Report for Assignment 3: "POS Tagger with MLP"

Baltzi Sofia (P3351915), Kafritsas Nikos(P3351905), Mouselinos Spyridon (P3351914)
May 20, 2020

0. Introduction

The purpose of this document is to report the results for the 3rd Assignment of class Text Analytics, April – June 2020. Exercise analyzed is "Exercise 10: Develop a Part of Speech Tagger using MLP". Code for the assignment can be found here.

1. 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 17 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	
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. Also, as a final preprocessing step, all sets where transformed according to the window size chosen. Attempted window sizes where 1, 3, 5 and 7. According to the window size, "x" sets where transformed to arrays with rows of length equal to the window size and "y" sets included only the label of the word that was in the middle of each training example.

2. 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.

3. MLP: 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 layers and number of units per layer. Tuning was preformed separately for the different window sizes.

Below are presented the best hyperparameter combinations per window size:

Window Size: 1
Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 1, 300)	2623800
flatten_1 (Flatten)	(None, 300)	0
dense_3 (Dense)	(None, 228)	68628
dropout_2 (Dropout)	(None, 228)	0
dense_4 (Dense)	(None, 132)	30228
dropout_3 (Dropout)	(None, 132)	0
dense_5 (Dense)	(None, 17)	2261

Total params: 2,724,917
Trainable params: 2,724,917
Non-trainable params: 0

Window Size: 3
Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 3, 300)	2623800
flatten_1 (Flatten)	(None, 900)	0
dense_3 (Dense)	(None, 132)	118932
dropout_2 (Dropout)	(None, 132)	0
dense_4 (Dense)	(None, 164)	21812
dropout_3 (Dropout)	(None, 164)	0
dense_5 (Dense)	(None, 17)	2805

Total params: 2,767,349
Trainable params: 2,767,349
Non-trainable params: 0

Window Size: 5
Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 5, 300)	2623800
flatten_1 (Flatten)	(None, 1500)	0
dense_3 (Dense)	(None, 228)	342228
dropout_2 (Dropout)	(None, 228)	0
dense_4 (Dense)	(None, 100)	22900
dropout_3 (Dropout)	(None, 100)	0
dense_5 (Dense)	(None, 17)	1717

Total params: 2,990,645
Trainable params: 2,990,645
Non-trainable params: 0

Window Size: 7
Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 7, 300)	2623800
flatten_1 (Flatten)	(None, 2100)	0
dense_3 (Dense)	(None, 196)	411796
dropout_2 (Dropout)	(None, 196)	0
dense_4 (Dense)	(None, 132)	26004
dropout_3 (Dropout)	(None, 132)	0
dense_5 (Dense)	(None, 17)	2261

Total params: 3,063,861
Trainable params: 3,063,861
Non-trainable params: 0

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. As for training, it was also performed for the different window sizes. Finally, best model for every window size was saved, so it can be used for the comparisons presented in the next sections.

4. Evaluation

The following table shows the F1 score achieved on Training, Development and Test sets versus the window size used. Window size 5 gives the best results in terms of Test set:

Window Size	F1 score Training	F1 score Development	F1 score Test
1	91.82%	85.94%	83.32%
3	97.81%	90.84%	88.97%
5	97.92%	90.88%	89.27%
7	99.28%	90.92%	89.12%

Below are presented classification reports on all sets for window of size 5:

Training Set
F1-score: 97.92%

	precision	recall	f1-score	support
0	0.96	0.98	0.97	5307
1	0.97	0.99	0.98	8502
2	0.95	0.95	0.95	3063
3	0.96	0.99	0.98	3681
4	1.00	0.99	1.00	2586
5	1.00	1.00	1.00	6859
6	0.90	0.86	0.88	105
7	0.97	0.99	0.98	14380
8	0.98	0.99	0.99	1746
9	0.99	0.99	0.99	1899
10	0.99	0.99	0.99	5620
11	0.98	0.93	0.96	6341
12	1.00	1.00	1.00	10900
13	0.94	0.88	0.91	1657
14	0.94	0.89	0.92	85
15	0.98	0.97	0.98	8608
16	0.92	0.73	0.81	206
accuracy			0.98	81545
macro avg	0.97	0.95	0.96	81545
weighted avg	0.98	0.98	0.98	81545

Development Set
F1-score: 90.88%

	precision	recall	f1-score	support
0	0.82	0.83	0.82	1110
1	0.95	0.97	0.96	1658
2	0.87	0.80	0.83	557
3	0.95	0.98	0.97	718
4	0.99	1.00	0.99	525
5	0.99	0.99	0.99	1324
6	0.82	0.60	0.69	15
7	0.83	0.92	0.87	2868
8	0.96	0.93	0.95	337
9	0.96	0.97	0.97	371
10	0.97	0.99	0.98	1119
11	0.76	0.55	0.64	940
12	1.00	1.00	1.00	2017
13	0.88	0.73	0.80	310
14	1.00	0.90	0.95	10
15	0.91	0.89	0.90	1635
16	0.45	0.17	0.24	30
accuracy			0.91	15544
macro avg	0.89	0.84	0.86	15544
weighted avg	0.91	0.91	0.91	15544

Test Set
F1-score: 89.27%

	precision	recall	f1-score	support
0	0.76	0.80	0.78	1102
1	0.95	0.98	0.96	1692
2	0.91	0.79	0.84	559
3	0.94	0.99	0.97	673
4	0.99	0.99	0.99	562
5	0.99	0.99	0.99	1307
6	0.00	0.00	0.00	1
7	0.80	0.90	0.85	3139
8	0.98	0.94	0.96	342
9	0.98	0.97	0.97	321
10	0.96	0.98	0.97	938
11	0.78	0.54	0.64	1329
12	1.00	1.00	1.00	1980
13	0.89	0.78	0.83	273
14	0.88	0.68	0.76	31
15	0.90	0.88	0.89	1587
16	0.60	0.19	0.29	32
accuracy			0.90	15868
macro avg	0.84	0.79	0.81	15868
weighted avg	0.90	0.90	0.89	15868

Obviously, there is a fair amount of overfitting since model performs much better on Training set compared to both Development and Test. Also, class 6 which appears only one time in the Test set (and a few times in the other sets) is nor correctly predicted and thus, has zero Precision, Recall and F1 scores.

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.

5. Bootstrapping

Taking all the above into account, after hyperparameter turning, the model with window size 5 was found to have the best macro-averaged F1 score on the test set. Bootstrap statistical significance test will be used to compare the aforementioned MLP 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 estimates 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:

A) H_0: The MLP classifier is not better than the baseline classifier
B) H_A: The MLP classifier is truly better than the baseline classifier.

The results are shown below:

Classifier	P-value
Baseline vs Window Size:5	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 MLP classifier is better.

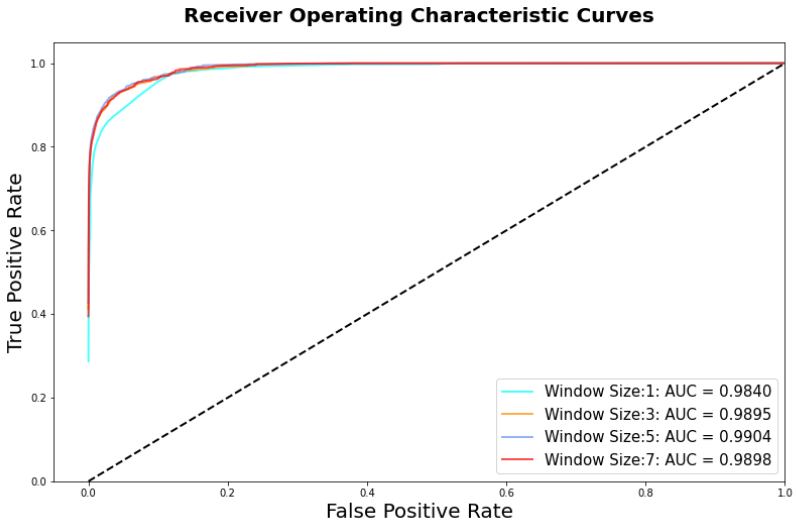
6. ROC/ Precision Recall Curves

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 for every window size.

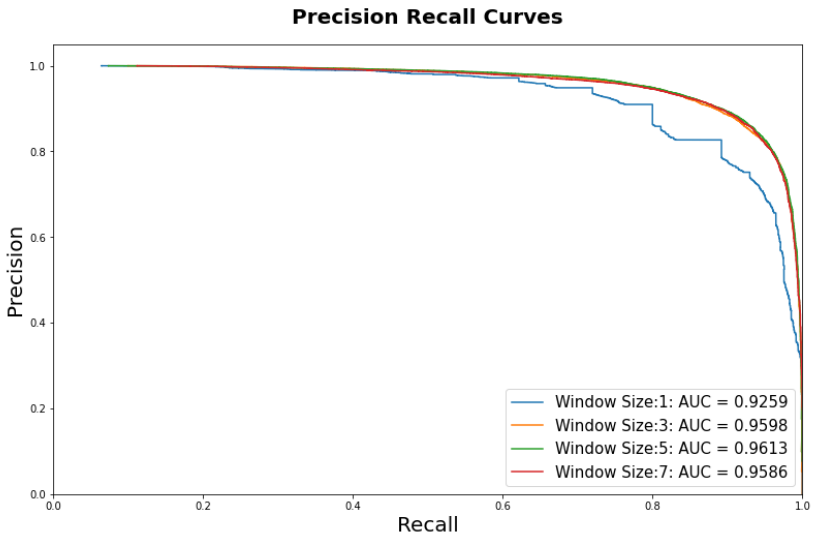
Observe that based on AUCs, classification seems be almost perfect, while F1 scores were about 90%. For this to be explained one should look at the definitions of AUC and F1:

- 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.



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.

Overall, a window of size equal to 5 has the largest Area Under Curve, as it can be observed in the diagram, making the use of it the most appealing.