Thinkful Unit 4 Capstone

Yelp Review NLP Classification
Predicting Author from Text Content

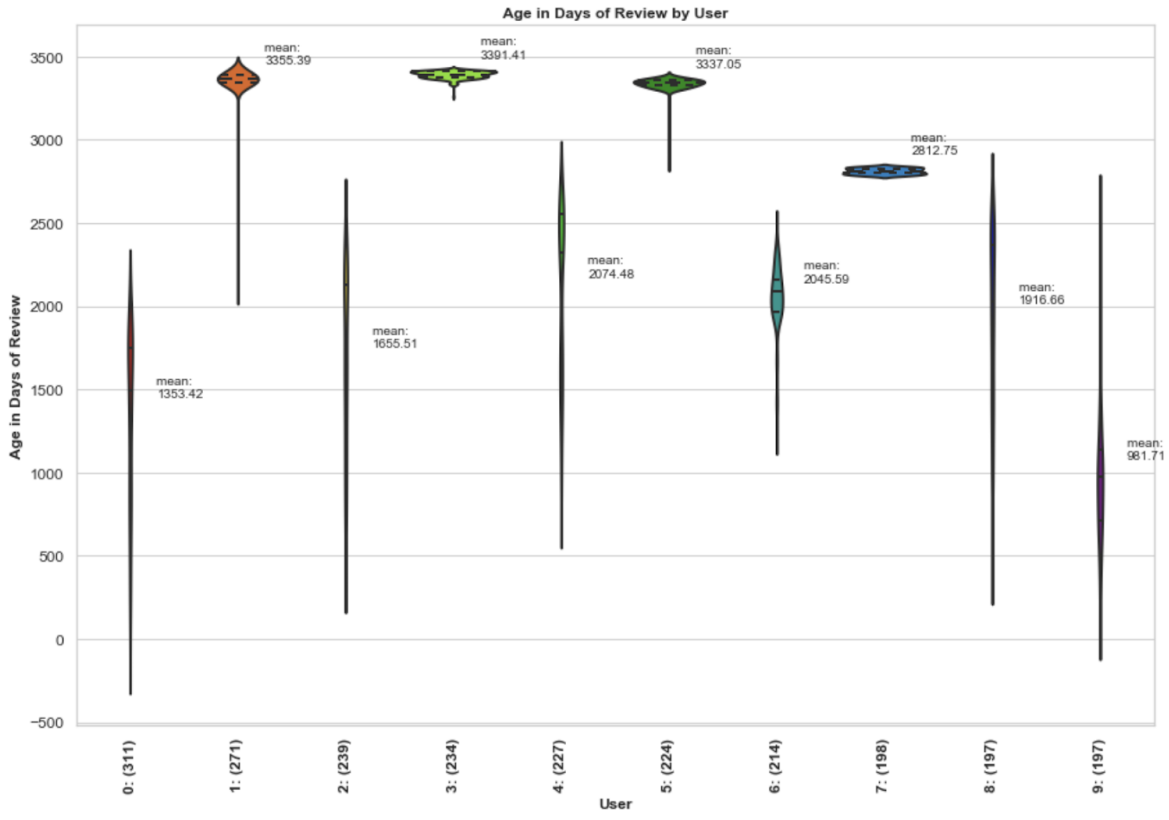
Introduction

Yelp reviews can cover a wide range of topics, depending on the types of businesses reviewed by users. It may be possible to identify users based on which types of businesses they typically review, the descriptive words they use, or by other features of the reviews themselves. Here we will choose ten prolific Yelp users, who have each written about 200

review or more, and attempt to build a classifier that can determine their identity using different sets of text features.

Such a model could be used to help prevent abuse of online ratings systems. In the case of Yelp, either irate customers or struggling businesses might sign up for multiple fake user accounts to artificially weight the star rating to be lower or higher than it ought to be, respectively.

Exploratory Data Analysis

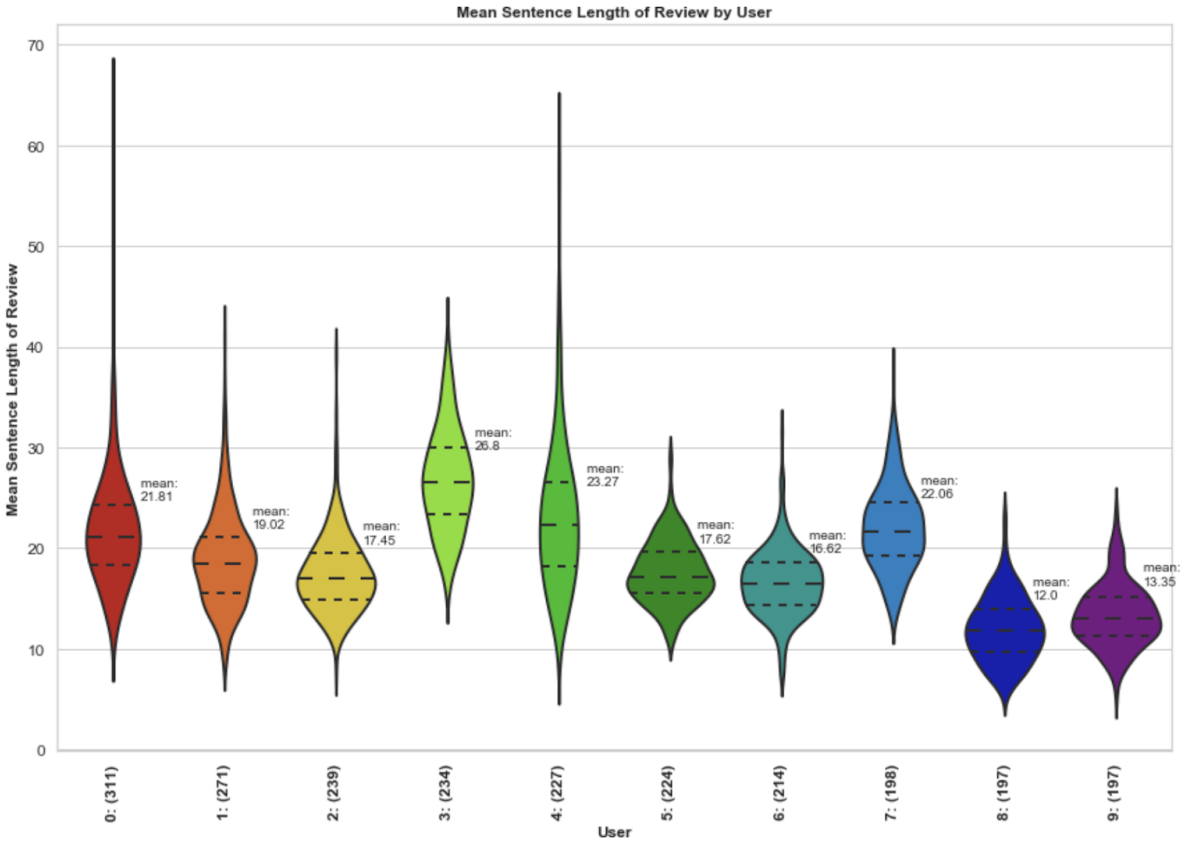


Using the information from the “date” column, we were able to extract the month of the year the review was written, as well as the age of the review in days. Users 1, 3, 5, and 7 were each active for a significantly shorter time than the others, with the bulk of the reviews of each of those users occurring within approximately a three-month period (each of these users posted a few reviews after their densest posting period, but appeared to trail off in their use of the platform.) The users displayed much more consistent user over a

longer period of time. These differences could have implications if we were looking at user retention, but that is not the main focus of this project.

The other immediately accessible features of this dataset are business_id, user_id, text (body of the review), stars (points assigned by user), “cool,” “funny,” and “useful” ratings (assigned by other users), and individual review_id. Of these, the information of the highest priority for the scope of this project is found in the user_id and text columns. Users 6 and 9 are standouts in terms of high mean “funny,” “cool,” and “useful” scores, while User 8 had the lowest mean score in all three of these categories.

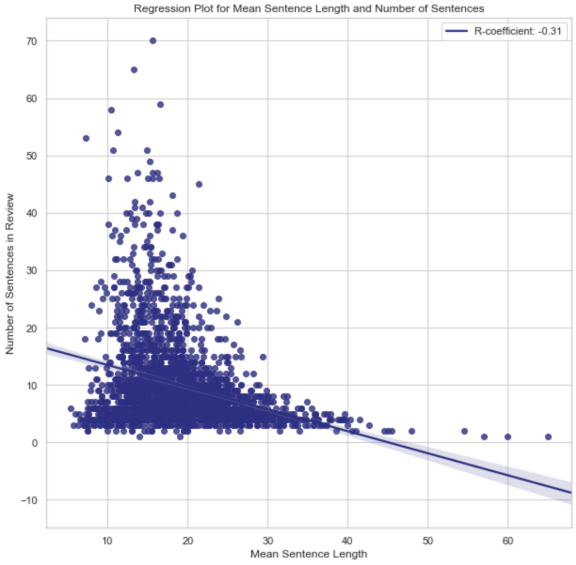
Without looking at the words of the reviews themselves, we were still able to extract features based on the length in sentences of each review, and on the sentence length. Users 6 and 9 had high mean sentence counts, with Users 3, 4, and 8 having the lowest. Users 0 and 4 had the highest mean sentence lengths, while 8 and 9 had the lowest.

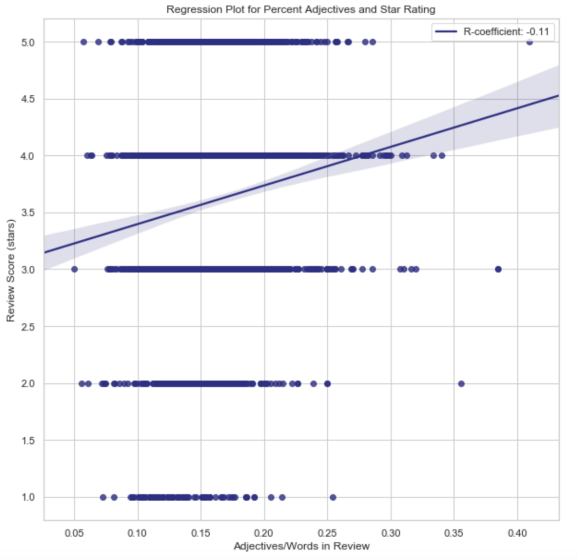


Users 6 and 9 had the highest overall descriptive word counts (adjectives and adverbs) in their reviews, but the mean of the ratio of descriptive word count to total word count was fairly consistent from user to user, except for user 8, whose reviews showed a wider variance and a higher mean adjective/adverb count per review. After a glance at the text of their reviews, it seems this is likely due to User 8's consistent (perhaps excessive) use of the words "pretty," "really," and "very" as modifiers for other adjectives.

Users 6, 8, and 9 wrote reviews that were distinct from the others in terms of document characteristics such as word counts, sentence length, and vocabulary, as well as the ratings ascribed to the reviews by other users. We'll be searching for features of the texts themselves that distinguish them from other users.

Looking at all the reviews summarily, there is an weak-to-moderate inverse correlation between the mean length of each sentence in a review the number of sentences in that review. Unsurprisingly, there is a strong positive correlation between number of sentences and adjective/adverb count, as well as between mean and maximum sentence length. There is a weak positive correlation between adjective/adverb count and star rating, and an even weaker positive correlation between mean sentence length and rating.





Clustering Non-Word Features

Before looking closer at the text, we attempted to use clustering models to find distinctions between the users in our data. After dropping highly-collinear features from the model, we tried K-Means clustering, with 10 expected clusters, but the vast majority of the data was assigned into one cluster. Mean-shift clustering performed poorly as well, lumping nearly all the data points into one cluster. Spectral clustering seemed to assign points to clusters somewhat evenly, not coinciding with the predetermined labels. DBSCAN split the data into two clusters, with users 0, 2, 4, 6, and 9 mostly in one cluster, and users 1, 3, 5, 7 in the other, with user 8 split more or less between the two. This division is most likely due to differences in the site activity over time for the two groups of users, not features of the reviews themselves. Clustering at this stage of the modeling process proved ineffective.

Word Data Preprocessing & Cleaning

Because we used using several different text modeling utilities, we had to clean the data in different ways to meet the requirements of each model. Our Bag of Words model used SpaCy-processed inputs, and our other models (TF-IDF, Word2Vec, and GloVe) all used inputs cleaned and processed using NLTK. We used SpaCy earlier to extract our sentence lengths and number of sentences, while using NLTK's word tokenizer to help get adjective/adverb counts.

Because we could implement a list of stop words directly in our Bag of Words function, we did not remove them in the "cleaned_text1" feature generated with SpaCy. For the NLTK-parsed "cleaned_text2" feature, we removed stop words using NLTK's default list for English.

Classification Modeling

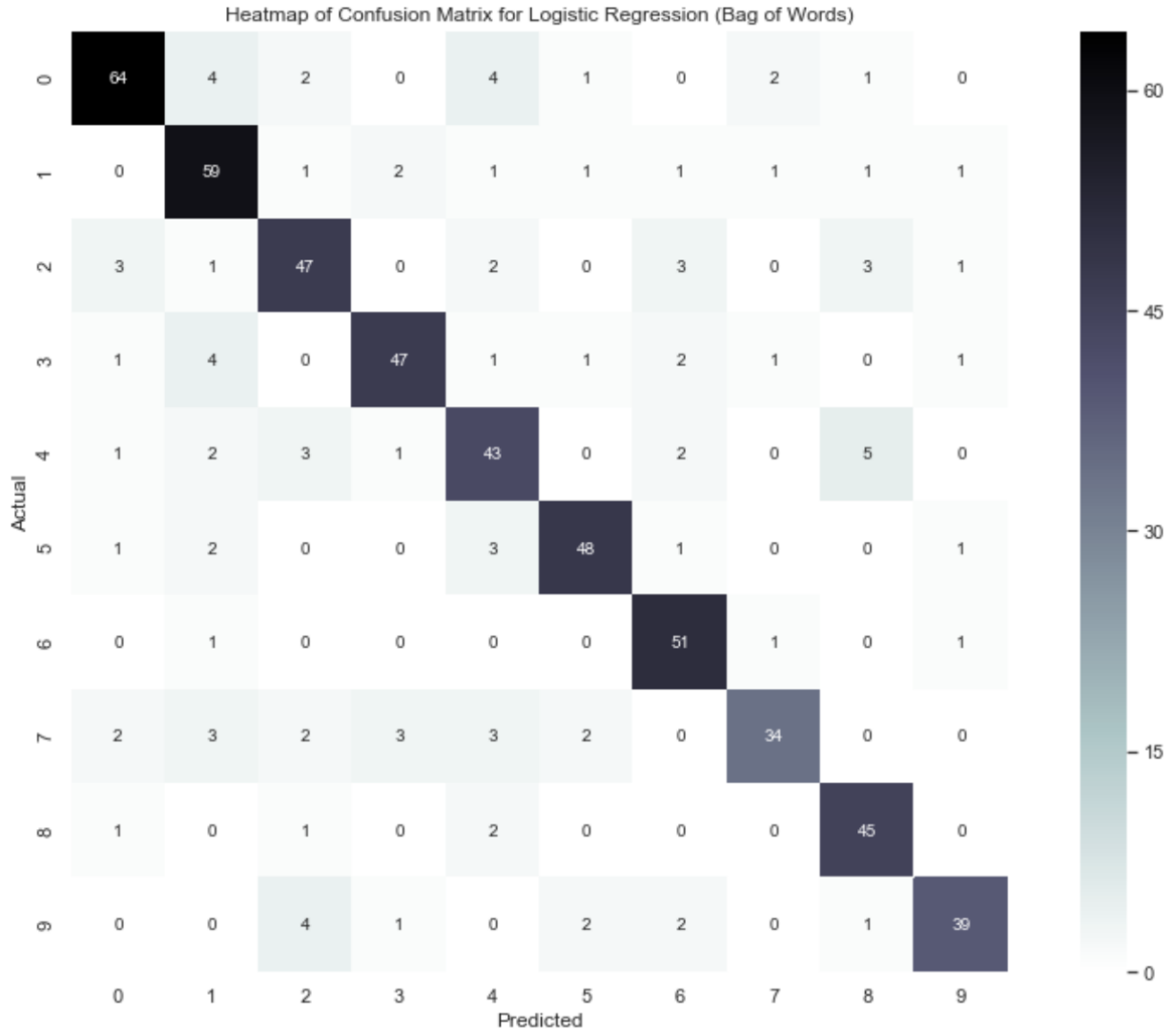
Our next step was to begin testing different classifiers applied to various NLP models. We started out with Bag of Words using SpaCy preprocessing, and then tried three vector models: a TF-IDF vectorization of our text corpus, and two pretrained models (GloVe trained on Twitter sources and Word2Vec trained on Google News) applied to the text corpus.

Bag of Words Model

To create our bag of words model, we took a number of unique words from each user's combined corpus of reviews equal to an arbitrary baseline number (in this case, 500 words) plus a proportion of the words above that threshold (in this case, approximately one third). We used this combined approach rather than the threshold or proportion by itself in an effort to avoid over- or under-representation of any given user's unique words. Once a bag of words had been created for each user, they were combined as a set, eliminating redundant words. From here, we created a sparse matrix with 5112 word features of the number of times a word in our combined bag appeared in each review.

The sparse matrix of word counts wasn't *precisely* binary data, but it was close enough to it that naive Bayes could be a good candidate for a classification model. Initial results with multinomial naive Bayes classification were quite promising, with an f1 score of 0.78 using the macro average method. The next classifier we tried was logistic regression, this time achieving an f1 score of 0.82, setting a new benchmark.

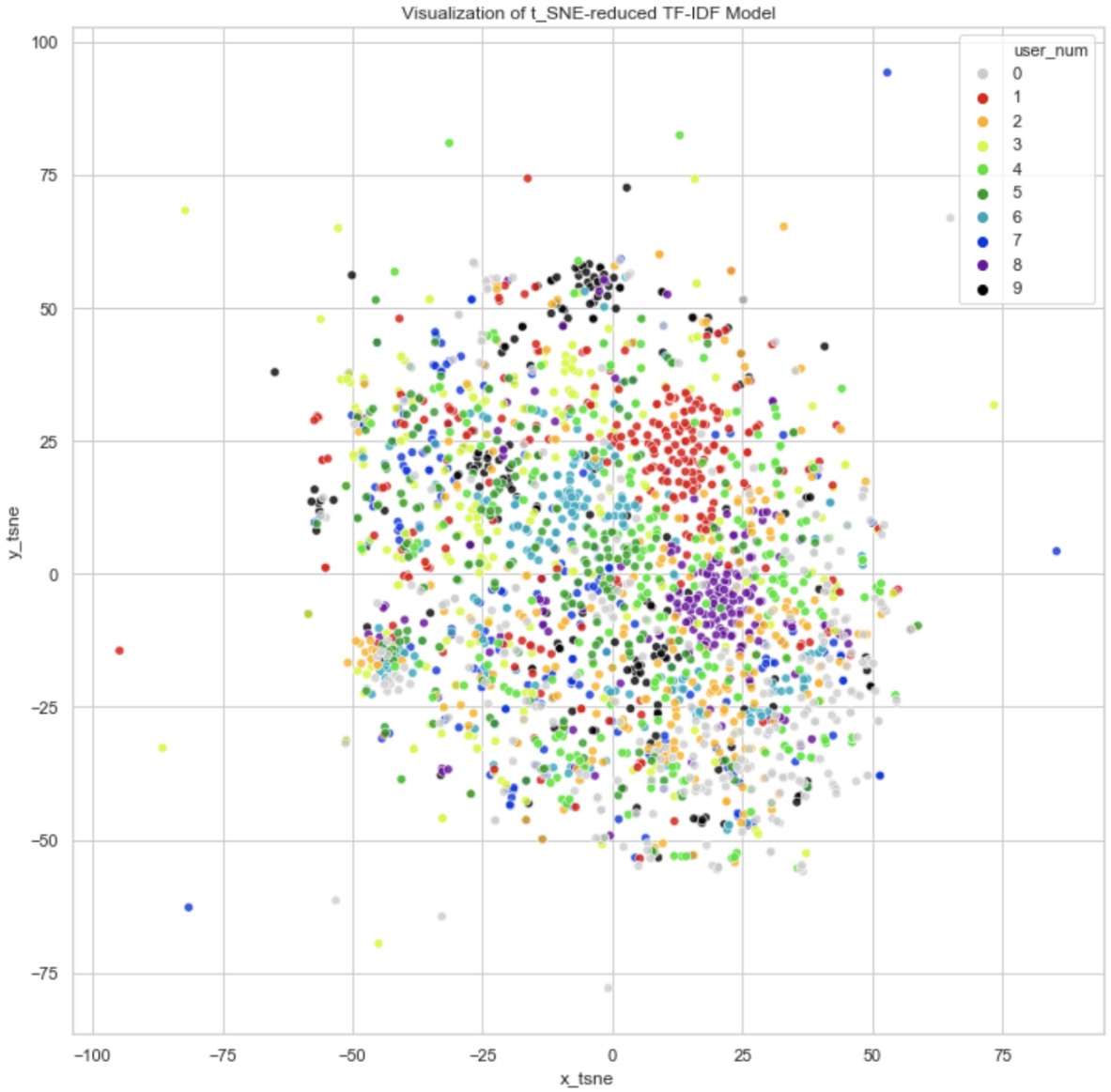
Random forest and gradient-boosted decision tree models failed to meet this benchmark, reaching 0.69 and 0.65, respectively. Oddly, after increasing the number of estimators, the gradient-boosted classifier's performance flagged even further, to the mid-50th percentile.



TF-IDF Vector Model

Next we tried using term frequency-inverse document frequency vectorization, in hopes of obtaining more precise features than simple word counts. We attempted classification both with the full TF-IDF feature set (8815 columns), and with a dimensionally-reduced feature set of 300 features (created using truncated SVD). The full feature set performed somewhat worse with the same classifiers as we used for Bag of Words, and the reduced set's performance was abysmally bad. We also tried using K-nearest neighbors classification for this set, since its values were not as discrete as Bag of Words, but the results still did not compare favorably to logistic regression.

Despite the lower accuracy of classification models using TF-IDF, the vectorization of the words let us visualize the similarity of the reviews by applying t-SNE to the multidimensional vector space. After tweaking the perplexity value, we were able to visually distinguish groupings of the different users' reviews. Some groupings of user data were clearly separable from those of other specific users. Although reviews from users 0, 4, and 5 were dispersed throughout, there was much less overlap between users 1, 2, 6, 7, 8, and 9. User 8's reviews, in particular, appeared in a cluster that did not show much overlap with other user's reviews.



After visualizing the TF-IDF model with t-SNE, we also tried using SciPy's hierarchical dendrogram clustering model to see if its label assignments corresponded more or less to individual users. They did not.

GloVe Model

Given the limited size of our combined text corpus, using an unsupervised neural net trained only on our data would be ineffective, so we used a pretrained model, in this case, a 200-dimensional model trained on Twitter (in hopes that it would have had sufficient input of atypical grammar, incorrectly spelled words, and slang, rather than being trained on a corpus of properly formatted published articles).

Before implementing any classification models, we returned to the sklearn clustering models we tried earlier on the non-word feature set. As before, their outputs did not follow any sort of consistent clustering by user, and their adjusted Rand scores were all close to 0, indicating that the groupings were more or less random with regards to user.

This time, multinomial naive Bayes classification was not an appropriate choice for modeling, given the continuous values in our matrix. Classification using logistic regression, random forest, gradient-boosted decision trees, and K-nearest neighbors all performed worse, the best among them being gradient-boosted decision tree, with an f1 score of 0.52.

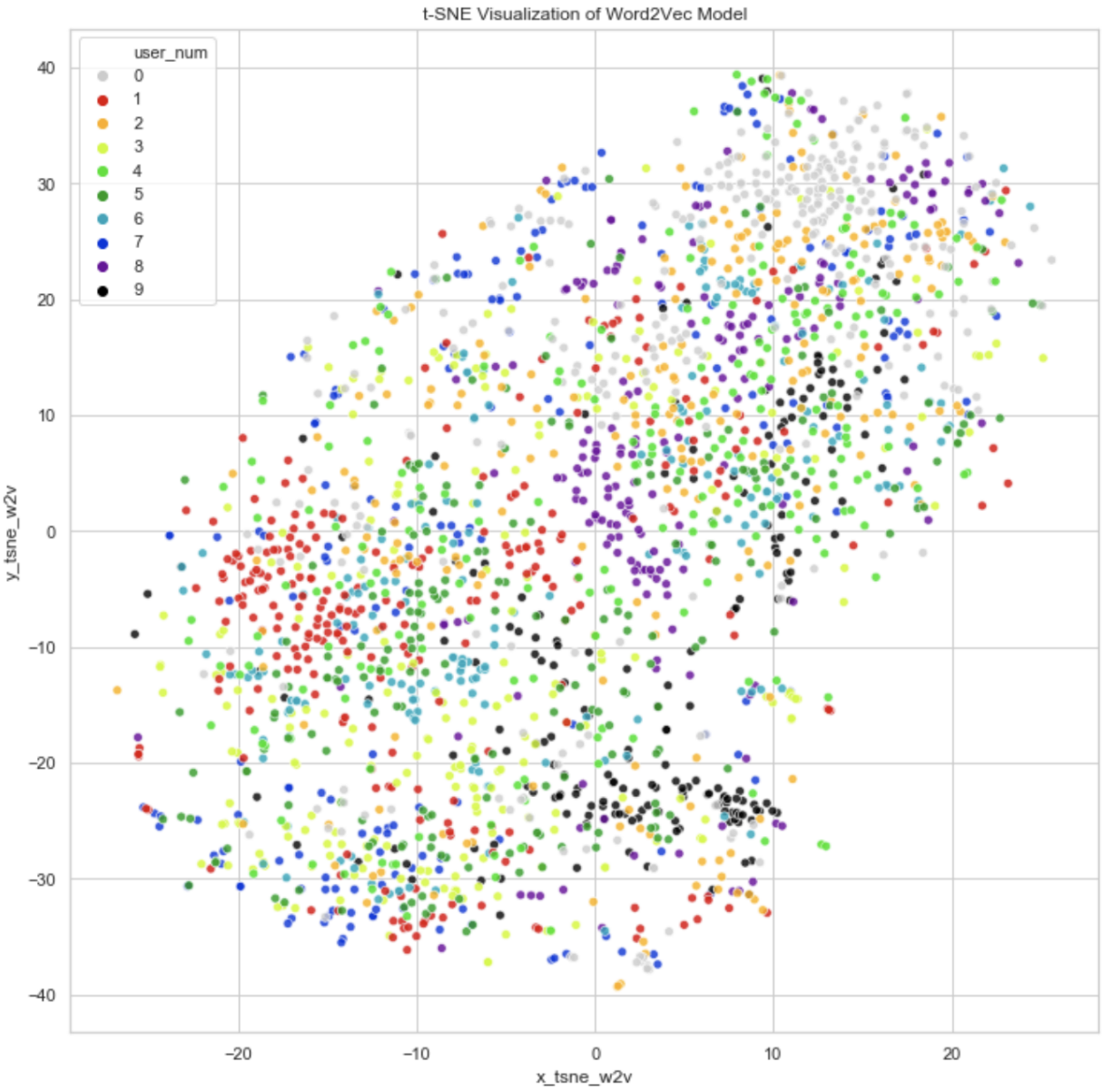
Visualization using t-SNE produced similar results to what it had yielded for TF-IDF, although User 8's reviews were not as isolated from the other users', and User 0's reviews were more isolated this time.

Word2Vec Model

Our final word vector model was Word2Vec, a 300-dimensional model trained on Google News articles. Assuming this model is trained on more orthodox language than

user-submitted reviews, we expected this to produce worse results than GloVe. Nevertheless, we attempted it for the sake of thoroughness.

t-SNE visualization again yielded similar results, with some users' reviews appearing to be distinct from each other, while reviews of the other users were spread throughout.



Classification results from the same models used on the GloVe matrix were slightly higher, contrary to expectations. This may be due to the higher dimensionality of this particular Word2Vec model as compared to our GloVe model. Still the highest f1 score recorded here

was for the gradient-boosted decision tree classifier, at 0.53, barely higher than the same model's score for GloVe.

Conclusion

Despite attempts to categorize Yelp reviews by author with more sophisticated models, logistic regression with Bag of Words was the most effective model overall. However, there was still a substantial difference between training and test scores, so even the most accurate model had issues with overfitting, likely due to relatively small sample size. Clustering models were nowhere near effective, but t-SNE visualization somewhat confirmed expectations about which users' reviews would stand out from the rest.

Sources:

Cover image: https://www.yelp.com/biz/niuda-hand-pulled-noodles-toronto

GloVe tutorial: https://www.kaggle.com/abhishek/approaching-almost-any-nlp-problem-on-kaggle/notebook