Machine Learning Project

Professor: Guillaume Wisniewski[1]

April 15, 2019

Maria Camila Remolina-Gutiérrez[2]

maria.remolina_gutierrez@telecom-sudparis.eu

Juan Manuel Iglesias[2]

juan.iglesias@telecom-sudparis.eu

[1] LIMSI-CNRS (Orsay)

[2] Télécom SudParis (Évry)

1 Data

We have 6 different datasets from the Universal Dependencies project [1] from which 4 are suitable for training a PoS tagger. We will label and describe them below:

NUMBER	NAME	CONTENT	TRAIN	DEV	TEST
1	EWT	Over 250,000 words of English weblogs, newsgroups, email, reviews and question-answers	X	X	X
2	GUM	Web texts including interviews, news stories, travel guides, how-to guides, academic writing, biographies, fiction and forum discussions.	X	X	X
3	LinES	Covers literature, an online manual and Europarl data.	X	X	X
4	ParTUT	Variety of text genres, including talks, legal texts and Wikipedia articles, among others.	X	X	X
5	pud	1000 sentences in different language, always in the same order. Taken from the news domain and from Wikipedia			X
6	foot	Tweets about football			X

We want to clarify that there was an additional test set available in the corpus. Its name is natdis and refers to data from natural disaster announcements. However, by a mistake in the website, the final is incorrectly named, and it is actually the same foot

dataset. So we did not take it into account as it was just a replication of foot.

1.1 Question 1

Describing shortly the different data sets we will consider : where do the data come from? what is the size (number of words or sentences) of the different of the train and test sets?

As we have already describe the origin of the content of the different data sets considered, we will now describe the size of each one:

- EWT train set has 12,543 sentences with a total of 204,585 words, 18,992 different ones.
- EWT test set has 2,077 sentences with a total of 25,096 words, 5,422 different ones.
- GUM train set has 2,914 sentences with a total of 53,686 words, 8,746 different ones.
- GUM test set has 778 sentences with a total of 13,326 words, 3,578 different ones.
- LINES train set has 2,738 sentences with a total of 50,091 words, 3,516 different ones.
- LINES test set has 914 sentences with a total of 15,623 words, 3,516 different ones.
- PARTUT train set has 1,781 sentences with a total of 43,491 words, 6,739 different ones.
- PARTUT test set has 153 sentences with a total of 3,404 words, 1,112 different ones.

- FOOT test set has 601 sentences with a total of 11,399 words, 2,304 different ones.
- PUD test has 1,000 sentences with a total of 21,176 words, 5,548 different ones.

Now, to characterize the differences between "in-domain" corpora and "out-domain" corpora, we consider 3 measures of the noisiness of a corpus. Their detailed description goes as follows:

1.1.1 Percentage of OOV words

The percentage of Out-of-Vocabulary (OOV) words, is the ratio of words appearing in the test set that are not contained on the train set.

1.1.2 KL divergence

The KL divergence of 3-grams characters distributions [2], estimated on the train and test sets, is defined as:

$$\mathrm{KL}\left(c_{\text{ test}} \| c_{\text{ train}}\right)=\sum_{n \in \mathcal{N}} p_{\text{ test}}(n) \cdot \log \frac{p_{\text{ test}}(n)}{p_{\text{ train}}(n)} \tag{1}$$

where the sum runs over $\mathcal{N}$ the set of all the 3-gram of characters in the train and test sets, and $p_d\left(c_{i-2}, c_{i-1}, c_i\right)=\frac{\#\{c_{i-2}, c_{i-1}, c_i\}+1}{\#\mathcal{N}+\#\mathcal{V} \cdot(\#d-2)}$ is the probability to observe the 3-gram $c_{i-2} c_{i-1} c_i$ in data set d with Laplace-smoothing; $\#\mathcal{V}$ is the number of distinct 3-grams of characters in the train and in the test sets and d is the number of characters in the corpus d (and, consequently $d-2$ is the total number of 3-grams in the corpus).

1.1.3 Perplexity

The perplexity on the test set of a (word level) Language Model estimated on the test set. The language model can be estimated by KenLM (this tools can also be used to compute the perplexity).

1.2 Question 2

Give an intuitive explanation of each of these metrics: what do we try to measure? why? what does it mean when a given metric has a low (or high) value? Compute the value of the different metric for the different combination of train and test sets. What can you conclude?

1.2.1 Percentage of OOV words

The percentage of OOV words tries to measure the similarities between the Train and Test sets. A high percentage means that most of the words of the Train set also appears in the Test set. On the other way, a low percentage means that most of the words in the first set do not appear in the second one. A good test set implies a high degree of OOV words, because it is important to analyze the model against different words and see which is the result in these cases.

In order to calculate this percentage we obtained the words of the test set and the words of the train set. Then, we obtained the difference between the both. That is, the words that were in the first but not in the second. Finally we compared the number of words against the total number of words to get the percentage. We did this for the different possible combinations between the train sets and the test sets to reflect how much this metric was affected. The results (rounded to two decimal places for more

readability) were:

Test sets \Train sets	EWT	GUM	LINES	PARTUT
EWT	31.43%	55.29%	62.21%	63.08%
GUM	31.25%	42.90%	53.83%	54.61%
LINES	30.49%	46.42%	37.94%	53.44%
PARTUT	20.05%	36.33%	37.05%	20.86%
PUD	34.10%	50.86%	58.90%	56.02%
FOOT	41.58%	55.77%	60.15%	62.98%

We can see better represented the information of the table in Figure 1.

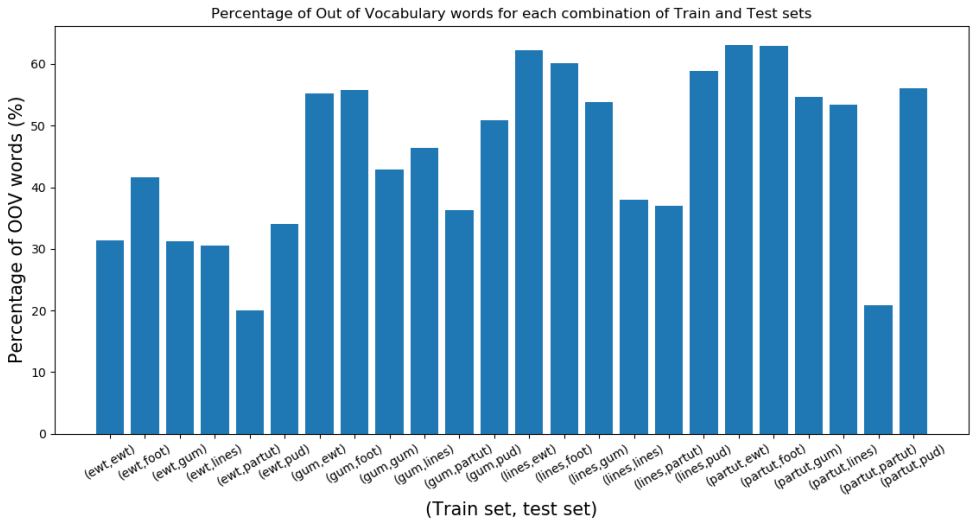

Figure 1: Percentage of OOV words for each combination of train and test sets

As we can see, there is a low percentage of OOV words for the combination of train and test set with the same type of information, which is obvious. For example, for (ewt, ewt), (gum, gum), (lines, lines) and (partut, partut). But also we could also found a low percentage between the ParTUT test set and all the train sets. The reason of this is that the ParTUT data set contains only 153 sentences and all the train sets

are longer. A cause of this, it is much probable that the words appear in these. Also, these sentences were taken from a variety of texts from legal ones to talks transcriptions.

1.2.2 KL divergence

In mathematical statistics, the Kullback-Leibler divergence (also called relative entropy) is a measure of how one probability distribution is different from a second, reference probability distribution. In case of a Kullback-Leibler divergence of 0, that would indicate that the two distributions are identical. In simplified terms, it is a measure of surprise.

Test sets \Train sets	**EWT**	**GUM**	**LINES**	**PARTUT**
EWT	5.60e-5	5.38e-5	5.59e-5	5.78e-5
GUM	8.51e-5	6.43e-5	6.94e-5	7.22e-5
LINES	8.98e-5	7.08e-5	6.19e-5	7.92e-5
PARTUT	39.25e-5	32.18e-5	31.72e-5	30.20e-5
PUD	4.51e-5	3.22e-5	3.48e-5	3.21e-5
FOOT	25.16e-5	22.82e-5	23.35e-5	24.90e-5

As can be seen, low values for the KL-Divergence are found when the test set and the train set are based on the same type of information. In addition, can be see that the test set PUD produces a low value for the KL-Divergence to all the Train sets.

1.2.3 Perplexity

Since a language model is a probability distribution over complete sentences, perplexity per word in a language model is a way to evaluate this distribution.
For example, if the average sentence can be coded with 50 bits this would produce a perplexity of 2^{50} per sentence. It is also possible to normalize for sentence length and

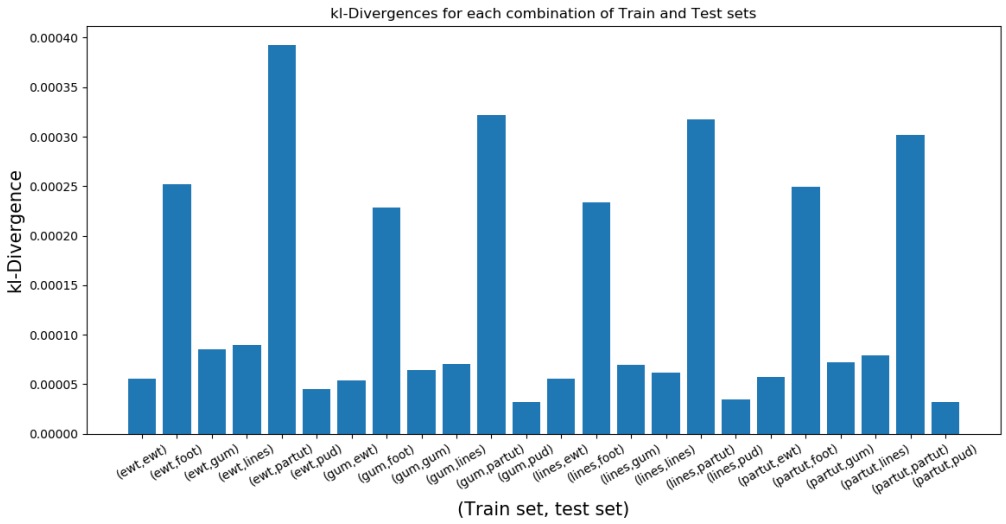

Figure 2: KL-Divergence for each combination of Train and Test sets

only consider the bit per word. In that case if we could code each word in 10 bits for example. The perplexity per word would be $2^{10} = 1024$. This means that for each word, the model has to choose between 1024 possibilities.

For the calculation of the Language Model and the calculation of the Perplexity we used KenLM library, following this steps:

- A text was generated for each of the test sets.
- With the installed program the arpa model for each of the test sets was generated,
- Using the arpa models obtained in the previous step and the text of the first step we intended to calculate the perplexity of each of the models.

2 Model

For the PoS tagger, the selected set of features for each word's feature vector are taken from the suggested by [2]. Explicitly they are:

- the word
- the 2 surrounding words (position $i-2$, $i-1$, $i+1$ and $i+2$)
- the 3 last characters of the word
- the last character of the word
- the first character of the word
- a binary feature indicating whether the word starts with a capital letter or not
- a binary feature indicating whether the word is made only of capital letters or not

2.1 Question 3

Explain (intuitively) how is each features relevant to predict the PoS of word. Implement and evaluate the different PoS taggers on the different combination of train and test sets.

2.1.1 Relevance of features

Let's analyze each feature's relevance in the word's feature vector, one by one:

The word

This is a obvious feature, and a very important one. We need to at least classify the word by itself. It is relevant because is the purest information we can obtain from a word. However by this stage, it lacks the contexts and allows ambiguities.

The surrounding words

The 2 previous and 2 following words are key features because they establish the word context within a phrase. They are the ones that allow the PoS tagger to know the effect of the word in its vicinity. One key example is: "The hunt was successful" VS "We hunt today". Here, the word "hunt" acts as a noun and a verb respectively. If we took only the word we wouldn't be able to distinguish between these 2 cases. Thus the relevance of the surroundings.

The 3 last characters of the word

The last 3 character of a word are relevant in the test cases we analyze because in latin/romance languages the verbs are conjugated. They usually have a common root and regarding of their ending they correspond to a different subject. They could indicate as well if the word is plural or singular, suggesting their part of speech, because just some type of words can be pluralized.

The last character of the word

The last character of the word is similar to the last 3, as it can indicate pluralization. It can also help distinguishing between actual words and punctuation or numbers.

The first character of the word

The first character has an analogous effect to the last character, except in the pluralization.

The capitalization of the first character of the word

This is a relevant binary feature because generally proper nouns have the first letter capitalised.

The capitalization of the word

This is similar to the first letter capitalized. If all the word is in capitals is either a noun or an abbreviation.

2.1.2 Implementation of the PoS tagger

For the PoS tagger we chose an averaged perceptron multi-class classifier. The training is carried out with the train dataset for a maximum number of 20 epochs. Then, we use the dev dataset to select the epoch with lowest dev error rate, and those are the weights associated with the perceptron.

Our approach for the training of the model is to train individually for the datasets 1-4, and then one more training with all 4 of them at the same time. Then we test on all the 6 sets and get their error rates. So it goes as:

EXPERIMENT #	TRAIN DATA	DEV DATA	TEST DATA
1	1	1	1, 2, 3, 4, 5, 6
2	2	2	1, 2, 3, 4, 5, 6
3	3	3	1, 2, 3, 4, 5, 6
4	4	4	1, 2, 3, 4, 5, 6
5	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4, 5, 6

Finally, all the code concerning the multiclass classifier, the post-processing and the plotting is implemented in Python. We send it attached to this document.

2.1.3 Evaluation of the PoS tagger

We evaluated each experiment's PoS tagger with the following quantities:

- test set error rate
- precision over the whole test set (i.e. the percentage of labels that have been correctly predicted)
- precision over the ambiguous words (i.e. the precision computed only on words that appear with more than one label in the train set)
- precision over OOV

So let's analyze each quantity one by one:

Test set error rate

After the perceptron's weights are selected, we perform a test stage with the error dataset for the same experiment. The table below shows the test set error rate for each experiment:

EXPERIMENT #	**TEST ERROR RATE**
1	8.73%
2	12.85%
3	8.17%
4	7.12%
5	1: 8.26% 2: 10.84% 3: 7.80% 4: 8.17%

So all errors fall in the same order of magnitude and in a close range, except for test case 2. This corresponds to the GUM dataset that is particularly the most varied of all, regarding types of information. Also, from the analysis in the Data section, is the most irregular dataset, so it makes sense that the variations within the same set influence in a negative way the error rates, even within itself.

Precision over the whole test set

For each experiment we plot in Fig. 3 the error rate of each dataset (1-6). In color is the experiment number and each x value corresponds to a different test set.

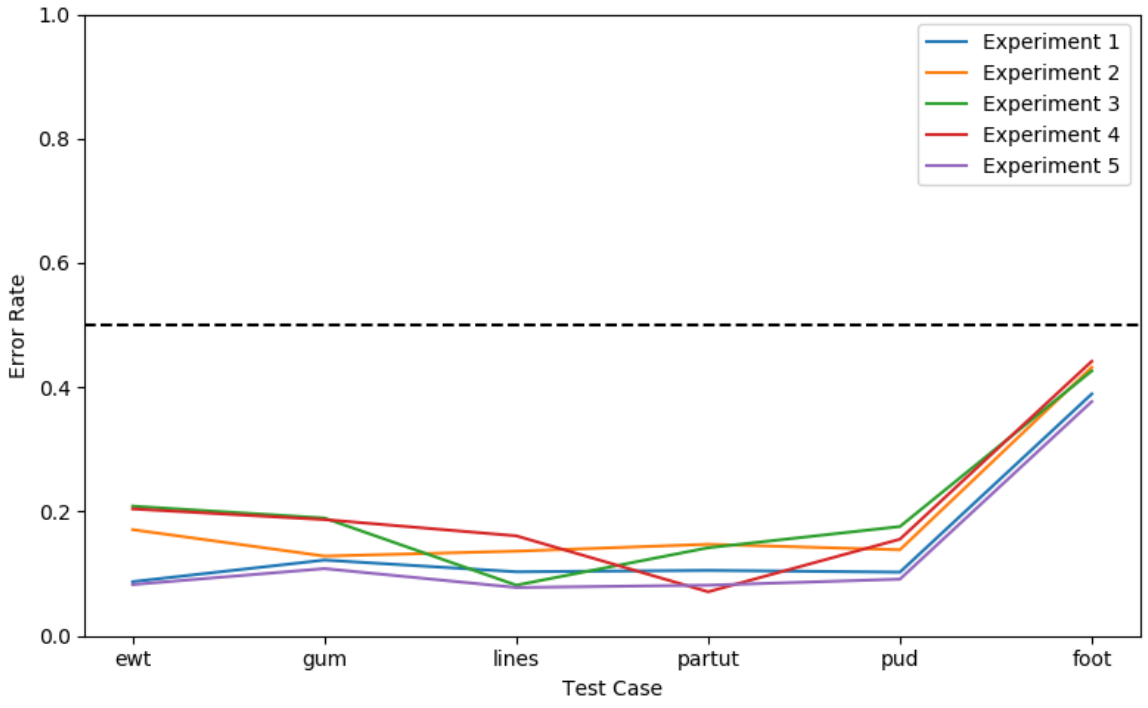

Figure 3: Precision over the whole test set

As expected, the experiment 5 performs the best in all test cases. This proves that the stronger the training corpus, the better the prediction. It's also very noticeable

that the football dataset is the worst performing in any PoS tagger. This, because it is data that hugely differs from the training one. However, with the experiment 5 we were able to obtain a error rate of 36.87%, which even though is very bad performance, it's better than random classification.

Precision over the ambiguous words

For each experiment we plot in Fig. 4 the precision of each dataset (1-6) over the words that appear with more than one label in the train set. In color is the experiment number and each x value corresponds to a different test set.

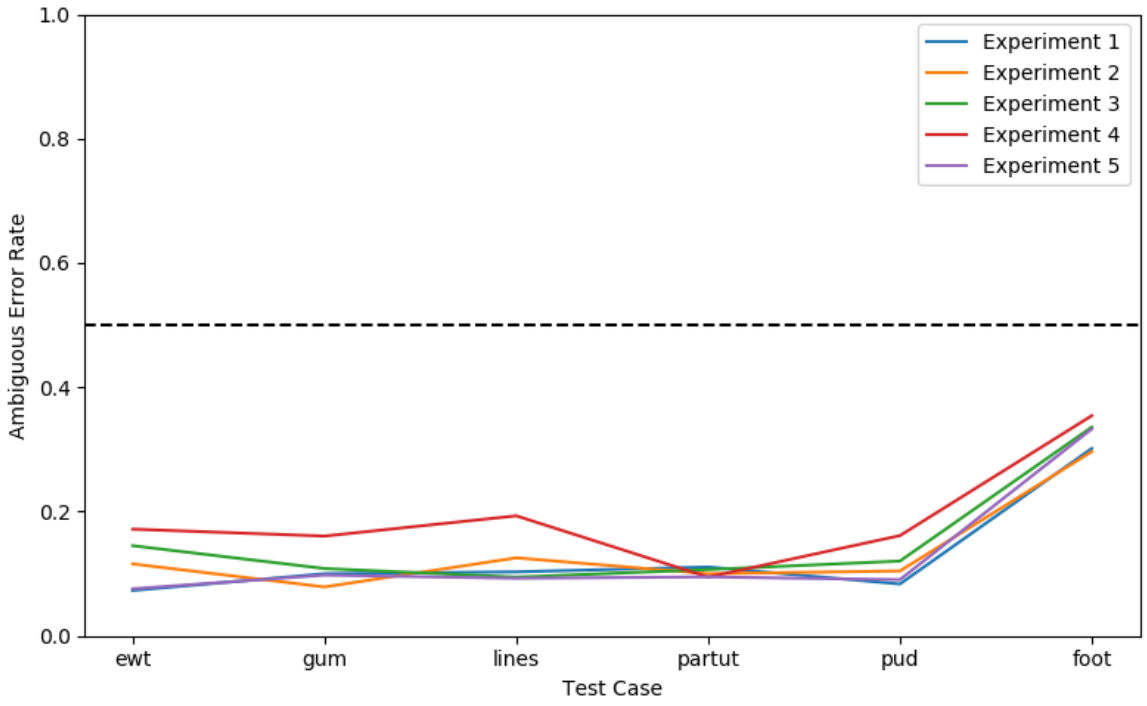

Figure 4: Precision over the ambiguous words

In this case, it is remarkable that the error rate decreases as compared to the whole test set cases. This is a proof that the features selected do a good job in distinguish-

ing the word label by its context, being able to fight the ambiguity of the word by itself.

Precision over OOV

For each experiment we plot in Fig. 5 the precision of each dataset (1-6) over the words appearing in the test set that are not contained on the train set. In color is the experiment number and each x value corresponds to a different test set.

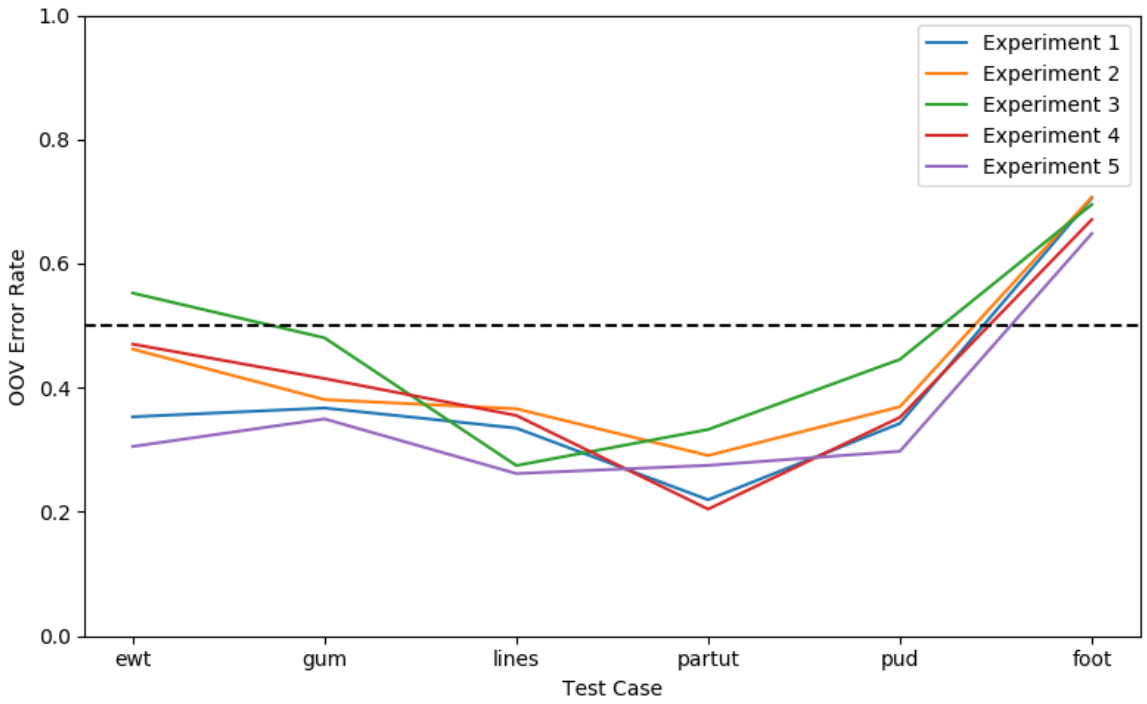

Figure 5: Precision over OOV

In this case it is very clear that for all the test cases the performance hugely decreases. In particular and surprisingly for the EWT dataset, the error rates go beyond 50%, so the perceptron behave worst than the random prediction for words that do not belong to the vocabulary. This shows that the EWT dataset is very closed. Also, and again, in the case of the football dataset, as it has the highest percentage of OOV

words, the prediction becomes useless, no matter how good the features selected are.

3 Discussion

From the work showed above, we can conclude the following concise statements:

- PoS tagging for new types of data (like tweets, texts or natural language) can be very expensive. This implies that training new classifiers might not be plausible. And even if it is, we would have to pay for the taggin job each time a new data type comes, and there are hundreds of them. For this reason, it is very relevant to find features in our available datasets that can distinguish PoS. There is a limit of course. Specially for OOV words, from where we obtain the worst performance. However in this work, we can see an acceptable approach. Even for the tweets dataset we were able to find error rates in the 30%.

- The best results were obtained with the fortified training dataset that include all the corpus. This, though logical, represents also a good alternative for PoS tagging for alternative data types. If we can heavily train the classifier with all available data it will give better weights to the same features.

- In the case of ambiguous words, the selected features make a very good job. In particular, they perform with an error rate even better that for the general test cases. The context definitely plays a key role in PoS classification.

- As a final remark, we want to indicate that the very high percentage of OOV in the train datasets (1-3) is very worrying. A key fact that we base our training in is that the train, dev and test are very similar to each other. However this feature

is higher that what we expected. In dataset 4 (ParTUT) the value is much more acceptable, but in the first 3 it could suggest that the training will not be as good and that could affect our following results. However this is just an intuitive guess. We require a further research in other corpus and in other literature articles in order to know if the values of OOV are normal or out of the ordinary.

3.1 Future work

For the future work we think it would be interesting to find more varied datasets to train and to test with. Specially to test. We would like to train as well with much more non-conventional features. However this could lead to an overfitting from our part. That effect should be quantified.

References

[1] J. Nivre. Universal Dependencies 2.3. *LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÃFAL), Faculty of Mathematics and Physics, Charles University*, November 2018.

[2] T. Schnabel and H. Schütze. FLORS: Fast and Simple Domain Adaptation for Part-of-Speech Tagging. *Transactions of the Association for Computational Linguistics 2*, pages 15–26, 2014.