DATA 612: Research Discussion Assignment 1 - Spotify Recommendations

Derek G Nokes

2019-06-13

Please complete the research discussion assignment in a Jupyter or R Markdown notebook. You should post the GitHub link to your research in a new discussion thread. Now that we have covered basic techniques for recommender systems, choose one commercial recommender and describe how you think it works (content-based, collaborative filtering, etc.). Does the technique deliver a good experience or are the recommendations off-target?

Introduction

Of the services that I use that come packaged with recommendations, Spotify is - by far - the most effective at producing recommendations that I find useful. This effectiveness has motivated my search for information over years about how they produce such great recommendations.

Spotify - the most popular global audio streaming subscription service - operates in 79 markets, has 217 million users (including 100m paid subscribers), and streams a catalog of 50+ million tracks. Spotify's user by track matrix is simply enormous. Not only that, but there are also 3+ billion user-created playlists on the platform - a goldmine for information about what Spotify's users like.

Employees of Spotify have publicly discussed aspects of the inner workings of their recommendation platform over the years, so there is some information about how the system works. Although their recommender continues to evolve, from what I can glean from publicly available information, they use a ensemble of at three types of recommenders (as-of last 2015):

All of the images in this discussion post are taken from a presentation made by Chris Johnson on 2015-11-16, which can be found at the following URL:
https://www.slideshare.net/MrChrisJohnson/from-idea-to-execution-spotifys-discover-weekly

1. Collaborative Filtering (CF)

From what I can gather, Spotify initially only used Collaborative Filtering (CF) to provide recommendations. This approach, which is outlined in the paper, Collaborative Filtering for Implicit Feedback Datasets (http://yifanhu.net/PUB/cf.pdf), has been presented at a handful of meetups over the years. Chris Johnson, formerly of Spotify, has even published a Python implementation of implicit matrix factorization (https://github.com/MrChrisJohnson/implicit-mf).

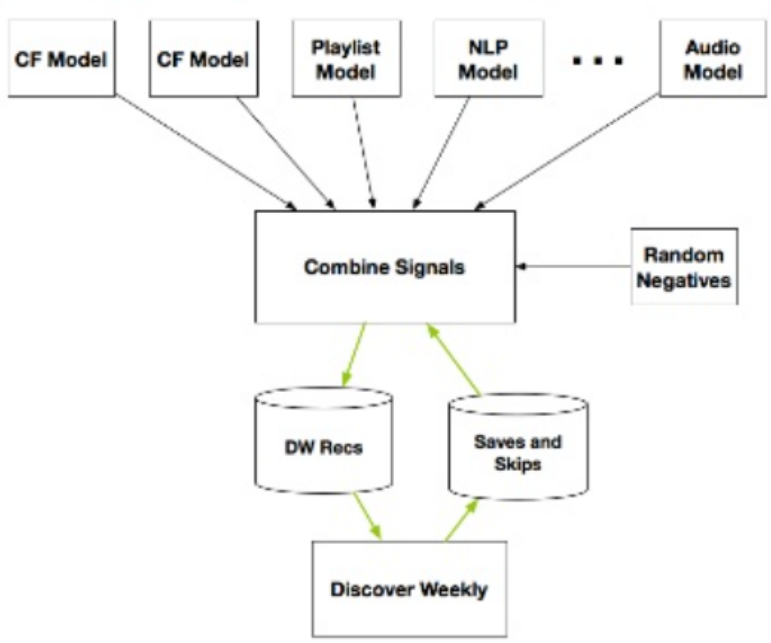

Figure 1: Recommender Structure

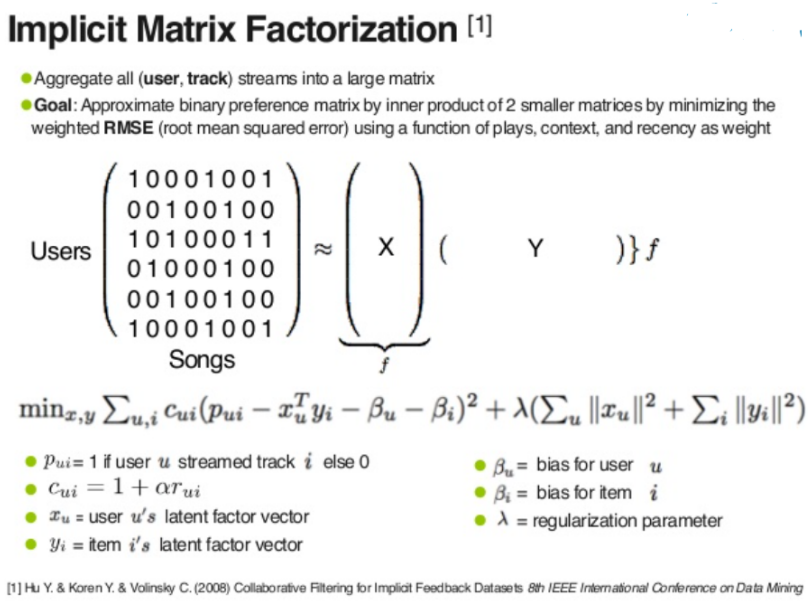

Figure 2: Matrix Factorization with RMSE

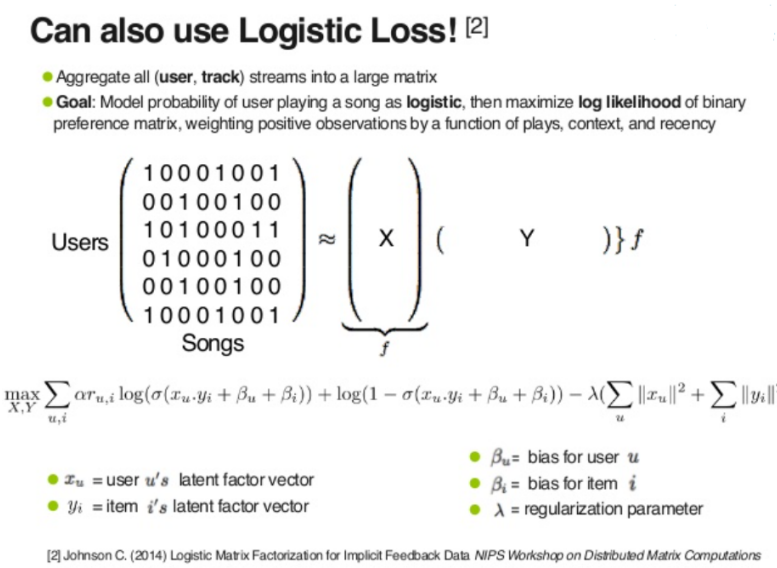

Figure 3: Matrix Factorization with Logistic Loss

Essentially, matrix factorization is applied to a binary representation of user/track preference where each element of the matrix represents whether a user has streamed a track. (Details can be found here https://www.slideshare.net/MrChrisJohnson/algorithmic-music-recommendations-at-spotify/19-Implicit_Matrix_Factorization19Replace_Stream_counts). The output of the method is a lower dimensional representation the universe of users and tracks. Each user has a vector that uniquely represents their musical preferences (i.e., it maps them to this space). An approximate nearest neighbors implementation is then used to search for similar users/items to get fast recommendations for each user (https://www.slideshare.net/erikbern/approximate-nearest-neighbor-methods-and-vector-models-nyc-ml-meetup). A Python implementation is again available (https://github.com/MrChrisJohnson/annoy)

2. Content-Based Filtering using Natural Language Processing (NLP)

NLP is applied to meta data associated with playlists.

3. Content-Based Filtering using Song Audio Files

This part of the system does Deep Learning on Audio files to learn latent embeddings for users and items, using implicit user/track feedback data. The network takes as input a user's history of implicit ratings (i.e., whether they stream a track for longer than a minimum time) and attempts to predict their listening history (approximately 2 weeks).

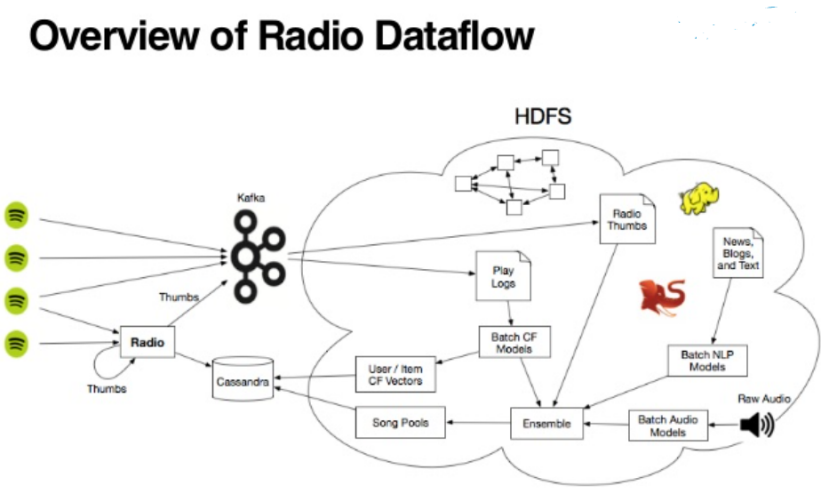

Figure 4: Ratio Data Flow

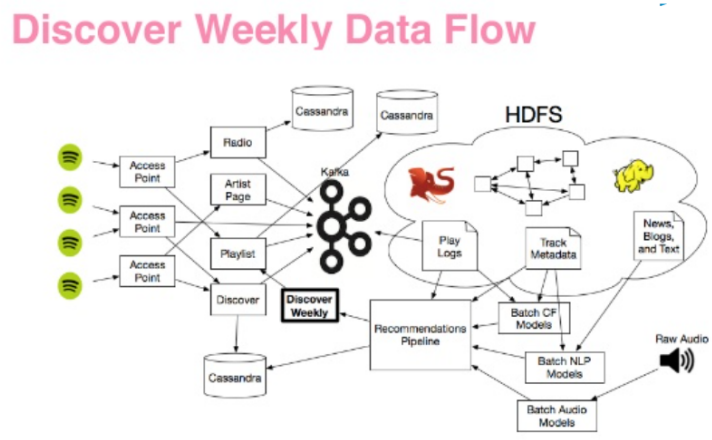

Figure 5: Discover Weekly Data Flow

Software

This discussion post was created using base R [R Core Team, 2018] and the R markdown [Allaire et al., 2018], tint [Eddelbuettel, 2018], kableExtra [Zhu, 2018], and tidyverse [Wickham, 2017] libraries.

Python Implementation Links

The following URLs provide Python implementations for some of the methods used in the ensemble:

https://github.com/MrChrisJohnson/implicit-mf

https://github.com/MrChrisJohnson/logistic-mf

https://github.com/MrChrisJohnson/deep-mf

https://github.com/MrChrisJohnson/annoy

Presentation Links

The following URLs provide fancinating details about featurs of the Spotify recommender system:

[2014-01-13] https://www.slideshare.net/MrChrisJohnson/algorithmic-music-recommendations-at-spotify

[2014-05-09] https://www.slideshare.net/MrChrisJohnson/collaborative-filtering-with-spark

[2014-06-27] https://www.slideshare.net/MrChrisJohnson/music-recommendations-at-scale-with-spark

[2015-01-06] https://www.slideshare.net/MrChrisJohnson/scala-data-pipelines-for-music-recommendations

[2015-09-25] https://www.slideshare.net/MrChrisJohnson/interactive-recommender-systems-with-netflix-and-spotify

[2015-09-25] https://www.slideshare.net/erikbern/approximate-nearest-neighbor-methods-and-vector-models-ny

[2015-11-16] https://www.slideshare.net/MrChrisJohnson/from-idea-to-execution-spotifys-discover-weekly

Key Papers

The following URLs provide details of matrix factorization on implicit data:

http://yifanhu.net/PUB/cf.pdf

http://stanford.edu/~rezab/nips2014workshop/submits/logmat.pdf

Software

References

JJ Allaire, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, and Winston Chang. *rmarkdown:*

Dynamic Documents for R, 2018. URL https://CRAN.R-project.org/package=rmarkdown. R package version 1.10.

Dirk Eddelbuettel. *tint: 'tint' is not 'Tufte'*, 2018. URL https://CRAN.R-project.org/package=tint. R package version 0.1.0.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018. URL https://www.R-project.org/.

Hadley Wickham. *tidyverse: Easily Install and Load the 'Tidyverse'*, 2017. URL https://CRAN.R-project.org/package=tidyverse. R package version 1.2.1.

Hao Zhu. *kableExtra: Construct Complex Table with 'kable' and Pipe Syntax*, 2018. http://haozhu233.github.io/kableExtra/, https://github.com/haozhu233/kableExtra.