Neural net final project

Kelly Maclauchlan
Devon Witol

For our project we chose to build a system for sentiment analysis, that can determine if the sentence given is positive or negative.  To achieve this we looked at the individual words rather than their placement in the sentence. Taking the data we mixed it all together to randomize the order, and using the Count Vectorize feature extraction functionality in sklearn we were able to create a dictionary of approximately 5000 words which we proceeded to map each input.  Using the count of each word for that sentence using a bag-of-words model. Once the data has been transformed it is ready to be entered into the Multilayer Perceptron (MLP).

We chose to use an MLP for our neural network because it provides a connected network on the single dimension of the input. With this the MLP can form connections between words that occur together to denote a positive or negative sentence. Although a Recurrent Neural Network (RNN) is typically used to solve this kind of problem we wanted to use a network that we had covered more in depth in the course. The bag-of-words had approximately 5000 words and differed between the two computers we tested on. Our theory was that this was caused by our use of different operating systems but since this was not the subject of our experiments we just ensured that our program was robust enough to deal with this variability and moved on.  To perform the train test split we used the built in tensorflow functionality with a test size of 10%.

Although the MPL was our final design we started out using a CNN. We took this approach while working with the sentence transformed by the dictionary to be an array of numbers that represented the words, with padding on the end to make them all a uniform length. We chose the CNN because we wanted to look at the spatial relation between the words, but quickly realized that we did not have enough data to gain meaningful results even with different configurations, as can be seen in the comparison graph. So we decided to implement different structures.

With the MLP we experimented with a number of different configurations of layer size along with the number of layers.  As can be seen in the comparison graph the network with 7 layers did not perform very well and ended up no better than random guessing. We started to get better results when using 4 layers and used a number of different strategies to play with the size of each layer. These different sizes can be seen on the comparison graph in brackets next to the labels.

Some of the strategies that we used were, having each layer be half the size of the one before it. Picking a consistent size for all of the layers. Using the general rule of having  the total number of weights be 1/5 - 1/10 the size of the input. Having a consistent decrement in the number of nodes in each layer. And making educated random guesses for the values at varied steps going down by roughly 1000 with each layer.

After graphing these results a divide can be seen between the networks that performed well and those that had around a 50% accuracy. Looking at the networks that performed better they all had a larger number of nodes compared to the networks with lower accuracy. We also noticed that the accuracy of the network over the epochs was more stable with a larger number of nodes and that they were the ones with the most improvement. Many of the networks with less nodes stayed below 55% accuracy throughout the 100 epochs and some consistently performed below 50%.

We noticed that the rate that our network was improving tended to happen in increments. It would make an improvement in accuracy and then stay at the same position for a few training cycles before improving again. We believe this was due to the small sample and testing size that we were using. Using the database suggested we were only able to get 3000 samples but our word set had 5000 words in it and we were basing our input layer on that number which resulted in better testing results over the 1/5 -1/10 weights to input rule of thumb.

Running these tests we determined that our datasize needed to be larger to produce better, and more stable results. Having a larger amount of data would also improve the ratio between training size and the number of weights. By having more data we are certain that the performance would improve and there would be less variation between epochs as can be seen with the 4500, 4000, 3500, 3000 MLP the changes in accuracy between epochs were drastic at times. Although it did not perform well the network that used an appropriate number of weights: 300,120,80,40 was the only network that consistently made improvements even though it did stall at results over a longer period of epochs. So it would seem that this ratio stabilized the results.

After getting these results with our system we ran the data through the Sentiment Analyzer in the Stanford CoreNLP Library. The Sentiment Analyzer performed with a 72% accuracy on our dataset. This was calculated by the number of sentences that were classified correctly, though it is important to note that this system also classifies neutral sentiments that were classified as wrong in this calculation. The results to this can be seen in the NPL results chart. Because of the output given by the network we manually calculate the accuracy by putting the results into a text editor and using the find function to see the number of occurrences of:
“ negative
0”

“ positive
1”

“ negative
1
“
and
“ positive
0”
Denoting correct negative, correct positive, false positive, and false negative respectively. Taking these numbers we can also see that 463 of the sentences were classified as neutral.

Four of the network configurations that we tested performed over this accuracy, all of which had a high number of nodes in the network. Where as the smaller networks had a poorer performance compared to the CoreNLP.

For optimization we used the tensorflow gradient descent optimizer and the adam optimizer. The adam optimizer had interesting behaviour where it very quickly achieved good results before suddenly returning to a 50% error. These experiments did not appear to be improving the network as a whole so that line of testing was abandoned.

The best results that we achieved were by using the MLP with 4 layers, each of size 3000, Which is the current configuration of the network. With this we were able to achieve a maximum accuracy of 82.3% and finished after 100 epochs at 81% accuracy. A graph of this network is included as chart 3. A larger image can be viewed directly from the code using tensorboard which is included in the submission.

References:

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60. [pdf] [bib]

Bag-of-words model. (2017, December 14). In *Wikipedia, The Free Encyclopedia*. Retrieved 04:12, December 21, 2017, from https://en.wikipedia.org/w/index.php?title=Bag-of-words_model&oldid=815406316

Your Bibliography: Anon, (2017). [online] Available at: https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences [Accessed 21 Dec. 2017].

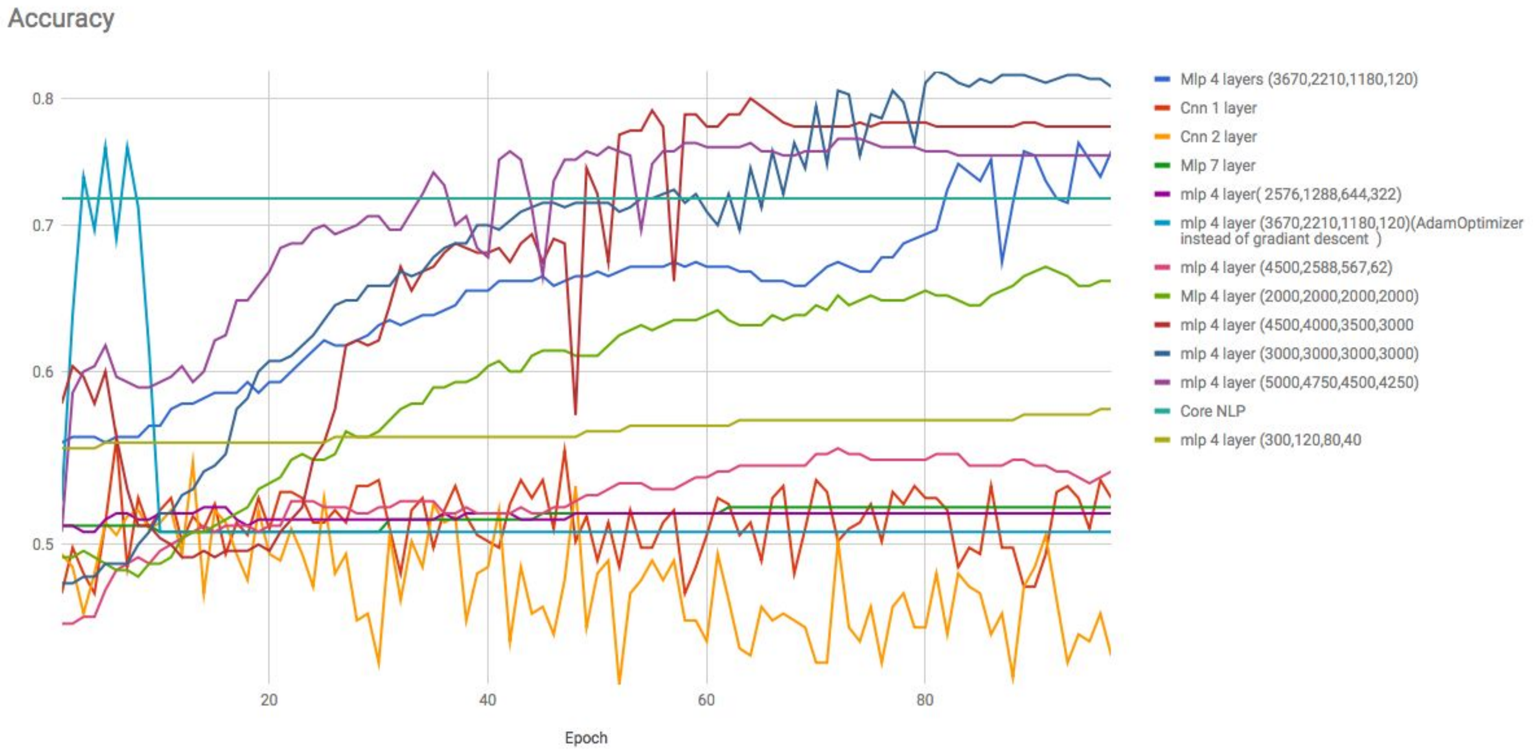

Chart 1. Comparison graph

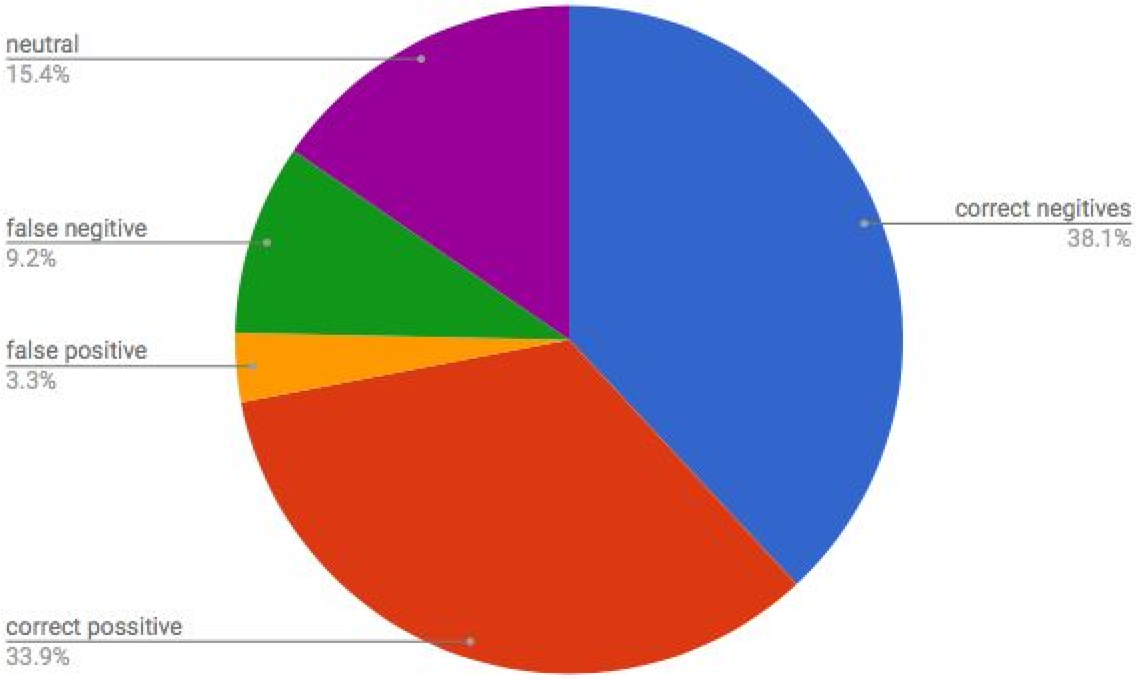

Chart 2.NLP results

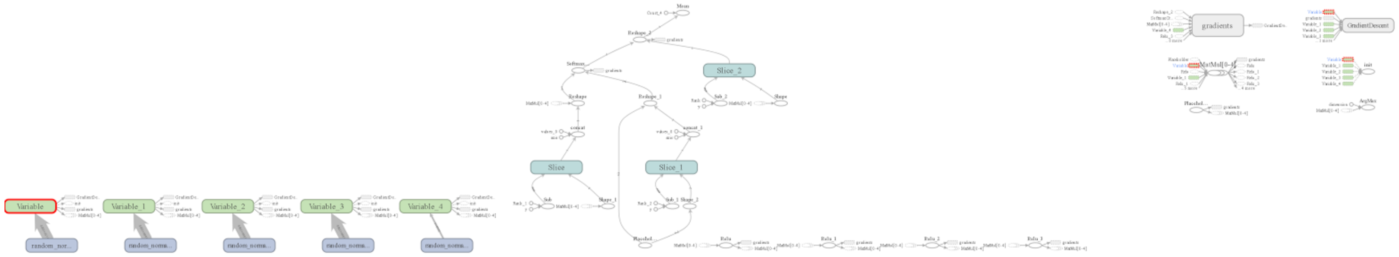

Chart 3.Network Structure