Sophia Nicolas

Prof. Vroman

CS361M

Final Results:

Animal Crossing Reviews Classification

1. What is the problem you worked on? Why did you choose this problem? Why is this problem interesting to the world?

The problem I worked on was classifying user reviews for the video game Animal Crossing New Horizons as holding either negative or positive feelings towards based on the frequency of certain words appearing the review.

For an Intercultural Communication class that I took for my Communications Studies major, I did an analysis on different cultural groups' attitudes towards music where I had sort through a lot of social media posts and comments during my research. I became interested in finding a way to automate this type of task, not so it can completely replace the reading process but to help lighten the load of manually sorting through extremely large datasets.

This problem can be interesting to businesses who draw upon reviews to see how audiences are reacting to their products, and how these insights should their future production and marketing endeavors. The cost of time and energy can become very high to sort through every single review manually, and so this tedious task can be made easier for people by automating the detection of whether a review is positive or negative.

Instead of having to sort through data manually and get very familiar with the data by one's own memory, a tool like this can be like a leveled-up chart or excel-sheet to keep track of categorizing user and critic feedback.

2. Has anyone else done ML on this dataset? What algorithms did they use? What were their results? How is your approach the same or different?

Phyllis Tay on Kaggle has also done ML on this dataset. She used hierarchical clustering represented by dendrogram graphs. Her language of choice was R.

Before applying the algorithm, she split the dataset into positive and negative reviews based on the user's grade for each review – 0-5 as negative and 6-10 positive.

The text data was preprocessed into a numerical form with a Document Term Matrix, each element representing a word by its frequency across each document or review. Dimension reduction was performed by removing sparse words or words that don't show up much across the dataset.

The top 10 words and their word count in each positive/negative grouping were graphed in a histogram to see what reviewers are talking about.

The main ML algorithm used was hierarchical clustering to classify the words from the Document Term Matrix into positive or negative sentiment groups.

Similarities with my approach:

- I split the dataset with spectral clustering and labelled negative reviews as graded 0-6 and positive reviews graded 7-10.
- I preprocessed my reviews with a TF-IDF matrix and attempted dimension reduction by setting the parameter TfidfVectorizer(min_df=.02).
- The patterns we noticed were similar, in that the complaint about players only having one island per console appeared in both negative and positive reviews.

Things that were different from my approach:

- Phyllis used hierarchical clustering where I used spectral clustering and random forest classification.
- I didn't get to the point where I could clearly find which words defined a positive or negative review, whereas Phyllis' clustering was able to find this successfully.
- Phyllis also used the "date" column in the dataset, finding that reviews written began positively, but became more negative as each day went on.

3. What ML algorithms did you use? Why? What parameters did you use? Why? How is the performance of your algorithms compared to the performance of previous approaches?

This study in sentiment analysis was divided into three parts:

- Clustering the data based on the grade attached to the user's review - Grades were scaled from 0 to 10 based on the user's judgment of the quality of the game. A word cloud was generated to get a visual on what words appear in each cluster.
- Training a model on random forest classification to predict a grade from 0 to 10, then running the model to make predictions on test data. This was then adjusted to predict only 0 for negative or 1 for positive.
- Mapping the frequency of each word in the vocabulary onto its count in the reviews clustered as positive or negative. This part was never executed, but I hoped the first two steps were able to set up well for this.

K-Fold cross validation was used at the beginning with default parameters to split the data in half for two folds. I found that more patterns were found consistently in the word cloud with a smaller portion of the dataset rather than in working with the entire dataset at once.

For preprocessing, I used the Wordcloud and nltk library to remove stopwords from, tokenize, and lemmatize the textual data to keep it general and easier to find patterns within. The text was then processed to be represented numerically through the TF-IDF vectorizer

algorithm. This defined the vocabulary of all words that appeared across reviews, the TfidfVectorizer(min_df=.02) parameter hoping to reduce its dimensionality to the most important or frequently occurring words. This would cut words appearing in less than 2% of all documents. A TF-IDF matrix was then constructed to hold each review as a vector, each element representing the frequency of a specific vocabulary word.

I chose spectral clustering because it seemed to perform best and consistently when I ran different clustering algorithms in Lab 10. Density-based clustering did not work as well since my grades data was 1-dimensional, but if run on my fully preprocessed or numerically-represented review data, it may perform better due to the imbalance of very low grades (0) and high grades (10).

Random forest classification was chosen because I had difficulties with understanding how to apply unsupervised learning for this dataset in the Clustering assignment, and also because I wanted to give a shot at a supervised learning algorithm. I chose 70 for forest depth because it was in the range of depths that worked best on both training and testing data.

I think my approach may have overcomplicated the process than previous approaches to this dataset that I found online. Both clustering and random forest classification were likely not necessary. I think what I have performs well in the preprocessing and categorizing steps, but I never got to figure out how to find and present the word count of each vocabulary word in each positive/negative cluster.

References:

Dataset:

https://www.kaggle.com/datasets/jessemostipak/animal-crossing/data

Other ML Approaches:

https://www.kaggle.com/code/phyllistay/acnh-text-analytics

https://public.tableau.com/app/profile/phyllis.tay/viz/AnimalCrossingNewHorizonsReviews/ACNHReviews

https://towardsdatascience.com/sentiment-analysis-of-animal-crossing-reviews-7ae98bc91dd3