Language Understanding System First Part

gabriel.roccabruna@studenti.unitn.it

December 2019

1 Abstract

This paper explains the techniques and the results obtained during the first project of Language Understanding System. The task is to build a name entity recognition classifier using a weighted finite state transducer (WFST). The first part consists in build a WFST that reaches the baseline of 76% of F1 metric. The second part consists in using the generalization of some concepts in order to tackle the problem regarding the overlapping and sparsity of some NER classes. The document is organized as follows:

- Introduction: where the theory of tools used in the projects is summarize,
- Dataset: where the dataset used is analyzed,
- Problems and solutions: in which an explanation of the problems and solutions of part one and part two of the project is provided.
- Results: in which the comparison of results obtained is shown,
- Error Analysis: in which the findings solution of the second part are shown.
- Conlusions.

2 Introduction

In this project, the task of NER tagging is undertaken as a sequence labeling problem. Therefore, examples and labels are represented as sequences of elements. In this project examples are sequences of words $w = w_1, w_2, w_3, ..., w_n$ and labels are sequences of concepts $c = c_1, c_2, c_3..., c_n$. In term of probabilities, the output of the WFST is the sequence C that maximizes the conditional probability $P(C|W)$, thus, $C = \text{argmax}_c\, P(c_1, c_2, ..., c_n|w_1, w_2, ..., w_n)$. To obtain this sequence using WFST, the sequence of words is converted into a final state acceptor (FSA). The FSA is composed with the WFST created from the training set, where input symbols are words and output symbols are concepts, and weights are computed as $\frac{Count(word,concept)}{Count(concept)}$. Finally, the resulting automaton is composed with the selected language model. Then, the result is obtained taking the shortest path of the final automaton.

3 Dataset

The dataset is called "NLSPARQL", it contains utterances regarding requests of information about the movie world, such as actors biography, movies characters, etc. The dataset is already split in training and test set, which are composed by 3338 and 1084 utterances respectively. In the training set the concept classes are 24. Furthermore, the most common classes are: "O" with 71% and "movie.name" with 14%. The remaining classes have a presence below 3%. While in the test set the concept classes are 23, their distribution is mostly the same of the training set. In overall, different concepts is 25, since in the training set there are two concepts, "movie.description" and "person.nationality", that are not present in the test set, while in the test set there is a concept "movie.type" that never appears in the training set.
Furthermore, the out-of-vocabulary (OOV) words are 246. These are distributed over the following concepts, where the number in the brackets is the percentage over the entire test set: "O" 92 (1.29%), "movie.name" 68 (0.95%), "director.name" 23 (0.32%), "actor.name" 14 (0.2%). Figures 1 and 2 show the other concepts distribution.

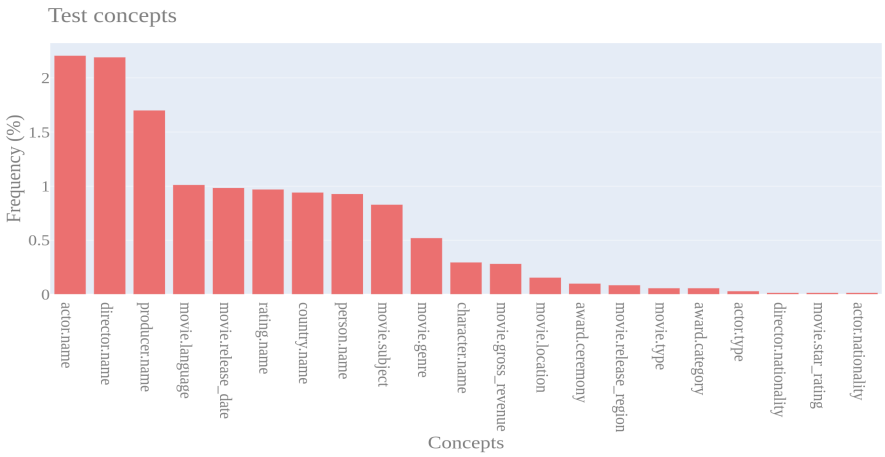


Figure 1: Test concepts without O and move.name

4 Problems and Solutions

4.1 First part

In the first part the assignment is to reach the 76% of F1 metric using a WFST and also experimenting different cut-off techniques. The results are reported in table 2.

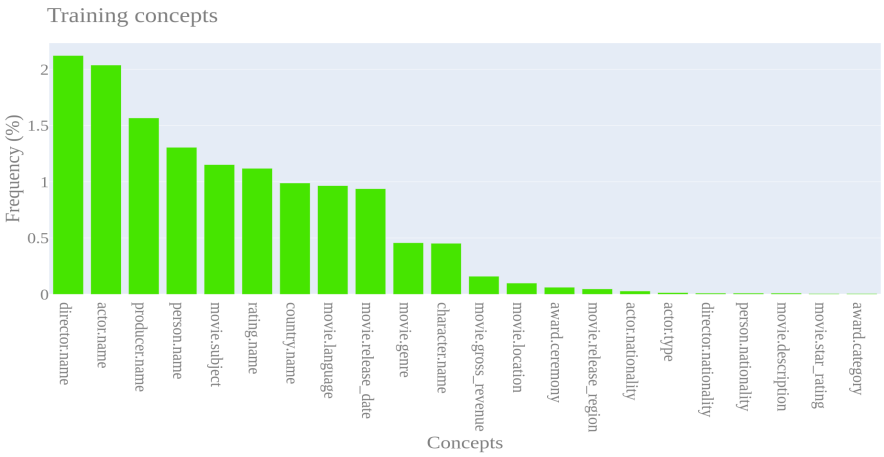


Figure 2: Train concepts without O and move.name

4.2 Second part

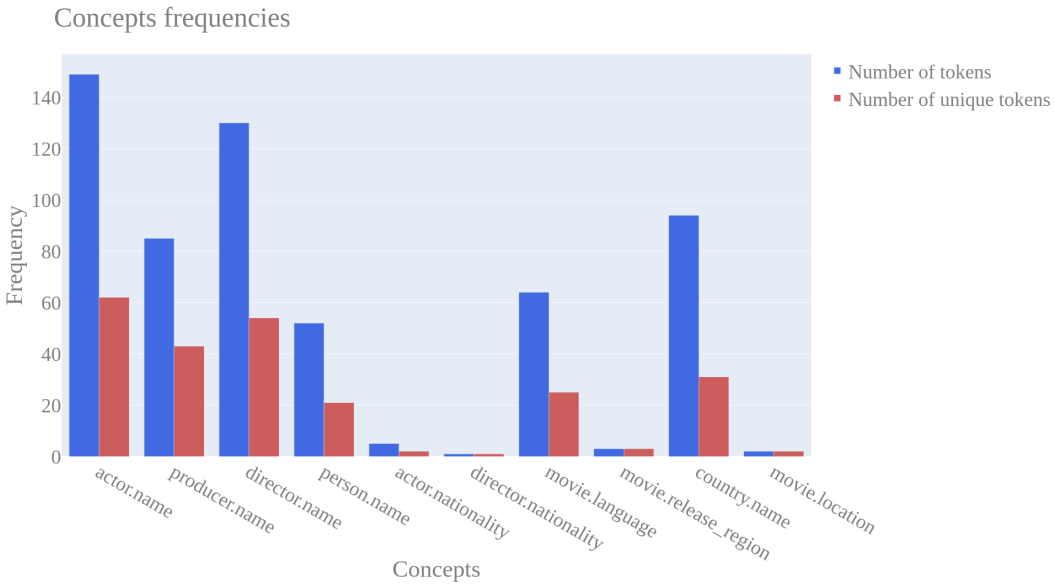
Overlapping and sparsity are issues that affect concept classes and they are related to every class classification task. The overlapping happens when two or more different classes have in common some elements. Formally, given a class A and a class B, they overlap if their intersection is not empty. On the other hand, sparsity problem happens when a class has few examples, this is due to two factors: very specific classes and lack of information. The specificity is intended as class that contains a subset of the main set, the smaller the subset the bigger the specificity is. Whereas, the lack of information is due to the size of the training set. It may happen that a class is not very specific but, in the training set there are few examples and then the sparsity of that class rises.
In the training set, analysing different classes it emerged that overlapping is a common characteristic since there are many IS-A of a main class. The biggest concept class "Name" contains six sub-concepts: person, actor, director, producer, country and character. Note that country name is totally an outsider in comparison with the other concepts and it is semantically closer to nationality class. However, there is also another hidden class that is nationality. Nationality has as sub-classes actor and director, but there is also the concept called movie.language which contains overlapping tokens, since in English language nationality and language have the same token. Note that this correlation is not one-to-one, since there exists nationalities where the corresponding language token is different, such as Bolivian nationality where there is not any Bolivian language. In the following table 1 and chart **??** the number of unique overlapping elements and frequency of the considered classes are reported. This analysis regards the training set.

The problems described above could be solved generalizing two or more con-

Table 1: Overlapping analysis

Concepts A	Concepts B	$A \cap B$
Person.name	Actor.name	5
	Producer.name	6
	Director.name	9
Actor.name	Producer.name	4
	Director.name	6
Director.name	Producer.name	14
Country.name	Movie.release_region	1
	Director.nationality	1
	Movie.language	6
Movie.language	Actor.nationality	1

Frequencies analysis



cepts. The generalization consists in merging two or more classes under the same concept, such as: "British people speak English language -> person.nationality people speak person.nationality -> person.nationality O O person.language". The next section describes how the generalization can be achieved.

4.3 Methodology

To implement the generalization task, two WSFTs are needed. The first one is needed to convert the given utterance into general concepts (generalization)

and the second one is needed to convert the generalized utterance with into NER concepts. Therefore, two training sets are needed, where the main difficulty consists of transferring the generalization to the training set. To do this, different attempts were tried. The first attempt was to use an already trained NER tagger. The selected NER tagger is constrained on StanfordNLP [2] since, it is the only one NER tagger that works with lower case words. Note that NER tags provided by StanfordNLP are generic and in many cases not pertinent to the concepts of the dataset (only NATIONALITY could be covered). Another attempt was to use gazetteers, taking some lists of actor and movie names from IMDB, but they did not give a good generalization result since, many tokens were missing from the dataset. Therefore, another approach was used, relevant tokens were extracted from the training set using the already existent concepts as a guide. However, even this approach is prone to error due to the existence of mistakes already present in the training set made by human annotators.
Furthermore, in the second WFST (generalization to concepts) the improvement 2018 is also implemented. This improvement consists in considering also the words and not only the concepts in the language model. In detail, only "O" concepts are replaced using the following policy: replace the concept with a word if $P(word|concept)$ is grater than a threshold, which is an hyper parameter to tune.

5 Results

In this section the obtained results are reported .

Table 2: Results first part

Cut off(upper-lower)	(None-None)	(1000-1)	(None-10)	(800-None)
Number of unique tokens	1728	948	250	1726
Best F1	76.37	72.64	54.31	75.58
Ngram	2	2	4	3
Smooth	absolute and witten-bell	absolute	witten-bell	kenser-ney

Cut-off: upper and lower bound are expressed in the number of occurrences of a token. For instance, in the range 1000-1 every word frequency grater and equal than 1000 and lower and equal than 1 are considered as "unk".
Number of unique tokens: it is the length of lexicon without considering "eps" and "unk" tokens.

Table 3: Results second part

Generalization	F1	F1(threshold > 0.07)	F1(threshold > 0.07 and NER)	F1 (only NER)
Base	76.37	82.79	82.62	75.18
actor.name ∪ *director.name*	76.27	82.56	82.60	75
actor.name ∪ *person.name*	76.37	82.52	82.56	75.09
actor.name ∩ *person.name*	76.31	82.70	82.53	75.09
director.name ∩ *producer.name*	76.40	82.79	82.62	75.18
director.name ∪ *producer.name*	76.52	82.47	82.47	75.28
actor.nationality ∪ *producer.nationality* ∪ *director.nationality* ∪ *movie.language*	76.31	82.88	82.62	75.18
country.name ∪ *movie.release_region* ∪ *movie.language*	76.09	82.79	82.62	75.18
country.name ∪ *movie.language*	76.09	82.79	82.62	75.18
$*NATIONALITY$	76.48	82.70	/	/
$*COUNTRY$	76.30	82.61	/	/

***NATIONALITY** and ***COUNTRY** are tags coming from StanfordNLP, which were selected since they are the most accurate and relevant. While PERSON tag, which is coherent with the dataset because it identifies names of people, does not cover a big slice of people's names (it covers 103 names of over 1000 occurrences in the training set). The other tags of StanfordNLP are not very coherent with the concepts of the dataset.
F1 : it reports the F1 metric measured on the same automaton used for the first part of the project.
F1 with threshold : the results reported are obtained using the improvement of 2018, thus using in words and concepts the language model.
NER: in these experiments all the tags provided by StanfordNLP were used as generalization concepts , although the precedence was given to the generalised concepts coming from the sets listed in the "Generalization" column.

Regarding the results reported in table 3, the language model used to obtain the best F1 is "Kneser-ney" except for the results reported in the "F1" column.

The language models used are not reported for the sake of clarity in the table, however all results are available on Github.

6 Error Analysis

The results obtained using the generalization of concepts are interesting since, it is clear that this technique is very effective to tackle the problem of sparsity. In spite of that, the problem of overlapping seems to remain unsolved. This is suggested by the error analysis reported in table 4. The two generalizations considered have the highest overlapping value as reported in table 1 (director.name and producer.name) or the highest values of mis-prediction as reported in table 4 (actor.name and person.name). As can be seen in table 4, the generalization in some cases increases the confusion between classes or it does not affect the phenomenon at all.

Table 4: Overlapping error analysis

Generalization	True label	Prediction	Base	with Gen.
$actor.name \cup person.name$	Actor.name	Actor.name	136	132
		Person.name	14	16
		Director.name	4	2
	Person.name	Person.name	48	46
		Actor.name	8	10
		Director.name	8	8
$director.name \cup producer.name$	Director.name	Director.name	126	126
		Producer.name	10	10
		Person.name	6	6
	Producer.name	Producer.name	86	86
		Director.name	22	22
		O	7	7

Base: the results obtained with improvement of 2018 are considered.

with Gen: the results obtained with improvement of 2018 and generalization are considered.

7 Conclusions

The analysis at the beginning was useful to understand which classes were most afflicted by the overlapping and sparsity problems. After that, different paths to implement the needed generalization were considered. Once the generalization process was concluded, many runs with different configurations and combinations were performed, producing the results reported in table 3. An error analysis was conducted on the result obtained in order to understand the critical issues of the classifier. With these issues in mind, the classifier was further modified obtaining the final version and final results that are presented in this paper. The software developed for this purpose allows the user to easily implement customizable and more complex experiments to perform on the dataset. Furthermore, the source is written in Python and it uses the library Pywarpfst[1] for the implementation of WFSTs and FSAs.

References

[1] Openfst. http://www.openfst.org/twiki/bin/view/FST/WebHome.

[2] Stanfordnlp. https://nlp.stanford.edu/software/CRF-NER.shtml.