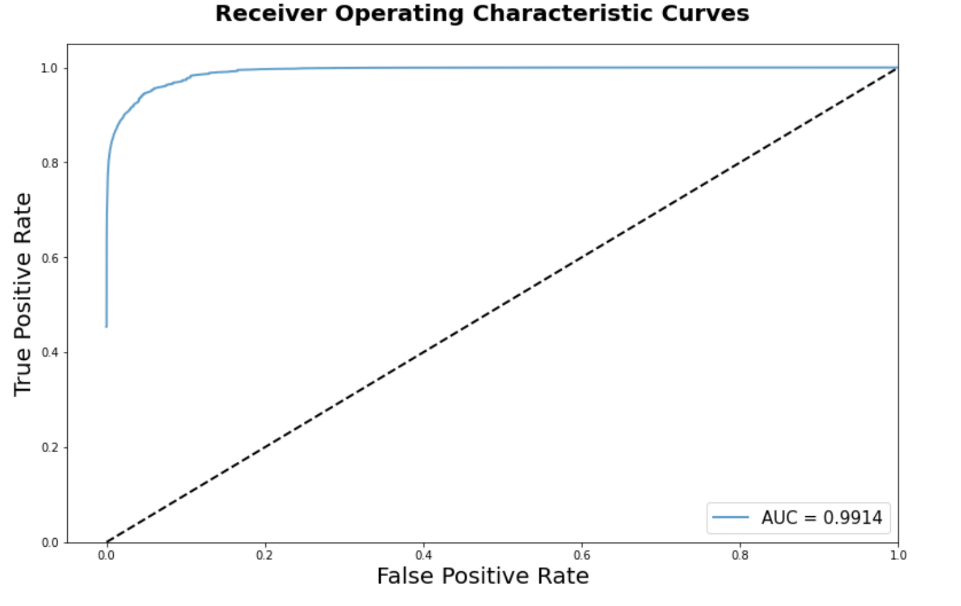
Obviously, the difference between the 2 classifiers is tremendous. The p-value is 0, which in turn means that the null hypothesis is rejected in favor of the alternative. In other words, the RNN classifier is significantly better.

## ROC Curves

Since the problem given is a multi – class problem, traditional ROC curves cannot be applied. Thus, the problem is binarized by implementing a one versus rest technic, where iteratively each class gets the role of the positive class and all the other 16 get the role of the negative class. False and True Positive Rates as well as Area Under Curve for each class are computed and then aggregated to give a macro – averaged perspective of the problem. That macro – averaged perspective is plotted to the diagram below.

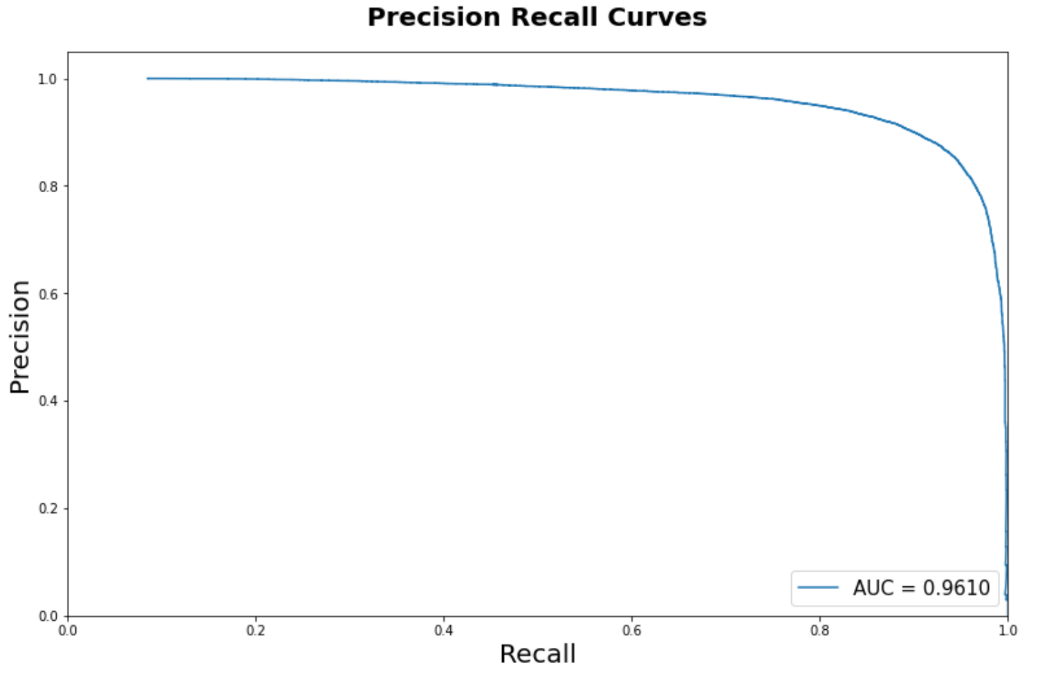
Observe that based on AUC, classification seems be almost perfect, while F1 scores were significantly lower. For this to be explained one should look at the definitions of AUC and F1:

• AUC: fraction of the negatively labeled sample that is correctly labeled (specificity)

• F1: fraction of the positively labeled sample that is correctly labeled

At the one versus rest approach, problem was heavily imbalanced, leaving negative class having much many more examples than positive, since it included examples of 16 classes. When computing AUC, focus was on negative class, rather on positive and even though classification was not perfect among the 16 classes (that constructed the negative) that was not depicted on the computation.

In conclusion, F1 scores are more representative of the algorithm performance.

For this reason, the use of Precision Recall Curves was also explored.

Precision is defined as the number of true positives over the number of true positives plus the number of false positives and Recall is defined as the number of true positives over the number of true positives plus the number of false negatives. Both give more attention to the positive class and thus, numbers seem much more realistic.