# Language Understanding Systems — Mid-Term project: FST & GRM Tools for SLU

Anonymous ACL submission

## Abstract

## 1 Data analisys

### Training and testing

The dataset provided contains the tokenized sentences and specifies the POS-tag related to all the tokens. From that file it was created a lexicon, using the tool *ngramsymbols*, and the corresponding text file in which the sentences are formed only with the IOB-tags. The lexicon included the epsilon and unkown tags, as well as all the IOB-tags. From the two files was created a weighted finite-state machine archive, which is a concatenation of the file representation of one ore more finite-state machines, using the tool *farcompilestrings*. Since one of the objective of the first part was to create a language model, two different tools were used to solve this problem: *ngramcount* and *ngrammake*. The tool *ngramcount* takes as input an archive of FST and gives as output an FST representing the count of the n-grams. The resulting fst is used as input for *ngrammake*, which create a language model. In order to get the most precise language model, ngramcount has a parameter that allows to change the number of token to count, while ngrammake allows to specify which smoothing method for the creation of the final FST. The last objective to complete before starting the testing phase is to create, with *fstcompile*, an FST using the lexicon and the graph matrix representation created at the beginning. While the first dataset was specific for the training, the second one was for the testing. It has the same format, which is word followed by the corresponding POS-tag. To test the language model and the FST it was necessary to create a different file containing a whole sentence in every row. Every line was transformed into an FST with multiple edges for the same couple of nodes, each of them weighted differently. After being composed with the trained FST and the language model, a shortest path algorithm is applied to the tested FST which results into a final FST with just one edge per couple of nodes. The file with the sentences was used for every type of FST and language model created before.

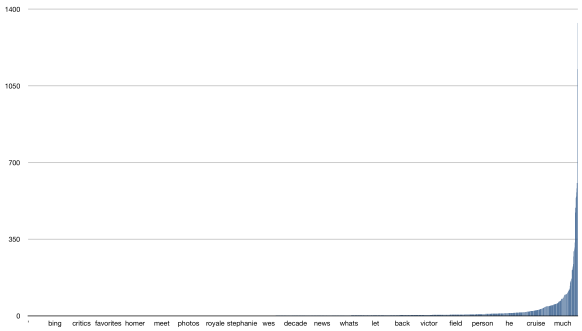


Figure 1: Zipf's law

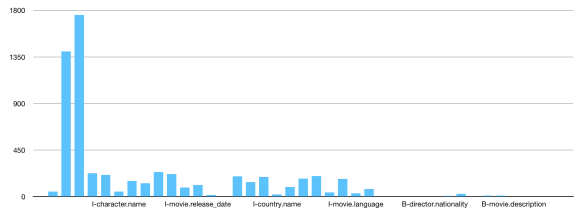


Figure 2: POS-tags distribution

## 2 Evaluation

The evaluation is the last practical phase before the analysis of the reults. There are four tipes of results: true positive, true negative, false positive and false negative. Computing the accuracy is done dividing the summation of the correct decisions by the total amount of instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

| | 1-gram | 2-gram | 3-gram |
|---|---|---|---|
| Absolute | 57.31 | **76.31** | 75.62 |
| Katz | 57.31 | 75.84 | 74.05 |
| Kneser-Ney | 57.31 | **76.27** | 75.67 |
| Unsmoothed | 57.31 | **76.15** | 75.46 |
| Presmoothed | 57.31 | **76.21** | 67.95 |
| Witten Bell | 57.31 | **76.31** | 75.52 |

Table 1: F1 Scores for every method and n-grams from 1 to 3

| | 4-gram | 5-gram |
|---|---|---|
| Absolute | **76.04** | **76.00** |
| Katz | 73.12 | 63.19 |
| Kneser-Ney | **76.04** | **76.01** |
| Unsmoothed | 75.85 | 75.90 |
| Presmoothed | 67.01 | 66.75 |
| Witten Bell | **76.07** | **76.17** |

Table 2: F1 Scores for every method and n-gram 4 and 5

Despite accuracy might seem to be enough, an unknown or infinite value of true negative can lead to a wrong result. Computing precision and recall is a good way to solve this problem.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

At this point it is possible to compute the harmonic mean of precision and recall, the F1 score.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

The output evaluation was made using a perl script called *conlleval*, which takes in input a file containing for every line the token, the real POS-tag and the predicted one, each of them separated by a single space.

The baseline F1 score is 57.31 for all methods, as showed on table 1, while the best performances were obtained with 2-grams and the method “absolute“ and “Witten Bell“.

## 3 Discussion