Ana Julia Bortolossi

Professor Lance Galletti

CS506

28 October 2024

Midterm: Amazon Move Score Predictor

For this project, I built a K-Nearest Neighbors model to predict review scores based on text reviews and their features from a huge. Dealing with such a massive dataset came with its own challenges, so to make processing manageable, I downsampled the training data to 10 percent of its original size. This gave me a set that was still pretty representative but way easier to handle.

To make the data useful for the model, I set up a custom feature extraction function called add_features_to that pulled out a bunch of meaningful numerical and text-based features. This included things like a helpfulness score (calculated from a ratio of helpfulness numerator to denominator), the length of each text, the total word count, and counts of punctuation like exclamation points and question marks. I even added sentiment features that counted positive and negative words to get a sense of how emotionally charged the reviews were. These added dimensions helped capture more of the subtle cues in the text, which ended up being pretty useful for the model.

Since KNN needs numbers to work with, I used TF-IDF vectorization to convert the raw text into a format the model could understand. I limited it to the top 500 features to keep it efficient. I stumbled on this technique through GeeksForGeeks after asking ChatGPT for advice,

but it also clicked because I had done something similar in my BA476 class last semester. Combining the TF-IDF results with all the numerical features I created gave me a solid feature set to work with.

Next, I standardized the numerical features with StandardScaler. This step was essential because KNN relies on distance calculations, and it helps if all features are on a similar scale. Without this, features with larger numbers could skew the model, making it less accurate.

For validation, I split my training data into a 75-25 training-test split to check how well the model was doing. I trained a KNN classifier with five neighbors and then used a confusion matrix to see how often it predicted correctly and where it missed. This showed not just the accuracy but also patterns in misclassification that I could work on improving.

One of the biggest issues I ran into was ensuring that my submission data kept all the necessary features even though I resampled my training data. To fix this, I merged the processed training data back with the submission rows, making sure all relevant features were there when it was time to make the final predictions. It took a lot of testing and tweaking to find that sweet spot between efficiency and accuracy.

I made a few assumptions along the way, like assuming the patterns in the downsampled data would reflect the full dataset, and that the features I picked would add significant value to the predictions. I also counted on the score labels to be accurate since they’re crucial in training a supervised model like this.

To improve this model, there are several additional steps and features I would have liked to explore. First, incorporating more advanced NLP techniques like word embeddings (e.g., Word2Vec or GloVe) could capture deeper contextual relationships in the review text, likely

enhancing the model's understanding of nuances. Another idea would be to include sentiment analysis using pre-trained models that score sentiment more precisely than a simple word count, as this could help identify the tone more accurately. Adding time-based features might also improve the model, like capturing seasonal trends or changes in review style over time. Finally, experimenting with different KNN parameters, such as tuning the number of neighbors or testing alternative distance metrics, could also lead to better accuracy. If computation resources allowed, training the model on the full dataset would likely yield improvements by exposing the model to a broader range of patterns and examples.

In the end, this KNN model, along with all the data processing and feature extraction, gave me a solid setup for predicting scores based on text reviews. The process balanced efficiency and accuracy pretty well and created a foundation I can build on for future improvements. The improvements I outlined above could have helped improve my position on the leaderboard which ended up being around 0.49 accuracy.

Works Cited

GeeksforGeeks. (2023, January 19). *Understanding TF-IDF (term frequency-inverse document frequency)*. https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

Jain, A. (2024, February 4). TF-IDF in NLP (term frequency inverse document frequency). Medium. https://medium.com/@abhishekjainindore24/tf-idf-in-nlp-term-frequency-inverse-document-frequency-e05b65932f1d