Language Understanding Systems - Mid-Term Project: FST and GRM Tools for SLU

Seyyed Arya Hassanli

[1]University of Trento

seyyedarya.hassanli@studenti.unitn.it

Abstract. *This report is provided to present the results of the first project of Language Understanding Systems course. The main goal of this project is to develop a pipeline as a concept tagger module for NL-SPARQL Dataset in Movie Domain.*

1 Introduction

In this project, it is tried to develop a concept tagger pipeline for NL-SPARQL dataset. First, the dataset is analyzed to have a general overview of it. Then a pipeline is created using a Hidden Markov Model. The final step is to evaluate the model with different parameters to achieve the best possible performance. For this reason, different cut-off frequencies for input words, and output tags, different N-gram sizes and all options of N-gram smoothing are considered.

2 Data Analysis

NL2SparQL4NLU is the main dataset in this project. It provides sentences in different formats in Movie Domain. The dataset is split in Train and Test. For both Train and Test subsets, Four files are provided: Two files for an utterance-per-line formatted datasets and its labels, one file that represents the utterances on CoNLL format and one other file to provide POS-tags and Lemmas.

The training subset consists of 3338 utterances and the test subset provides around one third of the training one. The detailed statistics of the dataset is given on Table 1.

Table 1: Dataset Size

	Utterances	Tokens
Train	3338	21453
Test	1084	7117

The Train subset includes 1729 words and the Test subset is made of 1039 words. More than 6% of the tokens in Train set are the word "the" which is the most frequent one. The next most frequent words of Train set are "movies", "of", "in", and

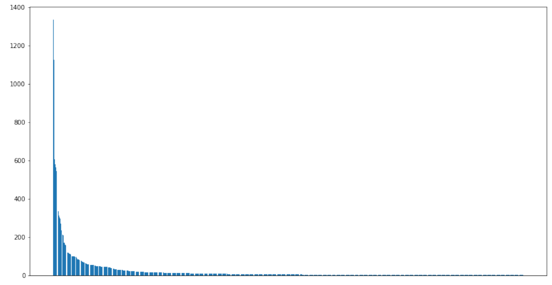

Figure 1: Distribution of Words

"movie" with respectively 1126, 607, 582, and 564 occurrences. The distribution of words in the train set is presented in Figure 1.

In order to manage out-of-vocabulary words, a lexicon cut-off is applied with different cut-off frequencies to find out the best performance. For example, by applying a min-cutoff-frequency of 2, the size of the lexicon dropped to 950 which is 45% less than the original lexicon. So, 45% of the words in Train set occurred only one time.

In CoNLL file, the number of concepts and labels are as follows:

Table 2: Number of Concepts and Tags

	Unique Concepts	Unique Labels
Train	23	41
Test	22	39

Among these 23 concepts in train set, the most frequent one is "movie.name" with 3157 occurrences which is 52% of all concepts available in training subset. The list of all concepts and their occurrences in training set are as follows:

When it comes to labels, most of the tokens are labeled as "O". To be exact, 71.74% of tokens are assigned the label "O". The most frequent non-O label is "I-movie.name" that holds a share of 8.18% of all labels. The distribution plot of concepts is presented in Figure 2.

In the training dataset, each utterance has at most 4 entities and there are utterances without any entity. On average, each one consists of 0.97 entity.

Table 3: Distribution of Concepts

Concept	Count
movie.name	3157
director.name	455
actor.name	437
producer.name	336
person.name	280
movie.subject	247
rating.name	240
country.name	212
movie.language	207
movie.release_date	201
movie.genre	98
character.name	97
movie.gross_revenue	34
movie.location	21
award.ceremony	13
movie.release_region	10
actor.nationality	6
actor.type	3
person.nationality	2
director.nationality	2
movie.description	2
movie.star_rating	1
award.category	1

The length of each entity varies between 1 and 9 tokens. On average, "movie.gross_revenue" is the longest one, with 4.25 token per entity length.

3 Concept Tagging

One common task in Natural Language Understanding is entity extraction or concept tagging. In this task, each sentence should be divided into segments of tokens; Then each segment should be tagged by a label (concept). In this project, the first goal is to create a concept tagging pipeline using Hidden Markov Models. In concept tagging module, it is needed to find the best tag sequence, given a word sequence:

$$t_1^n = \arg\max_{t_1^n} p(t_1^n | w_1^n)$$

By applying the Bayes's rule, the above equation can be written as:

$$t_1^n = \arg\max_{t_1^n} p(w_1^n | t_1^n) p(t_1^n)$$

The first part of the argmax, $p(w_1^n|t_1^n)$ relates to the emission probabilities and the second part, $p(t_1^n)$ would be the transition probabilities.

A common pipeline of a concept tagger is consist of three main components:

3.1 λ_W

The first part of the pipeline is the FSA representation of the observations, or simply the utterances of the dataset. In the code, first the dataset passes through sentence tagging and OOV handling and then a FAR archive is created out of the dataset. Different FAR archives are stored for different cut-off frequencies that applied to the dataset, e.g. trn_(mincut_3).far includes the train FSAs with minimum cutoff frequency of 3.

3.2 λ_{W2T}

λ_{W2T} represents the emission probabilities. By assuming that the probability of a word only depends on the related tag, it is possible to rewrite:

$$p(w_1^n|t_1^n) \approx p(w_1|t_1)p(w_2|t_2)...p(w_n|t_n)$$

Therefore, a file of tokens and their related tag is created to calculate the bi-gram probabilities of it. In this file, each utterance has two tokens. The second one is the word and the first one is its tag.

3.3 λ_{LM}

The last part of the pipeline would be an N-gram model of output tags to provide the transition probabilities. The transition probabilities are the probability of a tag given a sequence of tags observed before:

$$p(t_i|t_{i-n+1}^{i-1})$$

So, an N-gram model can represent this probability. In this project, N-gram models of order 1 to 6 are evaluated to assess the effect of N-gram size

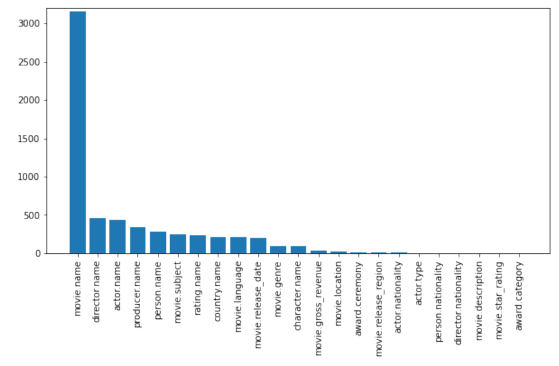

Figure 2: Distribution of Concepts

on the performance. Moreover, different smoothing methods have been used for smoothing the N-gram; ”Absolute”, ”Katz”, ”Kneser-Ney” ”Pre-Smoothed” ”Witten-Bell”. Using all combinations of these parameters, an N-gram model is created for being able to evaluate the parameters, e.g. t_(mincut_3)_(order_2)_(katz).lm is the N-gram model of output tags with cutoff frequency of 3, order of 2 and smoothed by Katz algorithm.

4 Evaluation

In this section, first, a baseline configuration is defined, and the pipeline is tested on the CoNLL test subset. Then, other parameters are tested with different values to achieve the best performance. At the very beginning of the project, the runtime of each evaluation assumed to be much lesser than what it was. Therefore, as it was possible for parameters to affect each other, for better evaluation, all combinations of cut-off frequencies and n-gram sizes were taken into account to create language models. So, that it was possible to calculate the performance for all combinations. However, at last the N-gram size and N-gram smoothing method were tuned dedicated and the input and output cut-off frequencies were tested in combination.

4.1 Baseline

To have a baseline performance, the pipeline is tested using the following parameters:

- Input Token Cut-Off: 2
- Output Tag Cut-Off: 2
- N-gram Size: 1
- N-gram Smoothing: No Smoothing (”unsmoothed”)

The whole pipeline tested over the test subset CoNLL file and evaluated using the given python CoNLL library during the lab sessions. Among the concepts, the pipeline performed on ”actor.type” without any error. In total, the *F1* of baseline is 56.1%, the *Precision* is 54.9%, and the *Recall* is 57.3%.

4.2 N-gram Size

The first parameter to tune is N-gram size of the transmission probabilities language models. The models have been generated with orders from 1 to 6. All models were tested with the same configuration as baseline, except for N-gram order. Among the tested N-gram orders, the best F1 and Precision achieved by using an N-gram of order 2. However, the Recall is slightly better using N=6. In the case of N=2, the *F1* equals 72.51%, the *Recall* is 72.41%, and *Precision* is 72.61%. The detailed results of evaluations are given in Table 4 and the chart of *F1* is provided in Figure 3.

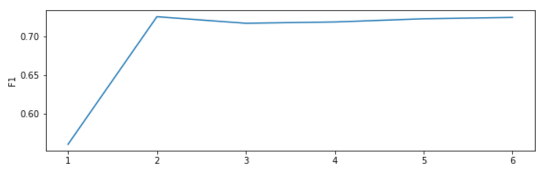

Figure 3: *F1*, Different N-gram Orders

Table 4: Evaluation of Different N-gram Orders

	N=1	N=2	N=3
Recall	57.28%	72.41%	71.86%
Precision	54.92%	72.61%	71.46%
F1	56.07%	72.51%	71.66%
	N=4	**N=5**	**N=6**
Recall	72.22%	72.04%	72.68%
Precision	71.44%	72.44%	72.15%
F1	71.83%	72.24%	72.42%

During the following tests, the N will be fixed to 2.

4.3 Smoothing Method

Using the best N-gram order found, which is 2, it is tried to find the best N-gram smoothing method among ”witten_bell”, ”absolute”, ”katz”, ”kneser_ney”, ”presmoothed”, ”unsmoothed”. In following evaluation, the input and output cutoff frequency will be fixed to 2, same as the baseline configuration. Both ”witten_bell” and ”absolute” smoothing methods preform the same, with *F1* of 72.66%, *Recall* of 72.59%, and *Precision* of 72.72%. Therefore, there is no difference between them in our point of view. The detailed performance of different are given in Table 5 and shown in Figure 4.

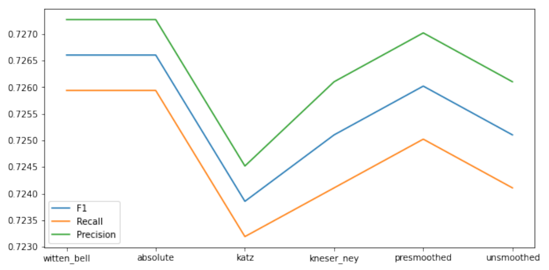

Figure 4: *F1*, Different N-gram Smoothing

Table 5: Evaluation of Different Smoothing

	witten_bell	absolute	katz
R	72.59%	72.59%	72.31%
P	72.72%	72.72%	72.45%
F1	72.66%	72.66%	72.38%
	kneser_ney	**pres...ed**	**uns...ed**
R	72.41%	72.50%	72.41%
P	72.61%	72.70%	72.61%
F1	72.51%	72.60%	72.51%

The "witten_bell" will be used as the smoothing method from now on.

4.4 Cut-Off Frequencies

Now that two parameters, N-gram order and smoothing method, are fixed, it is possible to evaluate the Tokens and Tags cut-off frequencies together. The minimum cut-off frequencies between 2 and 6 are evaluated. The values of *F1*s are shown in Figure 5.

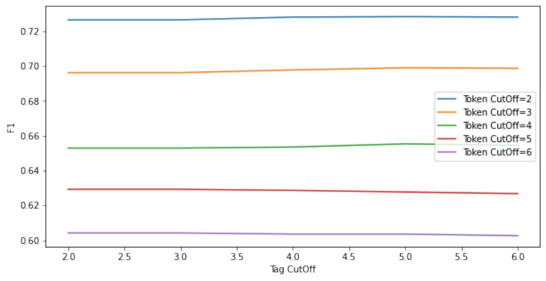

Figure 5: *F1*, Token and Tag Cut-Offs

According to the Figure 5, it is obvious that with Token Cutoff threshold equal to 2 we can achieve the best F1. Now for better comparison, let's focus on the Token Cutoff equal to 2. The Figure 6 illustrates that the *F1* has the highest value if Tag cut-off frequency equals 5.

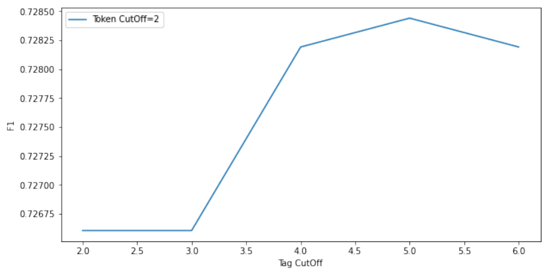

Figure 6: *F1*, Tag Cut-Off Frequencies

5 Conclusion

As presented in Table 6 the best achieved performance is *F1* of 72.84%. This performance is achieved by the following configuration:

- Input Token Cut-Off: 2
- Output Tag Cut-Off: 5
- N-gram Size: 2
- N-gram Smoothing: Witten-Bell

Table 6: Best Performance Achieved

	Final Performance
F1	72.84%
Recall	72.77%
Precision	72.91%