Natural Language Processing - INF210
Exercise 11 of part 3

Gketsopoulos Petros | $F3321804$
Gkikas Athanasios | $F3321808$
Euagelou Iordanis | $F3321801$
Kougia Vasiliki | $F3321805$

April 21, 2019

1 Dataset

For the implementation of our part-of-speech (POS) tagger we used the EWT corpus for English from the Universal Dependencies treebanks.[1] The corpus comprises 254,854 words and 16,622 sentences, taken from various web media including weblogs, newsgroups, emails, reviews, and Yahoo! answers. We employed the sentences of the dataset that are split to words, each assigned with one of the 18 available POS tags. The initial dataset consists of train, development and test sets. We used the pyconll library to load the conllu dataset files and randomly selected 10% of the provided training set to obtain our validation set.[2]. The resulted sets and the number of sentences they contain are shown in the table below. We trained our MLP on the train set and used the validation set for hyperparameter tuning, the development set to monitor the performance of the model during training and the test set to evaluate the results of our trained model.

	Sentences
Train set	11289
Validation set	1254
Development set	2002
Test set	2077

2 Architecture

POS tagging is a multiclass classification problem, so we address it by implementing an MLP that classifies each word of a sentence into one of the 18 classes, operating on windows of words. In the following subsections we describe the pre-trained word embeddings we used and how we handled any unknown words, the baselines we implemented, as well as the architecture of our MLP with the best hyperparameter values as they emerged from tuning.

2.1 Embeddings

For the training of our models, we embedded the pre-trained *Google news* feature vectors.[3] We used the gensim library to load the 300-dimensional word2vec embeddings and performed a lookup for every word in our corpus. Each word was replaced with its corresponding embedding, while the words

[1] `https://github.com/UniversalDependencies/UD_English-EWT`
[2] `https://pyconll.github.io/`
[3] `https://code.google.com/archive/p/word2vec/`

that were not present in the pre-trained embeddings' vocabulary were replaced with the default *UNK* embedding. We briefly summarize the per subset word distribution of our classes in the following table.

Train set										
ADJ	ADP	ADV	AUX	CCONJ	DET	INTJ	NOUN	NUM		**total sum**
11137	11243	9433	10846	1672	11438	596	30801	1439		
PART	PRON	PROPN	PUNCT	SCONJ	SYM	UNK	VERB	X		
1527	16604	11390	260	3287	362	41455	20556	439		184485
Test set										
ADJ	ADP	ADV	AUX	CCONJ	DET	INTJ	NOUN	NUM		**total sum**
1672	1465	1214	1449	206	1418	117	4020	156		
PART	PRON	PROPN	PUNCT	SCONJ	SYM	UNK	VERB	X		
201	2158	1973	69	363	61	5873	2638	44		25097
Development set										
ADJ	ADP	ADV	AUX	CCONJ	DET	INTJ	NOUN	NUM		**total sum**
1776	1462	1253	1465	238	1417	114	4079	137		
PART	PRON	PROPN	PUNCT	SCONJ	SYM	UNK	VERB	X		
210	2216	1785	35	374	44	5742	2743	60		25150
Validation set										
ADJ	ADP	ADV	AUX	CCONJ	DET	INTJ	NOUN	NUM		**total sum**
1209	1161	1051	1210	190	1252	79	3266	134		
PART	PRON	PROPN	PUNCT	SCONJ	SYM	UNK	VERB	X		
192	1958	1182	25	362	33	4461	2327	30		20122

2.2 Baselines

We implemented two basic baselines that perform POS tagging in order to conduct a meaningful comparison with our neural network architecture and assess its performance. First, we implemented a *most-frequent* classifier, which simply classifies each word of the test set to the most frequent POS tag it had in the train set. If a word is not present in the train set, it is assigned with the "UNK" tag. For the second baseline we used the *logistic regression* classifier of the scikit-learn library for multinomial classification with the Limited-memory BFGS solver. The model is trained on our train set operating on a window of 5 words, like our MLP. The input for each word of the sentence that is being classified is the concatenation of the word embeddings of the window, so it receives context from the two preceding and the two succeeding words.

2.3 Neural network structure

We used Keras to implement a window-based MLP model for POS tagging with a 5-word window. The main architecture of our model consists of 6 blocks. Each block is constructed with a *fully connected* layer, an *activation function* and a *drop out* layer. Finally, a fully connected layer with neurons equal to the number of our classes and a *softmax* activation is appended in order to extract the per class probability. The MLP is fed with the concatenated embeddings of words in the window and the word in the middle is classified into the class with the highest output probability.

We decided to tune our model using the *Hyperas*[4] API for 50 evaluation trials against the type of the global activation function in our network - Relu or LeakyRelu - and per block drop out probability, that is, 0.0 (no drop out), 0.2, 0.5. For efficiency and performance reasons the rest of the parameters were empirically chosen. We imbued the *Hyperas* optimizer with the validation subset in which we sliced a test subject prior to the initialization in order to monitor the accuracy during tuning. Our model architecture along with the selected tuned parameters are demonstrated in the following table.

	Dense size	**Activation**	**Dropout**
Input	5 * 300	-	-
Layer 1	512	Leaky Relu	0.5
Layer 2	256	Leaky Relu	0.5
Layer 3	128	Leaky Relu	0.2
Layer 4	128	Leaky Relu	0.2
Layer 5	256	Leaky Relu	0.5
Layer 6	512	Leaky Relu	0.0
Output	18	Softmax	-

3 Evaluation

3.1 Evaluation

In the following subsections, there is an in-depth analysis of our experimental results using the MLP approach which we then compare against the most frequent classifier and the logistic regression. Every model was trained with the train subset and tested against the test subset (described in section 1). Finally, every evaluation was done with the same embedding features that we described in an earlier section. For the purposes of this work and in order to be consistent with the project goals, we decided not to experiment with different window sizes nor embedding features.

3.1.1 MLP results

We have chosen the default *Adam* optimizer for both training and tuning, the development subset as the validation input, categorical crossentropy as the loss function, 128 batch size along with the *Keras* early stopping callback. The aforementioned callback was set to monitor validation loss with a patience of 2 epochs. Additionally, we take account for the unbalanced tag distribution by weighting our loss function. The formula is equal to the ratio between the median of the frequencies for each class and the frequency for each class separately [1] . The assigned weight for each class is reported in the following table and the behavior of our network during training is illustrated in Figure 1.

	% Representation	Weight			% Representation	Weight
ADJ	6.0	0.91		PART	0.83	6.6
ADP	6.1	0.9		PRON	9.0	0.61
ADV	5.1	1.1		PROPN	6.2	0.89
AUX	5.9	0.93		PUNCT	0.14	39
CCONJ	0.91	6.1		SCONJ	1.8	3.1
DET	6.2	0.89		SYM	0.2	28
INTJ	0.32	17		UNK	22	0.24
NOUN	17	0.33		VERB	11	0.49
NUM	0.78	7.0		X	0.24	23

[4] `https://github.com/maxpumperla/hyperas`

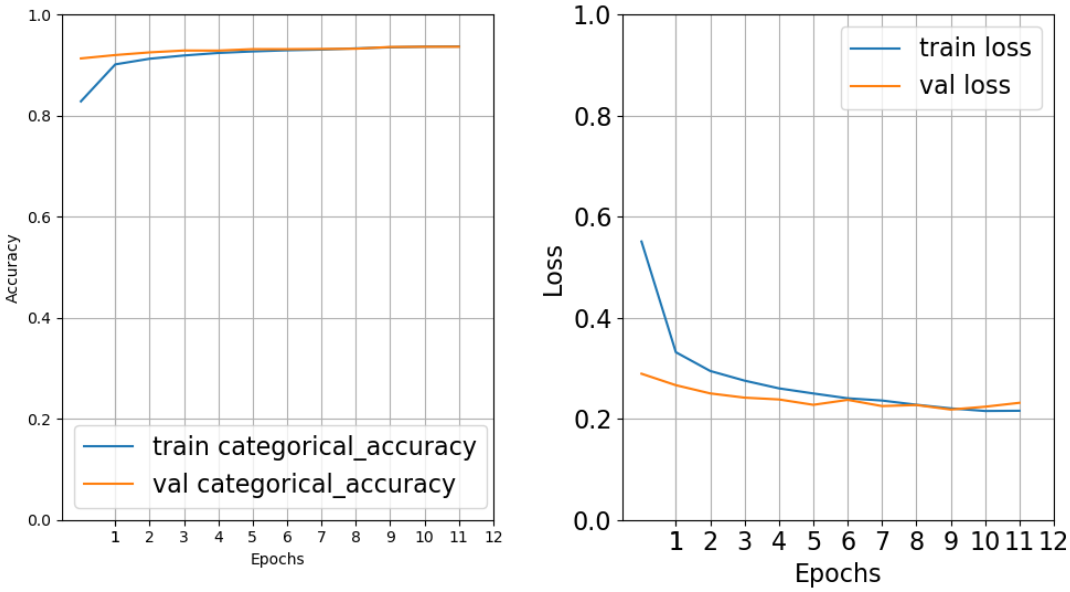

Figure 1: Per epoch progress of validation and train subsets. In the left figure, accuracy is monitored, while in the right one, the loss of the network.

Additionally, for each epoch we compute the macro F1 score, in the validation subset to measure the impact of the *UNK* token as illustrated in Fig. 2. Finally, we visualize the per class F1 scores in Fig. 3.

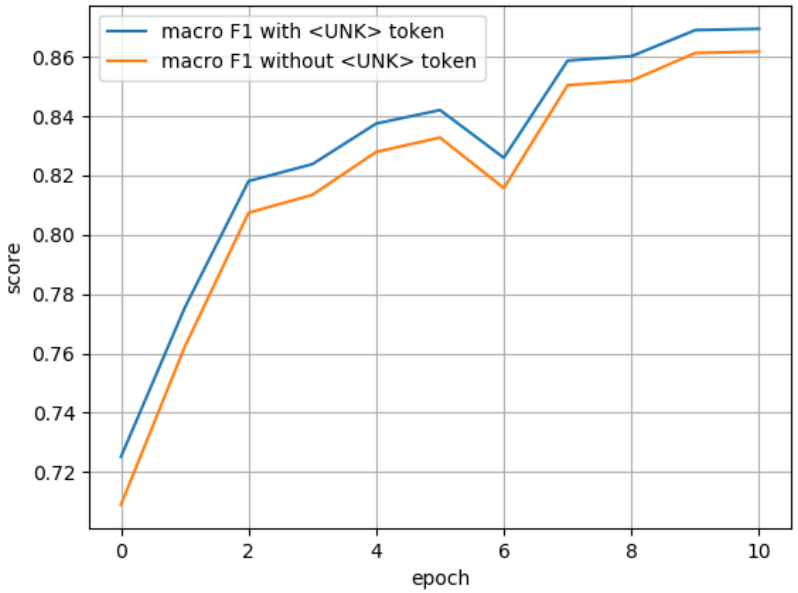

Figure 2: Global macro F1 score during training.

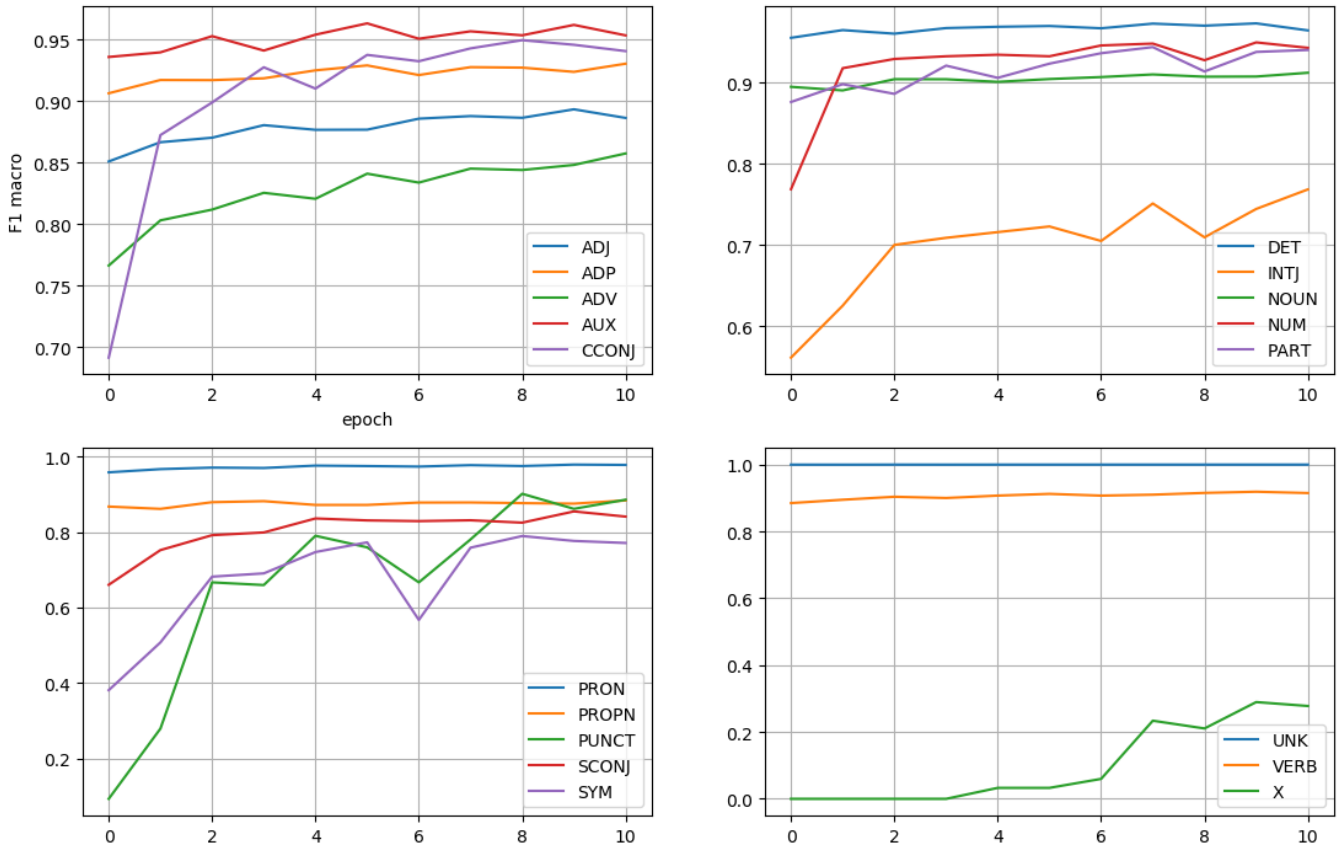

Figure 3: Per class F1 score during training.

3.1.2 Comparative results

In the following table, we analyze and compare the performance of our classifier against the base one and the logistic regressor. In order to assess our architecture we analytically present metrics individually for each class as well as average measurements per metric category.

	Most frequent classifier				Logistic regression				MLP			
	Precision	Recall	f1-score		Precision	Recall	f1-score		Precision	Recall	f1-score	Support
ADJ	0.90	0.81	0.85		**0.88**	0.87	0.87		0.87	**0.90**	**0.89**	1672
ADP	0.87	0.88	0.87		0.91	0.95	0.93		**0.91**	**0.96**	**0.93**	1465
ADV	**0.93**	0.77	0.84		0.87	**0.87**	0.87		0.91	0.83	**0.87**	1214
AUX	0.90	0.89	0.90		**0.95**	0.98	0.96		0.94	**0.99**	**0.96**	1449
CCONJ	**0.99**	**0.99**	**0.99**		0.93	0.96	0.94		0.93	0.96	0.95	206
DET	0.96	0.97	0.96		**0.98**	**0.99**	**0.99**		0.98	0.98	0.98	1418
INTJ	**0.99**	0.69	0.81		**0.95**	0.79	**0.87**		**0.91**	0.82	0.86	117
NOUN	**0.92**	0.76	0.83		0.90	0.91	0.90		0.90	**0.91**	**0.91**	4020
NUM	0.90	0.59	0.71		**0.90**	0.86	**0.88**		0.84	**0.88**	0.86	156
PART	0.66	**0.99**	0.80		**0.93**	0.94	**0.94**		0.92	0.94	0.93	201
PRON	0.97	0.93	0.95		0.98	0.98	0.98		**0.98**	**0.98**	**0.98**	2158
PROPN	0.89	0.53	0.67		0.88	**0.85**	0.86		**0.91**	0.83	**0.87**	1973
PUNCT	**0.99**	**0.99**	**0.99**		0.98	0.93	0.96		0.93	0.97	0.95	69
SCONJ	0.62	0.60	0.61		0.88	**0.80**	0.84		**0.89**	0.82	**0.85**	363
SYM	**0.99**	0.72	0.83		0.90	**0.92**	0.91		0.93	0.90	**0.92**	61
UNK	0.00	0.00	0.00		1.00	1.00	1.00		**1.00**	**1.00**	**1.00**	5873
VERB	0.88	0.81	0.84		0.92	0.92	0.92		**0.92**	**0.93**	**0.93**	2638
X	**0.56**	0.04	0.07		0.17	**0.16**	**0.17**		0.22	0.09	0.13	44
micro avg	0.83	0.83	0.83		0.94	0.94	0.94		**0.94**	**0.94**	**0.94**	
macro avg	0.83	0.72	0.75		**0.89**	0.87	0.88		0.88	**0.87**	**0.88**	
weighted avg	0.91	0.83	0.86		0.94	0.94	0.94		**0.94**	**0.94**	**0.94**	
accuracy	0.82				0.93				**0.93**			

3.2 Conclusions

In this report, we described a complete pipeline of an MLP based part-of-speech tagger. Furthermore, we compared our model's performance against a dummy classifier that uses the most frequent tags and a logistic regressor. We believe that the accuracy of the most frequent classifier is reasonably high, since the corpus lacks examples that diverge semantically. This particular property, enhances one to one relations between words and tags and minimizes the effort needed to learn context aware embeddings. In contrast, logistic regression combined with pre-trained feature vectors, achieved a higher performance and was able to learn a deeper and more meaningful representation. It was able to resolve several ambiguous examples and it becomes apparent that a window-based feature is a reasonable approach to the problem. Finally, the MLP model, with a more complex structure has marginal gains over the prior classifier. Apart from the vast amount of words that were unknown to the feature extractor, which we believe that had minor impact, the window-based approach might hurt the efficiency of our model since there exist classes (e.g PUNCT, CCONJ) that require less context to be estimated.

As shown in the results table, tags that have a less ambiguous content like punctuation (PUNCT) have higher results in all three taggers than classes that might be more difficult to decide for, like adjectives (ADJ). In particular the highest results for PUNCT are achieved by the most frequent classifier, as punctuation marks have the same semantic meaning in most cases and the context aware embeddings do not offer any improvement to their classification. To test how our models perform

in ambiguous cases we predicted the tags for two sentences with different semantic meaning of the word “love”. The ground truths for the two sentences are:

- I love you : [’X’ ’VERB’ ’PRON’]
- Let’s make love : [’VERB’ ’VERB’ ’NOUN’]

The tags predicted from the MLP for these sentences are:

- I love you : [’PRON’ ’VERB’ ’PRON’]
- Let’s make love : [’INTJ’ ’VERB’ ’VERB’]

As we see the MLP failed to recognize the different meaning of the word “love” in the second sentence and falsely classified it as a VERB.

References

[1] D. Eigen and R. Fergus, “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,” in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV ’15, (Washington, DC, USA), pp. 2650–2658, IEEE Computer Society, 2015.