UDACITY - Machine Learning Engineer Nanodegree

Capstone Project Proposal

Connor Phoenix
12.6.2021

Project - Artist Recommendation (via Sagemaker Custom Sklearn KNN Code)

Domain - Content recommendation is one of the most popular machine learning applications. Essential to the business model of many modern companies is the ability to offer users tailored content based on available user engagement data. This challenge is ubiquitous across many domains such as e-commerce and audio/video streaming services. Effectively solving the content recommendation problem was a key success factor for companies such as Amazon and Netflix. Interestingly the maturation of MLaaS platforms like AWS Sagemaker has lowered the overhead in developing and deploying such systems.

Problem Statement - Given a dataset of artist play counts by user, create a system to generate an array of artist recommendations given an user's favorite artist. The approach is loosely based an approach from Budiman and Giri of Udayana University (see paper in github repo).

Dataset - The dataset was sourced from the Last.fm API by Oscar Celma of Pompeu Fabra University. The data is available for non-commercial public use via the university website:

- https://www.upf.edu/web/mtg/lastfm360k

An ingestion and initial preprocessing script can be found in the artist_recommendation_KNN repository.

Solution Framework

1. Merge artist play and user files into single dataset
2. Filter data to users in the United States
3. Perform some exploratory data analysis
4. Filter to top n% of users (% to be determined)*
5. Filter to top n% of artists (% to be determined)*
6. Create a sparse matrix where rows represent artists and columns users
8. Build a custom Sklearn KNN model using custom train.py script
9. Create custom inference code to generate knn.neighbors in predict.py script
10. Use nearest neighbors output as recommendations for input artist

*Many artists in the dataset have very low play counts which could introduce noise into the system. The idea is that by filtering to the top artists we are ensuring a certain level of quality in the generated recommendations. This also has the side benefit of further reducing the dataset size.

Evaluation - Evaluation will be largely based on intuition from spot checking example input artists. As this prototype example is an unsupervised approach, standard regression and classification metrics won't apply. Typically, evaluation of these simple unsupervised systems is based on real user feedback. In their paper, Budiman and Giri used a soft evaluation method by surveying user satisfaction of the provided recommendations.

The quality of recommendations is inherently subjective and as such more difficult to evaluate than standard machine learning problems. In a production context, some sort of a/b user testing paradigm would be useful in monitoring recommendation quality. Alternatively, more advanced systems such as matrix factorization structure the recommendation problem in a classification/regression framework and could be a useful area of further exploration.

In the absence of robust evaluation, I will generate a “nice to know” gut check metric of hit rate of recommendations based on a user’s favorite artist. This metric will compute as follows:

```
recommended_artists = []
hits = []
```

For a random selection of users:
- Identify each user’s favorite artist (most played)
- Input the favorite artist to generate the top K artist recommendations
- recommended_artist.append(recommendations)
- For the top K recommendations determine how many have been played by the user
- hits.append(played_recommendations)

```
Total Hit Rate = len(hits) / len(recommended_artists)
```