**Week 7 Writeup - Develop Second modeling approach**

**Ryan O'Hara, Jack Metzger**

**Embeddings**

To improve the caliber of our recommendation system, we used precomputed user and tweet embeddings in this week's modeling method. We converted the vector files into Twitter embeddings, which could increase the model's complexity and its ability to make determinations. These embeddings served as structured input for our KNN model, which is useful for enhancing the model’s

This embedding consists of the raw and sparse model. Embeddings transformed these inputs into dense vectors that captured intricate associations, making them perfect for use with similarity-based techniques like KNN. To get the best tweet recommendations for each user, we specifically employed the cosine similarity between the user and tweet embedding vectors.

Using cosine similarity as the matching measure, a similarity matrix was built using user embeddings as rows and tweet embeddings as columns to get the top-N relevant tweets for each user. Our effective approach is to reduce the dimensionality by using raw data embeddings, which helps prevent overfitting and allows for the use of a weighted average. During similarity assessment, embeddings used bespoke weights to prioritize recent or highly engaged tweets.

We tracked the trade-off between suggestion accuracy and computing efficiency across these modifications. Weighted embeddings marginally increased relevance for trending content, whereas reduced-dimensional embeddings, as predicted, produced faster calculations with relatively little performance impact.

**Scoring/Accuracy Method:**

Due to the nature of our model, the output is a song recommendation rather than a traditional classification or numeric prediction. As a result, standard accuracy metrics did not directly apply. To evaluate the quality and relevance of each of our model’s recommendations, our group manually scored the outputs for a fixed testing sample of 100 songs, allowing for both a consistent and comparable evaluation of each model.

Test set song recommendations were evaluated on a binary scoring system: a score of 1 represents a relevant/strong song recommendation, while a 0 represents an irrelevant/poor song recommendation. Although we acknowledge this scoring system is imperfect and subjective, the best effort to score recommendations on lyrical content and overall “vibe” (i.e a listener who enjoyed the input song might reasonably enjoy the recommended one). It is important to reiterate accuracy scoring was based on perceived song similarity and recommendation quality, not based on personal music preference. Our test set contained a good variety of randomly selected songs, covering a number of genres and eras.

Given the subjective nature of grading a single recommendation as good or bad, we have added a +- 5% error to the graded score for each model. We felt in a 100 song sample, we were able to accurately grade positive or negative within 5 songs of error.

Given a longer project timeline and additional resources, we could have evaluated model accuracy using real-time user feedback to better capture preferences and listener behavior. This would have allowed us to incorporate a feedback/supervised training aspect in our model. However, due to the limited nature of this course and project, we chose an evaluation method that was both simple and immediately accessible. This allows us to develop, test, and compare multiple versions of the model while maintaining a consistent accuracy metric.

The test set contained songs we were both familiar and unfamiliar with. It is estimated that each song (input and recommended) was listened to for a minimum of one minute (longer for unfamiliar songs) with the lyrics on screen before making a scoring decision. Each round of scoring (done three times) is estimated to take approximately three hours.

**New Model 2: Twitter Embedding**

This new model we used this week was chosen to compare the efficacy of a new embedding against the one we used in last week's model. From this point, we could see which one gave more accurate recommendations and proceed to add our numerical features to the better of the two embedding models. We ended up proceeding with this new twitter embedding model because it was able to slightly outperform that of last week.

**New Model 3: Twitter Lyric Embeddings + Numerical Feature Integration**

Our second new modeling approach for the week worked to incorporate the other numerical features of the dataset beyond lyric embeddings. In this model the one-hot encoded genre and subgenre columns were omitted. Feature engineering and normalization was completed in week 5. The new variables include: popularity, square root of BPM, dance, square root of acoustic, happy, power (loud X energy), camelot\_sin, and camelot\_cos.

This model introduced a new layer of complexity compared to the Week 6 model, which relied solely on lyric embeddings and did not incorporate any musical or structural features. Our hypothesis was that combining lyrics with musical metadata would yield more holistic recommendations - matching not only the song's message but also its tempo and overall vibe.

The integration of the new features was straightforward, as it followed the same processing pipeline as our previous two embedding-based models. Due to the time-intensive nature of the manual accuracy scoring, we limited the combined feature and embedding model to the word2vec (Twitter) embeddings, which achieved a slightly better accuracy metric than GloVe. Additionally, we felt that word2vec, trained on informal Twitter language, aligned better with the structure and style of song lyrics and their informal nature.

Training time remained comparable to earlier models. The manual accuracy score for this model was 49 ± 5, reflecting a 44–54% relevant recommendation accuracy range after accounting for subjectivity in scoring.



**Figure 1:** Screenshot of the code incorporating the new numerical features into the model.

**Reason for the Hyperparameter**

Our group tackled many questions regarding hyperparameter tuning of K for our KNN model last week, and while we have made adjustments to the ways we are adding features and embedding the lyrics vector this week, nothing material has changed the way we are approaching hyperparameter tuning.

As our model is not one that relies on traditional accuracy measurements, we don’t see the utility in working to maximize our expected accuracy based on different values of K. We care more about the actual order of distance between our song and the recommended songs rather than the physical distance between the points. Changing our K only changes the distance and does nothing to the order at which the recommendations are provided.

We plan to address more hyperparameters in our model next week when we begin to tackle things like feature weighting for genres and other things, we will see an uptick in this section and the importance of handling hyperparameters, but for this week we reverberate the same sentiment we shared last week.

**Performance Comparison**

We saw tangible improvements in recommendation accuracy, gaining 6% accuracy from our previous best model. Nearing 50% accuracy is important for us, as this is still a very simple model with no added weights, no genre inputs, and no further tuning. We believe that there is still much more room for improvement in the coming weeks to expand upon this, but are happy with where we stand in this process.

While these measurements are subjective, the fact that our new score of 49% exceeds our hypothetical confidence interval of the previous model suggests that we are fairly confident that adding the numerical features did improve the quality of our model. The table below shows our models produced thus far and the improvements in accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Accuracy | Confidence Interval | K Parameter |
| KNN | 41/100 | 36 to 46 | 1 |
| KNN\_Twitter | 43/100 | 38 to 48 | 1 |
| KNN\_Full | 49/100 | 44 to 54 | 1 |

**Table 1:** Comparison of the three model performances with +- 5 confidence intervals. KNN\_Full showed the best performance.