Last.FM Musician recommendation

This repo is for this take home challenge.

This directory contains implementations of Last.FM Musician recommendation framework for the corresponding dataset. For more information about the data and the EDA analysis, please go to this google colab folder.

To run the pipeline for training and evaluation on the framework, simply run `python main.py`. See `args.py` for a list of arguments that you can pass.

Note that in order to run BiasedSVD for user-artist interaction embedding, you must install the package `pip install scikit-surprise`.

Example Commend

For the ease of reimplementation, I included saved data of user-artist interaction embedding. If you want to train the SVD model, you can simply run the following code, else it will use the saved data.

```
$ python main.py --runsvd True
```

For the same purpose of easy implementation, I also saved model weight for logistic regression. Run the following code for training this model

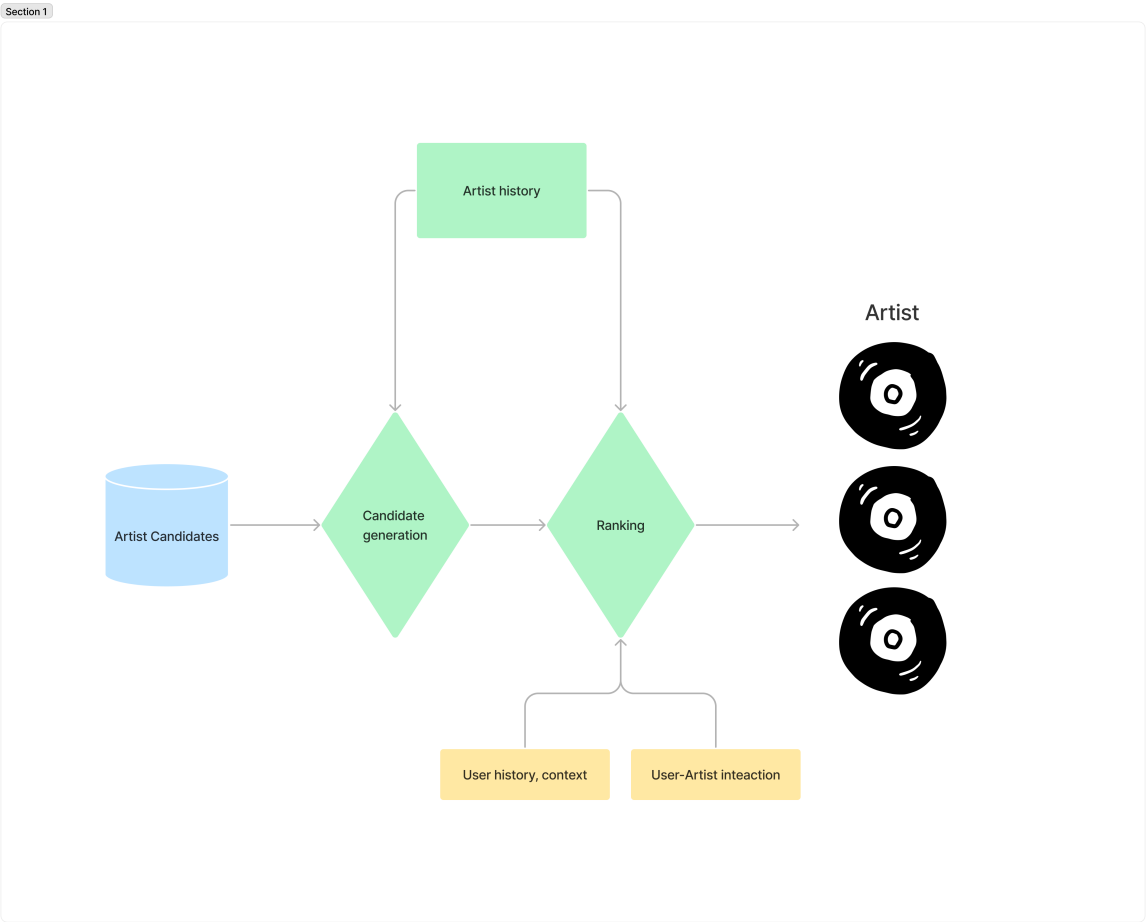
```
$ python main.py --judge True
```

Output

The output of this pipeline is a saved `.csv` file saved in `result` file with 20 artists (musician) recommended for each tested user. The score of each artist is also saved in this file. A recommendation system will recommend the artists to users in a descending order of the scores.

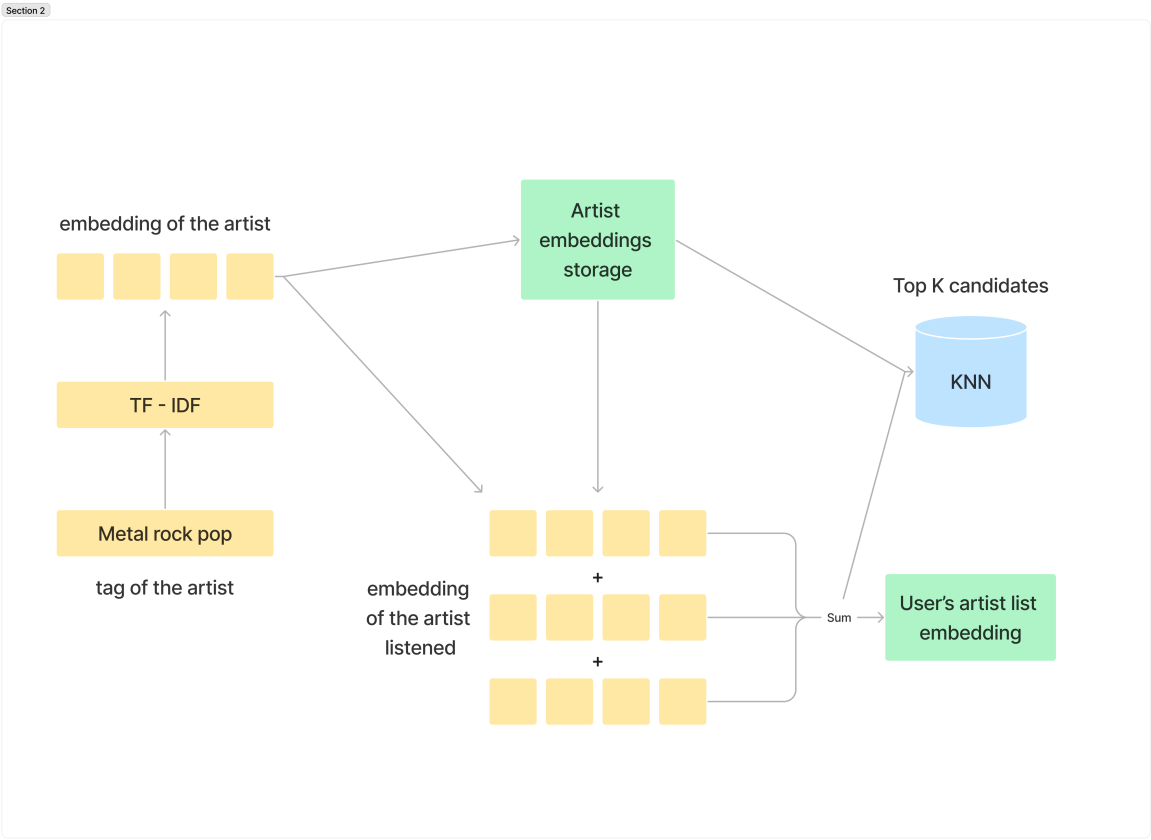
Model and System

Inspired by the paper [DNN for Youtube Recommendations](). The general architecture I used is a classical two stage 'funnel' architecture. and ranked before presenting only a few to users.

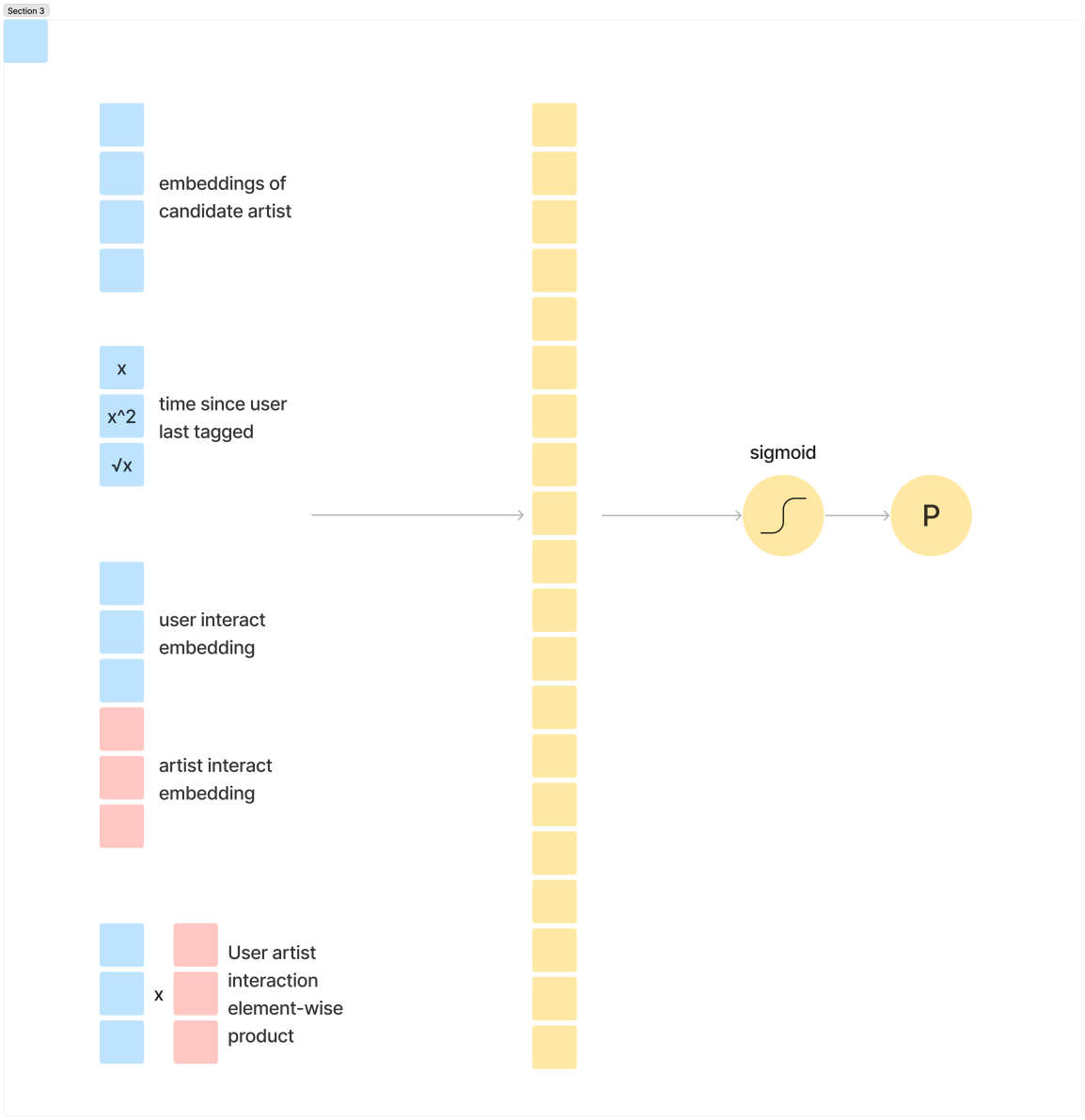


In the candidate generation stage, I mainly use the information from artist(musician)'s history and context. Artist will get tag from users, I made a corpus with the tag values and mapped the tag list of an artist to a 500-dim embedding space. Additionally, I also add other dimensions reflecting the popularity and freshness of the artist to the embedded vector. the vectors of the artists will be stored in an database. I sum up the embeddings of all artists a users listened before and use this

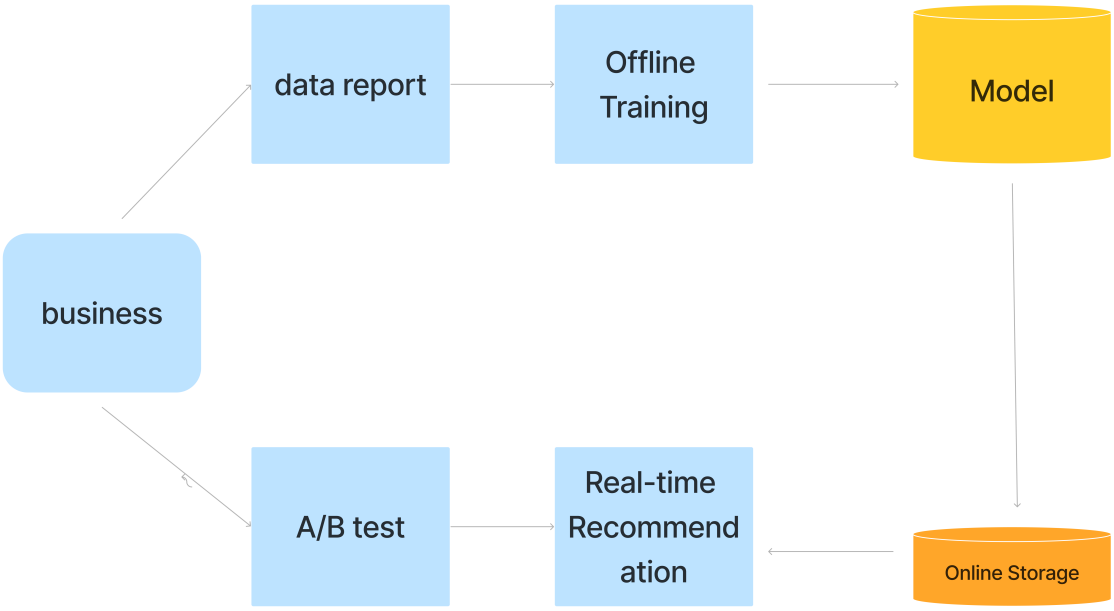
summed embedding find top K candidates based on similarity for recommending purpose later.



In the ranking stage, I leveraged more information from user side. for each user-artist candidate pair, besides from the embedding of artist, I also include other dimensions reflecting the freshness of the user and the user's interaction with the artists. Noteworthy, I got the interaction embedding of users and artist by a Matrix Factorization technique called BiasedSVD (FuncSVD). Concatenation of the two vectors will reflect their non-linear relationship, while element-wise product will reflect there linear relationship 2. Taking all these consideration, I concatenate all the dimensions and pass it through a sigmoid activation function to get a probability of how likely the user will interact with this artist. Artist will ranked by this probabilities in descending order.



A typical recommendation system includes the steps below. During the data report step, we will do the ETL process of the data reported and store the data in an offline data storage module. because of the high volume of the data, the module is always a distributed file system or platform. Offline training usually includes: data sampling, feature engineering, offline model training, similarity calculation, etc. Our first stage, candidate generation will occur in this step. All the information we obtain in this step will be stored in a online storage module (eg. cache) for quick response to achieve real-time recommendation. Besides offline training metrics, A/B test is another import metric to measure the impact from a new algorithm on customers.



Further Steps

This project demo of a recommendation system. Due to the scope of this project, besides the efforts of extracting features from raw_data, a simple sigmoid function is applied for final ranking which may not fully reflect non-linear relationship of the data. According to paper 2, build a deep learning architect to add MLP layers on the interaction data and make the model learn more about the non-linearity information.

Reference

1. Deep Neural Network for Youtube Recommendations

2. Neural Collaborative Filtering