Improving Yelp Predictions with NLP and clustering

Introduction to problem:

Yelp is a user-driver review platform for businesses in cities all over the globe. Many users are attracted to Yelp for new business discovery. They may be traveling and looking for a good restaurant, in the market for a new hair stylist, or looking for the best deals on car service. Yelp could improve their user return rate by improving improving the rating system and motivating users to return to the site. This strategy could also motivate users to write more reviews, because a larger number of reviews allows better classification of the user, and thus better recommendations.

One way to do this would be to show users businesses they are likely to rate highly based on similarity to other business they have previously rated well. An additional improvement could be to cluster reviewers into groups with similar preferences and show users businesses that were rated highly by other users like them.

The goal of this project is to identify methods to improve recommendations for novel businesses based on business and user similarity derived from review texts.   The client is Yelp. A more accurate recommendation process could increase the number of users on Yelp and encourage them to spend more time on the platform. Results may inform similar strategies with other recommendation platforms.

Part One: Initial Data Wrangling and Processing:

Yelp provides a subset of their review dataset for academic purposes that can be used to test various strategies for improving reviews. The subset is available at https://www.yelp.com/dataset and consists of over 5,200,000 reviews of over 174,000 businesses written by over 1,3000,000 Yelp users. The dataset is structured as multiple JSON files and this project use data from the business, user, and review datafiles.

For this project, reviews written for restaurants will be used because this is the most-represented category in the Yelp dataset.   To build the text corpus, reviews from the restaurant, food, and bar categories were used. The first step in the data processing was to identify the user and business ids for these categories and subset the dataset. This process is contained in the “yelp_subset_final” Jupyter notebook. The final subsets contained 33926 businesses and 197,837 reviewers that met the above criteria.

The review texts were subsetted using the ‘codec.open’ command, which can cycle through the reviews line by line without opening the full document. This is essential because the full file is large. Each review was tested against the list of businesses and users to include and written to a new output text file line by line. It was also essential to remove any escaped whitespace characters from the reviews, specifically newline (\n) and carriage return (\r). If these characters are not removed, they will produce a newline in the output text file and split single reviews into multiple entries.

Part Two: Can user approval be predicted with review text?

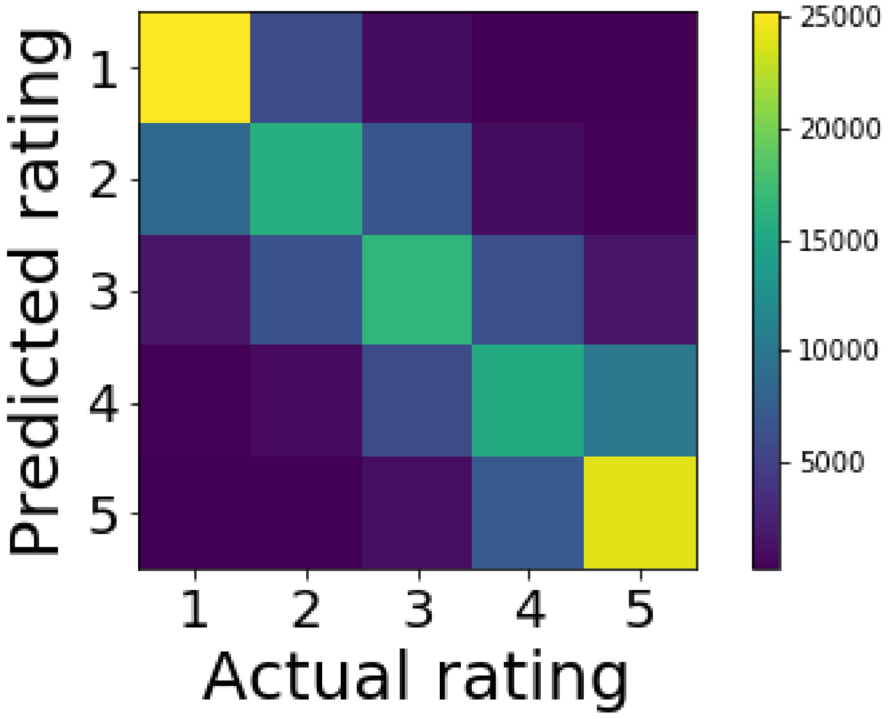
In the Yelp system, reviews consist of a numerical approval rating based on stars (1-5, worst to best) and a text-based review. The first step towards building the recommendation system is to demonstrate that the information in the text component of the review can be used to predict the star rating component of a review. This step is documented in the ‘star_prediction_from_text_final” Jupyter notebook.

First the review text and star ratings from the filtered dataset generated in part one are read into python and ‘balance_classes’ is used to generate a dataset with the same number of reviews in each of the star rating categories. Using a balanced dataset will reduce bias in the predictive model. In this case, we are limited by the number of one-star reviews and the balanced dataset contains 99,626 reviews in each of the five categories. The balanced dataset is then divided into a training (67%) and a test dataset (33%).

Next the text data are vectorized using a Term Frequency Inverse Document Frequency (TF-IDF) approach. This technique weights the words by the inverse of their appearance in the whole dataset. This means words that occur very frequently, like “the”, and weighted as less important than more uncommon words, like “pizza”. This algorithm is implemented with the ‘TfidVectorizer’ function.

Next a classification model is built using a linear support vector machine (SVM). This technique works well with text classification problems and is used to find a relationship between the review text vectors and their star ratings. The model was tested by predicting the star ratings of the test portion of the dataset and comparing the output to the actual ratings.

The accuracy score for the SVM classification model is 0.592. This indicates the model correctly classifies the number of stars the review will have based on the text of the review 59% of the time. This is substantially better than the 20% classification success rate expected if a star rating were chosen at random. The model was also evaluated using a confusion matrix comparing the predicted star rating to the actual star rating.



	Actual 1	Actual 2	Actual 3	Actual 4	Actual 5
Predicted 1	25212	5966	1029	229	220
Predicted 2	8665	16033	6920	1071	365
Predicted 3	1761	6516	16574	6360	1618
Predicted 4	364	940	5903	15430	10095
Predicted 5	289	245	1173	7351	24054

Most reviewers are interested in whether they will generally like or generally dislike a business, not in the distinction between a four-star and a five-star business. I built a second model that collapses the four- and five-star reviews into a category called “positive” and the one- and two-star reviews into a category called “negative”. This new model had an accuracy score of 0.96 and was also evaluated using a confusion matrix.

	Actual negative	Actual positive
Predicted negative	63059	2651
Predicted positive	2503	63341

This part of the analysis demonstrates that the rating of a restaurant can be predicted with a high degree of accuracy based on the text of the review.

Part Three: Using LDA to reduce text dimensionality.
In the previous step, the reviews were represented as vectors of individual words. Because, each possible word in the corpus was an item in the vector, the dimensionality was high, over 584041 words. The vectors were also very sparse, containing a lot of zeros, because each individual review contains a small percentage of the total possible words. Finally, representation of each word as an individual token does not preserve any information about the relationship between the words. For example, with each word represented at a token, “cake” and “pie” are no more similar than the words “cake” and “oil change”. In fact, “oil change” is not represented as a distinct concept, but as two individual words; “oil” and “change”.

In this section, I use lemmezation and topic modeling with Latent Dirichlet Allocation (LDA) to reduce dimensionality and group words into topics. This analysis is contained in the ‘Yelp_nlp_final’ Jupyter notebook. First, bigram and trigram phrases were identified in the dataset. Bigrams are two-word phrases that represent a single concept, like ‘ice cream’, and trigrams are three-word phrases, like ‘gluten free bread’. Next, I removed stop words, like “are”, “be”, and “do”, from the review text. These words re very common and unlikely to be useful in classifying the reviews. There are 305 words in the spacy library stop words list that were removed from the reviews.

After processing, topic modeling was used to summarize the reviews. Specifically, I used latent Dirichlet allocation (LDA) to identify 50 latent variables, or topics, that can be use to describe the dataset. This method is unsupervised and returns a list of words belonging to each topic. I then manually assigned a category name to each topic to make them more interpretable. For example, category four contained words like ‘sushi’, ‘roll’, ‘sake’, ‘tempura’, ‘wasabi’, ‘miso_soup’, and ’rice’. It was given the title “Japanese” because it contained words associated with Japanese style food. The topics identified fell into three major categories, food type, location, and service/ambiance, outlined below.

Food type	**Location**	**Service/ambiance**
Chinese	Toronto	smell
Thai	UK	service
healthy	New York	experience
Japanese	beach	parking
Asian soup	location	bar ambiance
grocery	coffee shop	good service
comfort food	general restaurant	fun ambiance
Greek	Las Vegas	young
Hot wing	Montreal	high end
breakfast	deli	wine & dine
sweet	airport & delivery	good taste
pubs		happy hour
non-alcoholic drinks		amazing

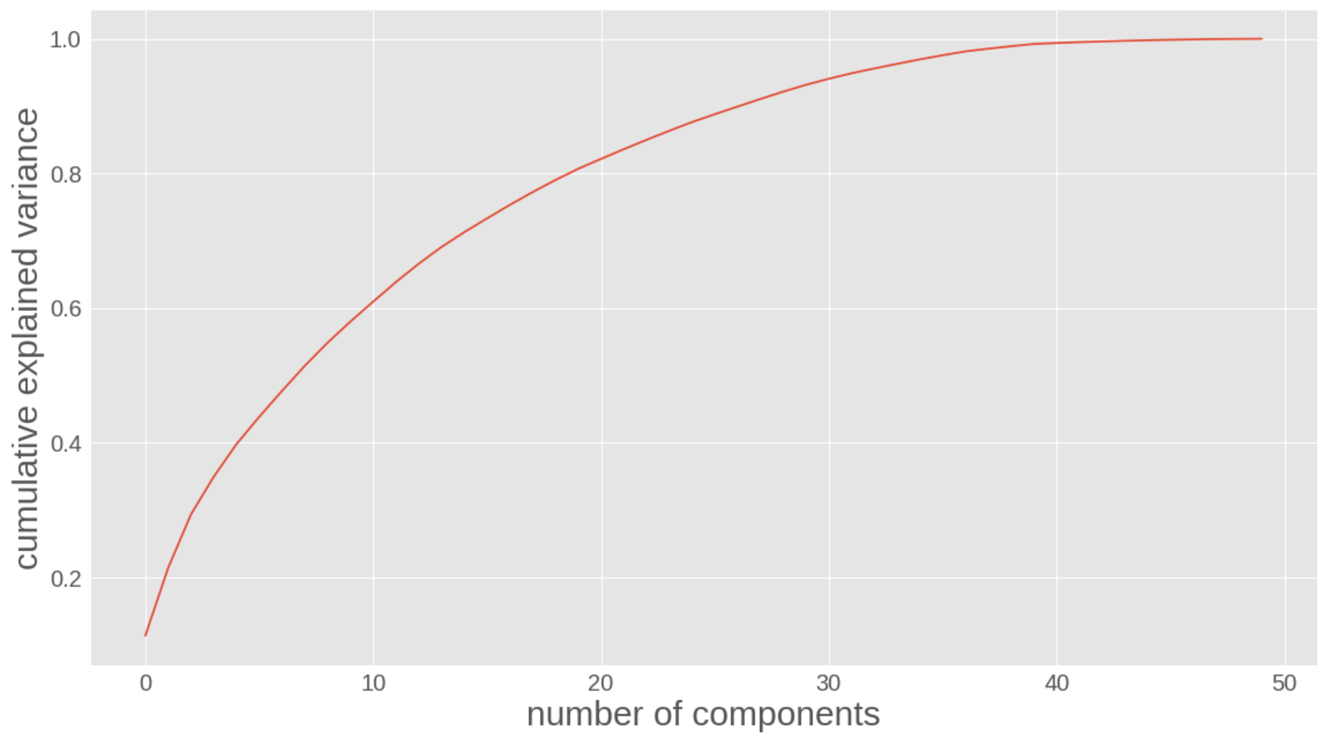
desserts		time
Mexican		buffet
Vietnamese		money
BBQ		general review
French		
pizza		
street taco		
Burger & fries		
Italian		

The LDA model was then applied to the review texts grouped in two ways. The user text file grouped all of the reviews by their author and produced a vector for each reviewer of the percentage of words in each topic in the reviews written by each user. The business text file groups the reviews by the business reviewed. The output vector contains categories of words frequently used to describe the business in the reviews. These analyses are contained in "user_subset_generator_csv_final.py" and "review_subset_generator_csv_final.py".

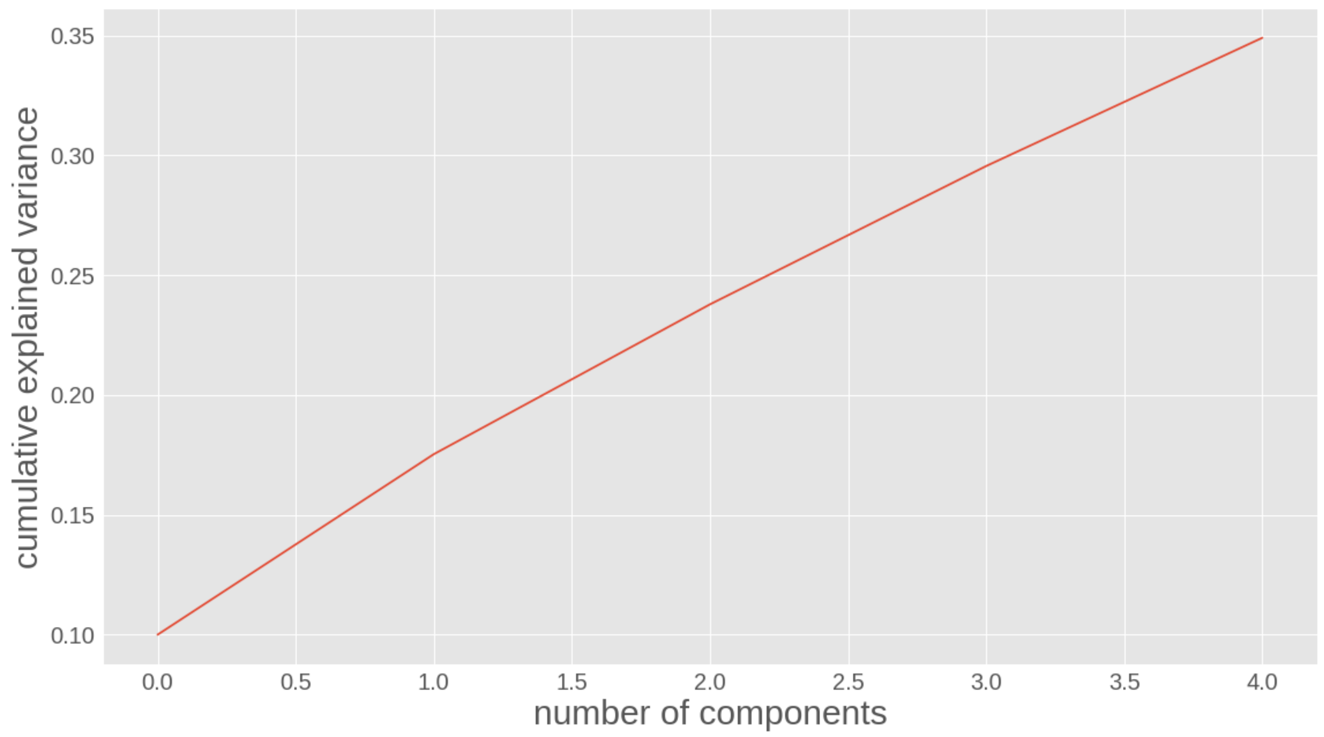
Part Four: Clustering Yelp Businesses and Reviewers
The final component of the analysis is using the LDA classified text to cluster users and businesses into groups based on similar text usage. This analysis is contained in the 'business_clustering_with_pca_final" jupyter notebook.

Principle components analysis:
Principle commonness analysis (PCA) was tried as an option for further reducing dimensionality before clustering.   Percentage of variance explained was plotted with the number of clusters used. Unfortunately, PCA dramatically reduces the variance explained by the variables, particularly for the user dataset.

PCA optimization: business dataset

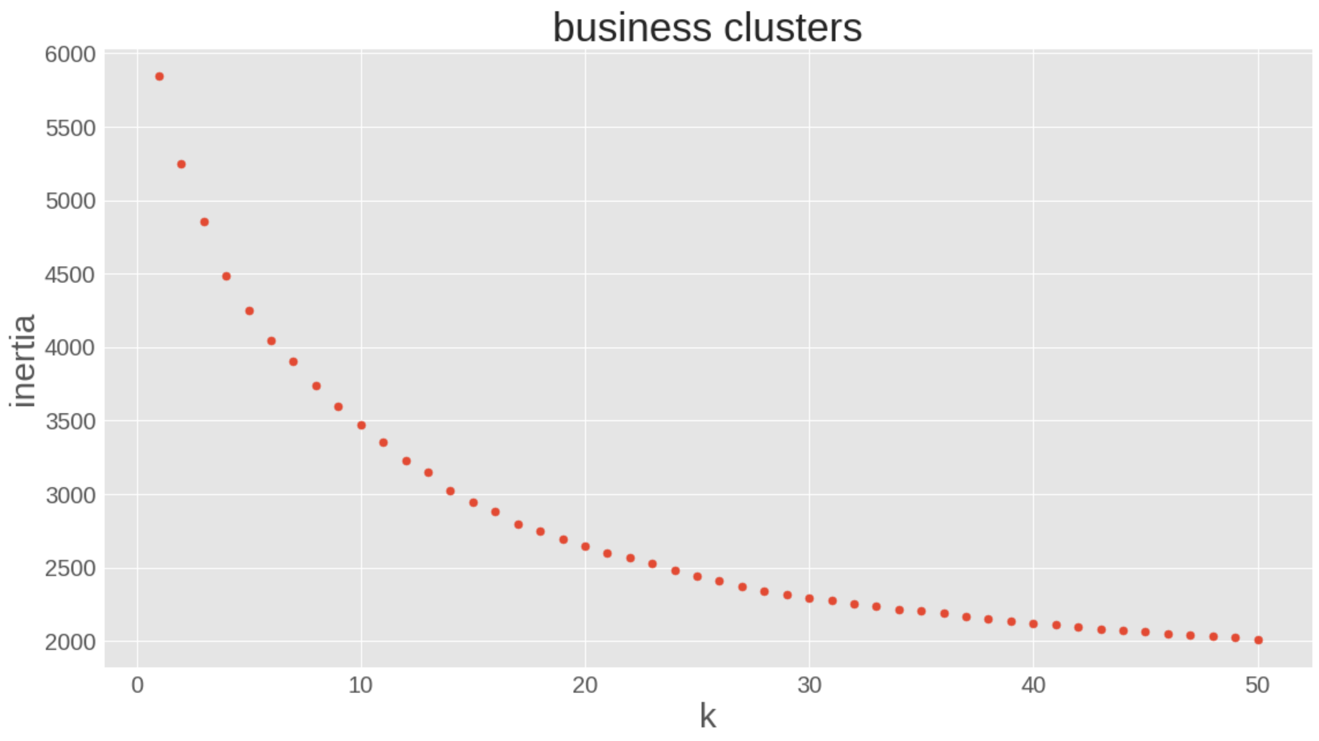


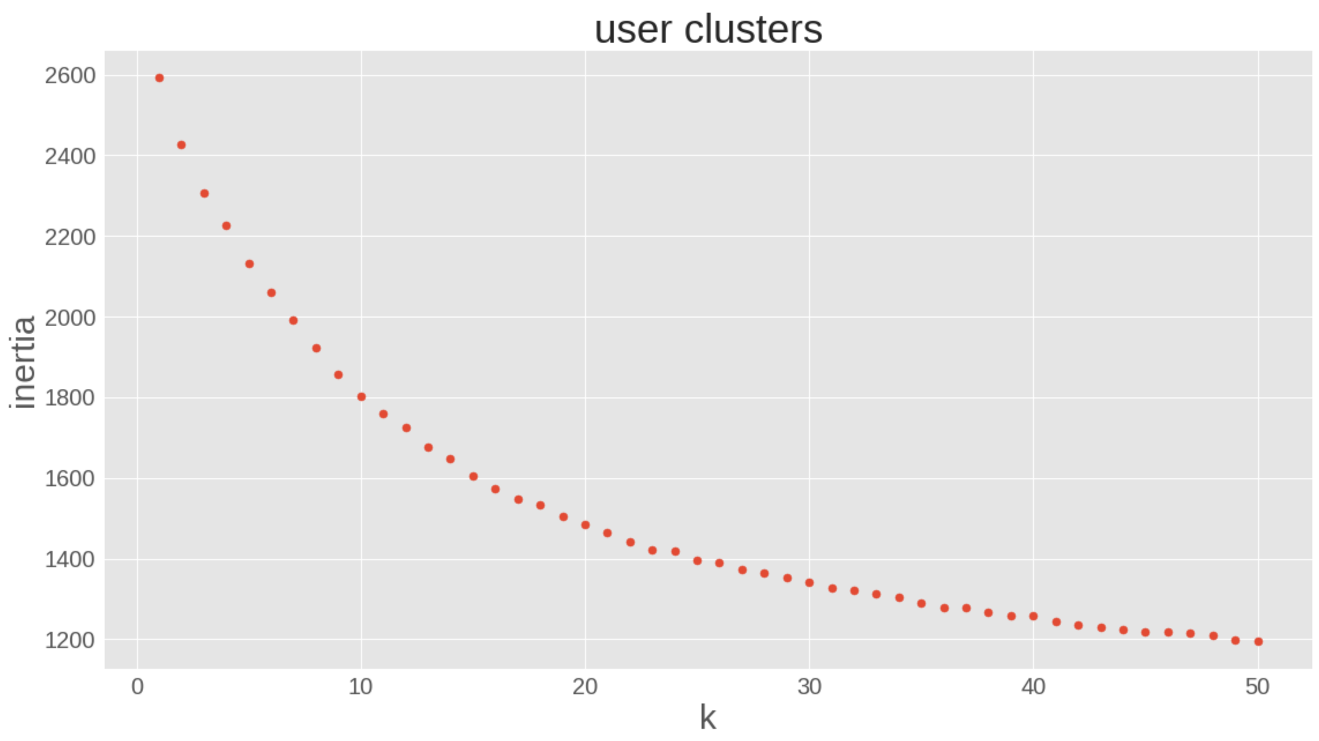
PCA optimization: user dataset



Clustering:

Several clustering methods were tried to generate business and user groups. Including k-means, affinity propagation, spectral clustering, and DBSCAN. All of these methods are unsupervised and contained in sklearn. Unfortunately, none of the methods tried produced high quality clusters, indicating that the features derived from the topic modeling and insufficient to cluster the dataset. Below are inertia vs. cluster number plots for the k-means clustering method. With this method, to optimize the number of clusters these plots are made to identify an ‘elbow’, the value for k where inertia stops decreasing as rapidly. In this case, there is no clear ‘elbow’.





Part Five: Conclusions and Recommendations

This analysis produced two major conclusions: that Yelp review text can be used to predict the star rating with a high degree of accuracy and that clustering business type and reviewers based on the LDA topics contained in their reviews is generally ineffective.   The first conclusion is indicated by the SVM model that correctly predicts from the text if the star rating will be positive or negative with 96% accuracy. This result suggests that the general idea of using text characteristics to predict whether a user with like or dislike a business is worth pursuing. However, the LDA topics identified did not yield high-quality features to be used in clustering analysis. My recommendations would be to investigate different methods of summarizing the text data (other than LDA) to develop more informative features. Other features encoded in the Yelp business files, such as operating hours and parking availability, could also be included.