# Language Understanding System

## A concept Tagger for Movie Domain using Finite State Transducer

Seyed Mahed Mousavi
Master Student of Computer Science
seyedmahed.mousavi@studenti.unitn.it

## 1. INTRODUCTION

In this project we try to create a system to be able to label movie domain related words found in unlabeled sentences. The system is trained by a dataset of NL-SPARQL which consists of 41 tags covering 23 concepts in IOB notation. Then exposed to unlabeled test sentences, the system could tag 817 concepts correctly at first step which obtains the accuracy of 92.61% and **F1 score of 76.37**.

Applying a heuristic change on the system, the performance improved to 880 accurate labels resulting to accuracy of 94.17% and **F1 score of 81.86**. The tools used in this project are OpenFst[1] and OpenGrm[2]. The codes are written in Python 3.6.

## 2. DATA DESCRIPTION

### 2.1 Data Size and Notation

The data proposed for this project is divided into two sets. A big set which we train our system on that contains 3338 sentences labeled in IOB notation, and a smaller set of 1084 unlabeled sentences to assess the performance.

IOB notation (stands for Inside, Outside & Beginning) is a common way for tagging the tokens in a chunking task [3]. For each concept, B and I stand for Beginning and Inside the span respectively; and O represents Outside any spanning tag. The format of the tagging is represented in table 1 by a related example.

The concepts we have are regarding the movies, actors, locations, release date and other related features. In general we are dealing with 41 tags (including O).

Table 1: IOB Notation

| Word | Tag |
|---|---|
| Actor | O |
| As | O |
| Luke | B-character.name |
| In | O |
| Star | B-movie.name |
| Wars | I-movie.name |

### 2.2 Frequency Distribution

One of the most informative statistics in linguistics is frequency distribution. It is scientifically proven that the frequency distribution in human language follows a mathematical form known as Zipf's law. Zipf law indicates that in a text resource, there are few very-high-frequency words that account for most of the tokens and many-low-frequency words for the rest. [4]

Our dataset is not an exception. Our analysis indicates that few words like "movie, in, the, of" occur considerably often (1680 for the word movie) and more informative words such as Dory, Troy, Indiana, Hobbit or Simpsons occur less than 10 times in the whole text.

Also 8 words of Show, Me, What, Movie and The occur 4600 times. In contrast, 778 words occur only once each in the dataset. Fig 1 shows a set of word samples and their frequency.

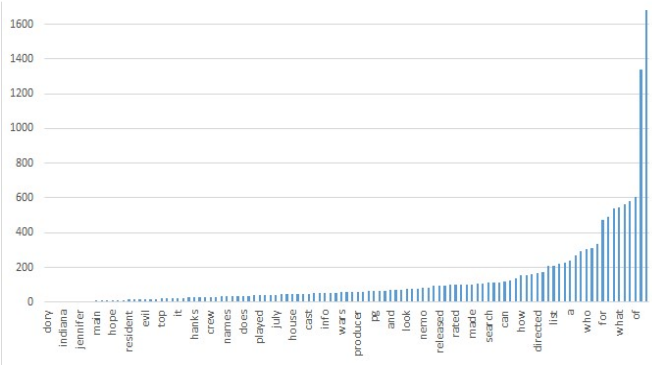


**Figure 1 Word Frequency Distribution**

In addition to our dataset words, our concept tags follow the Zipf's Law in their distribution as well. In our set of 40 tags (excluding O), the two tags of B-movie.name and I-movie.name have more than 3100 instances (1402 times for B-movie.name and 1755 times for I-movie.name), however tags I-actor.nationality, I-movie.release_region, B-movie.star_rating, B-award.category, I-actor.nationality, I-movie.release_region, B-movie.star_rating and B-award.category occurs only once.

Figure 2 illustrates the frequency distribution of the concepts in our dataset. At the first glance, one can infer that the concept tags also follow Zipf's Law in their distribution.

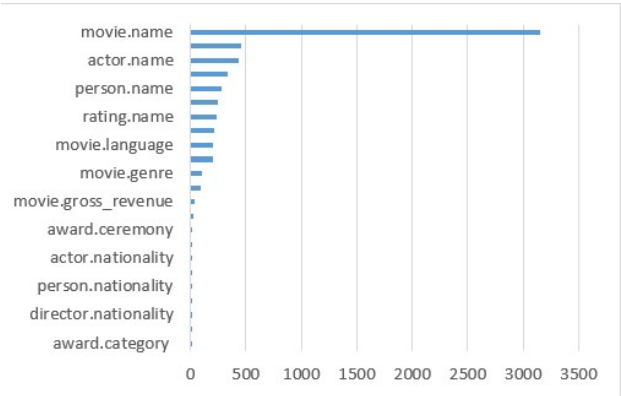


**Figure 2 Concept Frequency Distribution**

## 3. METHOD

We divide our method section to two parts of theory and implementation part. In the theory part we introduce the concept of the system and in the implementation part we describe the practical aspect of the concept.

### 3.1 Theory

We try to find the most probable tagging for the sequence. We can consider the sequence of words in a sentence as sequence of tags and try to predict the next tag with respect to the sentence structure. But using some simplifying assumptions, the tag for each word is determined according to the last word tag in the sequence. Therefore in our system, each word only depends on the previous word tag. The mathematical form of the concept is as follows.

$$t_1,\ldots,t_n = argmax \prod_i^n p(w_i \mid t_i) p(t_i \mid t_{i-1})$$

This formula indicates that the most probable sequence of tags for a sentence is computed by the most probable tag for each word as $p(w_i \mid t_i)$ i.e. the probability of observing the word $w_i$ given tag $t_i$ multiplied by the probability of the mentioned tag given the previous tag in the sequence $p(t_i \mid t_{i-1})$ .It is worth mentioning that our tags are the IOB tags.

### 3.2 Implementation

#### *3.2.1 Transducer*

The system has two main components. The first one is the implementation of $p(w_i \mid t_i)$ which is done using a transducer for each word given tag. The term is calculated as $p(w_i \mid t_i) = \frac{c(wi,ti)}{c(ti)}$ indicating the frequency of the word tag couple divided by the frequency of the tag. The negative log of the probability value for each word tag couple is then proposed as the cost of related arc in transducer from state 0 to 0 for that word and tag and is considered as the cost for the related transition. Figure 3 is an

example of the transition table, and figure 4 illustrates an abstract version of the transducer.

```
0  0  on    O             5.15
0  0  star  B-movie.name  3.22
0  0  wars  I-movie.name  3.44
0  0  new   I-movie.name  5.39
```

**Figure 3 Transition Table**

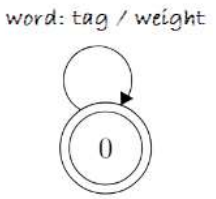


**Figure 4 Concept Machine**

Using the OpenGrm library we created a lexicon of the file and then using the OpenFst library we created the machine.

### *3.2.2 Language Model*

The next phase is related to the second part of the probability formula, the $p(t_i|t_{i-1})$ which we obtained it by creating a language model on the tags. So that based on the tags only, we could have a **language model** to obtain $p(t_i|t_{i-1})$. But regarding that, we have two parameters to take care of.

One is the **ngram** size and the other is the smoothing method. An n-gram is a type of probabilistic language model for predicting the next item in such a sequence. A simple sequence of tag in our system can be described as follows:

**Sentence:** O O B-movie.name I-movie.name

**Unigram**: O, O, B-movie.name, I-movie.name

**Bigram:** O O, O B-movie.name, B-movie.name I-movie.name

**Smoothing** is used to balance weight between infrequent and frequent grams and items not seen in the training data in order not to assign 0.0 to it [5]. According to help description of NGramMake command in OpenGrm community, it has six smoothing methods already implemented.

**Witten_bell:** smooths using Witten-Bell (Witten and Bell, 1991), with a hyperparameter k, as presented in Carpenter (2005).

**Absolute:** smooths based on Absolute Discounting (Ney, Essen and Kneser, 1994), using bins and discount parameters.

**Katz:** smooths based on Katz Backoff (Katz, 1987), using bins parameters.

**Kneser_ney**: smooths based on Kneser-Ney (Kneser and Ney, 1995), a variant of Absolute Discounting.

**Presmoothed:** normalizes at each state based on the n-gram count of the history.

**Unsmoothed:** normalizes the model but provides no smoothing.

Another matter we needed to take care was **Out Of Vocabulary (OOV**) words; the words which are not present in the training dataset but may be observed in the test data set. For that, we dedicated a tag of <unk> for these unknown words with the weight as the negative log of the possibility to be any of the tags.

## 4. RESULTS & ERROR ANALYSIS

### 4.1 Baseline

In order to be able to assess the system performance by comparison, we tried to tag words in the test set with a completely different approach than language models and related concepts. We tried to tag words in 3 different manners:

**Random:** Choose a tag randomly from the tag set and assign to the word.

**Probability Dominance:** The tag choice is influenced by its probability and contribution in whole.

**Majority:** Each word is assigned the major tag of the set which in our data set is ‘I-movie.name’

The result of this assignment is illustrated in table 2.

**Table 2: Baseline**

| Method | Acc. | Precision | Recall | F1 |
|---|---|---|---|---|
| Random | 2.35% | 0.40% | 2.47% | 0.68 |
| Prob.Dom. | 0.53% | 0.01% | 0.02% | 0.01 |
| Majority | 7.83% | 0.46% | 0.46% | 0.46 |

A quick glance at the table indicates that the system and the concept beyond it will succeed in winning over these three method even with badly tuned parameters.

## 4.2 First Results

Having the transducer, we tried various combination of ngrams and smoothing methods to obtain the best result regarding the F1 score. We composed each sentence of test set to the transducer and then the language model, eliminated the epsilon transitions, sorted the arcs and obtained the shortest path, **(fstcompose, fstrmepsilon, fsttopsort and fstshortestpath respectively)**, we obtained the set of predictions which by evaluating them, we obtained the results shown in Table 3.

**Table 3: First Results Based on F1 Score**

| | Smoothing Method | | |
|---|---|---|---|
| Ngram Size | Absolute | Witten_bell | Kneser_ney |
| 2 | 76.37 | 76.37 | 76.27 |
| 3 | 75.67 | 75.58 | 75.67 |
| 4 | 76.19 | 76.19 | 76.22 |
| 5 | 76.15 | 76.32 | 76.19 |

## 4.3 Error Analysis

The first results are interesting, but not satisfying. Therefore we tried to improve it. Since the Smoothing and Ngram size were obtaining the best result possible, this time we looked at the data once again. We realized that vast majority of tags are 'O' as they are **72%** of the tags (15,391 over 21,453).

Our hypothesis is that since the high frequency of 'O's, probability of observing an 'O' given an 'O' as the previous tag is high so predicting an 'O' mostly wins over all other possibilities. In this case, we substituted all 'O's in word tag couple to the word itself with an underline as prefix. An example could be 'who O' which changed to '`who _who`'.

Expectedly, this substitution causes major changes in all log probabilities considered as costs in our transducer.

Also, observing the results, we can see the system performance on some specific concepts are not as good as other tags. One of the examples for this concept is **movie.release_region** which the system performs poorly on it. Our hypothesis is that since the country names can be both considered as **movie.release_region** and **B-country.name**, our system is tricked and cannot distinguish the two. Therefore it is misguided in predicting the tag.

Another example is **director.nationality**. Considering that there is only two words for this concept (German, Turkish) besides the fact that these two can be representing the movie language as well related to **movie.language**, the system is misguided again in predicting them properly, predicting one for another.

Moreover, having less than 5 examples for **award.category**, **movie.type** and **movie.star_rating** concepts over the 21453 tokens has made it hard for the system to learn these concepts in order to tag them correctly.

At the time of writing this report, the writer does not have any reasonable idea why the system is not performing as expected on tags **gross_revenue** and **movie.release_date**.

## 4.4 Final Results

Once again we repeated the procedure of composition but this time with the new transducer. The result proved our hypothesis and we observed a huge increase in accuracy and F1 score. The best result was obtained by using **4gram size** and **Kneser_ney** smoothing method for the language model as shown in table 4.

**Table 4: Final Result**

| | |
|---|---|
| **Found:** | **1059 Phrases** |
| **Correct:** | **880** |
| **Accuracy:** | **94.17%** |
| **Precision:** | **83.10%** |
| **Recall:** | **80.66%** |
| **Fb1:** | **81.86** |

In the end we were able to predict the tag for 880 words correctly which infers the system is performing practically well.

## 5. REFERENCES


[1] OpenFst Library: http://www.openfst.org

[2] OpenGrm Library: http://www.opengrm.org

[3] EVALITA: http://www.evalita.it/2009/tasks/entity

[4] Zipf's word frequency law in natural language: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4176592/

[5] Wikipedia Article about Ngram https://en.wikipedia.org/wiki/N-gram