Spoken Language Understanding (SLU) Generative and Discriminant Models comparison

|  |  |
| --- | --- |
| 000  001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049 | 050  051  052  053  054  055  056  057  058  059  060  061  062  063  064  065  066  067  068  069  070  071  072  073  074  075  076  077  078  079  080  081  082  083  084  085  086  087  088  089  090  091  092  093  094  095  096  097  098  099 |

|  |
| --- |
|  |
| Rodrigo Joni Sestari  number: 179020  rodrigo.sestari@studenti.unitn.it |
|  |
|  |
|  |
|  |
|  |

Abstract

This document has the objective of illustrate the comparison between Generative compiled in the previous document and Discriminant models. This document consists in the second project for the course of Language Understanding System of the University of Trento. This project is available on **github.com/rodrigosestari/LUS.git**

Introduction

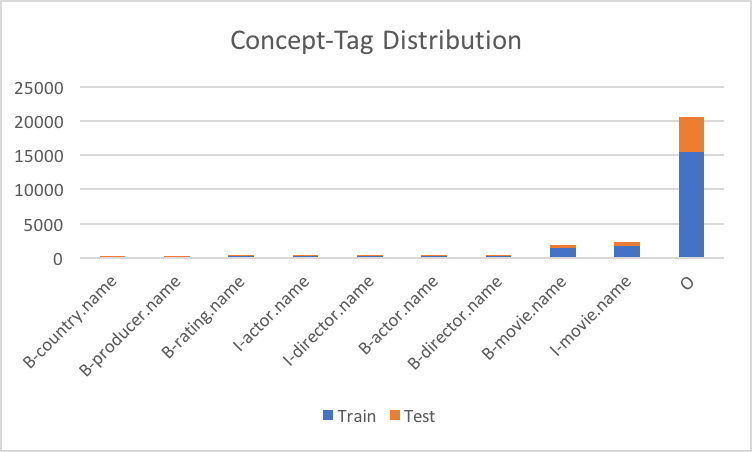
This document describes the steps necessary to implement a Discriminant model using recurrent neural networks and their comparison with Generative model where was used Finite State Transducers. The first part consists in a theory part about Neural Networks, the second one consist to apply the Discriminant model in different parameters, the last part consist to evaluate these models.

Dataset structure

The dataset available to this project are divided by Traning and Test:

* **NLSPARQL.train.data**: Dataset used to training the models. The file is format by the tuple *<word, concept>* the concept use **IOB Notation**. one for each line, and each of these sentences is separated by an empty line
* **NLSPARQL.test.data**: Dataset used to test, this dataset has the same structure of NLSPARQL.train.data
* **IOB Notation**: *Inside, Outside, Beginning* is a common tagging format for tagging tokens in a chunking task in computational linguistics. The B- prefix before a tag indicates that the tag is the beginning of a chunk, and an I- prefix before a tag indicates that the tag is inside a chunk. The B- tag is used only when a tag is followed by a tag of the same type without O tokens between them. An O tag indicates that a token belongs to no chunk.

Is possible see in the figure bellow the IBO distribution in the dataset:



Theory about natural language process

I this chapter introduces some principles about the natural language processing that come used in this document:

* **FST**: *Finite State Transducer,* is a FSA whose state transitions are labelled with both input and output symbols.
* **WFST**: *Weight Finite State Transducer,* A weighted transducer puts weights on transitions in addition to the input and output symbols.
* **LM**: *Language Model* compute the probability of a sentence or sequence of words:
* **Smoothing**: Assume that no n-gram of known words has 0 probability and redistribute probability mass from seen to unseen events, this is known as smoothing.
* **ANN**: *Artificial Neural Network*, is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems.
* **RNN**: *Recurrent Neural Network,* is a class of artificial neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behaviour. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs.

Tools

|  |  |
| --- | --- |
|  |  |

To execute the tasks about this document was necessary the following tools:

* **Conlleval**: A Perl script that allow evaluate the SLU result.
* **Theano:** A Python library for efficiently handling mathematical expressions involving multi-dimensional arrays (also known as tensors). It is a common choice for implementing neural network models. Theano has been developed in University of Montreal, in a group led by Yoshua Bengio, since 2008.

RNN Architecture

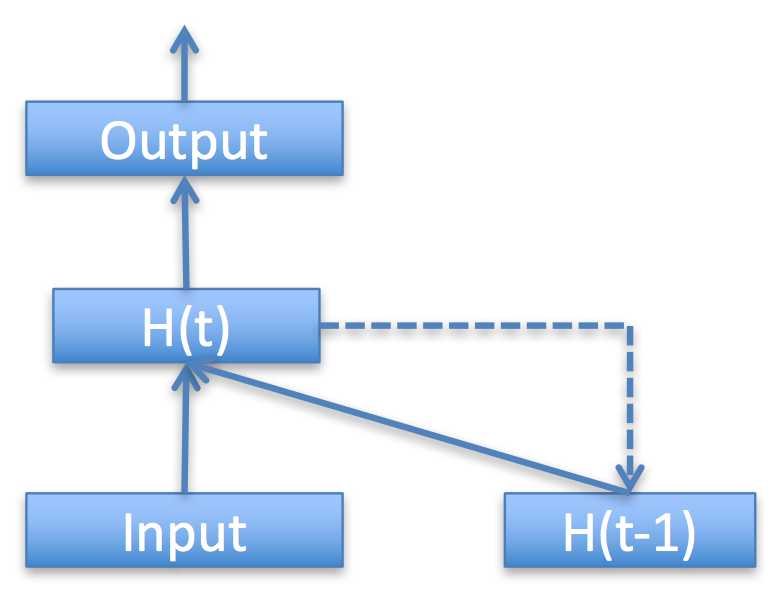
Research on language modelling for speech recognition has increasingly focused on the application of neural net- works. Two competing concepts have been developed: On the one hand, feedforward neural networks representing an n- gram approach, on the other hand recurrent neural networks that may learn context dependencies spanning more than a fixed number of predecessor words.

When a feedforward neural network (FFNN) is used, only the direct predecessor words are used to predict the probability of the current word . Although it is possible to include words from the previous sentence, most of the time the history is truncated at the beginning of the sentence in the n-gram approach. When a recurrent neural network is used, the full sequence of predecessor words is considered for predicting .

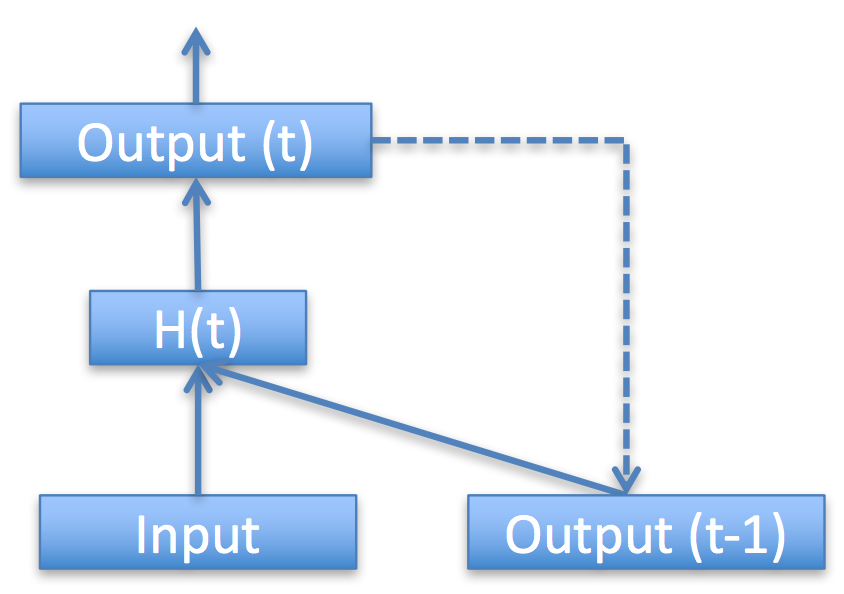
In another way, in a ANN the inputs and outputs are independent between their, the problem is that is not possible to explorer the sequential information. To explorer it, exist the RNN in which, the weighted connections feeding a neuron, also come either from the hidden units of the previous iteration, or from the outputs, been possible consider sequences.

In order to apply a RNN into SLU is used two type of networks:

* **Elman mode**: This network has a connection that feeds the activation of the hidden layer in the previous time step with the current input. The probability of label is estimates by where C is the label, the word an the previous state.



* **Jordan model**: this model connection that feeds the activation of the output layer at previous time step with the current input. The probability of label is estimates by where C is the label, the word an the previous output.



Implementation

The implementation consists to execute some scripts to training and test the LM, these scripts use an external tool Theano.

Scripts

Some scripts are implemented to elaborate chain of input/output files. The scripts are:

* **start.sh**: This script call Theano tool in order to execute all models, its requires one parameter, if negative number, run all models, if positive between 1..7, run a specific configuration file.
* **files.sh:** Create the label and word dictionary, using the initial training dataset, split it in a training dataset with the 75% of the sentences and in a validation dataset with the remaining 25%.

Configuration

The configuration file contains these parameters:

* **lr**: starting learning rate
* **win**: context window size
* **bs**: mini batch size
* **nhidden**: size of the hidden layer
* **seed**: random seed
* **emb\_dimension**: dimension of the word embedding
* **nepochs**: maximum number of backpropagation steps

was created 8 configuration files in order to test all parameters, bellow is possible see these models:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **mod** | **lr** | **win** | **bs** | **hid** | **seed** | **dim** | **epoc** |
| 1 | 0.1 | 9 | 5 | 100 | 3842845 | 100 | 25 |
| 2 | 0.1 | 9 | 5 | 100 | 3842845 | 100 | **10** |
| 3 | **0.5** | 9 | 5 | 100 | 3842845 | 100 | 25 |
| 4 | 0.1 | 9 | 5 | **200** | 3842845 | 100 | 25 |
| 5 | 0.1 | **7** | 5 | 100 | 3842845 | 100 | 25 |
| 6 | 0.1 | 9 | **10** | 100 | 3842845 | 100 | 25 |
| 7 | 0.1 | 9 | 5 | 100 | **300** | 100 | 25 |
| 8 | 0.1 | 9 | 5 | 100 | 3842845 | **200** | 25 |

Results

|  |  |
| --- | --- |
| 200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249 | 250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299 |

The idea is measure the output given by the SLU model.

Evaluation Methods

* **accuracy**: Accuracy refers to the closeness of a measured value to a standard or known value.
* **precision**: Precision refers to the closeness of two or more measurements to each other or also called positive predictive value, is the fraction of retrieved instances that are relevant.
* **recall**: recall is the fraction of relevant instances that are retrieved in another way is a measure of how many truly relevant results are returned.
* **FB1**: is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score

Evaluation

In the tables above is possible see the result of evaluation, the tables are divided by FST result, got from the first project and Elman/Jordan Networks, the first column for the FST represents the smoothing method, instead from the Elman and Jordan represents the model used, the next 3 columns represent the evaluation methods; accuracy, precision, recall and F-measure.

**FST**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **method** | **accuracy** | **precision** | **recall** | **FB1** |
| **absolute** | 92.88% | 76.97% | 75.34% | 76.15 |
| **Katz** | 92.62% | 78.03% | 73.88% | 75.89 |
| **Kneser** | 92.90% | 76.79% | 75.53% | 76.16 |
| **Pre-smo** | 92.65% | 78.41% | 74.24% | 76.27 |
| **Unsmoo** | 92.54% | 78.09% | 73.88% | 75.93 |
| **WittenB** | **92.90%** | 77.03% | 75.62% | 76.32 |

**RNN Elman**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **model** | **accuracy** | **precision** | **recall** | **FB1** |
| 1 | 95.05% | 75.64% | 75.99% | 75.81 |
| 2 | 94.36% | 76.29% | 70.49% | 73.27 |
| 3 | 90.46% | 63.70% | 53.07% | 57.90 |
| 4 | 94.98% | 78.48% | 75.53% | 76.97 |
| 5 | 94.82% | 78.53% | 73.42% | 75.89 |
| 6 | **95.33%** | 78.58% | 76.99% | 77.78 |
| 7 | 95.25% | 82.22% | 76.72% | 79.37 |
| 8 | 95.07% | 80.23% | 75.53% | 77.81 |

**RNN Jordan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **model** | **accuracy** | **precision** | **recall** | **FB1** |
| 1 | 94.71% | 80.08% | 74.43% | 77.15 |
| 2 | 94.71% | 79.78% | 74.15% | 76.86 |
| 3 | 89.31% | 55.67% | 48.12% | 51.62 |
| 4 | 94.77% | 80.60% | 73.88% | 77.09 |
| 5 | 94.98% | 82.06% | 75.89% | 78.86 |
| 6 | **95.08%** | 80.99% | 75.34% | 78.06 |
| 7 | 94.83% | 78.97% | 75.71% | 77.30 |
| 8 | 94.64% | 80.65% | 75.25% | 77.86 |

Conclusion

About RNN result, the networks Elman and Jordan are quite similar between them, but Jordan are slightly better than Elman, because Jordan model consider the previous output in order to label the given input.

|  |  |
| --- | --- |
|  |  |

With a learning rate higher, has the worsen result, I believe because the gradient descent is not fast enough to get better learning results.

About the comparison about the discriminant and generative modes, the RNN model is more accurate than FST, because consider the global decision on a sequence that is more accurate than local decision. The file results can be seen on **https://github.com/rodrigosestari/LUS/tree/master/Second/result**

References

John Wiley and Sons. New York, 2011. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*

Maccartney, B. *(2005).* Stanford.from *https://nlp.stanford.edu/~wcmac/papers/20050421-smoothing-tutorial.pdf*

Mohri et al. (1996) *FSM Toolkit Exercises*

Riccardi (2017). UniTN. Retrieved 10 April, 2017, from http://disi.unitn.it/~riccardi/

*Sundermeyer M, et al. (2013) Comparison of feedforward and recurrent neural network language models.*