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

Anonymous ACL submission

Abstract

The objective of this project is to develop a Spoken Language Understanding module for movie domain using NL-SPARQL Dataset. In particular training a concept tagger for sequence labeling via a language model and an FST.

1 Data analisys

The dataset provided contains the tokenized sentences and specifies the POS-tag related to all the tokens. In figure 1 it is shown the words' frequency distribution. While the most common words have a high frequency, the specific words, like names, are less frequent. From the dataset it was created two files: a lexicon, with the tool *ngramsymbols* and a file in which every line contains the sentences formed by the IOB-tags. The lexicon, besides all the words, includes the epsilon and unkown tag, as well as all the IOB-tags. A weighted finite-state machine archive was created using the lexicon and the IOB-sentences as input for the tool *farcompilestrings*. The objective for the first part was to create a language model and a fine-state machine (or FST).

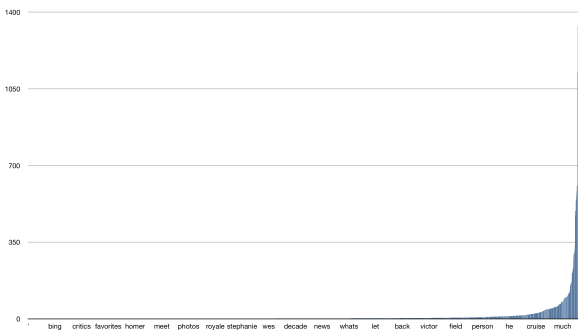

Figure 1: Zipf's law

Language model

To create a language model were employed two different tools: *ngramcount* and *ngrammake*. The first tool takes in input an archive of one or more FSTs and gives as output an FST representing the count of the n-grams. The resulting FST is used as input for *ngrammake*. Both the tools allows to specify some parameters. To create different type of language models were specified the order of n-gram, for the first tool, and the smoothing method for the second tool. This allows to test multiple language models and search for the most precise one.

FST

The FST was created using the lexicon and a file containing the matrix representation of the graph. Column one represent the start node, column two the end node, column three the token, column four the IOB-tag and column five the weight of the edge. The weight is computed as minus the logarithm of the probability of the *bigram-token-tag*.

Testing

Testing the language model and the FST was made creating a different file from the testset provided, containing in each row a whole sentence. 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.

Different training

Since the number of tag *O* is very large (figure 2), it's smart to associate the tag "O" with the related

token. For instance, if the token is “who“, the new tag will be “O-who“. In this way the precision is improved. Although the algorithm for the creation of the language model and for the FST is the same, it was increased the range of the n-grams from 1-to-5 to 1-to-9.

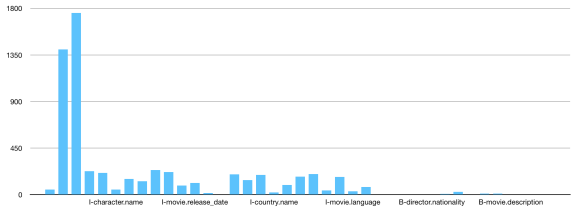

Figure 2: POS-tags distribution

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.

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

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.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

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

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

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.

Evaluation on first type of language model

Although conlleval gives all of the four types evaluation measures, the interesting result is the F1 score. On table 1 and 2 are listed the scores for each method and for each measure of n-gram. The results in bold are those that have passed the expeted performance. While comparing all method with each other the best one results to be “witten bell“, comparing the different quantity of n-grams, $n = 2$ gives the best performance.

	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

Figure 3 shows the learning scores for each method. For 1-gram and 2-gram the trend is almost the same for every method, but starting from 3-gram the performance go down, especially for “katz“ and “presmoothed“.

Evaluation on second type of language model

The language model trained over tags only shows higher performance with respect to the previous one. F1 scores for this language model are shown in table 3, 4 and 5. The results in bold show that “Kneser-Ney“ is the only method with F1 scores over the expected performance. Figure 4 shows that after 5-grams trend seems to be constant.

3 Conclusion

The language models trained for this project are two. The first one was trained over the dataset as it was provided, while the second one was trained over a modified dataset, which containes only tags. The second language model has better performance with respect to the first one and the

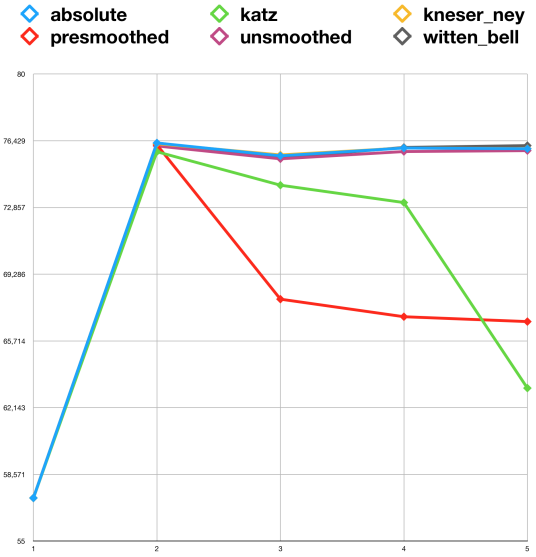

Figure 3: Performance trend for the first language model

	1-gram	2-gram	3-gram
absolute	55.31	78.88	81.37
katz	55.31	78.64	80.85
kneser_ney	55.31	79.45	**82.04**
presmoothed	55.31	77.61	79.54
unsmoothed	55.31	76.14	78.45
witten_bell	55.31	79.11	81.28

Table 3: F1 scores

F1 score remains almost constant from 5-gram up to 9-gram. This means that quality is not improved just by increasing n-grams.

	4-gram	5-gram	6-gram
absolute	81.57	81.34	81.34
katz	81.17	81.08	81.26
kneser_ney	**82.69**	**82.37**	**82.37**
presmoothed	79.77	79.47	79.56
unsmoothed	78.18	78.23	78.22
witten_bell	81.37	81.21	81.07

Table 4: F1 scores for language model trained on tags only

	7-gram	8-gram	9-gram
absolute	81.43	81.52	81.52
katz	81.17	81.08	81.08
kneser_ney	**82.61**	**82.61**	**82.70**
presmoothed	79.56	79.56	79.56
unsmoothed	78.45	78.77	78.58
witten_bell	81.52	81.48	81.48

Table 5: F1 scores for language model trained on tags only

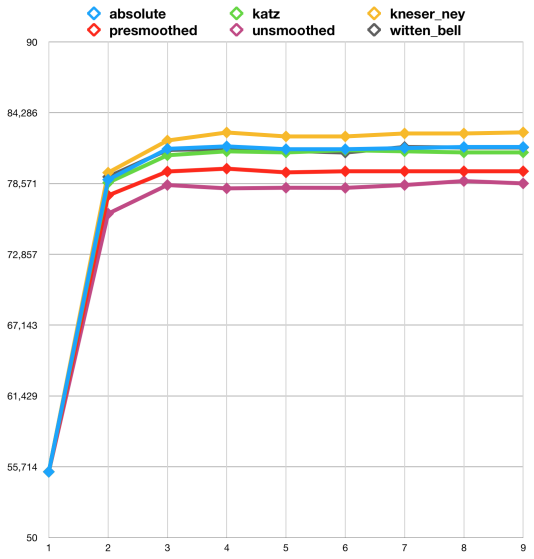

Figure 4: Performance trend for the second language model