Deep Learning Project: Election results

Deliverable: Recurrent Neural Network

Assigned by:

Andreev Alexandra (309234300)

Gropper Tom (312482037)

**Abstract:**

The report below describes the Deep Learning Project for prediction of election results, which is done based on Polls data of Israeli elections at 2006 (“many-parties” system). The input data comes as time-series of polls votes for each party. The accuracy of prediction is compared to the actual results of Elections. Three prediction methods had been tried for the prediction: Linear Regression, Perceptron Neural Network and RNN Neural Network using LSTM.

**Introduction:**

Predicting the results of Elections is a prediction of a time-series data set. The problem at such a prediction is luck of data: the longest input data set is about 50 polls, done daily, different polls suppliers. Based on such a short data as an input set, it is quite a challenge to make a precise prediction. Additional challenge is hidden at Israeli character of political system: being a relatively small country, having daily incidents and a politically active population makes the elections results very unstable and thus particularly hard to predict.

**Related Work:**

Related works found about Elections usually deal with American Elections, which is two-parties system. Usually it uses classification algorithms to predict the result (“Actionable and Political Text Classification using Word Embeddings and LSTM” by Adithya Rao, Nemanja Spasojevic). At the current work, the analysis was done without using classification.

There are a lot of discussions and publications about time-series analysis using the Deep Learning algorithms, though usually those have much bigger data sets.

**Background**

The report below is a summary for the Final Project at the Deep Learning Course at Ariel, University (lead by Amos Azaria). The purpose of this project was to learn Deep Learning different methods, implement it and to get as much intuition as it is possible within such a short time. The Elections topic had been chosen as a typical representative for time-series data type (the data-points’ order does matter), which is widely used in both Science Investigations & Experiments and Industry different fields.

Applying Deep Learning methods supplies additional instruments for results time-series data predictions (though usually it requires much bigger data sets).

On the other hand, the fact that the data used is not too big, allowed to debug much deeper, to understand the Deep Learning mechanisms, and to get better intuition about tuning parameters at different methods. Also, it gave a free hand to run massive number of different configurations as each run requires a small amount of computing resources. Performance time is shorter compared to other data models using Deep Learning algorithms, such as images, audio, etc.

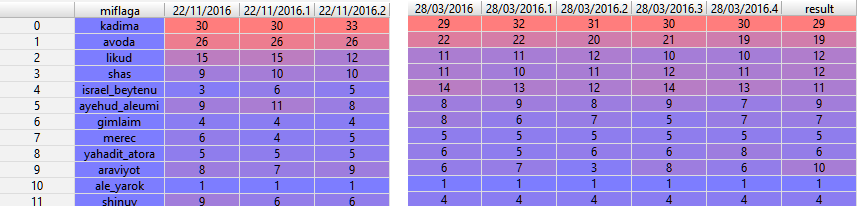
**Project Description:**

The best result (closest to the actual result) is predicted by RNN using LSTM.

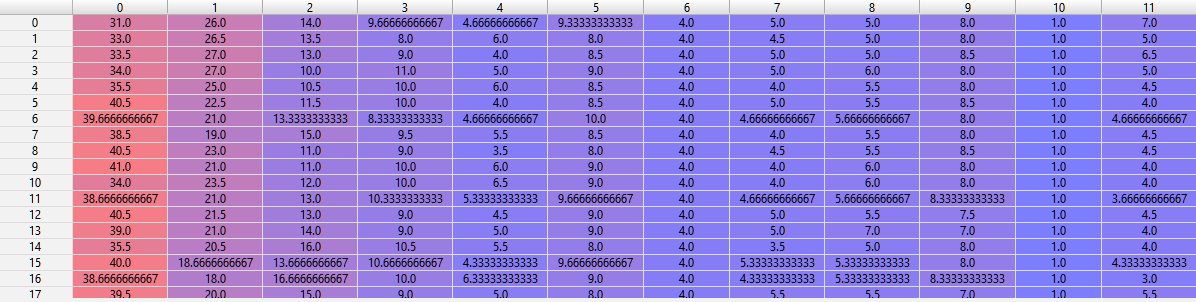
1. **Data**

The input data for the learning is multiple surveys before the Elections, performed by different institutes. There are 12 parties, 56 dates plus actual result of the Election.

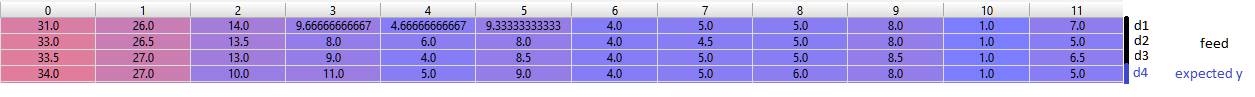
For each date it might be multiple results, by different institutes:



The data split is done by average result for the same date (the below is transposed from the previous screenshot):



The feed for each learning iteration is 3-grams (by dates), when the expected result is the result of the day following it:



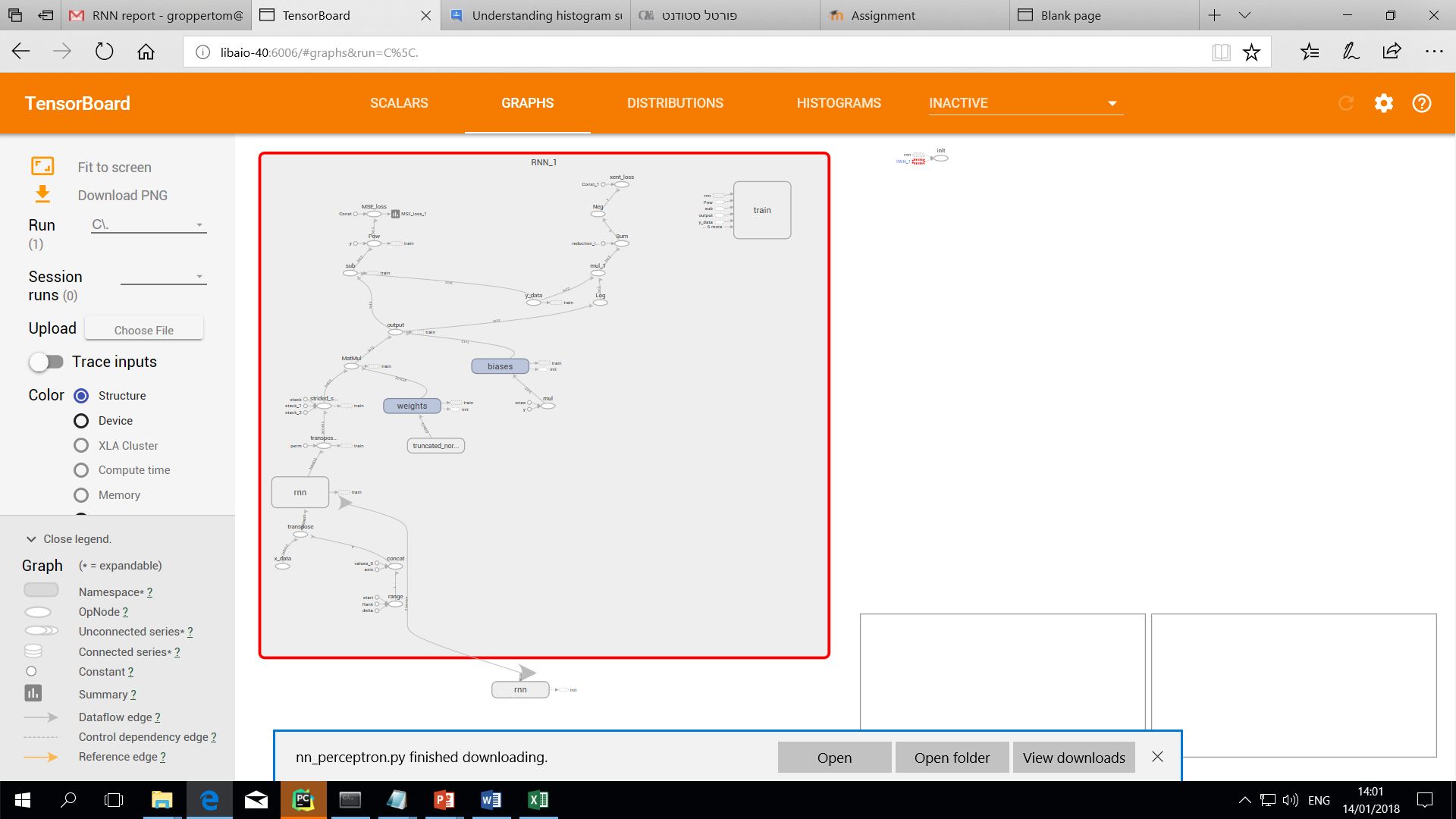
The whole data was split to training set (80%) and the test set (20%), based on random indexing of days.

1. **RNN architecture**

RNN processes (a batch of) records (batch size = n-gram size, 3 in this approach). The best approach had been achieved by the following configuration:

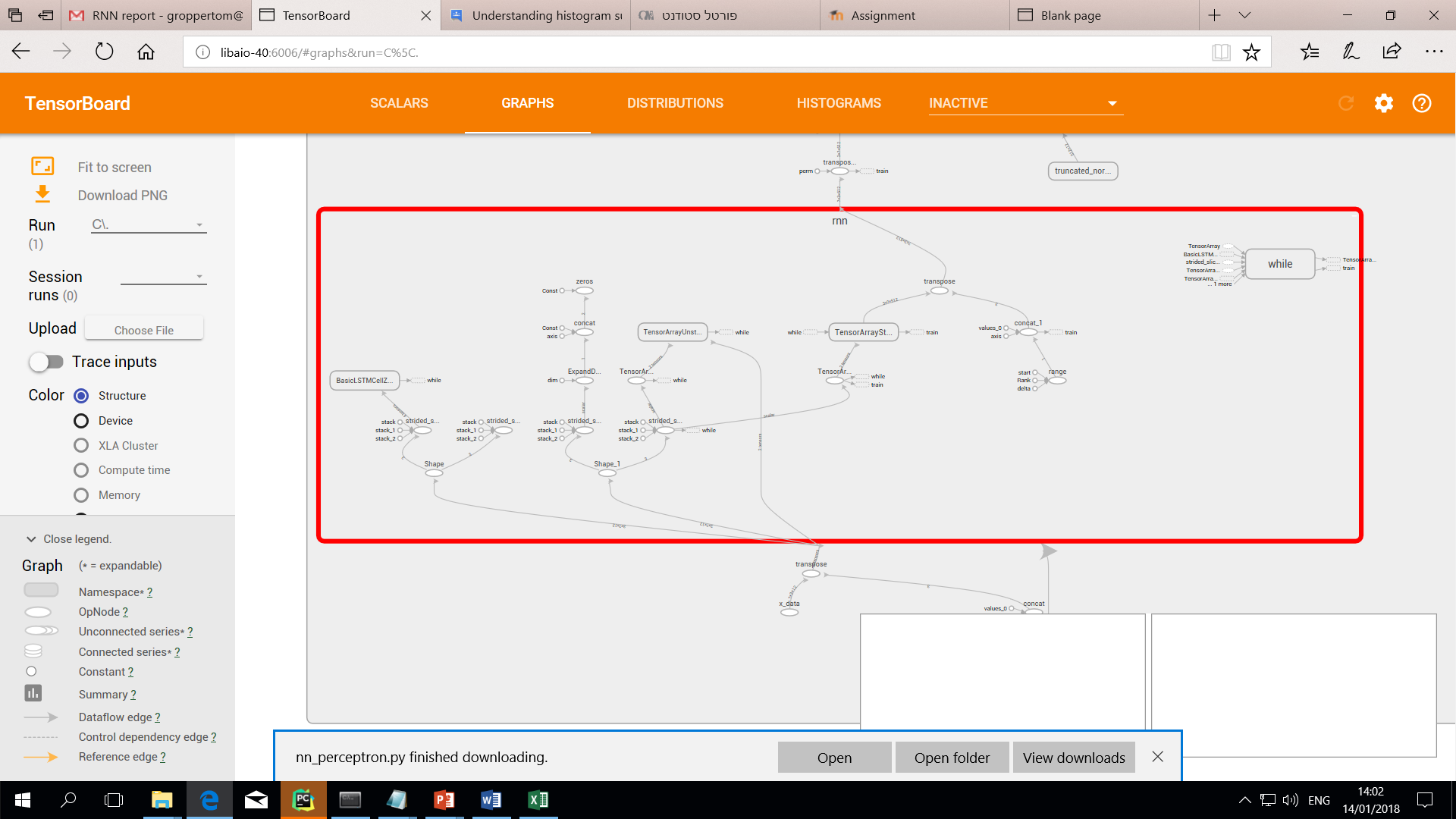
* Iterations number: 30K
* Learning Rate (for Gradient Descent at the loss calculations): 0.0001
* Neurons number: 512

Scheme (TensorBoard):



1. **LSTM architecture**

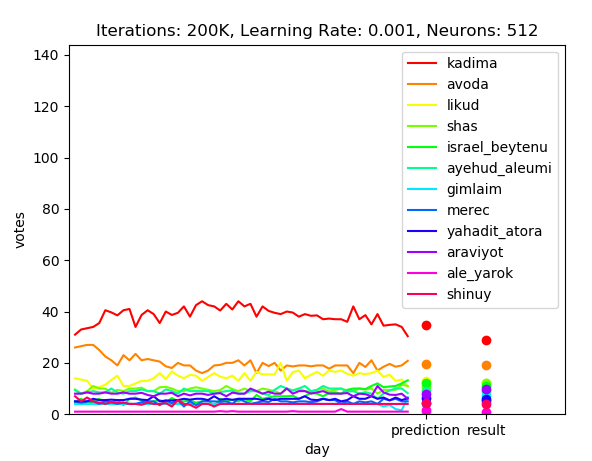
Using Tensorboard BasicLSTMCell, the architecture (by TensorBoard):



1. **Results**

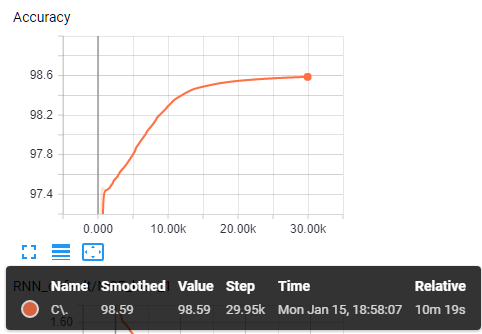
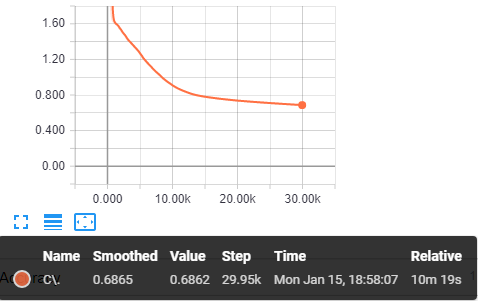
Predictions vs actual result for each party (different colors).

* Left line is vote polls by days, two points from the right:
* “prediction” – predicted by the RNN model result
* “result” – actual Elections result



Loss and Accuracy progress over iterations for the best result:

Loss progress: Accuracy progress (final = 98.59):



**Previous attempts:**

* **Linear Regression**

Linear Regression supplied not a bad result, mainly because the data’s nature was somewhat linear. Also, it was achieved faster (from perspective of performance measure of time elapsed). An input of a vector of 42 features was supplied (3 continues day’s measurements, the 4th day is the expected result).

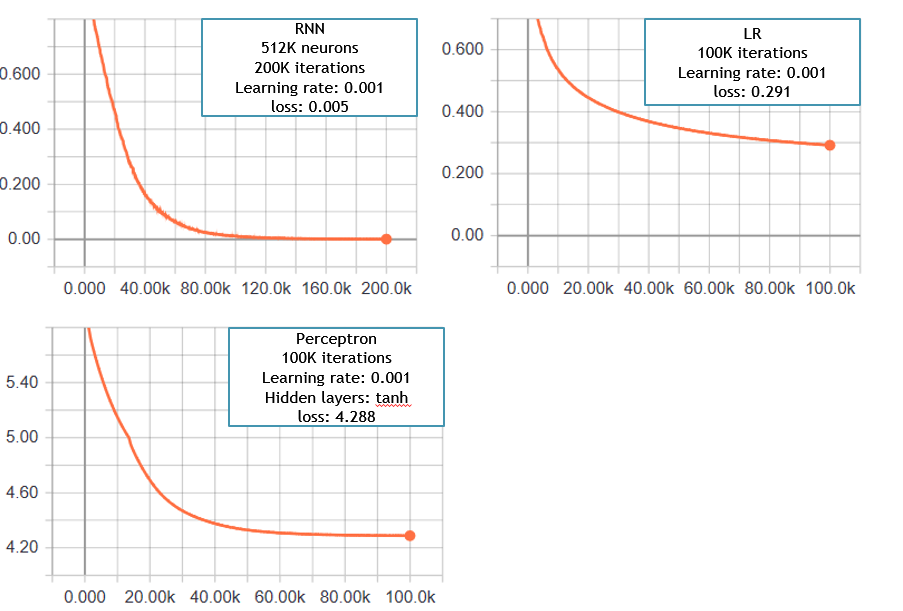
The best result achieved is: 96.52%

* **Perceptron NN**

Perceptron was found as not as effective model: it predicted a little better than Linear Regression. Performance/computing resources were heavier. The best result at the Perceptron was achieved by single layer with tanh activation function and MSE for the loss calculations

The best result achieved is: 95.25%

* Losses summary comparison for three methods (by Tensorboard):



**Experiments:**

Multiple configurations had been processed and compared each versus the other for tuning purposes. Tuning parameters validated:

* + Dropout percent
  + Loss type
  + Iterations number
  + Learning Rate (at Gradient Descent) options
  + Train percent
  + Neurons numbers
  + N-gram size
  + BasicLSTMCell vs MultiRNNCell (Tensorflow)
  + Activation function

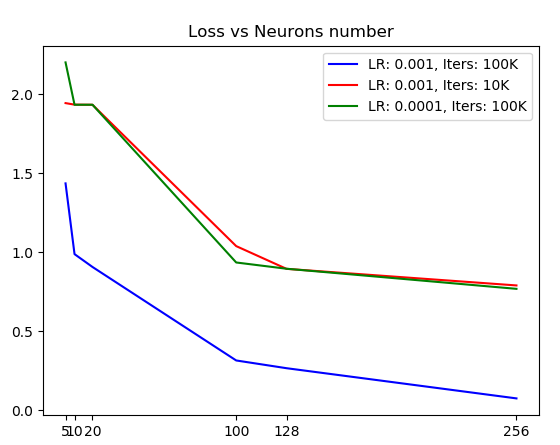
Below is summary of the 24 runs for specific parameters: no dropout, BasicLSTMCell, loss type=MSE, train percent = 80%, n-gram size = 3, activation function = tanh:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **neurons number** | **learning rate** | **iterations** | **loss** | **accuracy** | **time elapsed (secs)** |
| 10 | 0.0001 | 10,000 | 51.448 | 51.03253 | 42 |
| 10 | 0.0001 | 30,000 | 2.57992 | 97.24472 | 97 |
| 10 | 0.0001 | 80,000 | 2.03028 | 97.48814 | 241 |
| 10 | 0.001 | 10,000 | 2.02994 | 97.48017 | 35 |
| 10 | 0.001 | 30,000 | 1.69816 | 97.32049 | 119 |
| 10 | 0.001 | 80,000 | 1.78462 | 97.25609 | 285 |
| 64 | 0.0001 | 10,000 | 2.01973 | 97.48848 | 41 |
| 64 | 0.0001 | 30,000 | 1.83991 | 97.45845 | 127 |
| 64 | 0.0001 | 80,000 | 1.47261 | 97.15921 | 254 |
| 64 | 0.001 | 10,000 | 1.47491 | 97.2634 | 32 |
| 64 | 0.001 | 30,000 | 861233 | 96.35167 | 89 |
| 64 | 0.001 | 80,000 | 0.60089 | 96.14089 | 243 |
| 128 | 0.0001 | 10,000 | 2.01926 | 97.47955 | 48 |
| 128 | 0.0001 | 30,000 | 1.53911 | 97.34486 | 125 |
| 128 | 0.0001 | 80,000 | 0.999278 | 96.73925 | 335 |
| 128 | 0.001 | 10,000 | 0.943447 | 96.51217 | 43 |
| 128 | 0.001 | 30,000 | 0.601185 | 96.39088 | 125 |
| 128 | 0.001 | 80,000 | 0.241446 | 96.18057 | 339 |
| 512 | 0.0001 | 10,000 | 1.17133 | 96.97297 | 377 |
| 512 | 0.0001 | 30,000 | 0.917161 | 96.44704 | 1198 |
| 512 | 0.0001 | 80,000 | 0.700676 | 96.40462 | 6000 |

Conclusions based on results at above table:

* Neurons number does not improve the result by much, 10 neurons is enough to get a good accuracy.
* Training the model for 30k iterations yielded the best results. We observed that more iterations result in a decrease in the accuracy, probably due to over-fitting.
* As number of neurons is grows, elapsed time increases, especially if there is a big number of iterations.

Loss is highly dependent on configuration, here are some acceptable (in the prediction result meaning) configurations:



**Conclusions**

* It is very important to set up the input data, both for Linear Regression (to find appropriate features to feed the Learning Mechanism), and for the NN (n-grams sizes).
* The result is highly dependent on the configuration of the parameters, the nature of the data (linear, non-linear), and the following parameters:
  + Loss type
  + Iterations number
  + Learning Rate (at Gradient Descent) options
  + Train percent
  + Neurons numbers (NN)
  + Hidden Layers number (NN)
  + N-gram size (at this case)
  + Activation functions
  + Dropout percent (RNN)
  + Implementation
* Training by weeks and by days had been processed, the days probing resulted in better accuracy value, since it had more samples.
* Different methods had been used: Linear Regression (LR), Perceptron NN (PNN) and RNN using LSTM(RNN). The worse result had been received by LR, the best one by RNN. Though the difference at the accuracy is not too big due to linear character of the probing data.
* Different Tuning had been applied for each method. For the RNN the crucial tuning was: Neurons Number, Learning Rate, Iterations Number, Loss Type. Did not have an effect (except of performance time): dropout, regularization, MultiRnn definitions, n-gram size.