Concept Tagging for the Movie Domain

Giovanni De Toni (197814)
University of Trento
Via Sommarive, 9, 38123 Povo,Trento TN
giovanni.detoni@studenti.unitn.it

Abstract

This work focuses on the well-known task of concept tagging phrase elements. This present a relatively important challenge in the NLP field since it is the starting point for more complex application. Weighted Finite State Transducer and statistical method were used and an analysis of their performances is also provided.

1 Introduction

One of the first step of some NLP task is to analyze a given phrase to understand its underlying concept. For instance, imagine we are using a vocal application to order something to eat (e.g., "I would like a tortel di patate delivered at Piazza Trento 1, please"). In order for a machine to understand correctly what we want, firstly it needs to convert our utterances to a word representation, secondly it needs to assign concept to each of the word it heard (e.g., "tortel di patate" equals to requested food and "Piazza Trento 1" equals to the delivery address). This basic operation is of utmost importance for all the application which can be built upon this. For instance, the quality of you food assistant is dependent on the quality of this concept tagger. Imagine what could happen if the machine were to swap the delivery address with the requested food!). The scope of this project is to provide a simple concept-tagger by developing a WSTF (Weighted Finite State Transducer) applied to the movie domain. The report is detailed as follow: in the first section we define more formally the problem statement, then we proceed with an analysis of the given dataset and with the description of the models employed. Ultimately, we discuss the obtained results while underlying their stengths and limitations.

2 Problem Statement

Given a sequence of tokens $< t_1, \dots, t_n >$ and given a pool of concepts $< c_1, \dots, c_m >$, we want to find the most likely assignment $< t_i, c_i >$ such that it maximizes the following probability:

$$c_1, \dots, c_n = \underset{c_1,\dots,c_n}{\arg\max} P(c_1, \dots, c_n | t_1, \dots, t_n) \tag{1}$$

Thanks to the Markov assumption (the probability of the i-th concept depends only on the $(i-1)$-th concept and the probability of the i-th token depends only on the i-th concept), we can simplify the previous formula. Moreover, we can also compute its component by MLE (Maximum Likelihood Estimation):

$$c_1, \dots, c_n = \underset{c_1,\dots,c_n}{\arg\max} P(t_i|c_i)P(c_i|c_{i-1}) \tag{2}$$

In the previous formula, $P(t_i|c_i) = \frac{C(c_i,t_i)}{C(c_i)}$ and $P(c_i|c_{i-1}) = \frac{C(c_{i-1},c_i)}{C(c_i)}$ (where $C(x)$ counts the occurrences of x inside the given dataset).

3 Data Analysis

The dataset used is called NL2SparQL4NLU. This work used only the "*.conll.txt" files (more specifically, one for training the language model and one for testing it). Each file is written using the token-per-line CONLL format with tokens and NLU concept tags. An analysis of the content of the two files follows:

The distribution of the tokens was analyzed. It was found to behave accordingly to the Zipf's Law, however, for brevity, it was not reported here since it was not showing anything interesting. The OOV rate for the tokens is around 23.68%. The distribution of the various concepts was also analyzed (both for the training and the test datasets). The distribution of the various concepts is shown in Figure **??**. It is possible to

see how the *movie.name* concept is one of the most common (followed by *actor.name* and *person.name*). However, it can also be noted that the "*O*" concept represent the 70% of the dataset, which represent an issue for the following tagging procedure.

Both train and test set contains concepts which are not present in the other dataset (and viceversa). The train dataset contains *person.nationality* (2 occurrences) and *movie.description* (2 occurrences) which are not present in the test set. The test dataset contains the concept *movie.type* which is missing from the train dataset (4 occurrences). These again could cause mistagging issues and therefore lower the final performance. As a final note, there are also some tokens which are the results of mispelling (e.g., "dispaly", "Scorcese") which therefore could lead to mistagging.

4 Models

In this work we devised four separated language models. The first language model was developed by using directly the dataset provided (without any other addition). It implements directly Formula **??**. The second model implements a solution provided by Gobbi et al. [add citation]. As the data analysis showed us, the majority of the dataset concepts are "*O*" which are not informative enough. Gobbi et al. proposes to substitute each occurrences of the "*O*" concept with the corresponding lemmatized token such to increase the final performances. The third and fourth models add to their pipelines an entity recognition tool which converts certain token(s) to an entity definition. This entity definition replace the previous token. This improvement was performed such to reduce the variability inside the dataset. Some tokens infact refer to the same entity, while they may have different values (e.g., *nick fury* and *robin* are both character names, even if they have different tokens). See Figure 1 for an example of entity recognition.

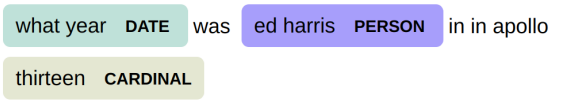


Figure 1: Example of entity recognition. For instance, the "*ed harris*" tokens were recognized as a PERSON while "*thirteen*" was recognized as number.

Generally, the entire pipeline was composed by three main components:

- First
- Secon
- Third

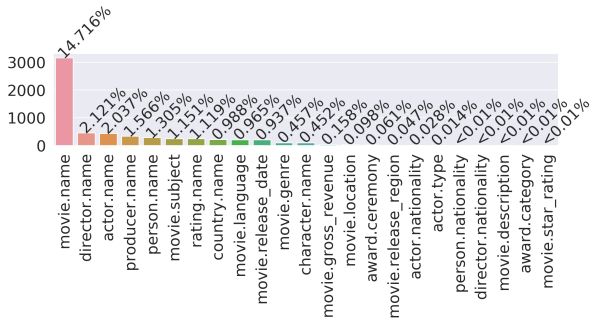
5 Experiments

Several experiments were run to assess the quality of the concept taggers. As a metric, we tried to find the solution which maximize the *F1-score* over the test set. For each provided solution, we perfomed an extensive hyperparameter search by trying several smoothing techniques and ngram sizes. Each solution was evaluated by using k-fold cross-validation and the best result was then recorded. Firstly, we evaluated the Language Model based directly on the given dataset without any further improvements. The best combination was then used as a baseline. Secondly, we computed a second improved baseline performance by using the solution provided by Gobbi et al. [add citation]. Ultimately, we evaluated the two SpaCy model variants (dataset with entity recognition and dataset with entity recognition plus "*O*" tags replaced. The results are summarized in Table X.

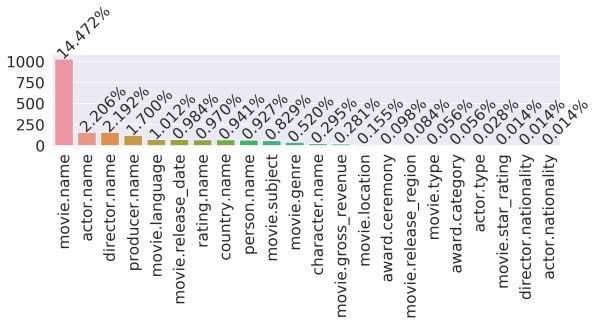
The code employed for the project is publicy available on Github [add link]. The script were written using Python 3.6 and bash. OpenFST [add citation] and OpenNGRAM [add citation] libraries were used to build the WFST and the language model. SpaCy [add citation] library was used to perform entity recognition. The experiments were run on a 8th-core Intel i7 CPU with 16GB of RAM.

6 Results

7 Credits



(a) Train dataset.



(b) Test dataset.

Figure 2: Distributions of the various concepts in the train and test datasets. The *O* concept was removed to make it easier to understand the weight of the other concepts.

NL2SparQL4NLU.trai.conll.txt	
# of lines	24791
# of sentences	3338
# of unique tokens	1728
# of unique concepts (without the prefix)	24

(a) Description of the content of the train dataset.

NL2SparQL4NLU.test.conll.txt	
# of lines	8201
# of sentences	1084
# of unique tokens	1039
# of unique concepts (without the prefix)	23

(b) Description of the content of the test dataset.