Language Understanding Systems
First midterm report

Williams Rizzi
Master in Computer Science
University of Trento
williams.rizzi@studenti.unitn.it

Abstract

Language Understanding Systems (LUS) intended as the set of technologies can supports analysis of data in dialog or text format. This systems between many other capabilities are able to keep track of what is described into a sequence of words. Those sequences of pairs concept word can be represented statistically by math distributions if given a fair amount of data to learn from. Is in fact possible to set up a bottom up machinery that running inference over the dataset learns the most frequent structures and patterns of tuples $(word, concept)$. The final purpose of this system's analysis is to enable machines better comprehend what a sentence truly means. Giving machines access to the most common format of expressions used by humans, sentences. And, viceversa ease the interaction of humans with their computers.

1 Introduction

A large part of the work in building statistical models from data is about the pre-processing of the data itself and the fine tuning of the algorithm parameters. Data, being the beginning and the main driver of the entire system creation, needs to synthesize, as far as possible, all the variants that the system might encounter. Nevertheless, occupying as low space as possible and being robust towards variations that might happen over time. On the other side, for what regards the model building and the tuning of the respective parameters. The parameter represent the adjustments that can be done during the construction of the model following a specific algorithm. Those might seem simple to grasp and easy to fix but the reality is that the parameters make a very big difference and usually are extremely difficult to tune effectively. In the following sections is described the dataset statistical peculiarities, is then depicted the approach followed, are reported and discussed the scored results and, finally are drawn the conclusions discussing possible further approaches.

2 Dataset

This section wants to tackle the dataset composition and its peculiarities. The dataset used to train the model comes from the movie domain. The corpora is already divided in train and test set. The train set is composed of 3338 sentences whereas the test set is composed of 1048 sentences. Each sentence is of an average length of 7 words with a standard deviation of 3 words. This dataset is presented in the form of a list of sentences and, for each word in each sentence is given the respective Part of Speech(PoS) (Martinez, 2012), Lemma (Gross, 1998) and IOB-tag (Tjong Kim Sang and De Meulder, 2003). In the rest of the work the words word and token will be used interchangeably.

2.1 IOB-tags

The concepts in the dataset follow the CoNLL IOB notation (Tjong Kim Sang and De Meulder, 2003) in which each concept is classified as the part of the multi-word span it belongs to, as begin (B), in (I) or out (O) of the concept span. The words $concept$ and IOB_tag will be used from now on interchangeably.

When training statistical a machinery like the one described in this work the concept distributions gather a lot of importance. In particular w.r.t. how the concept distribution stand out between each other and how similar are the data distributions between the train and the test set. both

those points will be tackled by the following subsections.

2.2 Balance of Concepts Distribution

Considering the Figure 1, where is plotted the comparison between the most common positions in which is possible to find a concept inside a sentence. The figure shows that the amount of $movie.name$ concepts is quite high compared to its most numerous counterparts, respectively $director.name$, $actor.name$ and $producer.name$. Even if the situation is not ideal and the cut is not very clear between the three distributions, there is some spikes in the figure that can tell us some interesting insights, one instance of this is represented by the $producer.name$ that tend to be more concentrated in in the sixth token of the sentence, where as the $actor.name$ is more concentrated in the fourth one and the $director.name$ in the fifth with a significant drop between the fifth and the sixth token.

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

Table 1: Comparing between the concept occurrences in the train and test set

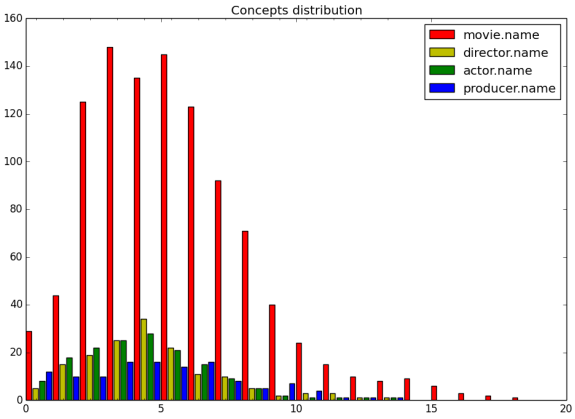

Figure 1: The figure shows the label distribution over the position in the sentence. By the ordered is represented the number of occurrences. And the abscissa represents the token position.

2.3 Previously Unseen Concepts

Building a model with finite sets of data forces to select the amount of information fed to the building algorithm. Being this set of combinations limited leaves space for unseen combinations of words and concepts. Comparing the train and test sets there is a concept, $movie.type$, that is not covered by the train set, but there is other situations like the one of the tag $movie.release\_region$ that occur so few times that even if they have been seen enough times into the train set there is only 4 sentences in the test set that contain the concept. Viceversa, the tag $character.name$ shows very few occurrences, 21, in the test set if compared with the 97 in the train set. Furthermore, there are 4 evidences of $actor.type$ in the train set and 2 in the test set. These three latter situations are not always easy to be found but can manage to lower the overall performances.

In the next section is discussed the developed approaches and the solutions designed to overcome the discussed situations.

3 Approach

This work features two approaches to the proposed situation. The first approach is quite basic and exploits the encountered sentences and their label distributions. Whereas the second approach handles the text keeping track not only of the words and labels but also of the Parts of Speech (PoS) and the Lemmas. Is interesting to notice that the second approach even if more conscious of what is actually the context of the tokens is handling is

not always better performing. A fine tune of the parameters is necessary to have evidences of the superiority of one approach or another. A more sophisticated algorithm is in fact not necessarily better performing since sophistication often implies more parameters to tune and more awareness needed to positively answer to the situations encountered by the model. The structure underlying the two models is common and, is composed by four steps, (i) loading, (ii) tuple computation, (iii) likelihoods estimation and, (iv) finite state transducer building (FST) (Mohri et al., 2000). As mentioned in Section 2 the dataset is already splitted in train and test and for both are provided the data in the form of lists of tuples $< word, IOB\_tag, PoS, Lemma >$. The main difference of the implemented approaches is the tuple computation, in the following subsections are described the differences in the designed approaches.

3.1 Baseline Approach

This approach exploits the aforementioned tuples in order to create the $word\_to\_concept$ as the tuple $< word, IOB\_tag >$. This approach is quite simple and is used as a baseline and proof of concept of the correlation between concept patterns and words used. In the next subsection is depicted a development of this approach specifically addressing the un-even data distribution.

3.2 Threshold Approach

To address the data distribution this approach is taking two actions, it filters the train set to discriminate noisy information and, generalise the form of the words with the Lemma in the $word\_to\_concept$ ending up with a tuple $< Lemma, IOB\_tag >$. To perform the train filtering is used a threshold set empirically to discriminate if either a tuple occurs enough times, or not, to be informative during the training. Whereas the Lemma to substitute to the word in the $word\_to\_concept$ is already provided in the train set. The rationale behind this two choices is motivated in the Section 2.2 and Section 2.3. The low presence of some concepts makes them not important enough to be learned properly, but still enough to confuse the model lowering performances. Whereas the decision to use the lemma instead of the word to label the sentences is due to the capability of the lemma to lower the entropy in terms of words that occour to have the same meaning but with different conjugations. And, in this way augment the number of the example with same $word\_to\_concept$ tuple.

3.3 Likelihood estimation and build of FST

After the tuples are created, both the approaches share the same following implementation where are computed the likelihoods and then built the FSTs. Computing the likelihood starts from counting the occurrences of the $word\_to\_concept$ and IOB_tags. Those count will then be used to compute the likelihoods. As mentioned in Section 2, when building the model is mandatory to bear in mind that the train set is partial w.r.t. what words and concept a model might encounter during the test. To answer this situation the $word\_to\_concept$ is integrated with a set of tuples with the $< unk >$ token and every known IOB_tag in order to allow the model to parse also unknown tokens. With the previously computed counts, namely the IOB_tag count, the $word\_to\_concept$ count and, the dropped $word\_to\_concept$ count, are computed the likelihoods.

$$P(concept|token) = \frac{P(token, concept)}{P(concept)} \quad (1)$$

Formulae 1 shows that if the $word\_to\_concept$ is composed by a known word then the likelihood is equal to the amount of occurrences of the tuple $< word, IOB\_tag >$ divided by the amount of words associated to the given IOB_tag.

$$P(concept| < unk >) = \frac{P(drop\_tuple)}{P(concept)} \quad (2)$$

Otherwise, Formulae 2 shows that if the word is not part of the known dictionary then the likelihood is equal to the amount of tuples dropped by the previously mentioned threshold divided by the amount of words associated to the given IOB_tag.

The final FST built needs to be able to keep track both of the fact that a given word is most likely associated to a given concept and of the fact that the aforementioned likelihood vary slightly depending on the context in which is placed. To achieve this two FST are created and then composed. The first FST models the likelihoods associated to the single $word\_to\_concept$ tuple. The second FST models the likelihood of sequences of words. To keep track of sequences the approach has been to model the text in small units, called N-grams. The size of

the N-grams is decided as a parameter. The likelihood of each N-gram is modeled as the Bayes Posterior Probability (Walters and Ludwig, 1994).

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \quad (3)$$

As shown in Formulae 3, where D is representing the N-gram and θ is the next word composing our sequence.

In the next section is reported the experimental results of the variants of the algorithm.

4 Results

The following results are computed using the CoNLL evaluation script, which scores the predictions in terms of accuracy, precision, recall and F1. The techniques tested to smooth the model are: (i) unsmoothed (unsm), (ii) presmoothed (prsm), (iii) absolute (abs), (iv) kneser ney (kn), (v) witten bell (wb) and, (vi) katz. In the tables is reported the accuracy and F1 of each configuration. In Table 2, is reported the results of the baseline algorithm, showing that with very little tuning the results are very interesting. Whereas in Table 3,

N-Gram	Smoothing	Acc	F1
5	unsm	91.04%	72.65%
5	wb	91.97%	72.65%
4	kn	91.58%	71.46%
4	abs	91.96%	72.73%
2	prsm	91.60%	71.66%
2	katz	91.57%	71.36%

Table 2: This table reports the results from the baseline approach in Section 3.1. In the given approach is possible to set only the N-Gram parameter and the Smoothing method used by the FST generation algorithm.

are reported the results from the enhanced tuple generator that ends up performing similarly tending to fit better on some concept instead that on some others.

In the next section will be further discussed the motivations behind the performances of the two approaches.

5 Discussion

This section will better define the limits of the described approaches investigating into the reasons of their performances. The proposed approaches

L-H	N-Gram	Smoothing	Acc	F1
0-5	6	unsm	92.13%	71.87%
0-5	4	wb	91.78%	70.87%
0-5	6	kn	92.07%	71.15%
0-5	6	abs	92.16%	72.15%
4-5	2	prsm	91.18%	70.04%
4-5	2	katz	91.15%	69.76%

Table 3: This table reports the results from the threshold approach in Section 3.2. The given approach allows to set the previous two parameters, N-gram and smoothing method, and also the two threshold discriminating the tuples occourring not enough times to be considered by the building procedure the settings are here reported as Low(L) and High(H) threshold.

show the feasibility of modeling sequences of text using statistical distributions, some of most common issues in dealing with them and, how to take effective counter measures to deal with them. Is interesting noticing as lowering the amount of concepts entropy slightly increase the quality of the performance. Is still difficult for the model to correctly figure out some challenging associations, for instance in the situation in which a person name in some cases, depending from the context, is both for instance an actor and a director. Is left as future work to try to model a similar solution but with many model competing to label a concept in a many versus one manner.

In the next section are discussed the possible further developments of the proposed approach.

6 Conclusions and Further Developments

Overall the results where satisfying, but the amount of OOV tags is significantly lowering the results of the algorithm. Regarding this situation might be interesting taking into consideration designing an enforcer for the IOB-tagger for instance exploiting WordNet [1]. A better tagger might significantly improve the amount of useful information given to the building procedures allowing a better model to be built. Nonetheless, the hyperparameters used to run the algorithms were tuned by hand, and therefore not necessarily optimally, might be an interesting further development automatically tuning parameters with Grid

[1] Princeton University "About WordNet.". Princeton University. 2010. https://wordnet.princeton.edu/

Search and Random Search (Bergstra and Bengio, 2012) or with Genetic Algorithms (Anderson-Cook, 2005).

References

Christine M Anderson-Cook. 2005. Practical genetic algorithms.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13(Feb):281–305.

Maurice Gross. 1998. Lemmatization of compound tenses in english. *Lingvisticae Investigationes* 22(1):71–122.

Angel R Martinez. 2012. Part-of-speech tagging. *Wiley Interdisciplinary Reviews: Computational Statistics* 4(1):107–113.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science* 231(1):17–32.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pages 142–147.

Carl Walters and Donald Ludwig. 1994. Calculation of bayes posterior probability distributions for key population parameters. *Canadian Journal of Fisheries and Aquatic Sciences* 51(3):713–722.