

Language Understanding System

A concept Tagger for Movie Domain

Seyed Mahed Mousavi 196272

I. 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 consist 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 75.75**.

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 Python 3.6.

II. Data Description:

A. Data size & 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 is 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.

Table 1: IOB Notation

<u>Word</u>	<u>Tag</u>
Actor	O
As	O
Luke	B-character.name
In	O
Star	B-movie.name
Wars	I-movie.name

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).

B. Frequency distribution:

One of the most informative statistic in linguistics is frequency distribution. It is scientifically proven that the frequency distribution in the human language follow 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 most of the tokens and many-low-frequency words for the rest. [4]

Word	Frequency (approx.)
dory	10
indiana	10
jennifer	10
man	10
hope	10
resident	10
evil	10
top	10
it	10
hank's	10
crew	10
names	10
does	10
played	10
july	10
house	10
cast	10
info	10
war's	10
producer	10
pg	10
and	10
look	10
memo	10
released	10
rated	10
made	10
search	10
can	10
how	10
directed	10
liz	10
a	10
who	10
for	10
what	10
of	1600

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 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.

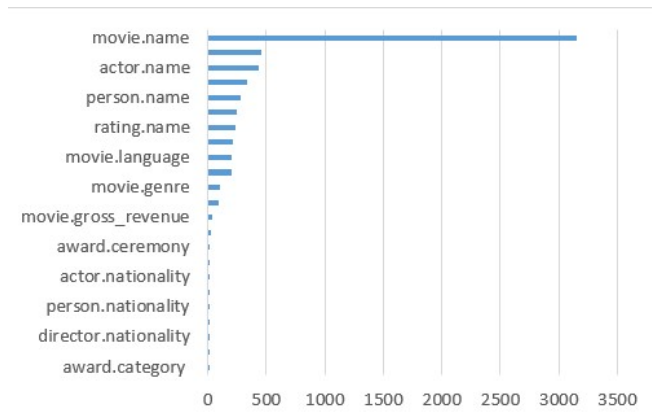


Figure 2: Concept Frequency Distribution

We divide our method to two parts of concept 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.

In theory, we try to find the most probable tagging for the sequence. But using some simplifying assumption, 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.

$p(w_i | t_i)$: the probability of observing the word w_i given tag t_i .
 $p(t_i | t_{i-1})$: the probability of observing tag t_i given the tag t_{i-1} . It is worth mentioning that our tags are the IOB tags.

B. Implementation:

1. Transducer

The system has two main components. The first one is the implementation of $p(w_i | t_i)$ which is done using a transducer for each word given tag. The term is calculated as $p(w_i | t_i) = \frac{c(w_i, t_i)}{c(t_i)}$ 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 as an example of the transition table, and figure 4 illustrates an abstract version of the transducer.

0	0	on	0	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

word: tag / weight

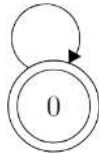


Figure 4 Concept Machine

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

2. Language Model, Ngram Size, Smoothing And OOV Words

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.

For our system, after trying various combination of ngrams and smoothing methods, we found out that 4-gram size with **Kneser_ney** smoothing method will provide us the best result.

IV. Results & Error Analysis

a) 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.

b) First Results

Having the transducer, and the Language model of 4gram size and Kneser_ney smoothing method, we composed each sentence of test set to the transducer and then the language model, eliminating the epsilon transitions, sorting the arc and obtaining the shortest path, (**fstcompose**, **fstrmepsilon**, **fsttopsort** and **fstshortestpath** respectively), we obtained the set of prediction. Evaluating them we obtain the results as follows:

Found:	1066 Phrases
Correct:	817
Accuracy:	92.61%
Precision:	76.64%
Recall:	74.89%
Fb1:	75.75

c) Error Analysis

The first results are interesting, but not satisfying. Therefore we tried to improve it. But 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', probability of observing an 'O' given 'O' is high so predicting an 'O' mostly wins over all other possibilities. In this case, we substitute all 'O's in word tag couple with a word with an underline as prefix. As example could be 'who O' which changed to 'who _who'.

Expectedly, this substitution cause major change 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 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, it 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 way system is not performing as expected on tags **gross_revenue** and **movie.release_date**.

d) Final Result

Once again we repeated the procedure of composition this time with the new transducer. The result proved our hypothesis and we observed a huge increase in accuracy and F1 score as follows:

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 well practically.

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] <https://en.wikipedia.org/wiki/N-gram>