Concept Tagging for the Movie Domain by using Named Entity Recognition Tools.

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. It is the starting point for more complex techniques and it represents a relatively important challenge when building Spoken Language Understanding applications. This report details the realization of two SLU modules by using Named Entity Recognition tools for concept tagging sentences taken from the movie domain. We also show the performance obtained over various experiments.

1 Introduction

One of the first and most important of any NLP task is to analyze a given sentence to understand its underlying meaning. More specifically, we want to find the most likely interpretation which maps the utterances to given concepts. For instance, imagine we are using a vocal assistant to order something to eat (e.g., "I would like a *tortel de patate*[1] to be delivered at *Piazza Trento 1*, please"). Firstly, a machine needs to convert our utterances to a word representation. Secondly, it needs to assign a concept to each of the words it heard to understand correctly what we want (e.g., "*tortel de patate*" equals to a hypothetical "FOOD" concept and "*Piazza Trento 1*" equals to the "DELIVERY ADDRESS"). This basic operation is of utmost importance for all the applications 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 Spoken Language Understanding Model applied to sentences in the movie domain by using statistical tools and Named Entity Recognition (NER) techniques. The report is structured as follow: the first section defines more formally the problem statement, then we proceed with an analysis of the used dataset and with the description of the models employed. Ultimately, we discuss the experiments and the evaluation results while underlying the models' strengths and limitations.

2 Problem Statement

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

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

The previous formula can be made easier to compute thanks to the *Markov assumption*. Namely, 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 parameters by *Maximum Likelihood (MLE)*:

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

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

We used the dataset called *NL2SparQL4NLU*[2] (Chen et al., 2014; Gobbi et al., 2018). It contains several English sentences related to the movie domain. This work used only the "*.conll.txt" files.

[1] A typical northern-Italy dish.

[2] https://github.com/esrel/NL2SparQL4NLU

NL2SparQL4NLU.train.conll.txt	
# of lines	24791
# of sentences	3338
Dictionary size (# unique tokens)	1728

(a) Train dataset.

NL2SparQL4NLU.test.conll.txt	
# of lines	8201
# of sentences	1084
Dictionary size (# unique tokens)	1039

(b) Test dataset.

Table 1: Brief description of the content of the dataset used.

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. Table 1 provides a general description of the two files. The average length of the sentences is around 6.42 and 6.52 for the train and test dataset respectively. The *OOV rate* for the tokens between the train and the test dataset is around 24%.

Figure 2 shows the distribution of the various concepts. There are 23 distinct concepts which define 43 tags in IOB tagging format. We noted that the "O" concept represents the 70% of the dataset, which poses a significant problem for the tagging procedure. The concept graphs do not show the "O" concept percentage 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 *director.name*.

Moreover, both the train and test set contains concepts which are not present in the other dataset (and vice-versa). 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 training dataset (4 occurrences). These again could cause mistagging issues and therefore lower the final performance of the SLU model. As a final note, there are also some tokens which are the results of misspelling errors (e.g., "dispaly", "Scorcese") which could lead to other tagging inaccuracies.

4 Models

In this work, we devised and evaluated two separated SLU models. The first SLU model was trained by using directly the dataset provided (without modifications) and it implements Formula 2. This first model was used as the baseline. The second model uses named entity recognition tools which convert certain token(s) to an entity definition before training the complete language model. See Figure 1 for an example of named entity recognition. These entity definitions replace the previous tokens. We chose two external libraries which provides two pre-trained NER classifier: **NLTK**[3] and **spaCy** [4]. We performed these improvements to mitigate some ambiguity and variability issues which are presents in the dataset. More specifically, some tokens refer to the same entity and the same concept while they may have different values (e.g., "*nick fury*" and "*robin*" are both "*persons*" (entity) and "*character names*" (concept), even if they have different tokens). As the dataset analysis showed us, the majority of the concepts are equal to "*O*" which is not informative enough. A possible solution already used in the literature is to substitute each occurrence of the "O" concept with another value. We tried three possible substitution policies. We used: directly the corresponding **token value**, the **token stem** and the **token lemma**.

4.1 Spoken Language Understanding Pipeline

The entire SLU pipeline of both models includes two main components: the *Concept Tagger (WFST)* and the *Language Model (LM)*. Moreover, the *Named Entity Recognition Classifier* can be optionally used before building the WFST and the LM.

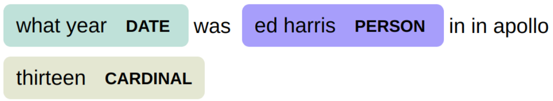
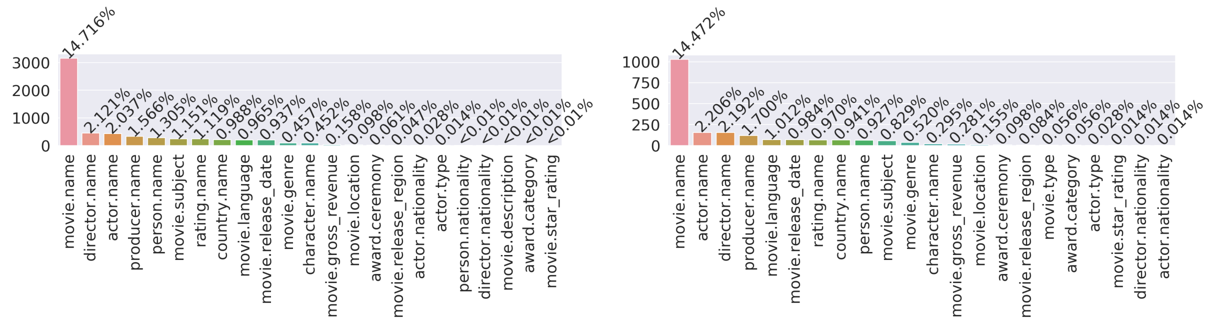


Figure 1: Example of named entity recognition. For instance, the "*ed harris*" tokens were recognized as a PERSON while "*thirteen*" was recognized as number.

[3] https://www.nltk.org/

[4] https://spacy.io/



(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.

4.1.1 NER Classifier

The *Named Entity Recognition classifier* was used to convert certain tokens (or group of tokens) into *entities*. After applying the classifier and after having removed the possible duplicates entities (e.g., *ed harris* is resolved into *PERSON PERSON* instead of just one entity *PERSON*), we obtained a final phrase which was to build the SLU model. Below we show an example of the process:

1. *what year was ed harris in in apollo thirteen*
2. *DATE was PERSON in in apollo CARDINAL*
3. *O O B-actor.name O O B-movie.name I-movie.name*

The first phrase is the original. The second represents the "translation" made by the NER and the third represents the final concepts. Apart from the double "*in*", we can already seen the limits of this approach (e.g. *apollo thirteen* is not recognized as a definite entity).

4.1.2 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 weight given to the transition from the given token to the concept (see Figure 3). To prevent numerical stability issues, the negative log was applied to the probability. Therefore, the weights assigned to the transitions are computed by using $-log(P(t_i|c_i))$. An additional token called *<unk>* was added to the transducer to account for Out of Vocabulary words (namely, tokens which are present in the test dataset, but not in the lexicon). Moreover, for each of the possible concepts c_i, a transition *<unk>*:c_i was added with a score of $\frac{1}{\#concepts}$ (this means that each unknown token has an equal probability of "representing" each concept).

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 means finding a concept c_i conditioned on having seen concept c_{i-1}.

4.3 Final SLU Model

The previous components need to be composed together with the sentence we want to tag to be used.

$$\lambda_{CT} = \lambda_{sentence} \circ \lambda_{WFST} \circ \lambda_{LM} \tag{3}$$

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

5 Experiments

We run several experiments to assess the quality of the SLU models. For each provided solution, we performed an **extensive hyperparameters search** by trying several **smoothing techniques**, **ngram sizes**, **pruning frequencies** and the "O" concept replacement methods. As a metric, we tried to find the solution which maximizes the *F1-score* over

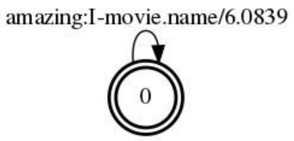


Figure 3: Example of a possible transducer in which the token *amazing* is mapped to the concept *I-movie.name* with an associated transition score.

Table 2: Best K-fold cross-validation results ($K = 3$). The bold values indicate the best score. The underlined values indicate the overall best hyperparameters combination. The baseline is shown on the first line from the top.

Ngram Size	Smoothing	Pruning	Replace O	K-fold F1 Score	K-fold F1 Score (spaCy)
4	*kneser_ney*	*5*	*keep*	*75.0467*	*74.4367*
4	kneser_ney	5	lemma	**83.45**	83.2167
4	kneser_ney	5	stem	**83.6067**	83.5033
4	kneser_ney	5	token	**83.73**	83.5467
4	witten_bell	5	token	83.3633	**83.4567**
4	katz	5	stem	82.8733	**83.1033**

the test set. 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 evaluated the NER SLU model. The code employed for the project is publicy available on Github[5]. OpenFST [6] and OpenGRM[7] libraries were used to build the WFST and the language model. The experiments were run on an 8th-core Intel i7 CPU with 16GB of RAM.

6 Results

Table 2 shows the best results over all the hyperparameters combinations tested. We were able to increase the performances of around 8% from the baseline by replacing the "O" concepts with either the **stem/lemma** of the token or the token itself as expected. However, **we did not notice any significative improvements by using the NER classifiers** (apart from a reduction of the *perplexity* from 2492.06 to 1815.36). We were able to improve the precision and recall of certain concepts when using a NER classifier (e.g., *movie.gross_revenue*, *movie.release_date* or *movie.language*), but we also reduced the precision and recall of other concepts with a higher distribution (e.g., *movie.name* or *producer.name*), thus lowering the overall performances. The NER SLU model performed a little better when using the Witten-Bell/Katz smoothing method, but we did not gain any important improvement. We also discovered that **the spaCy NER classifier performs better than the NLTK NER classifier**. Therefore, the performance with NLTK was not reported in this report since they were almost similar to the non-NER SLU results.

We think that adding the spaCy NER classifier does not improve the performances mainly because it was trained on the OntoNotes dataset [8] which is a general source for annotated text and **it is not specific for the movie domain**. Therefore, the entities used are quite general (e.g., PERSON instead of ACTOR or DIRECTOR). This caused some errors when doing named entity resolution (e.g., "*apollo 13*" is the title of a film, but just 13 was recognized as a CARDINAL entity, while both the tokens should have been recognized). Furthermore, the original dataset had some issues regarding its concepts definition. Some of them have a very similar meaning and they represent the same entity in the end. For example, *movie.location* and *movie.country* both resolves into GPE (GeoPolitical Entity). Therefore, it happened that some tokens were tagged as *movie.location* instead of *movie.country*.

One possible solution would be to build a **Custom NER Model** specifically trained on the movie dataset such to categorize more efficiently the various tokens. In addition, more complex models could be explored, like deep neural networks (Gobbi et al., 2018), and we could try to devise new and better features from the training dataset.

References

Yun-Nung Chen, Dilek Z. Hakkani-Tür, and Gökhan Tür. 2014. Deriving local relational surface forms from dependency-based entity embeddings for unsupervised spoken language understanding. *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 242–247.

Jacopo Gobbi, Evgeny Stepanov, and Giuseppe Riccardi. 2018. Concept tagging for natural language understanding: Two decadelong algorithm development.

[5] https://github.com/geektoni/concept_tagging_NLP

[6] http://www.openfst.org/

[7] http://www.opengrm.org/

[8] https://catalog.ldc.upenn.edu/LDC2013T19