Team Project Report - Rough Draft

Blake Cromar

Jordan Tway

Kimberlee Simpkinson

Source of the data:
https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences
UCI Machine Learning Repository

For this project we wanted to build a text prediction algorithm. We started searching the web for text databases we could use. Amazon, Yelp, and IMDB datasets were found, which consisted of average everyday words. Given a word, and a few words preceding it in a sentence, we wanted to build an algorithm that would predict the next word in the sentence. Finally, we wanted to bring this all together in a nice wrapper to be user friendly and easy to read. While not a perfect word predictor, we have successfully accomplished the goals of this project.

The first dataset we found was a set of random sentences gathered from a collection of Wikipedia sites. After a while of pre-processing and cleaning the data we wanted to look for a better dataset that included words that were used in every day conversations. After searching a little more, we found review data from Amazon, Yelp, and IMDB.

There were many things we had to do to clean this dataset up. It was not 'off-the-shelf' ready. There were spaces where there shouldn't be, and a lack of spaces where there should be. There were formatting issues, and ways we had to arrange the data in a custom way to fit the algorithm we wanted to run. To be specific, we rearranged the data to include only one word per

cell. Furthermore, we removed unnecessary punctuation and reorganized the data so there was only one sentence per row.

A data modification step was required so the data could be fed into the machine. We ran an algorithm that picked a random word in the clean dataset, locate the previous 3 words along with next word, and make an array out of those words. This algorithm would repeat this step many times and finish by concatenating those arrays together to compose rows of the "machine ready" dataset. The data was composed of columns that were labeled by word position. In order those columns were -3, -2, -1, 0, +1, where 0 was treated as the last used word in the sentence.

There were a few algorithms we thought about using and that would be best for the problem we were trying to solve. We considered the apriori and decision tree algorithms. We thought maybe a decision tree would work if we had an attribute that labeled words as nouns, adjectives, etc., but we couldn't find a data dataset with that type of labeling. When it come down to using the apriori algorithm we shied away from it because we were concerned about the fact it didn't account for word order. At one point, we strongly considered a naïve Bayes classifier, due to it's low computational expense. We later changed our mind and decided to use neural network because we felt it would handle the complexity of are data better.

When it came down to designing the neural network we did a lot of experimentation on how to design the network. We tried many combinations of layers, node numbers, and activation functions. The only thing that proved to be useful was having a rectifier function for the output layer. The number of layers and node numbers didn't contribute to the accuracy of the machine. Because of this, one conclusion that we made was we needed much larger database so that the machine had more to learn from.

One challenge we ran into was when there was a word in the sentence the user was typing, that wasn't in our database. To solve this, we would label encode that word as the same number we used as a blank space. Another problem we ran into was if we were trying to predict a word when 3 or fewer words had already been typed. In this situation there aren't a complete set of 3 words before that current words. To solve this, we made it so that the dataset had blank spaces in the -3, -2, and -1 whenever needed.

Using a neural network algorithm, we were successfully able to predict the next word in a given sentence, and not only that, we were able to build a wrapper that works with this algorithm, making it more user friendly. As the user types their phrases into the textbox, our algorithm predicts the next word and creates a suggestion for them.

A text predictor has the potential to make a lot of revenue due to the reality of many people typing and messaging. It could be used for messaging, e-mail, word processing, etc. It could be sold for cheap and easily formatted to other software. While our text predictor isn't perfect a stakeholder would find this of value because they would see the economic potential as we improve the algorithm.

The results of our project are interesting because it really shows us how complex human speech is. One of the things we learned was the solution space for an algorithm like this is incredibly large. Careful attention to grammatical rules along with the individual's mannerisms is important for an accurate text prediction. It was always amusing to have the machine guess a word that made no sense to the context of the sentence you composed.

To improve the results the dataset would need to be composed of sentences the user has previously used. The algorithm will likely be limited in its prediction ability as long as a generic dataset is used. Making a dataset based on your text history creates an obvious ethical issue. If your text history was accidently leaked then unwanted social consequences might follow.

Even though made a good effort to make a text predictor, the predictions still weren't fantastic. While the predicted word went well with the previous word it didn't always fit the sentence as a whole very well. Another downside to our algorithms is updating and training the dataset with the neural networking is time consuming. Creating a more efficient text prediction algorithm, especially one that could be updated frequently with ease, would be a worth challenge.

If we were to start this project again, we would use a larger dataset that was more inclusive with types of and categories of words. With our machine ready dataset, there were many words that were repeated in the many sentences we had. To add on, there wasn't a large of a variety of words. Having a dataset that resolved those two issues would be something we would of included. The last thing we would of done was explored more potential algorithms that more efficiently processed the data while maintaining a specific threshold of accuracy.