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 sentences. This represents a relatively important challenge when doing Natural Language Processing applications since it is the starting point for more complex techniques. This report details the realization of an SLU module for concept tagging on the movie domain. It shows also the performance obtained and the results over various techniques.

1 Introduction

One of the first and most important of any NLP task is to analyze a given phrase to understand its underlying meaning. More specifically, we want to find the most likely interpretation which maps the phrase to given concepts. 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"). 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 a concept to each of the words it heard (e.g., "tortel di patate" equals to an hypothetical "FOOD" concept 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 your 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 structured as follow: the first section defines 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 strengths 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:

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

The previous formula can be made easier to compute thanks to the Markov assumption. The probability of the i-th concept c_i depends only on the $(i-1)$-th concept c_{i-1} and the probability of the i-th token t_i depends only on the i-th concept c_i. We can estimate the various parameter by Maximum Likelihood (MLE):

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

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 [citation]. 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.

First of all, we analyzed the distribution of the tokens. and it was found to behave accordingly to the Zipf's Law. The OOV rate for the tokens between the train and the test datasets is around 23.68%. Secondly, we analyzed the distribution of

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

(a) Train dataset.

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

(b) Test dataset.

Table 1: Description of the content of the dataset used.

the various concepts. In such case, we performed a more in-depth analysis. Figure **??** shows the distribution of the various concept. We noted that the "*O*" concept represent the 70% of the dataset, which pose a big problem for the following tagging procedure. The concepts graph do not show the "O" concept distribution to make it easier to visualize the distribution of the other concepts. It is possible to see how the *movie.name* concept is one of the most common (followed by *actor.name* and *person.name*).

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.

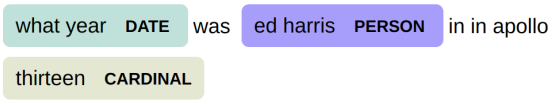


Figure 1: Example of entity recognition. For instance, the "*ed harris*" tokens were recognized as a PERSON while "*thirteen*" was recognized as number.

4.1 Spoken Language Understanding Pipeline

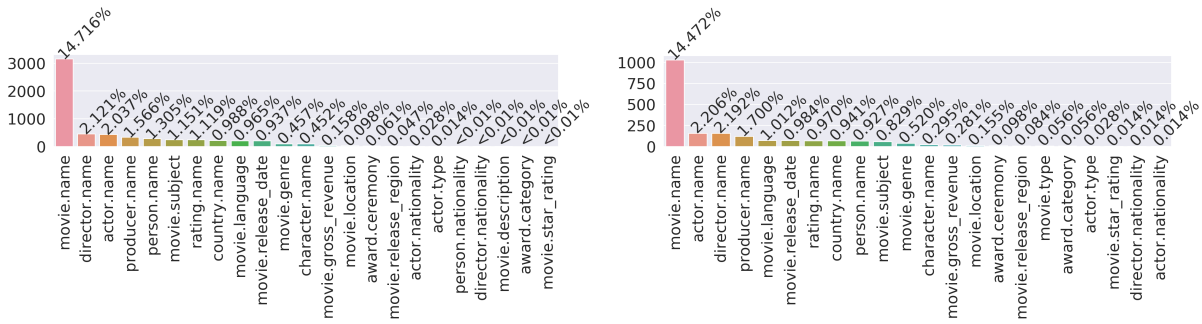
The entire pipeline is composed of two main components.

4.1.1 Concept Tagger (WFST)

It is a transducer which encodes the probability $P(t_i|c_i) = \frac{C(c_i,t_i)}{C(c_i)}$. The probability (score) is the weigth given to the transition from the given token to the concept. In order to solve possible numerical stability issue, the negative log was applied to the probability. Therefore, the weigth assigned to the transitions is computed by using $-log(P(t_i|c_i))$. An additional token called *<unk>* was added to the transducer to account for unknown words (namely, tokens which are present in the test dataset, but not in the train dataset). Moreover, for each of the possible concepts c_i, a transition c_i:*<unk>* was added with a score of $\frac{1}{\#concepts}$ (this means that each unknown token has an equal probability of "represeting" each concepts).

4.2 Language Model (LM)

It is a transducer which encodes the probability $P(c_i|c_{i-1}) = \frac{C(c_{i-1},c_i)}{C(c_i)}$, which mean finding a concept c_i conditioned on having seen concept



(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.

c_{i-1}.

4.3 Final SLU Model

The previous components needs to be composed together to be used. The sentence we want to tag needs to be transformed into a FST and then it will be composed with the previouse outlined components:

$$\lambda_{CT} = \lambda_{sentence} \circ \lambda_{WFST} \circ \lambda_{LM} \qquad (3)$$

Then, we need to find the path which minimize the cost within the final transducer (since we are using $-log()$ to compute the weigths, minimizing the path cost is the equivalent of maximizing the probability of that tagging sequence). This final operation corresponds to computing the equation 2.

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 ($k = 3$) and the best result was then recorded. Firstly, we evaluated the SLU 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.

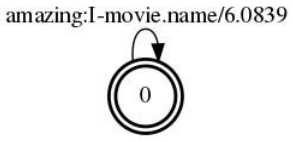


Figure 3: Example of a possible transducer in which the token *amazing* is mapped to the concept *I-movie.name* with an associated transition score.

6 Results

7 Credits

Table 2: Evaluation results

ngram size	smoothing used	pruning	replace O	entity resolution	k-fold F1 score
4	kneser_ney	5	keep	OFF	75.04
4	kneser_ney	5	keep	ON	70.6
4	kneser_ney	5	lemma	OFF	83.45
4	kneser_ney	5	lemma	ON	82.32
4	kneser_ney	5	stem	OFF	83.61
4	kneser_ney	5	stem	ON	82.64
4	kneser_ney	5	word	OFF	83.73
4	kneser_ney	5	word	ON	83.37
4	witten_bell	5	keep	OFF	75.32
4	witten_bell	5	keep	ON	70.39
4	witten_bell	5	lemma	OFF	83.29
4	witten_bell	5	lemma	ON	82.48
4	witten_bell	5	stem	OFF	83.38
4	witten_bell	5	stem	ON	82.98
4	witten_bell	5	word	OFF	83.36
4	witten_bell	5	word	ON	82.24