

Language Understanding Systems Mid-Term Project 2017

FST & GRM Tools for SLU

Nicola Garau (189086)
nicola.garau@studenti.unitn.it

Abstract

The aim of this project is to develop a **SLU** (Spoken Language Understanding) module capable of assigning **Movie Domain** tags to words and to analyse its performance with different configurations.

To achieve these goals, a language model was trained to predict unseen utterances from a testing data set. For both the training and testing phase, a set of natural language utterances for movie domain was used. The results show that, with this method, training the SLU even on higher order n-grams (greater than five) with the **Kneser Ney** smoothing algorithm leads to a good **Accuracy** and **F1 Score**, obtaining a model that doesn't overfit the training data.

1 Introduction

A **Concept Tagger** is a model that assigns a token, called tag, to different words belonging to sentences. In order for it to work, it needs to be trained with several training data composed of couples word-tag.

For the purposes of this project, the following were given:

- A data set for movie domain, split into train and test set
- A data set with additional features, split into train and test set
- A Perl script to analyse the tagger performances and give different scores as output (like Accuracy, Precision, Recall and F1 Score)

2 Data Analysis

As said before, a data set for the Concept Tagger training and testing was used. The given dataset is the Microsoft NL-SPARQL data set ([Hakkani-Tur et al., 2014](#)). All the data was already split into train (3338 utterances) and test (1084 utterances). Four files were given:

- **NLSPARQL.train.data**, containing training utterances composed of tab-separated couples of word-concept tag
- **NLSPARQL.train.feats.txt**, containing training utterances composed of tab-separated triples of word-POS tag-lemma
- **NLSPARQL.test.data**, containing, containing test utterances composed of tab-separated couples of word-concept tag
- **NLSPARQL.test.feats.txt**, containing test utterances composed of tab-separated triples of word-POS tag-lemma

Each utterance in the files is separated by the others with an empty line.

Here is an example of utterance from the given data set:

- From the data file

who	O
's	O
in	O
star	B-movie.name
wars	I-movie.name
episode	I-movie.name
four	I-movie.name

- From the feats file

who	WP	who
's	VBZ	be
in	IN	in
star	NN	star

wars		NNS	war
episode	NN	episode	
four		CD	four

2.1 Data Distribution

As expected, *movie.name* is the most popular tag found (**Zipf's Law**). A graphic representing the concept distribution can be found below. [2](#)

It is interesting to notice that more than 80% of the tags are out of span concept tags.

2.2 Concept Tags

Concept tags are defined using an **IOB Notation**. The notation (the prefix notation B-NP is the one used in the dataset) is used to label multi-word spans in token-per-line format.

The prefixes are:

- **B** : beginning of a span
- **E** : end of a span
- **I** : inside of a span
- **O** : outside of span

2.3 POS (Part Of Speech) tags and lemmas

Both POS tags and lemmas in the feats files are additional features that can be used to improve the training phase of the concept tagger. They didn't really help raising the F1 Score value of the tagger in this solution, though.

2.4 Dataset Issues

During the development phase there were three main misunderstandings with respect to the additional features data sets.

The first was that it was not clear how to cope with the fact that the same word could be mapped to different lemmas ('s-be and 's-'s for example) in the transduction from word to lemma. But then it emerged that the transduction was not necessary because it led to a worst training performance.

The second issue was that was not clear how to take advantage of the test set NLSPARQL.test.feats.txt. The fact is, probably it can't be exploited during the training phase without cheating.

The third issue was that it seemed like that the additional features could help improve the performance by calculating the cost of each transition with the following formula:

$$P = \frac{C(word, lemma, POS_{tag}, Concept_{tag})}{C(Concept_{tag})}$$

But, in the end, applying that formula always led to a worst performance.

3 Development, Structure and Implementation

The whole system was entirely developed using a combination of Bash and Python 2.7. No third party libraries were used, apart from:

- **OpenFST** library ([Ope, a](#))
- **OpenGrm** NGram library ([Ope, b](#))

The main script, called *conceptTagger.sh*, can be split into 6 parts:

1. Variables definition and clearing of the previous computation's files and folders
2. Pre-processing: the training data set was augmented by appending the data set words to the concept tag whenever an out of scope concept tag was found. The pre-processing script is called *preTraining.py*. By doing so, the overall best F1 score was increased because some out of span tags that before were mapped as inside the span were now mapped correctly as out of span
3. Utterances and tags sets extraction: two lists of sentences and of the associated concept tags were needed in order to create a far archive and extract FSTs for every single utterance later. The sets can be found inside the data folder. The script responsible for the extraction is called *extractSets.py*
4. Lexicon creation: a lexicon (list of couples element-index) was created, where each element was chosen amongst (training set):
 - Words
 - POS tags
 - Concept tags

The lexicon can be found inside the data folder. The script responsible for the lexicon creation is called *createLexicon.py*

5. Word-To-Concept transducer creation: a transducer which maps words into Movie Domain concept tags was created using a Python script called *wordToConcept.py* and then compiled using OpenFsts command *fstcompile*

During this phase, the cost of the transition from a word to the corresponding concept tag was calculated as follows:

$$Cost = -\log(P)$$

Where P is the probability of the word given the concept. It was calculated by taking into account just the lemmas and POS-tags occurrences (but not the words and concept tags ones):

$$P = \frac{C(word, Concept_{tag})}{C(Concept_{tag})}$$

Where C stands for counts. With this probability calculation, the best F1 score (**82.76**) was reached with a 9-gram or 10-gram, Kneser Ney smoothing method.

6. Methods and n-gram orders iterations, tagging and evaluation: for each smoothing method offered by OpenFst (witten_bell, absolute, katz, kneser_ney, presmoothed and unsmoothed) and for each n-gram order from 1 to 10, the following pipeline was applied for each utterance in the dataset, using the appropriate generated language model:

- Composition of the current utterance FST with the Word-To-Concept FST
- Composition of the FST from the previous point with the language model (for the m-th method and the i-th n-gram order)
- Epsilon removal
- Shortest path calculation
- Topological sort
- Creation of the input file for the Perl script and evaluation of all the utterances for the m-th method and i-th n-gram order

Each result for all the combinations methods-n-gram order were saved in a file. All the results can be found in the folder *results/outputData*. The points 2,3,4 and 5 are defined by calls of Python scripts from the main Bash script. To call the main script is necessary to execute the command *bash conceptTagger.sh* from inside the scripts folder.

4 Failed attempts

Different method were tested before reaching the best result. Among them, two are worth mentioning:

1. Creation of intermediate transducers, like Word-To-Lemma and Lemma-To-Concept or Word-To-POS and POS-To-Concept. All those transducers only led to a loss of information and therefore of performance, especially when trying to map a Part-Of-Speech information into a Concept tag (intuitively, because are two unlinked kind of information). The creation of a Word-To-Lemma seemed like a good way to generalise and therefore improve the over-all performance but instead led to a worst F1 score for each combination.
2. Taking into account prior probabilities and assigning non-uniform probabilities to the transitions from **unk** words to concept tags. In the end, assigning cost zero to all the unknown transitions produced better results overall. Also, taking into consideration the additional features did improve the F1 score, but not when applying the pre-processing phase.

5 Evaluation

As mentioned before, the Perl script *conlleval.pl* gives as output a text file containing different evaluation measures, some of which are:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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

$$F = (1 + \beta^2) * \frac{Precision * Recall}{(\beta^2 * Precision) + Recall}$$

In particular, in the script, with $\beta=1$,

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

In the diagram 1 is it possible to see all the best obtained F1 scores of the possible combinations method - n-gram order for the first 10 orders of n-grams (except the first).

6 Conclusion

The method provided by this document describes a method for Movie Domain concept tagging given a training data set, which allows to obtain a high F1 score for high order n-grams, by taking into account just the words and concepts, but not additional feature sets like POS tags and lemmas. It was possible to achieve these results by applying a pre-processing phase in which each word was appended to each out of span concept tag before the training.

The best F1 score obtained by this method was 82.76, given by a Kneser Ney smoothing method with an n-gram order of 9 or 10.

Figure 1: F1 score for different smoothing methods and n-grams

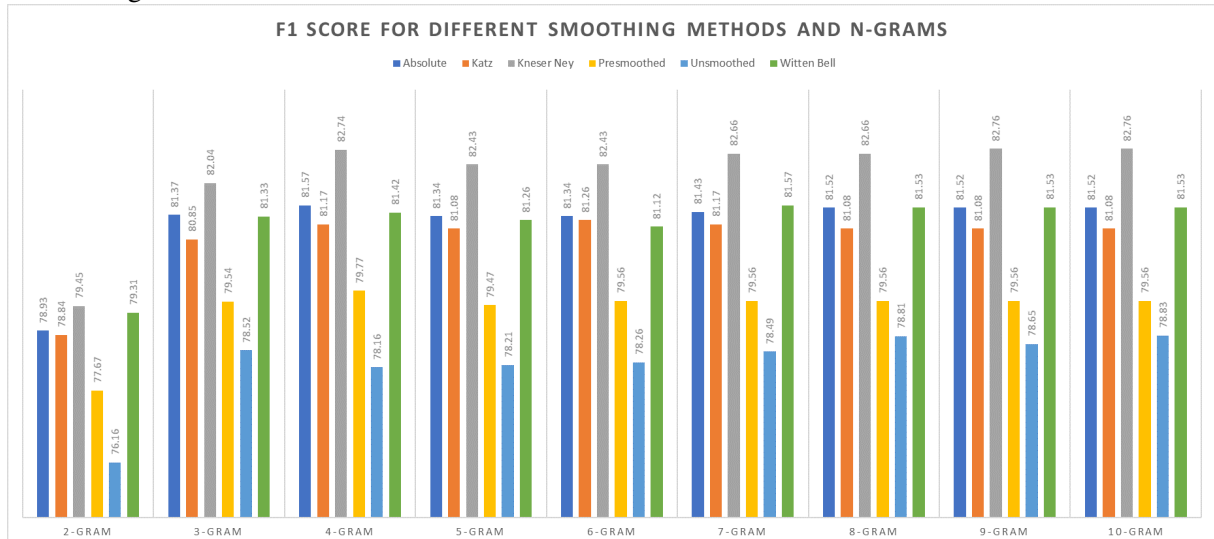
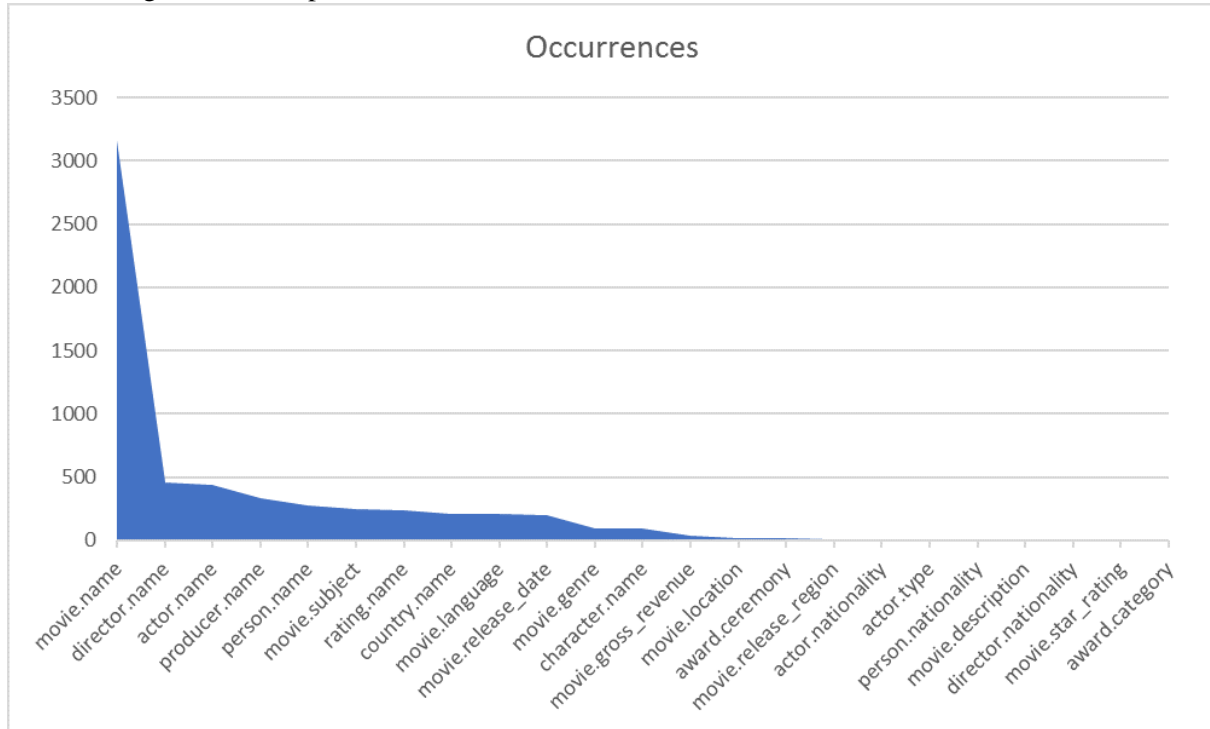


Figure 2: Concept occurrences



References

Openfst. <http://www.openfst.org/>. Last Update: 2017-03-14.

Opengrm. <http://opengrm.org/>. Last Update: 2017-01-24.

Dilek Hakkani-Tur, Asli Celikyilmaz, Larry Heck, Gokhan Tur, and Geoff Zweig. 2014. [Probabilistic enrichment of knowledge graph entities for relation detection in conversational understanding](#). In *Proceedings of Interspeech*. ISCA - International Speech Communication Association, page 1. research.microsoft.com/apps/pubs/default.aspx?id=219835.