Votes Prediction In Stackoverflow Answers
Faculty of Science and Technology
University of Tartu

Aqeel Labash
aqeel.labash@gmail.com
Supervisor
Tambet Matiisen

May 3, 2016

abstract
The aim of this project is to see if we can truly predict number of votes a stackoverflow answer can get. That is depending only on the question and the answer text using LSTM [1] network.

Keywords
Deep Learning;Stackoverflow votes prediction;Long Short Term Memory

1 Introduction
Every day around 10-20M [2] views, 4-10M users visit stackoverflow website [1]. Every minute 4.6 answers,2.84 questions. With 11,573,980 questions and 18,713,658 answers, 5,509,974 users ,56,372,889 comments.[2]This huge amount of data exist on stackoverflow website. Where people can post questions and they are answered by the users. It has a feature which is, each answer can get votes based on the quality of the answer. Consequently, Voting is the vital factor that determine how good is an answer for specific question. Therefore, being able to predict the expected number of votes before even submitting it is the major motivation for this project.

2 Background
2.1 Deep Learning
There is many definitions for deep learning one of them is:"A class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification."[3] another definition explain it as an algorithms based sub-field in machine learning to learn multiple levels in order to model complex relationship between features.[3] One of the methods used in deep learning is Recurrent Neural Networks.

2.2 Recurrent Neural Networks
It is a class of artificial neural network. It is renowned for it's dependency on previous information (sequences).[4][5]Which traditional neural networks lake because it treat all input,output as independent from each other which is not suitable for all cases.[5] Specially when we are working with natural languages where each word depend on series of previous words.

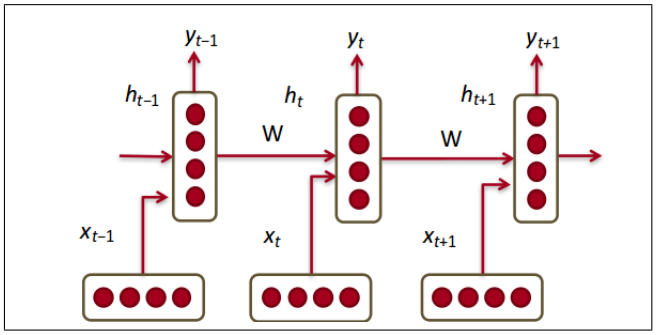

Fig. 1: Recurrent Nerural Network

In figure 1 we can see that the result at certain time y_t depend on x_{t-1}, x_t.But it does not depend on x_{t-1} directly, weights have to be applied first

2.3 Long Short Term Memory
Long Short Term Memory is a Recurrent neural network that introduced memory cell.Memory cell prevent gradient vanishing and explosion by using 4 gates Shown in Fig 2.Those gates modulate environment-memoery cell interaction.

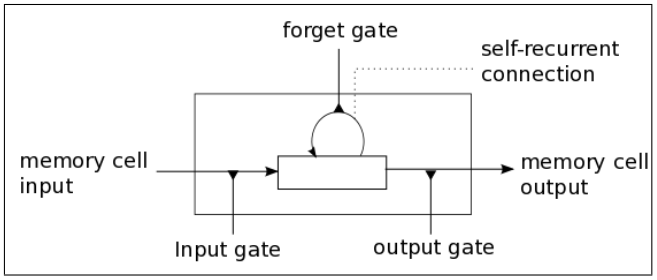

Fig. 2: Memory cell [7]

Self-recurrent connection:it has a weight of 1.It's main task to keep memory cell state constant regardless of any interference.

[1] Long-Short-Term-Memory
[2] Million

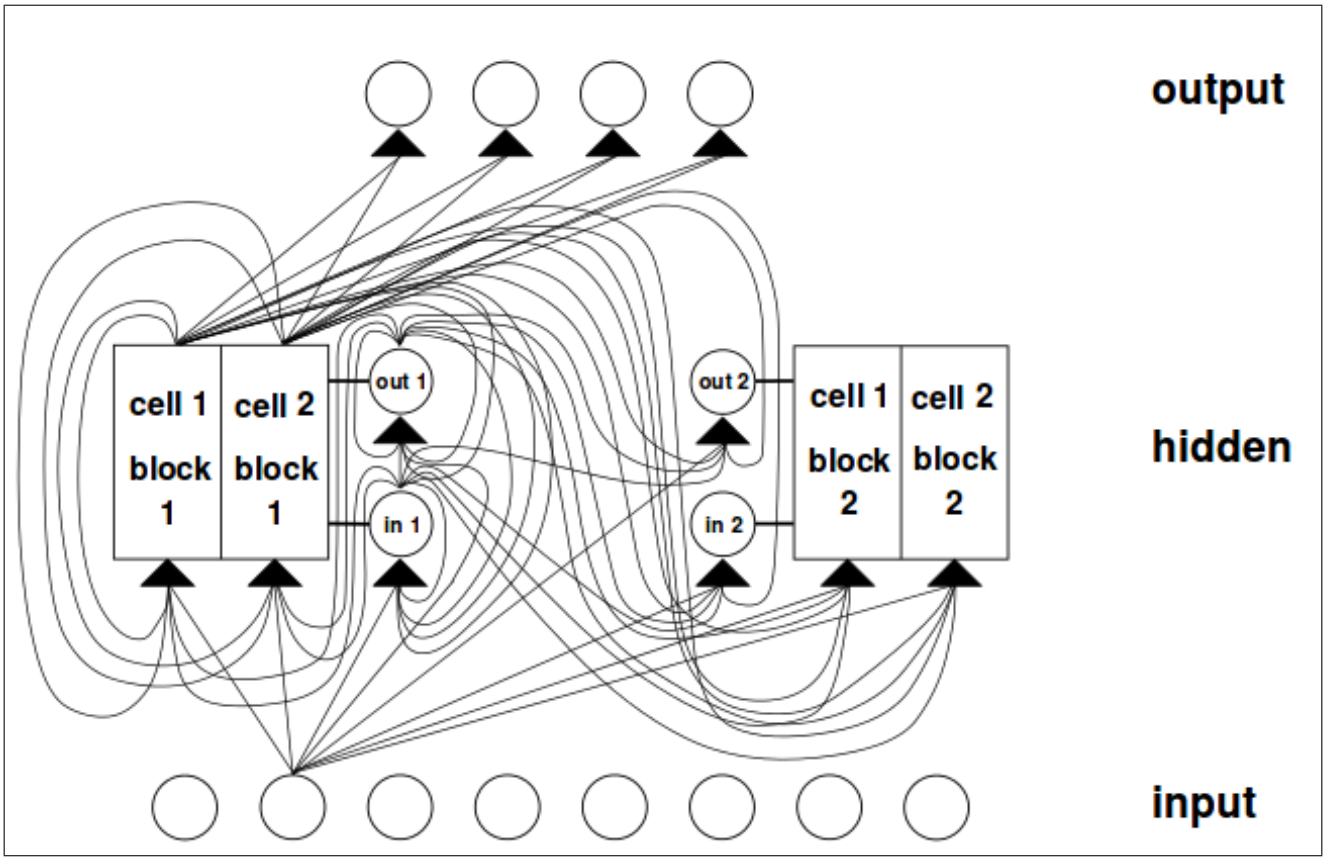

Fig. 3: Example of a net with 2 memory cell blocks of size 2[8]

Input gate: allow or prevent incoming signal from changing the memory cell state.
Output gate: responsible of allowing the memory cell to affect other neurons.
Forget gate: this gate inflect self-recurrent connection to allow or prevent it from remembering the previous state.

2.4 Fully connected layer

A fully connected layer is when every neuron from previous layer is connected to every neuron on the next layer.As shown in Figure 4

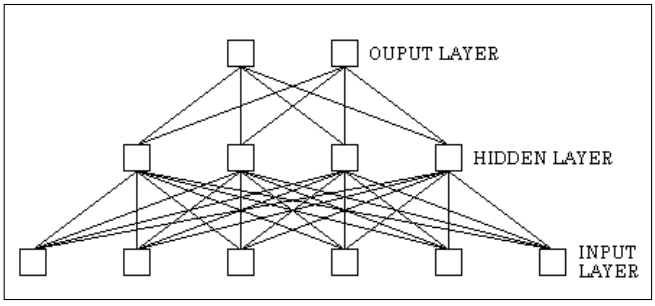

Fig. 4: Fully connected neural network [9]

3 Dataset

The data set I used to optimize the parameters were from astronomy.stackexchange.com. It has around 4536 answers and 2738 answered question. I picked this because it is not so large to take long time in processing nor very small.

3.1 Preprocessing

Before using the data It should be organized and cleaned to get better result.

3.1.1 Organize Data

In this level the task was to get the data that we might use, from XML files to CSV file.

3.1.2 Remove HTML tags

When working with data from the web usually it contain html tags which was the first thing to do. For this I used Regular expression library in python

3.1.3 Remove Special Characters

The answers and questions contained new lines and quotes which needed to be removed to generate a valid CSV file.

3.1.4 Remove Stop Words

Stop words usually have high frequency which lead for them to act as noise more than features.To achieve this task I used nltk library.

3.1.5 Stem Words

Stem words help in decreasing the dictionary size.I used Porter algorithm to stem the words.

3.2 Simple Statistics

The following table so some measures for words in questions and answers text.

	Min	Mean	Max	Median
Q	3	50	3095	38
A	3	117	2195	86

The answers **average** score : 3.34,**median**:2.

The Questions **average** score : 4.82,**median**:3.
The total number of unique words : 60644.

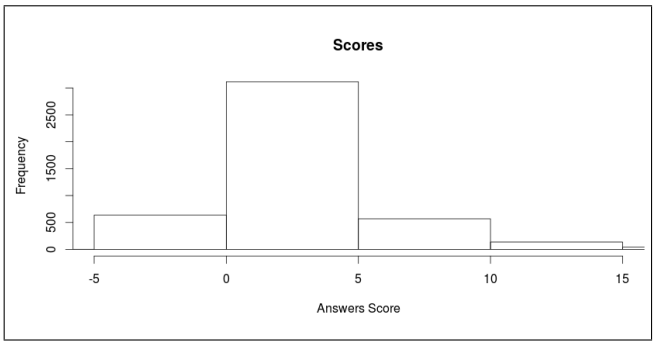

Fig. 5: Distribution of answers over scores(Votes)

Figure 5 shows the distribution of answers over votes. where we can clearly see that most questions lay between 0-5 votes.

4 Model

4.1 Question Answer Representation

To represent the data, I used dictionary index so each word would be weighted as a unique word.Using count,binary representation for the words might loose the uniqueness of the words.
For the length of question and answer vectors I used the mean value of length.The increase in vectors length will lead to increase of MSE (Mean Square Error).After that the result from both networks were merged to be fed to full connected layer.

4.2 LSTM Model

The questions and answers were fed to two LSTM networks with the same hidden layer size.

4.2.1 Hidden layer size

To decide the best hidden layer size many experiments were done to optimize it.I tried the values (5,15,25,100,150), results in figure 6

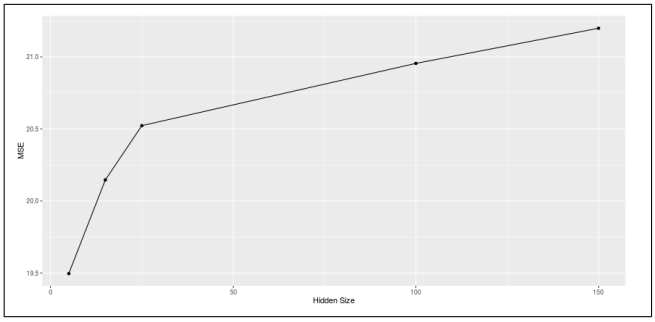

Fig. 6: Shows test set MSE changes over different hidden layer size

From the figure we can see that the best hidden layer size was 5.

4.3 Full connected layer

At the end there is a fully connected layer that take as input the merged vector of questions and answers networks and output a single value that represent the prediction.

4.4 EPOCH

To select the best epoch number I tested (2,50,100,125,200) and the test MSE was as the following plot:

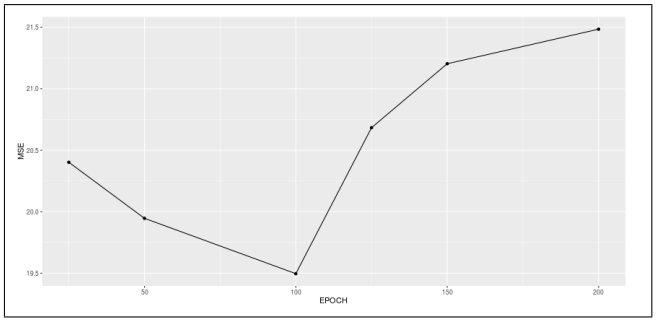

Fig. 7: Shows test set MSE changes over different epoch

From figure 7 we can see that the error decreases while going from 25 to 100 and after that is start increasing.

5 Results

After finalizing the parameters the lowest MSE achieved was 20.27 for testing and 2.15 for training.Here is some examples:
Link: Dark Matter Particle Candidates
Score:6
Predicted Score:6.19
Link: Is it odd that our Sun has so many planets?
Score:15
Predicted Score:11.20
The previous examples score was close some how.
To save the space I'll just mention the links. The following example is the worst prediction in test set :
Link: Does the Sun rotate? **Score:**62
Predicted score:5.20
Link: How probably is it that galaxies will extinguish?
Score:-2
Predicted score:3.33
Link: Is extraterrestrial mining more difficult or impractical for bodies without plate tectonics?
Score:0
Predicted score:-1.21

6 Discussion

The final result achieved was 20.27 MSE for testing and 2.15 MSE for training which indicate a clear over fitting. I believe this problem is due to the amount of data used.I'll try to use larger amount of data (as much the server can handle) to inspect the parameters more.But regardless of that point if we take a look at the histogram of prediction and test (figure 8 & 9) we can see that the problem happens with the outliers.

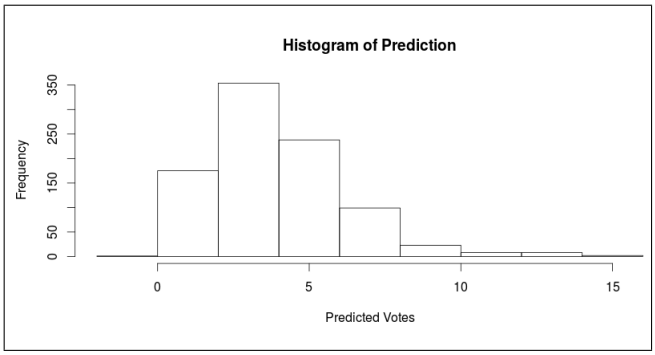

Fig. 8: Histogram of votes predicted

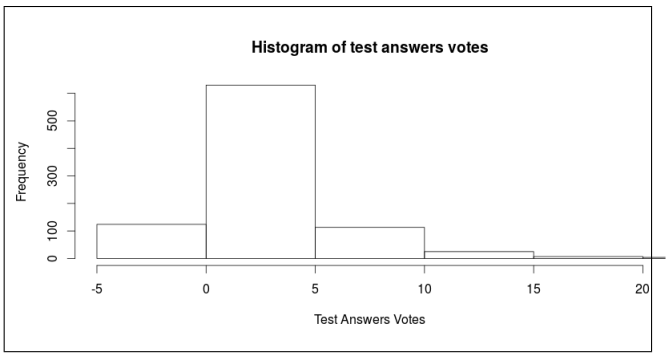

Fig. 9: Histogram of test set answers votes

The following figure shows the density of predicted votes and actual votes in testing set.

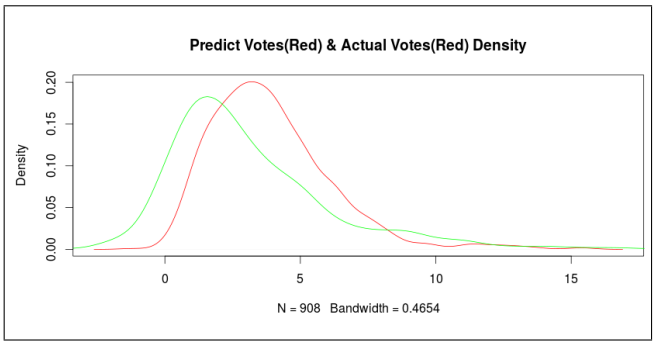

Fig. 10: Density of predicted votes vs actual votes

7 Future work

Firstly, more cleaning for the data would improve the results. I was focused on generating a valid CSV and keep latex code. Which distracted me at some point to do more special characters cleaning.
Secondly, I think there is many feature that would decrease the error like:

1. Question view count: Which represent how much popular the topic is. some times the answer is quite good but it's not popular topic which lead to small amount of votes.
2. Percentage between question view count and question score: This feature would give how good the question is.
3. Date : the older the post is the more chance it has to have more votes.
4. User reputation: reputation of the user who answered question have influence as well on the votes.Users with high reputation usually provide good answers.
5. User badges: each badge has specific meaning which also can help in determining the quality of the user answers.

Use larger data set and try different Stackoverflow forums would help optimizing the model even more.But it'll need more resources as well.

8 Conclusion

In this project Stackoverflow answers votes were used to train a LSTM network to predict the votes.Depending on the current findings the votes could be predicted with acceptable margins considering that only the text were used. Popular topics won't be predicted correctly. The results could be improved by using more samples or including more features related to topics popularity.
Note: All code,.tex, .py,.ipython etc.. files available on Github

References

[1] https://api.stackexchange.com/docs/info

[2] https://www.quantcast.com/stackoverflow.com

[3] Deng, L.; Yu, D. (2014). "Deep Learning: Methods and Applications"

[4] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, 2009.

[5] The Unreasonable Effectiveness of Recurrent Neural Networks, karpathy.github.io,May 21st, 2015

[6] Recurrent Neural Networks Tutorial, Part 1 Introduction to RNNs, Denny Britz, September 17, 2015

[7] http://deeplearning.net/tutorial/lstm.html

[8] Sepp Hochreiter and Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation 9 (8): 1735–1780.

[9] Mary Jo Creaney-Stockton,1996, Isolated Word Recognition Using Reduced Connectivity Neural Networks With Non-Linear Time Alignment Methods.