Language Understanding Systems Project

Maria Pia Natale (189160)

University of Trento
mariapia.natale@studenti.unitn.it

Abstract

This document explains all the work done for the Language Understanding Systems course. The aim of this project is to develop a Spoken Language Understanding module based on Movie Domain. The first section of this document is dedicated to the data analysis, then the report proceeds by dealing with the different implementations developped and ends with a comparison between the obtained results.

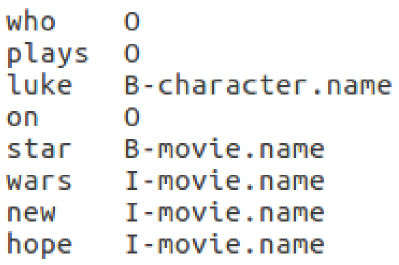
1 Introduction

This project was created with the aim of creating a Spoken Language Understanding (SLU) module for the domain of Movie. The dataset analyzed as test is composed by brief sentences, which are the input for the module developped. This must associate each word of the sentence to a predicted concept tag using the IOB notation.
The first section describes the dataset on which the SLU module has been implemented. In the Section 3 will talk about the different solution develod with the goal of improving the performance of the project, which results will be shown in the last part of the report. The tools used in this project are Python and Bash for the scripts. **OpenFST**, a tool used for the creation of the transducers and **OpenGrm** used to train the model Lanague using different smoothing method such as absolute, presmoothed, unsmoothed, katz, kneser_ney and witten_bell and different n-gram size for the n-gram model.

2 Data Analysis

The dataset provided is NL-SPARQL Data Set. The data were divided into two main sets:

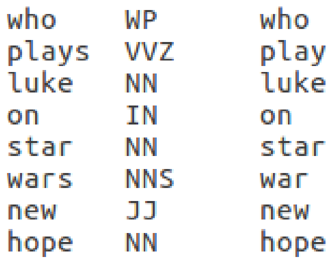
- Basic set that contains the words of the sentences associated to a IOB concept tag, as shown in the Figure 1;



```
who     O
plays   O
luke    B-character.name
on      O
star    B-movie.name
wars    I-movie.name
new     I-movie.name
hope    I-movie.name
```

Figure 1: Example of basic sentence

- Set of data with extra features that includes Lemmas and Part Of Speech tags in addition to the words. The Figure 2 shows an example of sentence.



```
who     WP      who
plays   VVZ     play
luke    NN      luke
on      IN      on
star    NN      star
wars    NNS     war
new     JJ      new
hope    NN      hope
```

Figure 2: Example of sentence with extra features

Each of these section includes both data for training and for testing. The training file is composed by 3338 sentences, for a total of 21453 words (1728 unique); while the testing set contains 7117 words (1039 unique) with 1084 different sentences. Basic and additional features sets contains the same numebr of words and sentences. Regarding the extra features, in the training file the words are associated to 1582 different lemmas and 49 Part Of Speech tags, while in the testing file there are 952 lemmas and 49 tags.
The Figure 3 shows the distribution of the fre-

quency of the words in the training file. As can be seen, the most frequent words are the words more common in the Natural Language, as *"the"*, *"of"*, *"in"*. Furthermore, it is possible to notice that we treat with Movie Domain because in the list of the most frequent words there are also context-words, like *movies*, *movie* or *directed*.

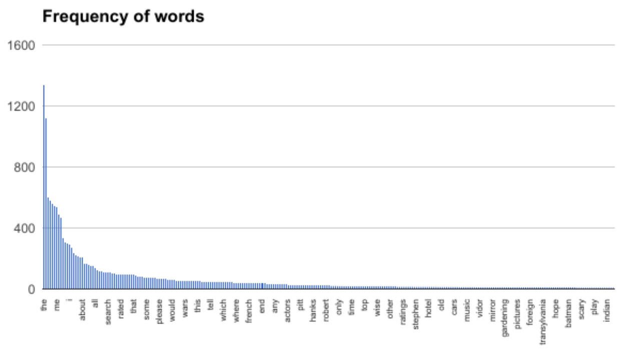


Figure 3: Frequency of words

2.1 Concept tags

The aim of the project is to associate a concept tag to each words in the dataset. The concept tags in this project follow the *Inside*, *Outside*, *Beginning* notation (IOB). In the Table 1 the frequency of each concept tag in the train and in the test file is shown. As can be seen, the most frequent tag is *movie.name*. In the table is not present the concept tag *O*, (Outside), that represents all the words that are outside of the span.

Concept	**Train**	**Test**
movie.name	3157	1030
director.name	455	156
actor.name	437	157
rating.name	240	69
producer.name	336	121
country.name	212	67
movie.language	207	72
movie.subject	247	59
person.name	280	66
movie.genre	98	37
movie.release_date	201	70
character.name	97	21
movie.location	21	11
award.ceremony	13	7
movie.release_region	10	6
movie.gross_revenue	34	20
actor.nationality	6	1
actor.type	3	2
director.nationality	2	1
movie.description	2	0
person.nationality	2	0
movie.star_rating	1	1
award.category	1	4
movie.type	0	4

Table 1: Frequency of concept tags

3 Basic Implementation

In this section will be described the different implementations developed for this project using the basic part of the dataset. For each version of transducers have been used all types of smoothing (`absolute`, `presmoothed`, `unsmoothed`, `katz`, `kneser_ney` and `witten_bell`) with different n-gram order (from 1 to 5).

3.1 Basic transducer

The first version of the project takes in consideration only the basic part of the dataset. The transducer generated is composed by a single state 0 with transitions *Word to IOB tag*.
This transducer assigns the IOB concept tag with the highest probability to each word in the test file. To compute the probability associated to the transition has been used the following formula, based on the *Bayesan's rule*:

$$P(w|t) = \frac{C(w,t)}{C(t)} \quad (1)$$

where *w* represents a word, *t* is a concept tag, *C(w, t)* is the number of times the pair *(word, IOB)* appears in the training set and *C(t)* is the frequency of that specific IOB tag. The cost of the transition of the transducer is computed as the negative logarithm of the probability of the pair *(word, IOB)*.
Unknown words have been associated to all the IOB concept tag. The probability is equal for all of them and it is given by:

$$P(<unk>|t) = \frac{1}{\#IOBs} \quad (2)$$

3.1.1 Frequency cut-off

The first implementation has the possibility to choose between using frequency cut-off or not. For the frequency cut-off, two parameters have to be used: a lowerbound and an upperbound. Once these parameters have been choosed, all the words with frequency higher than the upperbound (or lower than the lowerbound) are deleted. The probabilities and the costs of the transitions are computed as described in the previous section.

4 Additional Features

This section will show the implementation done using the additional features provided by the dataset.

4.1 Lemma to IOB tag transducer

The transducer generated in this part is similar to the transducer described in the Section 3.1. As before, this automaton has a single state but it does not take word as input, but their lemmatization. In this way, the transducer assigns to each lemma in the training file an IOB concept tag, as shown in the Figure 4.

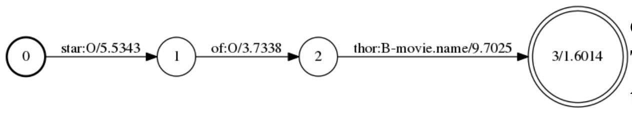


Figure 4: Example of a transducer

The proabability associated to each transition is computed using the following formula:

$$P(l|t) = \frac{C(l,t)}{C(t)} \tag{3}$$

where *l* represents a lemma, *t* is a concept tag, *C(l, t)* is the number of times the pair *(lemma, IOB)* appears in the training set and *C(t)* is the frequency of that specific IOB tag. As before, the cost is the negative logarithm of the probability of the pair *(lemma, IOB)*.

4.2 Part Of Speech tag to IOB tag transducer

This transducer takes as input a Part Of Speech tag (POS), like `NN`, `WP`, `IN`..., and maps it to an IOB concept tag obtaining automaton similar to the following:

State	State	Input	Output	Weight
0	1	NN	O	2.26389503
1	2	NNS	O	2.61333418
2	3	:	I-movie.name	7.81569195
3	4	NN	O	2.26389503
4				2.00485039

Table 2: Example of automaton POS to IOB

The probability is computed, as usual, with this formula:

$$P(p|t) = \frac{C(p,t)}{C(t)} \tag{4}$$

where *p* is the POS tag and *t* represents the IOB concept tag.

4.3 Lemma_POS to IOB tag transducer

This transducer has been created with the purpose of trying to improve the performance of the project. Here, the input for the transition is a combination of lemma and POS tag used as a single pair. These pairs are mapped to an IOB concept tag based on the training phase and on the merge of the basic file and the training with features.
As in all the transducers described till now, the Maximum Likelihood Estimation is computed by dividing the number of times that a pair *(lemma, POS)* associated to the same *IOB tag* appears in the training set with the frequency of the IOB concept tag.
The $< unk >$ pairs are associated to any concept tag with the same probability given by the formula 2.

4.4 Independent features

This experiment generates three different transducer:

- 1st transducer: *Word to Lemma*;
- 2nd transducer: *Lemma to POS*;
- 3rd transducer: *POS to IOB*.

These trasducers divide all the links between features and treat them independently.
The first transducer takes as input a word and associate it to the lemmatization version of that word. Here, the cost of the transition is 0 if a word is always transduced into the same lemma. Otherwise, the cost is given by the number of times a given pair *(word, lemma)* appears divided by the total number of times that the word appears.
The second transducer assign a POS tag to each of the lemma in the training set. The Maximum Likelihood Estimation associated to each transition is given by the following formula:

$$P(l|p) = \frac{C(l,p)}{C(p)} \tag{5}$$

where l represents a lemma, p is the POS tag associated to that lemma, the numerator is given

by the frequency of the pair *(lemma, POS)* and the denominator is the number of times that a specific POS tag appears in the training set.
The third transducer associates the POS tag to an IOB concept tag. The cost of the transition is computed by the negative logarithm of the fraction between the frequency of the pair *(POS, IOB)* and the frequency of that specific IOB concept tag.
Once these three transducers have been generated, they have been composed into a single automaton using the tool provided by **OpenFST**, `fstcompose`. The resulting transducer takes as input a word and maps it to the corresponding IOB concept tag.

5 Evaluation

The evaluation of the various implementations of the module has been done using four metrics: *Accuracy*, *Precision*, *Recall* and *F1 measure*. The best measure among these metrics is the F1, which is a combination of Precision and Accuracy.
The baseline of the project to start evaluating the performances comes from the *Word to IOB* transducer described in Section 3.1 with a unigram model and `unsmoothed` as smoothing method:

- `Accuracy: 88.84%`
- `Precision: 55.51%`
- `Recall: 60.04%`
- `F1 measure: 57.68%`

In the following tables, the performances of all the transducers will be shown based on the F1 measure.

Smoothing	**Order 1**	**Order 2**	**Order 3**	**Order 4**	**Order 5**
Absolute	57,68	76,37	75,67	76,19	76,15
Presmoothed	57,68	76,27	67,95	67,01	66,89
Unsmoothed	57,68	76,21	75,52	76	76,05
Katz	57,68	75,89	74,11	73,33	63,52
Kneser_ney	57,68	76,27	75,67	76,19	76,16
Witten_bell	57,68	76,37	75,58	76,22	76,32

Table 3: F1 measure of Word to IOB transducer

Smoothing	**Order 1**	**Order 2**	**Order 3**	**Order 4**	**Order 5**
Absolute	57,02	74,08	73,34	74,18	74,36
Presmoothed	57,02	74,14	65,28	63,24	63,55
Unsmoothed	57,02	74,08	73,21	73,71	74,03
Katz	57,02	73,48	71,07	69,42	65,53
Kneser_ney	57,02	74,14	73,55	74,21	74,32
Witten_bell	57,02	74,17	73,49	74,2	74,55

Table 4: F1 measure with frequency cut-off (upperbound 1000)

Smoothing	**Order 1**	**Order 2**	**Order 3**	**Order 4**	**Order 5**
Absolute	38,45	52,48	52,53	52,81	52,69
Presmoothed	38,45	52,36	40,8	40,07	39,84
Unsmoothed	38,45	52,36	52,44	52,62	52,7
Katz	38,45	52,36	50,14	48,21	41,94
Kneser_ney	38,45	52,36	52,12	52,58	52,32
Witten_bell	38,45	52,6	52,21	52,47	52,15

Table 5: F1 measure with frequency cut-off (lowerbound 10)

Smoothing	**Order 1**	**Order 2**	**Order 3**	**Order 4**	**Order 5**
Absolute	51,38	75,72	74,72	75,19	75,25
Presmoothed	51,38	75,68	67,22	66,38	65,76
Unsmoothed	51,38	75,62	74,7	74,86	74,94
Katz	51,38	75,4	72,98	72,16	62,69
Kneser_ney	51,38	75,59	74,85	75,35	75,52
Witten_bell	51,38	75,72	74,76	75,09	75,31

Table 6: F1 measure for Lemma to IOB tag transducer

Smoothing	**Order 1**	**Order 2**	**Order 3**	**Order 4**	**Order 5**
Absolute	1,26	14,31	17,71	20,37	21,54
Presmoothed	1,26	14,31	4,8	4,7	4,8
Unsmoothed	1,26	14,28	17,8	20,15	21,41
Katz	1,26	14,28	9,95	8,5	4,74
Kneser_ney	1,26	14,31	17,86	20,38	21,65
Witten_bell	1,26	14,31	17,81	20,36	21,63

Table 7: F1 measure for POS tag to IOB tag transducer

Smoothing	**Order 1**	**Order 2**	**Order 3**	**Order 4**	**Order 5**
Absolute	54,59	74,7	73,59	74,35	74,99
Presmoothed	54,59	74,7	65,48	65,01	64,78
Unsmoothed	54,59	74,55	73,43	74,21	74,59
Katz	54,59	74,38	73,26	71,68	62,34
Kneser_ney	54,59	74,42	73,64	74,42	74,91
Witten_bell	54,59	74,8	73,64	74,39	74,88

Table 8: F1 measure for Lemma,POS to IOB tag transducer

Smoothing	**Order 1**	**Order 2**	**Order 3**	**Order 4**	**Order 5**
Absolute	0,17	17,35	20,78	24,39	25
Presmoothed	0,17	17,34	4,41	4,6	4,7
Unsmoothed	0,17	17,34	20,66	24,27	24,51
Katz	0,17	17,34	8,85	4,77	3,09
Kneser_ney	0,17	17,34	20,79	24,39	24,73
Witten_bell	0,17	17,36	20,65	24,17	25,01

Table 9: F1 measure for independent features transducer

As can be seen from Table 3, all the parameter configurations of the Word to IOB transducer improve the performance of the baseline and have the best performance among all the simulations. The best results are given by *witten_bell* as smoothing method and bi-grams.

Tables 4 and 5 show that the two cut-off strategies do not improve the models. Moreover the performance, especially for the lowerbound cut-off, decrease by over the 20% and so it can not be taken in consideration as possible solution to handle unknown words.

Using Lemmas instead of Words to train the model do not help the performance, in fact 6 shows that Lemma to IOB transducer has worse performance compared to the Word to IOB transducer. Again bi-grams resulted in the best n-gram model, while *kneser_ney* and *witten_bell* are the best smoothing methods.

POS to IOB transducer can not be taken in consideration as a possibile solution as it has very bad performance among all the configuration as can be seen in Table 7

Using a combination of Lemma and PoS for the training phase does not give the expected results. Again, it has worse performance than Word to IOB and Lemma to IOB transducers as shown in Table 8

Lastly, Table 9 shows that using a composition of three transducers gives the worst performance among all the simulations. As expected using the data as independent features is not the right choice due to the dependecy among the features.