**Using Yelp Business Reviews to Predict Potential Business Attributes**

CS525 Final Project

December 17th, 2021

Patrick Critz

Conor McDonough

Christopher Guerrette

Yash Patel

# Table of Contents

[**Table of Contents**](#_2l35m2exu08f) **2**

[**Introduction**](#_mahavnsveeu1) **3**

[**Literature Review**](#_7h6ki4hs1euf) **3**

[**Methodology**](#_n6agw7vojzgz) **4**

[**Results and Findings**](#_1nhb6c67jghk) **6**

[**Conclusions**](#_f1j8wjdkonkg) **9**

[**Works Cited**](#_nvn2nwz49sfs) **10**

# 

# Introduction

In this paper we discuss using Yelp business review data to try and predict potential business attributes. In total, the Yelp business data set contains 39 possible attributes that businesses can have. Not every business has a value defined for each attribute so our research can be used to try and fill in these gaps. To do the predictions, we are using multi variable label classification with both single and multi output layer models. We attempt to predict a set of seven binary attributes and four binary attributes and also investigate how only using review data for business with at least 1000 or at least 500 reviews affects performance. Finally we discuss our results and explore how future work could build upon what we have done here.

# Literature Review

Yelp has provided 5 different json files that contain information on businesses, check ins, reviews, tips, and user data. In our research we will only be exploring the business data and review data. The businesses dataset contains various info about a business like their name, where they are, and what attributes they have. From this data we were mainly interested in the provided *business\_id* and list of *attributes*. The attribute data consists of both binary and string related attributes. For example *BikeParking* can either be a *True* or *False* value but the *Alcohol* attribute might be one containing *‘beer\_and\_wine’*. Our focus was with binary attributes. The review dataset contains information about individual reviews like who wrote it and what business it is for, and of course, the review text itself. From this we needed the *business\_id* since it would tell us what business it was for and the *text* since that was the review itself.

Multi-Label Classification is a type of binary classification that allows the predicting of zero or more class labels. This is different from typical classification where an object is classified into one group. In this case, multiple mutually non-exclusive labels can be given to an object (Brownlee, 2020). This is useful if you are looking to predict what subset of attributes a business might have from a much larger group of potential ones.

Global Vectors for Work Representation, or GloVe, is an unsupervised learning algorithm for obtaining vector representations for words (Pennington, 2014). Glove uses the Euclidean distance (or cosine similarity) between two word vectors to provide an effective method for measuring linguistic or semantic similarity of words (Pennington, 2014). Figure 1 below shows an example of the relation between words that GloVe can make.

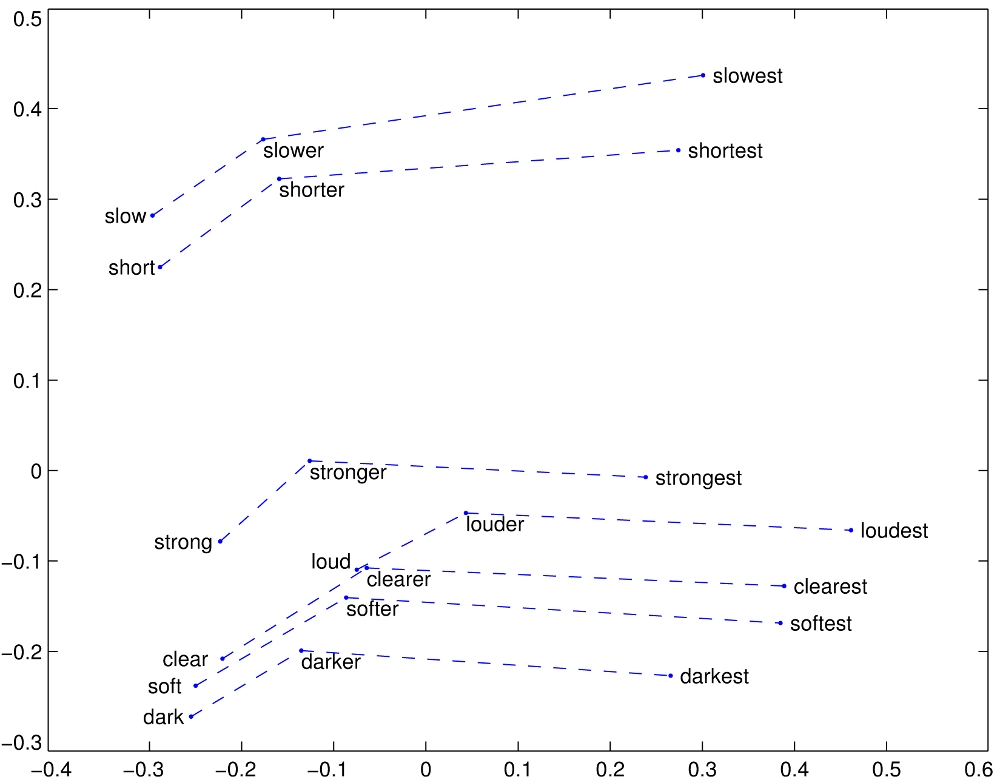


Figure 1: GloVe relation for comparative and superlative words

# **Methodology**

The work flow for our research can be roughly broken down into three separate parts:

1. Determining what attributes we want to predict and how to predict them.
2. Filtering the Yelp data and turning it into something we can use.
3. Writing the model code and testing it with our data.

Figure 2 below shows an overall design figure of our research.

Figure 2: Design Figure

Our first step on executing this project was planning what attributes to predict and what information and algorithms we would use to predict them. We defined two sets of attributes we wanted to predict. The first set had seven binary attributes while the second only had four. We believed we would be able to accurately predict all seven attributes of the first set; however, we wanted to ensure if we did not get accurate results, we could simplify the algorithm by using a subset of the original seven attributes. Figure 3 below outlines the attributes used in each set.

| **Set One: Attribute (Ranking, #Of Business)** | **Set Two: Attribute (Ranking, #Of Business)** |
| --- | --- |
| *BusinessAcceptsCreditCards (1/39, 120177)* | *GoodForKids (7/39, 56850)* |
| *BikeParking (4/39, 76480)* | *OutdoorSeating (9/39, 50128)* |
| *GoodForKids (7/39, 56850)* | *WheelchairAccessible (19/39, 29370)* |
| *OutdoorSeating (9/39, 50128)* | *DogsAllowed (22/39, 18308))* |
| *HasTV (13/39, 44495)* |  |
| *WheelchairAccessible (19/39, 29370)* |  |
| *DogsAllowed (22/39, 18308)* |  |

Figure 3: Prediction attributes for each set

Our thought process behind choosing these attributes was that they were somewhat central aspects of a business and would be addressed in reviews in some fashion. We would then be using the review text to help predict the business attributes. The Yelp review data is about seven gigabytes in size and we wanted to narrow down what reviews that we actually wanted to use. We decided to have two sets. One set would include all reviews from businesses with at least 500 reviews and the other would have reviews from all businesses with at least 1000 reviews.

Now that we knew that data we wanted to use, we had to filter the Yelp json datasets and turn it into something that we would be able to use. Using a C++ script, we filtered the Yelp review and business datasets to create four csv files that we would use for testing. Each attribute set that we determined earlier would be associated with both the 500 review and 1000 review set, thus there were four files in total. Figure 4 below shows how the data in each csv file was formatted. We also use the “|” character as a delimiter since many of the reviews contained commas.

| Review | Business\_id | Attribute 1 | Attribute 2 | … | Attribute n |
| --- | --- | --- | --- | --- | --- |
| Review | 24bxH8U1DRu1 | 0 | 1 |  | 1 |
| Review | 24bxH8U1DRu1 | 0 | 1 |  | 1 |
| Review | 24bxH8U1DRu1 | 0 | 1 |  | 1 |

Figure 4: Table showing organization of .csv data

Converting to a csv file was an important step because it would make the data much easier to deal with when training our models.

With these reviews, we needed to present the text to the model in a fashion where it could effectively utilize its information in order to predict the presence of binary attributes. In order to do so, we decided to use Global Vectors for Word Representation (GloVe). There are several reasons why GloVe was chosen to encode the text. One reason is its ability to identify the nearest neighbor of words (Pennington, 2014). We thought this would be useful when dealing with synonyms of attribute related words that show up in the review text.

Now that the data was properly formatted, we were ready to analyze the information and predict the attributes. We did this by creating a multi-label text classification model. To help with the creation of our model code, we referenced the tutorial “Python for NLP: Multi-label text classification with Keras” by Malik Usman. In one model, we utilized a single dense layer and had 4/7 outputs, depending on which set of attributes we were attempting to predict. We also used sigmoid activation functions along with binary cross entropy loss functions. We also ran the models with 5, 10, and 50 epochs. Once our model was created, we were ready to run the model.

# Results and Findings

Before running all of our models, we had a few expectations on how we believed that they would perform:

1. We believed that it would be easier to predict the set of four attributes more accurately than the set of seven attributes because the training would be less complex and there would be less ways for the model to produce an incorrect output.
2. We also believed that we would see a higher accuracy with the models that considered business with at least 500 reviews rather than 1000 reviews simply because there would be more data.
3. Finally, we thought that the more epochs we ran the models for, the higher it’s accuracy would be.

We found that, as we thought, our first expectation was correct about the models. Figure 5 shows the results from the Single Output Layer Model Test Accuracy. The six models that were predicting 4 attributes had an average test accuracy of 70% whereas the six models predicting 7 attributes had a test accuracy 21%. It is quite obvious in this case that the model struggled with having more attributes to predict.

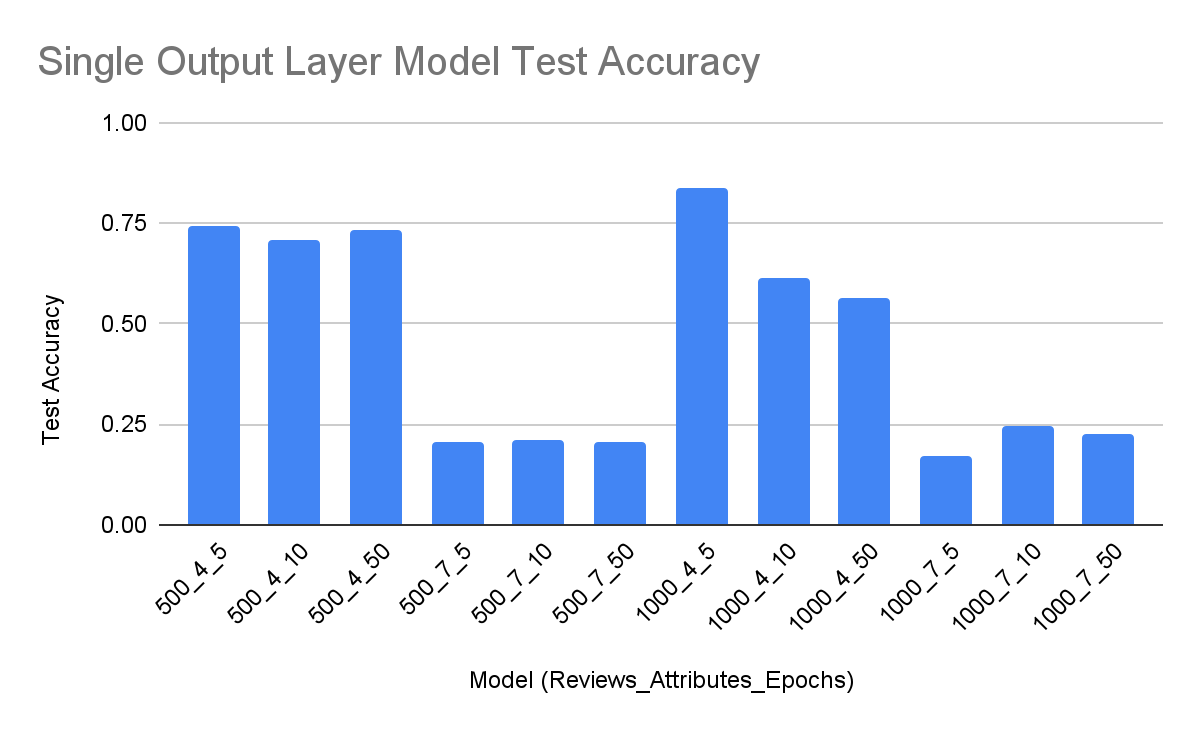


Figure 5: Single Output Layer Model Test Accuracy. X-axis data is defined by <number of minimum business reviews>\_<number of attribute predictors>\_<number of epochs>

We also found that including more review data in our model did not help the accuracy as much as we thought it would. Based on the data in Figure 6, the test accuracy only increased by 6% when including all reviews from businesses with at least 500 reviews when predicting four attributes. When predicting seven attributes, we even saw a decrease in accuracy from 22% to 21%.

| **Model Data** | Four attributes | Seven attributes |
| --- | --- | --- |
| At least 500 reviews | 73% | 21% |
| At least 1000 reviews | 67% | 22% |

Figure 6: Test accuracy average of each model for each unique group of data

For our final assumption, we learned that increasing the number of epochs used did not mean a better model performance. There was not a single model out of the four groups of data where the 50 epoch run had the best performance and only two out of four where the 10 epoch run had the best performance.

We also tested the same data with Multi Output layers to see how it would compare to the Single Output layer. Based on the results in Figure 7, we did not see better overall performance, however the data was much more consistent. The standard deviation of the Single Output Layer models were 0.26 whereas it was only 0.05 for the Multi Output Layer models. This means that the Multi Output Layer models were much better at handling more attributes and could perhaps be used to try and predict even more.

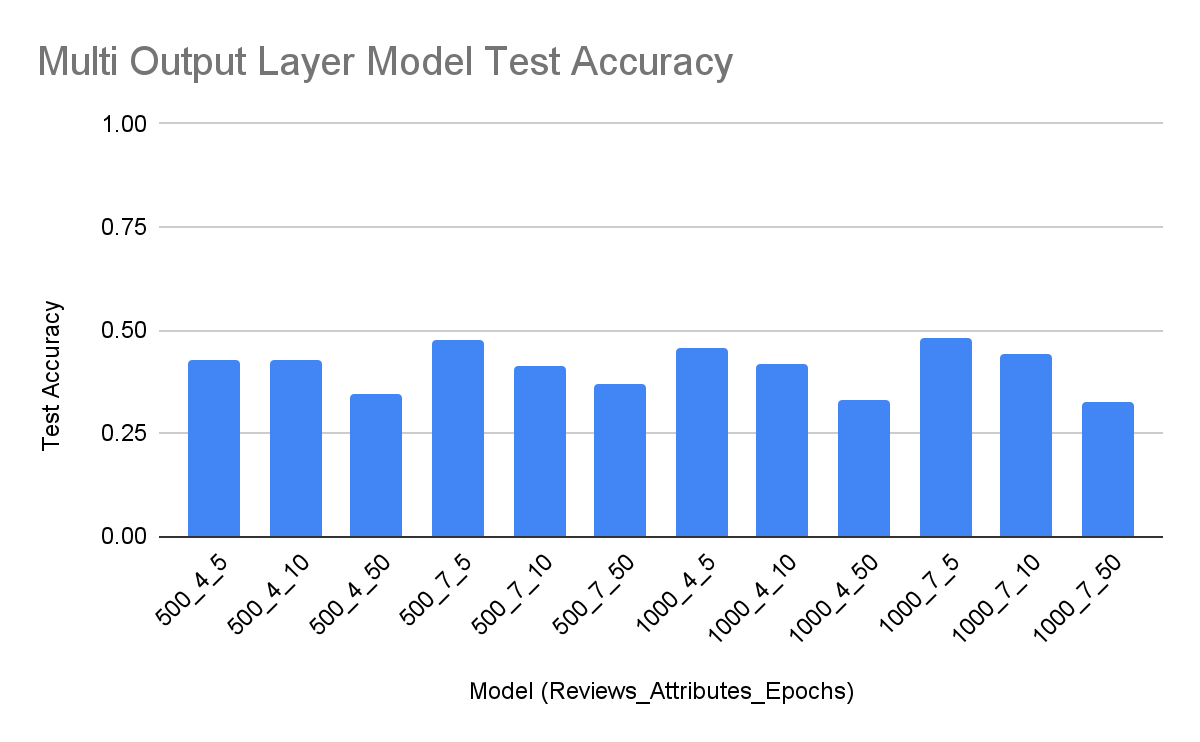


Figure 7: Multi Output Layer Model Test Accuracy. X-axis data is defined by <number of minimum business reviews>\_<number of attribute predictors>\_<number of epochs>

As with the Single Output Layer models, the Multi Layer Output models did not benefit from using more epochs when running and for each of the four unique groups of data, the test accuracy decreased as the epochs increased.

# Conclusions

Overall, we were able to produce a few models that could predict a subset of attributes with an accuracy of 70% - 75%. We were happy to see this success but know that there is much more that can be done. If we want to try and predict more attributes at once, we can either move forward with the Multiple Output layers and tweak the way that the model runs or change the way that we encode review text data. Our current model looked at individual reviews but we have discussed the possibility of concatenating an entire set of business reviews and using that as input for the model. Furthermore, we can try different encoding techniques like using word2vec rather than GloVe. Going even more beyond what we have done in this paper, different business data could be included in the prediction process, such as tips. All in all, we were satisfied with our results and we look forward to pursuing alternative ways to improve our model in the future.

# Works Cited

Brownlee, J. (2020, August 30). Multi-label classification with Deep Learning. Machine Learning  
 Mastery. Retrieved December 14, 2021, from  
 https://machinelearningmastery.com/multi-label-classification-with-deep-learning/

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word

representation. Retrieved December 14, 2021, from https://nlp.stanford.edu/projects/glove/.

Usman, M. (2019, August 27). Python for NLP: Multi-label text classification with Keras.

Stack Abuse. Retrieved December 14, 2021, from https://stackabuse.com/python-for-nlp-multi-label-text-classification-with-keras/.

| Name | Contribution |
| --- | --- |
| Conor McDonough | * Multi output layer classification code * Collecting Results * Paper * Presentation |
| Christopher Guerrette | * Single output layer classification code * Collecting Results * Presentation * Paper |
| Patrick Critz | * Parsing and formatting code into csv * Collecting Results * Presentation |
| Yash Patel | * Single output layer classification code * Multi output layer classification code * Ran the different datasets * Obtained the results * Presentation |