Find Similar Users Based On Music Taste

Ziyue Wang, Chamanthi Aki
wang.ziyue1@northeastern.edu, aki.c@northeastern.edu

Problem Definition

In the audio streaming and music services industry, platforms like Spotify or Apple Music have already built industry-leading recommendation systems that preciously provide songs that we might like. However, there are two problems: first, they only focus on the music listening experience, while ignoring the social needs and community values when people are drowning in the ocean of music. Second, suggestions generated by algorithms cannot match our wide music tastes. People tend to discover new music mostly through friends' recommendations. We want to bridge the gap and fully discover the power of music that could potentially bring people together, build relationships, and further expand their music community. Because of the strong social connections, this would strengthen the tie between users and the music streaming platform, reducing the churn rate and increasing profits as a result. In our project, we recommend new friends and their favorite songs to users based on their similarity of music tastes.

Dataset

This dataset is a 2.53 GB tsv file that contains up to 1k users' music listening records from 2005 to 2009 on the platform Last.fm. There are 6 columns: userid, timestamp, artid(artist id), artname (artist name), traid (track id), and traname (track name). We only took the listening records in March 2008, assuming we recommend users and music to them at the beginning of April 2008. There are 482386 entries, including 590 users and 178423 tracks from 29197 unique artists.

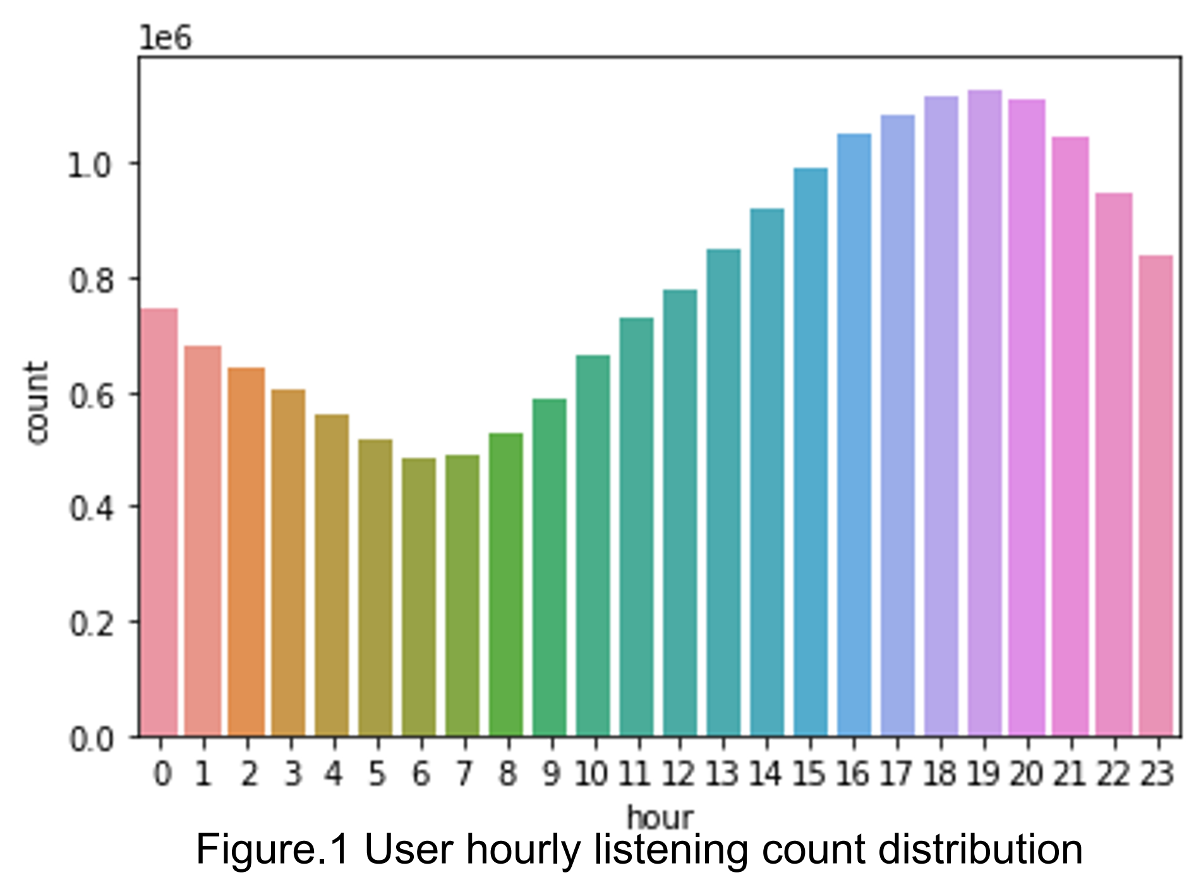

Figure.1 User hourly listening count distribution

Find Similar Users

Naïve solution: calculating Euclidean distance between users according to their listening counts.

Our process: Aggregating listening counts -> Scaling -> Averaging -> Dimensionality Reduction -> Build Similarity Matrix

Since it's hard to measure the result of similar users we found explicitly, we tried to use different clustering algorithms to see how well we can cluster these users. We tried K-means, HAC (Hierarchical Agglomerative Clustering), DBSCAN, and Spectral Clustering. One of the challenges to use parametric clustering–clustering algorithms that need k as a component parameter–is to determine how many clusters we want. In our case, we want to recommend up to 5 similar users to the target user. Therefore, 100 will be a proper estimate of the cluster number we want. We also used T-SNE to visualize the data distribution.

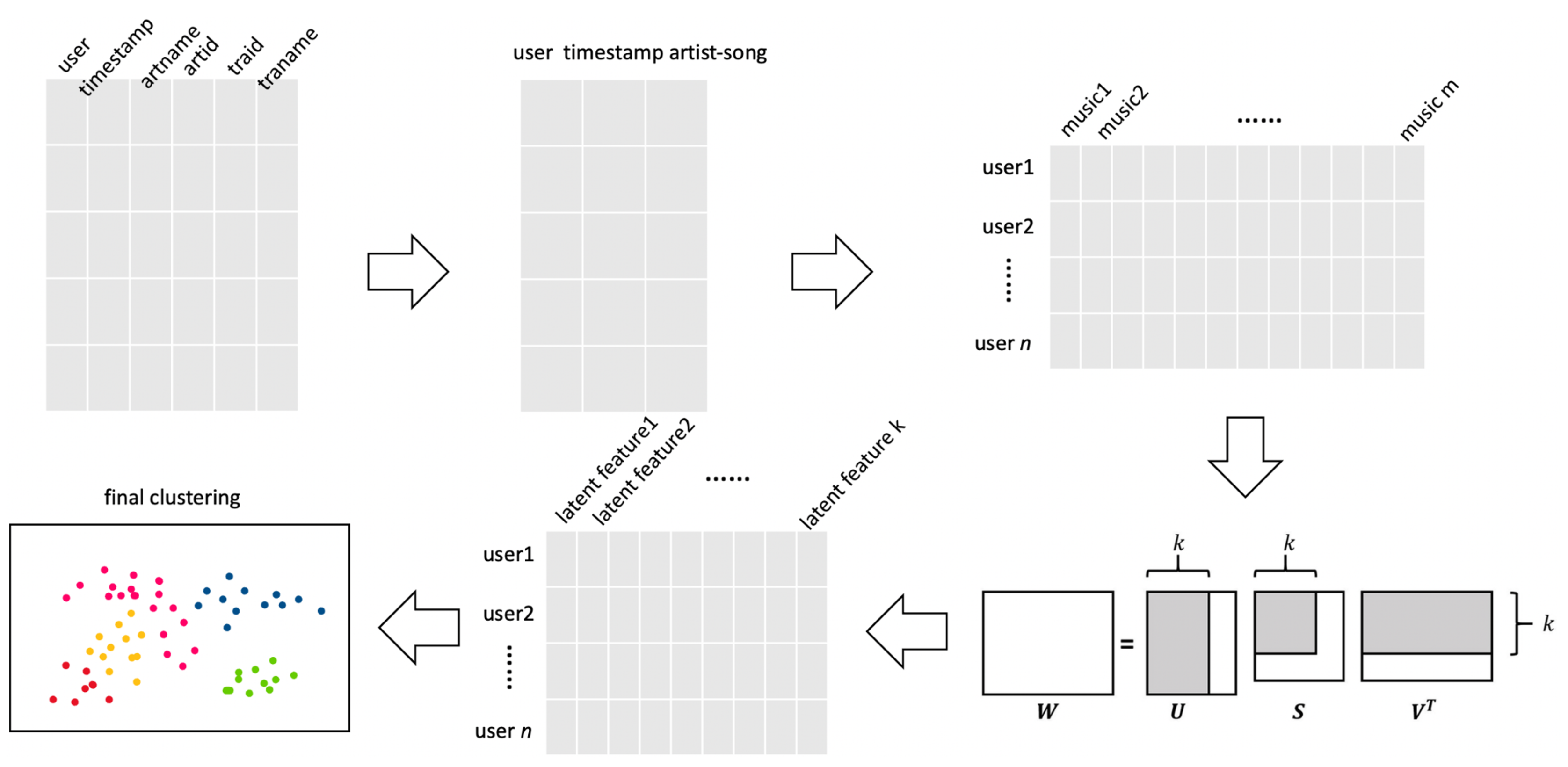

Figure.2 Flow chart to find similar users

Recommendation

SVD (Singular Value Decomposition) is a matrix factorization algorithm that can be used in recommendation systems in a way that the matrix of listen counts (utility matrix) is factored into a product of matrices representing latent factors for the music tracks and the users. In the first approach, we utilized scipy library in implementing SVD algorithm. Developed a pivot matrix and normalized the matrix by each user's mean and converted it to an array. Implemented SVD choosing k=50 as the number of latent factors. Decomposed the matrix to find the predictions on listen count of the music tracks for each user. In the second approach, the dataset is trained and cross-validated (cv=3) on a model where Surprise package is used to implement SVD in order to build the recommendation system. The model is trained on the entire dataset using fit() method after cross-validation into a Surprise Trainset. Further, implemented GridSearchCV to tune the hyperparameters on the SVD algorithm.

Evaluation

To measure if scaling the listening count and averaging each user's habit contributes to our final result explicitly, we measure the silhouette score of clustering based on the data derived from the above processing. For the naive solution mentioned in the Method section, silhouette scores of DBSCAN and Spectral Clustering are negative, which represents potential wrong clustering(see Table.1). As a comparison, silhouette scores increase in varying degrees after performing the scaling and averaging. Meanwhile, the distortion error of k-means when k is 100 decreases 56% after averaging and 70% after scaling. In addition, from the T-SNE plot, we can tell small clusters are distributed denser and slightly far from each other (see Figure.3). The Recommendation model is evaluated based on the RMSE metric, where the model showed 3.01 RMSE without cross-validation and 2.41 RMSE value after cross-validation where the model not only recommended music tracks but also users based on similar music taste.

	K-means	DBSCAN	Spectral Clustering	HAC
Without processing	0.5315	-0.6053	-0.0823	0.5124
After Averaging	0.5621	0.4213	0.4213	0.6263
After Scaling	0.3240	0.4284	0.4284	0.3112

Table.1 Comparison of Silhouette score of different clustering algorithms

Conclusion

Our project focuses on finding similar users and recommending their favourite music based on music listening history on Last.FM. Since it's difficult to measure the similar users we found for the target user, we conduct experiments on clustering to evaluate the validity of our preprocessing. The result demonstrates the effectiveness of scaling and averaging, and the authenticity of the similar users we recommended implicitly.

We also developed a Recommendation system to recommend music tracks and users using Singular Value Decomposition.

- In the first approach, scipy library is used for matrix decomposition and implementing SVD. Second approach used Surprise library to build a utility matrix and decompose it to recommend music and users such that the music is recommended only when the estimated listen count is above the mentioned threshold.
- The model is evaluated on RMSE before and after cross-validation where the implementation of cross-validation showed a great improvement in the model.

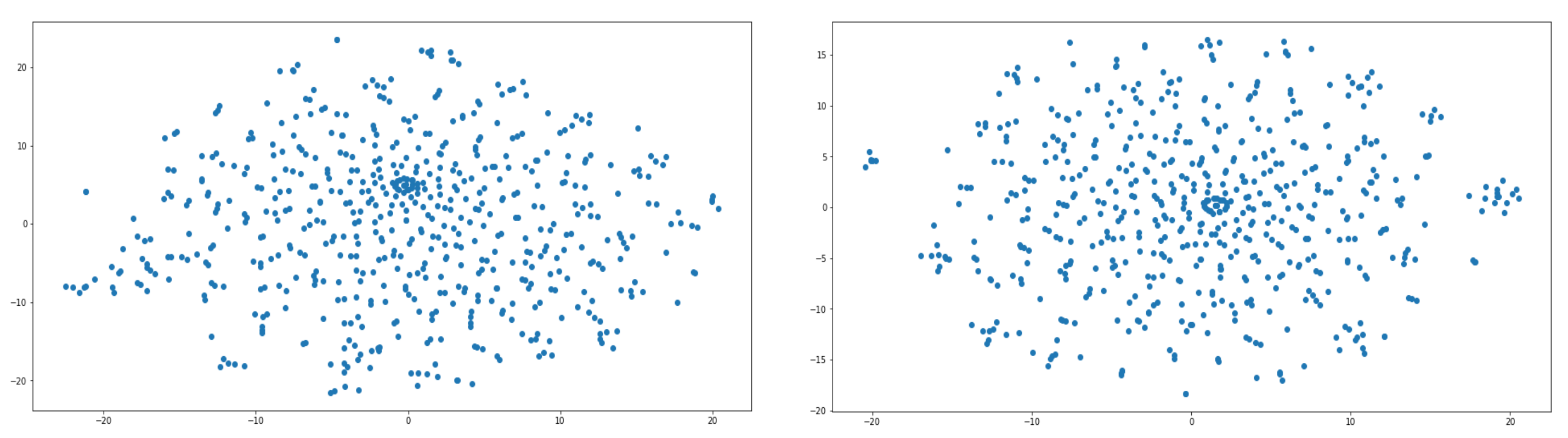

Figure.3 T-SNE plot of reduced data before and after processing

Future Work

Further work can be done to incorporate more social data so that we can recommend friends of recommended users. Additionally, we need to address the cold start problem since new users will not be able to recommend any users as they might not have listened to the music tracks above the given threshold value in the mentioned algorithm. Gray sheep problem might be also faced as there might exist some users whose opinions consistently would not match with any group of people and so their opinions will not benefit from collaborative filtering.