Concept tagging module for movie domain

Michela Lorandi - 207522
michela.lorandi@studenti.unitn.it

Abstract

This report has the goal to describe the project Concept tagging module for movie domain for the course Language Understanding Systems.

1 Introduction

The goal of the project is to develop a concept tagging module for the movie domain. The module has the goal to, given a sentence, find the IOB tags associated to the sentence. An example can be found in Table 1. The concept tagging module uses the concept tags in IOB format and the dataset used is the NL-SPARQL[1]. The module has been implemented in three different versions: the baseline, the first improvement and the second improvement. The code of the project can be found at https://github.com/michiL96/concept_tagging_module.

2 Data analysis

The NL2SparQL4NLU dataset is divided in train and test. The train dataset contains 3338 utterances for a total of 21453 words, while the test dataset contains 1084 utterances for a total of 7117 words.

The entities identified by the Spacy's NER module in the training dataset are: PERSON (1432 words), NORP (187 words), FAC (17 words), ORG (552 words), GPE (187 words), LOC (15 words), PRODUCT (15 words), EVENT (20 words), WORK_OF_ART (90 words), LAW (10 words), LANGUAGE (43 words), DATE (197 words), TIME (48 words), PERCENT (0 words), MONEY (4 words), QUANTITY (19 words), ORDINAL (8 words), CARDINAL (133 words). The three most identified entities are:

1. PERSON, 6.7% of total words;
2. ORG, 2.6% of total words;
3. DATE, 0.9% of total words.

	Sentence
Base	how many oscars has meryl streep won
IOB	O O B-award.ceremony O B-person.name I-person.name O

Table 1: An example of sentence with its IOB tags.

The NORP entity represents nationalities, religious or political groups. The FAC entity represents buildings, airports and other faculties. The ORG entity represents companies, agencies, institutions and organizations. The GPE entity represents countries, cities and states. The LOC entity represents non-GPE locations, like mountains.

3 Method description

In this section, the three versions implemented for the concept tagging module are presented. The first one is the baseline version, followed by the first improvement and the second improvement. All the versions are trained using the NL2SparQL4NLU dataset and they use the OpenFST[2] and OpenGRM[3] libraries.

3.1 Common implementation

In Figure 1, we can see the abstraction of the methods implemented in all the versions.

First, we need to load the training dataset, then we need to count how many times each tag appears in the dataset and how many times each couple word-tag appears in the dataset. After these

[1] https://github.com/esrel/NL2SparQL4NLU

[2] http://www.openfst.org/twiki/bin/view/FST/WebHome

[3] http://www.opengrm.org/twiki/bin/view/GRM/NGramLibrary

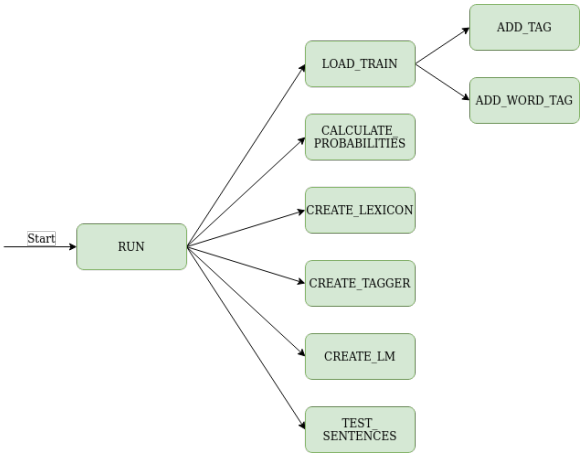

Figure 1: The visual representation of the methods implemented in the program.

counts, we compute the probability of each word-tag as follow:

$$-\log \frac{\#word\_tag}{\#tag} \quad (1)$$

where $\#word\_tag$ is the number of times the given couple word_tag appears in the dataset and $\#tag$ is the number of times the given tag appears in the dataset.

In order to correctly manage the test dataset, we need to take into account the possibility to have unknown words, i.e words that are not present in the training dataset. For these words, we do not know the correct association word-tag, for this reason we have equal probability that the unknown word appears with one of the tag in the tags set. So, we calculate the probabilities of the unknown words as follow:

$$\frac{1}{tags} \quad (2)$$

where $tags$ is the number of tags present in the dataset. In this way, we associate the unknown word (identified by the token $< unk >$) at each tag present in the dataset because we don't know at which tag it is correctly assigned.

Then, we need to build a lexicon, that contains all the symbols accepted by our system. In order to build it, we use all the words and all the tags present in the dataset and the unknown token.

The probabilities obtained and the lexicon are used to build a Weighted Finite State Transducer (WFST) that translates the words into the respective IOB concepts, this WFST is our tagger.

The next step is to create a language model using the sentences composed by the tags. First, we compile the sentences in order to obtain a FST that represents the sentences using the lexicon previously created. Then, we count the ngrams of the sentences, which are used to make a Finite State Transducer representing a ngram back-off stochastic Language Model. In the LM creation, we can specify the smoothing method and the order of the count ngram to be used.

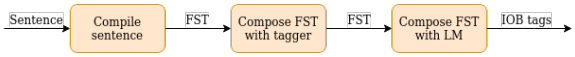

Figure 2: Given a sentence, the sentence is compiled using the lexicon to obtain a FST. The FST is composed with the tagger to obtain a new FST, which is composed with the Language Model (LM). This last step outputs the resulting IOB tags.

At this point, we can use the test dataset in order to test our system. Given each test sentence, we want to obtain the respective sentence containing the IOB tags, the process that generates this result is shown in Figure 2. For each sentence in the test dataset, we compile the sentence in order to obtain a FST that represents the sentence using our lexicon and verify that the sentence is accepted by our system. Then we compose the obtained FST with the tagger we created. In this way, for each word we obtain the respective tag based on a probability. At this point, we compose the result with the LM, in order to obtain the final result containing the tags for each word.

When we obtain the final result, we can evaluate our result.

3.2 Baseline

The baseline version of the concept tagging module is implemented as explained in Section 3.1 and it is trained taking into account all the IOB tags. This means that the O tag is not removed from the dataset.

3.3 First improvement

The implementation of the first improvement is almost equal to the baseline implementation, the only difference is in the language model training and in the construction of the WSFT. In fact, the O tag is replaced with the actual word, because the O tag is the majority of tags.

3.4 Second improvement

In the implementation of the second improvement, we introduce the use of Entity Recognition.

Words	NER	IOB
who	who	O
plays	plays	O
luke	PERSON	B-character.name
on	on	O
star	star	B-movie.name
wars	wars	I-movie.name
new	new	I-movie.name
hope	hope	I-movie.name

Table 2: An example of sentence in which there are the original words, the generalized concepts (NER) and the IOB tags.

The training dataset is used to generate two new datasets, one containing the generalized entities and one containing the IOB tags. The first dataset contains the association from the original sentences to the sentences with the generalized concepts. The second dataset contains the association from the sentences with the generalized concepts to the IOB tags. In Table 2, the columns **Words** and **NER** represent the first dataset, while the columns **NER** and **IOB** represent the second dataset.

In order to generate the sentences containing the generalized concepts, Spacy and CoreNLP are used. First the sentences are reconstruct, then we use CoreNLP TrueCaseAnnotator[4] to estimate the correct case of all the sentences words. The result of the TrueCaseAnnotator is passed to the Spacy EntityRecognizer[5] that extract the generalized entities. This final result is used to split the original dataset and construct the two final ones: the NER dataset and the IOB dataset.

For the dataset generation, different entity sets have been taken into account. The configuration of the entity sets is explained in Section 3.5.

The implementation of this method is similar to the one explained in Section 3.1, but we need two WFSTs: one that translates the words into the generalized entities and one that translates the generalized entities into the IOB tags. First, we generate a NER tagger, this means we use the generated NER dataset to translate the words into the generalized concepts. The unknown words are maintained as unknown. Furthermore, the NER Language Model is generated using the sentences containing the generalized concepts. Then, we generate a IOB tagger using the generated IOB dataset to translate the generalized concepts into the IOB tags. In this case, the unknown words are treated as explained in Section 3.1. Finally, the IOB Language Model is generated using the sentences containing the IOB tags and replacing the O tag with the generalized concepts or words.

[4] https://stanfordnlp.github.io/CoreNLP/truecase.html

[5] https://spacy.io/api/entityrecognizer

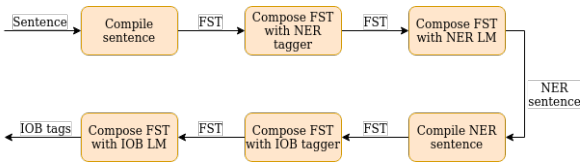

Figure 3: In the second improvement, given a sentence, the sentence is compiled using the lexicon to obtain a FST. The FST is composed with the NER tagger to obtain a new FST, which is composed with the NER Language Model (LM). The resulting sentence containing the generalized concepts is compiled using the lexicon to obtain a FST. The FST is composed with the IOB tagger to obtain a new FST, which is composed with the IOB LM. This last step outputs the resulting IOB tags.

At this point, we can use the test dataset in order to test our system. Given each test sentence, we want to obtain the respective sentence containing the IOB tags, the process that generates this result is shown in Figure 3. For each sentence in the test dataset, we compile the sentence in order to obtain a FST that represents the sentence using our lexicon. Then we compose the obtained FST with the NER tagger we created. In this way, for each word we obtain the respective generalized entities based on a probability. The resulting FST is composed with the NER LM that generates the sentence containing the generalized entities. At this point, we compile the result using the lexicon and the resulting FST is composed with the IOB tagger. Then, we compose the obtained FST with the IOB LM, in order to obtain the final result containing the IOB tags for each word.

3.5 Training parameters and settings

The methods can be executed using some smoothing methods and specifying the order, that is the number to be used for the ngram count. The smoothing methods are: absolute, unsmoothed, witten bell, katz, kneser ney, presmoothed.

Furthermore, the second improvement has been executed using different type of training dataset. The datasets were generated taking into account

different entity sets, that are:

1. person (pers);
2. person-work of art (pers_art);
3. person-language (pers_lang);
4. person-event (pers_ev);
5. person-org (pers_org);
6. person-work of art-gpe (pers_art_gpe);
7. person-work of art-event (pers_art_ev);
8. person-norp-org-gpe (pers_norp_org_gpe);
9. person-art-gpe-norp (pers_art_gpe_norp);
10. all the entities explained in Section 2 (all).

4 Evaluation

The evaluation has been executed using the script *conlleval.pl*[6]. The three versions of the module have been tested using all the smoothing methods and the order in the range 1-5.

The versions of the concept tagging module are evaluated using some metrics: Accuracy (A), Precision (P), Recall (R) and F1 score. The Accuracy is calculated taking into account all the correct labeled words with respect to the total amount of words. The Precision is calculated considering the words predicted to be a certain tag and checking how many are actually labeled with that tag. The Recall is calculated considering the words labeled with a certain tag and checking how many of them are correctly identified by the system. The F1 measure is the harmonic average between Precision and Recall.

The results of the metrics for all the versions of the concept tagging module are visible in the Table 3. The best setting for the baseline version is the Kneser-Ney method with an order of 4. Also for the first improvement, the best setting is the Kneser-Ney method with an order of 4. The second improvement has been tested with different entity sets using all smoothing methods and the order in the range 1-5. The resulting best setting is the Kneser-Ney method with an order of 4 and entity set composed by all the entities. The results of the second improvement using different entity sets are shown in Appendix A.

[6] https://github.com/esrel/LUS/tree/master/extras

	A	P	R	F1
Baseline	0.93	0.77	0.75	0.76
1º improvement	0.95	0.82	0.83	0.83
2º improvement	0.97	0.88	0.90	0.89

Table 3: The task success calculating Accuracy (A), Precision (P), Recall (R) and F1 measure. The metrics are calculated for the baseline, the first improvement and the second improvement. The metrics refer to the best configuration of each method.

5 Conclusions

In conclusions, we have seen different versions for the implementation of a concept tagging module for movie domain. With the second improvement, we have obtained the best performances. The NL2SparQL4NLU dataset contains only lowercase words, which don't help the NER module in the identification of generalized concepts. An improvement would be use a dataset containing words with the proper case that help the NER module in the concepts identification. Nonethless, the performance gained are very good and the system performs well using this datasets.

A Appendix

In this section, the results of the evaluation of the second improvement using different entity sets are shown in Table 4.

	A	P	R	F1
pers	0.97	0.86	0.88	0.87
pers_art	0.97	0.86	0.88	0.87
pers_lang	0.97	0.86	0.88	0.87
pers_ev	0.97	0.86	0.88	0.87
pers_org	0.96	0.85	0.88	0.87
pers_art_ev	0.97	0.86	0.88	0.87
pers_art_gpe	0.97	0.86	0.89	0.87
pers_norp_org_gpe	0.97	0.86	0.89	0.88
pers_art_gpe_norp	0.97	0.86	0.89	0.88
all	0.97	0.88	0.90	0.89

Table 4: The task success calculating Accuracy (A), Precision (P), Recall (R) and F1 measure. The metrics are calculated for second improvement using different entity sets.