Language Understanding Systems final project

Alessia Tovo - 189192
University of Trento - DiSi
alessia.tovo@studenti.unitn.it

Abstract

The aim of this document is to describe the work done for the final project of Language Understanding Systems course.
The purpose of the project is to develop a *Spoken Language Understanding* model using different approaches like WFST, CRF, RNN trying to get better results.
The WFST approach has been done as mid-term project, but in the Conclusion section is possible to see all the results compared.

1 Introduction

The aim of this project is to develop a *SLU* model using the movie domain, the *NL-SPARQL* dataset with basic and additional features like POS tag and lemmatization of words.
To build this model three different approaches are used:

1. WFST
2. CRF
3. RNN

The first part has been done in the mid-term project that you can find in my Github repository `LUS_first_midterm_project_unitn`.
For the second part of the project, the same dataset has been used to build *Conditioanl Random Fields* model and *Recurrent Neural Network* model. In the section 3, afer a brief recap of the crf model, is possible to see which models have been built and which results have been obtained.
The same structure has been adopted for section 4.
In the Conlcusions section is possible to see the overall results of the two projects.

2 Dataset

The dataset given to build the *SLU* model is *NL-SPARQL* movie domain and is composed in this way:

- train with basic features, so the word and the *IOB*tag;
- train with additional featues, lemmatization and *POS* tag are added;
- test with basic features that are the same of the corresponding train;
- test with additional features that are the same of the corresponding train;

For the CRF model, other two features are added, the `prefix` and the `suffix` of the words.

3 Conditional Random Field

3.1 Concept of CRF

The Conditional Random Fields are part of the of the Discriminative Sequence Models.
CRF extract featured based on undirect graphs and maximize the performance of sequence labeling modeling the conditional distribution

$$p(y|x) = \frac{1}{Z(x)} exp(\sum_{c \in C} \sum_{k} \lambda_k f_k(y_c, x, c))$$

with

$$Z(x) = \sum_{y} exp(\sum_{c \in C} \sum_{k} \lambda_k f_k(y_c, x, c))$$

where *x* is the all the words to be labeled and *y* is the label to be assigned.

With the CRF model is possible to encode long dependencies over the observations of x. In order to do that we need to decide on a set of *feature functions*. Each feature function in CRF is a function that take in input [1]:

- a sentence s;
- the position i of a word in the sentence;
- the label l_i of the current word;
- the label l_{i-1} of the previous word

3.2 Methodology and results

In order to implement CRF the `CRF ++` tool has been used. This is a simple and open source implementation of Conditional Random Fields for segmentinglabeling sequential data [2].
To model CRF with this template, is necessary build a template in order to generate a model with the command `crf_learn`.
The template is composed of a sequence of line like this: `U00:%x[0,0]`, where `[0,0]` is the window size. Is possible to do conjuction of window size in order to have conjuction of conditional distribution of the word observed.
Changing the parameter inside the view, different words are observed to model the CRF. For example the window written above considers only the current word, while the window `[-1,0]` obsrvers the word before the current word.
The second number in the view indicates which colum of the train set has been taken, so for the standard train we have only the `0` column, while in the others train set we can have different columns.
Moreover, is possible to add the bigram feature that allow to combine the current output with the previous one. The difference between unigram and bigram (**?**) is:

- unigram: (output tag) · (all possible strings expanded with a macro)
- bigram: (output tag) · (output tag) · (all possible strings expanded with a macro)

[1] http://blog.echen.me/2012/01/03/introduction-to-conditional-random-fields/

[2] https://taku910.github.io/crfpp/

3.2.1 Dataset

In order to do different experiments, an additional train set has been created, merging the two NLSPARQL train set given. Inside the ”new” train set is possible to find 5 columns:

- word;
- lemma;
- PoS tag;
- suffix;
- prefix;

In the following section is possible to see which are the results obtained considering different features of the same word and an explanation of the prefix and suffix feature.

3.2.2 Results

Word feature : results obtained using only the words and no additional features.

Accuracy %	Precision %	Recall %	FB1
94,30	86,20	79,01	82,45
94,23	83,38	78,46	82,23
94,23	86,22	78,55	82,21

Table 1: The three best result using only word

Table 1 is order by best result of FB1 score.
To obtain the best result a template like the one in Table 2 has been used.

Unigram	Bigram
U00:%x[-2,0]	B00:%x[-2,0]
U01:%x[-1,0]	B01:%x[-1,0]
U02:%x[0,0]	B02:%x[0,0]
U03:%x[1,0]	B03:%x[1,0]
U04:%x[2,0]	B04:%x[2,0]

Table 2: Template that give best performance of Table 1

Word - lemma features : results obtained using words and their lemmatizaions.

Table 3 is order by best result of FB1 score.
To obtain the best result a template like the one in Table 4 has been used.

Accuracy	Precision	Recall	FB1
94,27	87,46	78,64	82,90
94,28	87,20	78,64	82,70
94,30	87,03	78,74	82,68

Table 3: The three best result using word and lemmatization

Unigram	Bigram
U00:%x[-2,0]	B00:%x[-2,0]
U01:%x[-1,0]	B01:%x[-1,0]
U02:%x[0,0]	B02:%x[0,0]
U03:%x[1,0]	B03:%x[1,0]
U04:%x[2,0]	B04:%x[2,0]
U05:%x[3,0]	B05:%x[0,0]%x[0,1]
U06:%x[0,1]	
U07:%x[-1,0]%x[0,0]%x[0,1]	

Table 4: Template that give best performance of Table 3

Accuracy	Precision	Recall	FB1
94,20	86,42	78,19	82,10
94.18	85.94	78.46	82,03
94.28	86.23	78.09	81.96

Table 5: The three best result using word and PoS

Word - Pos features : results obtained using words and their PoS tags.

Table 5 is order by best result of FB1 score.
To obtain the best result a template like the one in Table 6 has been used.

Unigram	Bigram
U00:%x[-2,0]	B00:%x[-2,0]
U01:%x[-1,0]	B01:%x[-1,0]
U02:%x[0,0]	B02:%x[0,0]
U03:%x[1,0]	B03:%x[1,0]
U04:%x[2,0]	B04:%x[2,0]
U05:%x[0,1]	B05:%x[0,0]%x[0,2]
U06:%x[0,0]%x[0,2]	
U07:%x[0,0]%x[-1,0]	
U08:%x[-1,0]%x[0,0]%x[0,2]	

Table 6: Template that give best performance of Table 5

Word - Lemma - Pos features : results obtained using words, lemmatization and PoS tags.

Table 7 is order by best result of FB1 score.
To obtain the best result a template like the one in Table 8 has been used.

Accuracy	Precision	Recall	FB1
94,58	87,35	79,74	83,87
94,58	87,16	79,65	83,24
94,45	87,24	79,56	83,22

Table 7: The three best result using word, lemma and PoS

Unigram	Bigram
U00:%x[-2,0]	B00:%x[-2,0]
U01:%x[-1,0]	B01:%x[-1,0]
U02:%x[0,0]	B02:%x[0,0]
U03:%x[1,0]	B03:%x[1,0]
U04:%x[2,0]	B04:%x[2,0]
U05:%x[0,1]	B05:%x[0,1]
U06:%x[0,0]%x[0,1]	B06:%x[0,0]%x[0,2]
U07:%x[0,0]%x[0,2]	B07:%x[0,0]%x[0,1]
U08:%x[3,0]	

Table 8: Template that give best performance of Table 7

Word - Lemma - Pos features - prefix - suffix : results obtained using words, lemmatization, PoS tags, prefix and suffix.
Prefix and suffix are used with the aim of reduce errors in word's transcription in the train set. The lenght of both prefix and suffix is 3 and has been obtained after different experiments.

Accuracy	Precision	Recall	FB1
94,74	87,37	80,93	84,02
84,02	87,24	80,84	83,92
94,69	87,14	80,75	83,82

Table 9: The three best result using word, lemma, PoS, prefix and suffix

Table 9 is order by best result of FB1 score.
To obtain the best result a template like the one in Table 10 has been used.

4 Recurrent Neural Network

The second model used to build a SLU model are the *Recurrent Neural Networks*. They work like a Multilayer Percepton but, if this one create a connection from previous input vector to the different output vectors, RNN use recurrent connections and can connect the whole history of previous input to each output. There are two types of RNN that we have seen:

Unigram	Bigram
U00:%x[-2,0]	B00:%x[-2,0]
U01:%x[-1,0]	B01:%x[-1,0]
U02:%x[0,0]	B02:%x[0,0]
U03:%x[1,0]	B03:%x[1,0]
U04:%x[2,0]	B04:%x[2,0]
U05:%x[0,1]	B05:%x[0,1]
U06:%x[0,0]%x[0,1]	B06:%x[0,0]%x[0,2]
U07:%x[0,0]%x[0,2]	B07:%x[0,0]%x[0,1]
U08:%x[3,0]	

Table 10: Template that give best performance of Table 9

- Elman type: feeds the activation function of the hidden layer at the previous step
- Jordan type: feeds the activation function of the output layer at the step

4.1 Methodology and results

In order to train RNN model codes given during the lesson are used, one for the Elman type and another for Jordan type. The code is based on Theano and is possible to find more information on the online manual for RNN's implementation [3].

4.1.1 Dataset

For this part of the project not large mofies have been done to the two train set. Four different file have been created to merge different features in order to obtain different and better results:

- standard train;
- train with word - lemma
- train with word - PoS
- train with lemma - PoS
- train with word - lemma -pos

They both used a backprogragation called Backpropagation Through Time (BTT) where the network during training is unrolled in time backwards and backpropagation is applied.
For the Elman type, since it was necessary to find the best configuration for the RNN model, only the first three train set have been used, and once the best configuration has been found, all the train sets have been used for the Jordan type.

[3] http://deeplearning.net/tutorial/rnnslu.html

4.1.2 Results

In the first part of this section are reported the results of the Elman's application, while in the second part there are the results obtained with Jordan's application.

Elman type : like written above, for Elman only three of the five train sets are used, trying firstly to find the better configuration of the file necessary to build the RNN model.

To find the best configuration, the standard train have been used. Not only the parameters in the `config.cfg` file were modified, but also the size of the validation set.
Firstly, a validation set that is 20% of the entire train set has been used, then a train set with 10% of the words of the original train set and finally a validation set with dimension equals to 5% of train set.
Without changing the configuration file, better results are obtained with a validation set equal to 10% of the train set.
Moreover, in order to have more inhomogeneities on the topic of the sentences of validation set, the entire train has beeb shuufled before obtain the validation set. For the learning part, the untouched train set has been used.
Once found the best configuration for the `config.cfg` file, the *Word - lemma* and *Word - PoS* train set have been trained.
In Table 11 is possible to find the best result obtained. If no parameters are written in the Features column, means that the standard configuration file given during the course has been used.

Features	FB1 score
10% validation	79,53
20% validation	77,94
10% validation, `win` = 13	77,52
10% validation, `seed`=345, `nepochs` = 15	76,83
5% validation	76,42

Table 11: Best results obtained with Elman model.

Jordan model : this model has been trained with all five train sets, since the best configuration have been found thanks to Elman. The configuration model used is the one given during the course and is reported in Table 13.

In Table 12 is possible to find all the results obtained training Jordan model.

Train set	FB1 score
Standard train	78,12
Lemma - PoS train	77,62
Word - PoS train	77,48
Word - Lemma - PoS	77,42
Word - lemma	77,35

Table 12: Results obtained with Jordan model.

Parameters	Value
lr	0,1
win	9
bs	5
nhidden	100
seed	3842845
emb_dimension	100
nepochs	25

Table 13: Configuration model that gives best results for both types.

5 Conclusions

As said in the Introduction section, the whole prject consists of building a SLU model with different techniques and compare them.
In the mid-term project genrative Weighted Final State Transducers have been adopted. The three best results are reportd in Table 14.

Implementation	FB1 score
Standard	76,37
Lemma	75,82
Lemma_PoS	74,99

Table 14: Three best results obtained with WFST.

As you can see from Table 15, using standard train, the best result is obtaine thanks to CRF model, while the worst scenario is the one with WFST model.

CRF and RNN model have clearly better results but the training time is very long, also for small dataset like the ones given for project's purpose.
As future work, other model can been used for RNN like Hybrid type and the Bi-directional Elman type.

Model	FB1 score	Training time
CRF	82,45	$\sim$ 10 min
RNN	78,12	$\sim$ 40 min.
WFST	76,32	$\sim$ 2 min

Table 15: Overall results with standard train