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[1] (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. 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

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

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 train and test dataset respectively. The OOV rate for the tokens between the train and the test datasets is around 0.24%.

Figure 2 shows the distribution of the various concepts. There are 43 concepts tag in the dataset (with the IOB tags) and 23 without the prefix. We noted that the "*O*" concept represent the 70% of the dataset, which pose a big problem for the next tagging procedure. The concepts graph do not show the "O" concept 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*.

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 of the SLU model. As a final note, there are also some tokens which are the results of mispelling (e.g., "dispaly", "Scorcese") which therefore could lead to other tagging errors.

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). It implements Formula 2. This first model was used as the baseline. The second model uses entity recognition (ER) tools which convert certain token(s) to an entity definition prior training the complete language model. See Figure 1 for an example of entity recognition. These entity definitions replace the previous tokens. We choose two external libraries which provides two pre-trained ER classifier: NLTK (Bird et al., 2009) and spaCy (Honnibal and Montani, 2017). 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 analysis showed us, the majority of the dataset concepts are equal to "*O*" which is not informative enough. A possible solution is to substitute each occurrences of the "*O*" concept with another value, such to increase the final performances. We tried three possibile substitution policies. We used: directly **the token**, the **token's stem** and the **token's lemma**.

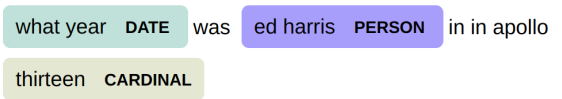


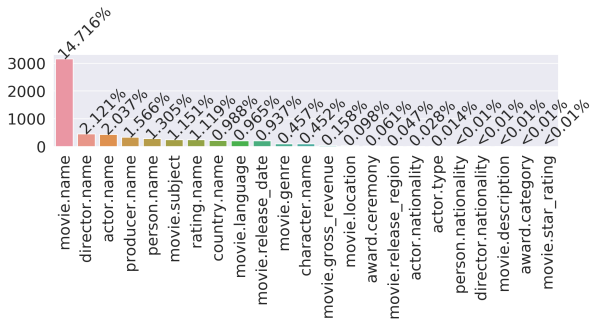
Figure 1: Example of entity recognition. For instance, the "*ed harris*" tokens were recognized as a PERSON while "*thirteen*" was recognized as number. Therefore, the "*ed harris*" tokens will be replaced by the entity "PERSON"

4.1 Spoken Language Understanding Pipeline

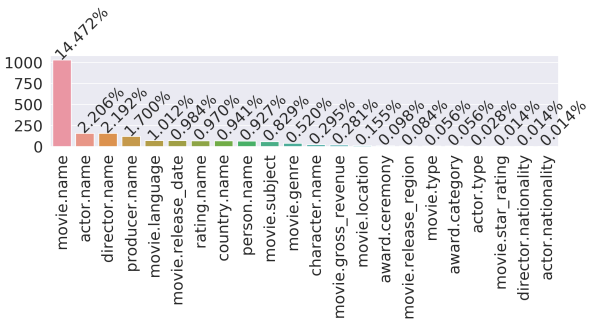
The entire SLU pipeline of both models is composed of two main components: the Concept Tagger (WFST) and the Language Model (LM). Moreover, the Entity Recognition classifier can be optionally used before building the WFST and the LM.

4.1.1 ER Classifier

The Entity-Recognition classifier was used to covert certain tokens (or group of tokens) into entities. After applying the ER classifier and after having removed the possible duplicates entities (e.g., *ed harris* is resolved into *PERSON PERSON* in-



(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.

stead of just one entity *PERSON*), we obtain a final phrase which will be used to build the SLU model.

- **Original Phrase:** *what year was ed harris in in apollo thirteen*
- **ER Phrase:** *DATE was PERSON in in apollo CARDINAL*
- **Final Concepts:** *_date O B-actor.name O O B-movie.name _cardinal*

From this example 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 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 *<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 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 c_{i-1}.

4.3 Final SLU Model

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

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

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 shown in Formula 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**, **ngram sizes** and **pruning frequencies**. 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. Sec-

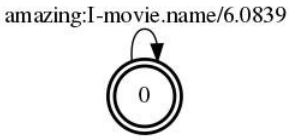


Figure 3: Example of a possible transducer in which the token *amazing* is mapped to the concept *I-movie.name* with an associated transition score.

Table 2: K-fold cross-validation results ($K = 3$). The bold values indicate the best combination found with respect to the baseline which is shown on the first line from the top.

ngram size	smoothing used	pruning	replace O	k-fold F1 score	k-fold F1 score (ER)
4	kneser_ney	5	keep	75.04	74.41
4	kneser_ney	5	lemma	83.45	83.27
4	kneser_ney	5	stem	83.61	83.47
4	kneser_ney	5	word	83.73	83.49
4	witten_bell	5	keep	75.32	74.53
4	witten_bell	5	lemma	83.29	83.43
4	witten_bell	5	stem	83.38	83.55
4	witten_bell	5	word	83.36	83.40

ondly, we evaluated the ER SLU model. The results are summarized in Table 2.

The code employed for the project is publicy available on Github[2]. The script were written using Python 3.6 and bash. OpenFST (Allauzen et al., 2007) and OpenNGRAM (Roark et al., 2012) libraries were used to build the WFST and the language model. The spaCy[3] and NLTK[4] libraries were used to perform entity recognition. The experiments were run on a 8th-core Intel i7 CPU with 16GB of RAM.

6 Results

We performed an intensive hyperparameter search and we discover that the best results were obtained using the: **Kneser-Ney** smoothing method, an **ngram size of 4** and a **pruning threshold of 5**. Moreover, we discover that by replacing the "O" concepts with either the **stem** of the token or the token itself we were able to increase the performances of 8%. However, we did not notice any significative improvements by using the entity-resolution classifiers. We also discover that the spaCy's ER classifier performs much better than the provided NLTK ER classifier.Table X shows the complete results for the best combinations.

We think that the ER classifier does not improve the performances for several reasons. First of all, the spaCy entity resolution tool is not very precise. It was trained on the OntoNotes dataset [add citation] which is a general source for annotated text. It is not specific for the movie domain. Therefore, the label used are quite general (e.g., PERSON instead of ACTOR or DIRECTOR). For example, this caused some wrong entity recognition tagging (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 had recognized). Secondly, the original dataset has some issues regarding its concepts definition. Some of the them are very similar and they represent the same entity in the end. For instance, movie.location and move.country indicates states and it can be difficult to differentiate between them. Therefore, it happens that some tokens are tagged as movie.location instead of movie.country.

One possible solution would be to build a custom Entity Recognition model specifically trained on the NL2SparQL4NLU dataset such to categorize more efficiently the various tokens. Moreover, more complex model could be explored (e.g., deep neural networks like LSTM) as several works have already done.

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

[3]https://spacy.io/

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

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. Openfst: A general and efficient weighted finite-state transducer library. In *Proceedings of the 12th International Conference on Implementation and Application of Automata*, CIAA07, page 1123, Berlin, Heidelberg. Springer-Verlag.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*, 1st edition. OReilly Media, Inc.

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.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai. 2012. The opengrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, ACL 12, page 6166, USA. Association for Computational Linguistics.