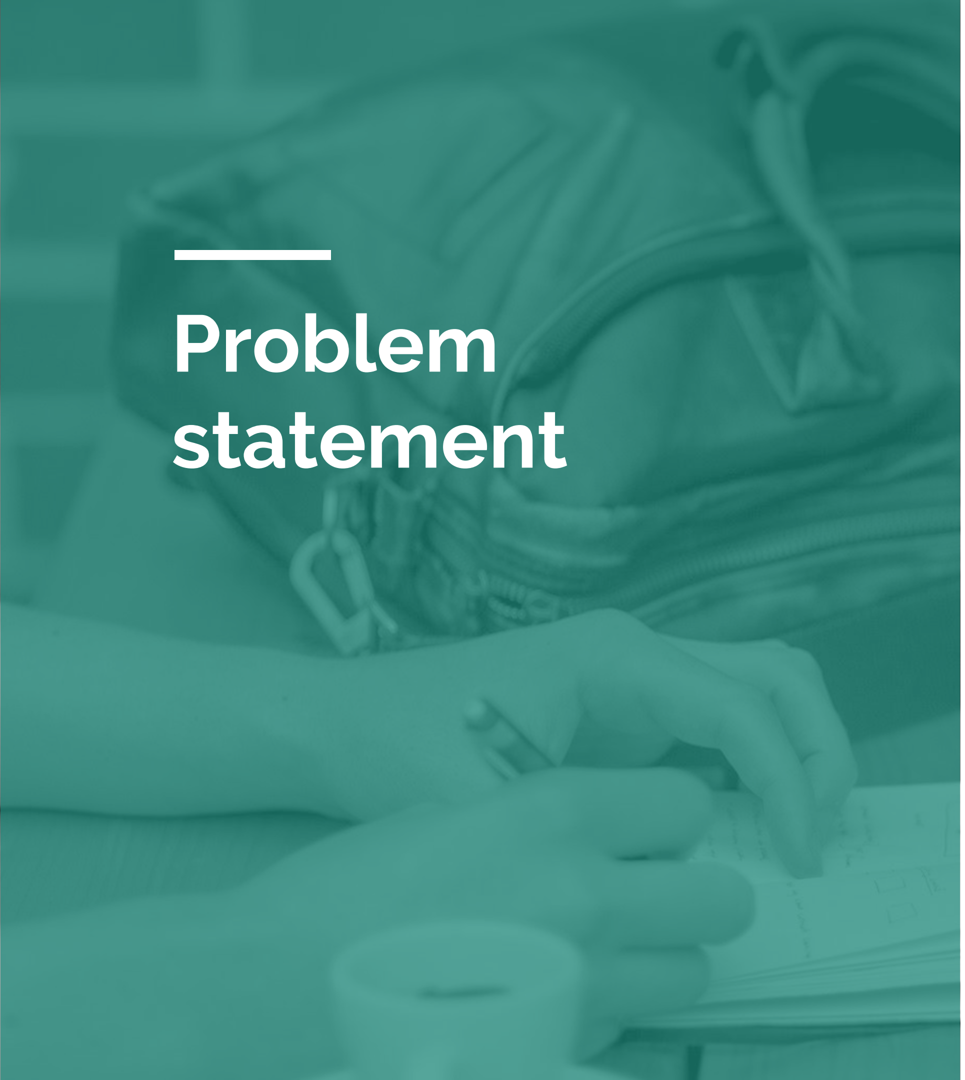
Hybrid
Music Recommender System

By Yan Zhu

Outline

The Problem



Problem statement

There are **0.3 Million** unique users using Music Box;

Choosing from **1.6 Million** unique songs;

Expanding users and songs!

They need some help to find the next favorite song.

Solution Proposal

A personalized recommender system

A personalized recommender system

- ➔ That would learn users' preferences from their play history
- ➔ Recommend based on similar users' preference
- ➔ Recommend songs similar to songs they have liked

Choices of Recommender

- ➔ Collaborative filtering
- ➔ Content based / Item-item filtering
- ➔ Demographic or other contextual filtering

Choices of Recommender

- ➔ **Collaborative filtering (CF)**
- ➔ **Content based / Item-item filtering**
- ➔ Demographic or other contextual filtering

Recommender with CF

Example of cleaned aggregated data

uid	song_id	freq	download
12333	5237384	52	0
12333	706155	14	0
60183	445491	1	0
60183	1117264	2	0
…			

Recommender with CF

Example of labels and prediction on test set

uid	song_id	freq	download	**label**	**prediction**
12333	5237384	52	0	**6**	**4.370734**
12333	706155	14	0	**4**	**-0.08947**
60183	445491	1	0	**-1**	**-0.23749**
60183	1117264	2	0	**1**	**0.067943**
			…		

Recommender with CF

Example of labels and prediction

uid	song_id	freq	download	**label**	**prediction**
12333	5237384	52	0	**6**	**4.370734**
12333	706155	14	0	**4**	**-0.08947**
60183	445491	1	0	**-1**	**-0.23749**
60183	1117264	2	0	**1**	**0.067943**
...					

Example of recommendation based on labels

12991703	Sham	Archive	20.82
6324025	我们能不能不分手(伴奏版)	花儿乐队	19.03
5890375	恋爱百分百(34秒铃声版)	蔡依林	18.92
163185	第一人选	何俐恩	18.58
502758	朱砂泪	夏一可	18.58

Item Based Recommender Prototype

Example of song attributes in the dataset

uid	song_id	down	freq	**label**	type	song_name	singer
12333	708428	0	61	**6**	1	酒吧英文**慢摇**舞曲	**DJ舞曲**
12333	875447	0	54	**6**	1	有雨的日子	六哲
12333	5237384	0	52	6	2	逆流成河	金南玲
12333	706155	0	14	**4**	1	Music劲爆DJ	**DJ舞曲**
12333	708261	0	12	**4**	1	俄罗斯**慢摇**	nasa

Item Based Recommender Prototype

Songs user 12333 liked from top 500

song_id	freq	down	label	type	song_name	singer
5237384	**52**	**0**	**6**	**2**	**逆流成河**	**金南玲**
5114569	46	0	6	2	没有你陪伴真的好孤单	梦然
90519	6	0	3	2	情人	刀郎
55219	6	0	3	2	水手	郑智化
21596231	2	0	1	1	凉凉-(电视剧《三生三世十里桃花》片尾曲)	花舞倾歌&灰菟

Example of songs recommended from top 500 songs

song_id	type	song_name	singer
157767	2	等你等到我心痛	张学友
6477086	2	小水果	筷子兄弟
7196022	2	时光笔墨-(电视剧《青云志》片尾曲)	张碧晨
23497506	2	画心(Live)	张靓颖
7153193	2	别把疼你的人弄丢了	雨宗林

Top K songs

- ➔ General non personalized recommender
- ➔ For inactive users with only 1 play record
- ➔ For inactive user played single song multiple times but not in the top 500
- ➔ Recommend Top K most frequently played songs

Technical details

Data cleaning

Collaborative filtering

Item-item filtering

Efficiency and scalability

Data cleaning

- ➔ Validate: song id and user id
- ➔ Remove duplicates from song table
  - ◆ Under same song id, get majority vote for song name, song type and singer name
- ➔ Remove songs only have one entry in play history
- ➔ Remove user with abnormally high play frequency
  - ◆ Cutoff at 0.999 percentile, song played frequency around 1000
- ➔ User: 264715; Song: 1559987

Collaborative Filtering

- ➔ Key idea: recommend based on other users who share similar preferences
- ➔ Challenge:
  - ◆ Only implicit feedback => experiment with various design of utility matrix
  - ◆ Extreme sparsity: 0.014% => need hybrid recommender
- ➔ Final Utility matrix formula
  - ◆ S1: **normalize play frequency to 0 ~ 10**
  - ◆ S2: **penalize the song only played once**
  - ◆ S3: **award download**
- ➔ CF with Alternating Least Squares minimization

Evaluation

- ➔ RMS: 2.91 (on scale -1~10);
- ➔ Rank 1, no regularization gives lowest error
- ➔ Problem: ineffective “collaboration”
  - ◆ Most users don’t have much play history
  - ◆ 25% of users only have one entry in play history
- ➔ Solution: supplementary recommender for inactive user

Item-item filtering

- ➔ Key idea: recommend similar songs based on their content to inactive users
  - ◆ ~70k
- ➔ Features used:
  - ◆ Song type
  - ◆ Vectorized singer name, vectorized song name (Word2Vec)
- ➔ Challenge: big matrix 0.8 M * 0.8 M) that slows down system significantly
- ➔ Solution: downsample to only 500 songs

(80 % completion)

Efficiency and Scalability

- ➔ CF: after training, recommend for each user takes ~40s
- ➔ Highly scalable with Apache Spark's MapReduce model
- ➔ Item-based: very slow because of matrix multiplication involved

Recommend for single user

Future Work

Future improvement

- ➔ Recommender for inactive users with song preference not in top 500
- ➔ Better word embedding (Chinese or mixed language) for Song name and Singer
- ➔ Collect more information about the song to improve explainability and effectiveness of recommendation
- ➔ Streaming
- ➔ User interface

Questions?

References

[1] Schedl, Markus, et al. “Music Recommender Systems.” SpringerLink, Springer, Boston, MA, 1 Jan. 1970, link.springer.com/chapter/10.1007%2F978-1-4899-7637-6_13.

[2] “Large Scale Matrix Multiplication with Pyspark (or - How to Match Two Large Datasets of Company...” Yodas Labs, Yodas Labs, 2 Aug. 2016, labs.yodas.com/large-scale-matrix-multiplication-with-pyspark-or-how-to-match-two-large-datasets-of-company-1be4b1b2871e.

Implementation for this project can be found at Github repository.