# Recommendation Engines for Music Services

#### Peter Klipfel

## Abstract

The rise of music streaming services in recent years has seen an arms race by companies to produce high quality recommendations to users. The two most common approaches for implementing recommendation engines are content based filtering and collaborative filtering. These two techniques can utilize similar algorithms, but are different in the way that they classify similar content. Pandora, Echonest, and Spotify offer a good spectrum to analyze the different approaches with. Spotify uses the most diverse set of techniques for recommendations, and Echonest is the most constrained in what it can do.

## Introduction

Proliferation of data across the internet has pushed us into a new era of machine intelligence. The systems that we design have to leverage facts to come to a reasonable conclusion. Facts, here, are data points that are absolutely true. For example, when capturing click data, it is absolutely true that a user clicked a button. This area is often referred to as “Big Data”. The techniques for dealing with trillions of data points vary based on problem domain. This paper focusses on recommendation systems in music services.

There are many methods for recommending music, but it is most likely that a given music service uses a hybrid method. The commentary on music streaming services in this paper is mostly speculative, as the recommendation systems that are used are proprietary. However, the information that is not cited is from my conversations with employees at that company. The most common approach to recommendation engines is collaborative filtering, and because streaming services are listener-centric, collaborative filtering is the most common recommendation algorithm. Collaborative filtering has many variants and subsets that are beyond the scope of this paper.

## Methods

Of the numerous techniques used for recommending music, collaborative filtering is the most popular and successful technique [1]. Collaborative filtering comes in two flavors: memory based and model based. It is also possible to combine the two collaborative filtering techniques to create a hybrid technique. Because neither technique is absolutely better than the other, it often makes the most sense to use a hybrid system. The other way that services make recommendations is via content based filtering. In contrast to collaborative filtering, content based filtering uses similarity between items for recommendations [2]. That is, content based filtering recommends content that is quantitatively similar to content that the user has already shown interest in.

### Algorithms

The algorithms used in recommendation engines can usually be used for both collaborative filtering and content based filtering because the algorithms are just doing classification. The classifications are then applied in different ways. Classification algorithms are not new to the field of computer science, but they have gained increasing importance and relevance with the rise of data driven services. Because we are presented with so much data, it is important to have some kind of automated filtering for it.

The k Nearest Neighbor (kNN) algorithm was first proposed in 1967 as a general classification algorithm for large datasets. It probabilistically determines the neighborhood that a vector belongs in by testing the neighborhoods of k of its most closely related neighborhoods [3]. If we choose k=1, the nearest neighbor algorithm will place a vector in the neighborhood of its closest neighbor. It is important to note that kNN relies on a precomputed model of the data. This model must be updated as often as possible. The model classifies a training set into neighborhoods. The following classifications are made against the training set. Classifying a vector takes O(*knd*) time where *n* is the number of examples and *d* is the dimensionality of the example vectors [8]. Even if we let k be 1, this problem can take a long time.

If we consider the example of two users: User A and User B, suppose that the vectors that the kNN algorithm would be comparing are (*A B C D*) and (*A B C D E)*. For concreteness, we could say that (*A B C D E*) is in the “People Who Like Hip Hop” neighborhood. Then when we classify (*A B C D*) as being in the “People Who Like Hip Hop” neighborhood, we can reasonably suggest elements from other vectors in the Hip Hop neighborhood.

The amounts of data that large music streaming services have to deal with are so large that kNN becomes unwieldy. The approximate nearest neighbor (ANN) algorithm can be solved in O(*n* log *n*) time [4]. This is a small enough overhead that analyzing billions of vectors becomes a more reasonable size. However, using ANN forces the user to set k=1.

There are many other algorithms that can be used for classification like locality sensitive hashing[5], classifier chaining [6], and naïve Bayes classification [7]. Many of these recommendation techniques are only subtly different. This is an active area of research, and industry is pouring resources into it. In 2009 Netflix gave $1M to the winner of their challenge to improve recommendations [8].

### Collaborative Filtering

When recommendations are based on finding similar users, the technique is called collaborative filtering. Collaborative filtering matches users based on their ratings. If User A has shown preference for items *A, B, C,* and *D* and User B has shown preference for items *A, B, C, D,* and *E*; it seems reasonable to suggest item *E* to User A. In music services, a “show of preference” could be manifested as a thumbs-up in the case of Pandora or a rating in the case of Spotify. The more data points that are available, the better recommendations can be made. The problem reduces to one of vector analysis. The most common technique for solving this problem is kNN [9].

Music services that cater to the general public are a good use case for collaborative filtering because they have users who can rate music. Interestingly, the larger the service is, the more effective recommendations are. This is one of the primary reasons that datasets are so valuable. When services are new, and have very few listeners, recommendations are difficult. Intuitively, this makes sense. The fewer people you know, the harder it is to tell people what they might like.

Some services offer recommendations without ratings. Sometimes it is possible to collect data via implicit ratings. Implicit ratings are collected facts that imply a user preference. In the case of music; number of plays, shared links, and number of playlist appearances could constitute implicit ratings.

### Content Based Filtering

For services that don’t receive ratings, collaborative filtering is usually not an option. Instead, they use content based filtering. Rather than finding similar users, content based filtering finds similarities between different items. In an abstract sense, this can be compared to suggesting red items to a person who has only ever bought red objects. In music, this is done by analyzing songs and using the characteristics obtained from analysis to suggest similar songs. As a simple example, if all the songs that I have rated highly have a tempo between 130 and 140 bpm, content based filtering will suggest songs that are in a similar tempo range. This example assumes that the models we have developed incorporate the tempo data.

The algorithms discussed for collaborative filtering are also useful for content based filtering. However, the algorithms for identifying items are different. When classifying music, it is usually necessary to use the frequencies in a song (as opposed to amplitudes) for identification. This process is known as spectral analysis and is an open area of research. Two common approaches are wavelet analysis [10] or Fourier transformation [11]. These approaches allow the computer to find the key of a song and can help predict the kinds of instruments are present in a song. The process of identifying songs in this manner is sometimes called fingerprinting.

## Implementations

Most digital music services use a hybrid approach to recommendations. They will incorporate both collaborative filtering and content based filtering in to find appropriate music. Among the services that use content based filtering, the primary differentiator is the input to the models that they use. At Pandora, there are trained listeners who enter values for different attributes of a song [12], and at Echonest the process is automated. Spotify uses both collaborative filtering and content based filtering. Much of the commentary on internal workings of these companies is based on conversations that I have had with employees and should be regarded as speculation.

### Pandora

When a song is submitted to Pandora, it is placed in a queue of songs which are each listened to by trained musical classification employees. These employees assign a value to each of over 400 attributes. The process takes approximately 20 minutes per song [13]. Though this approach is not as scalable as automating the process, it allows Pandora to more thoroughly screen music. Pandora does not accept all the music that is submitted to them.

Because Pandora is classifying items, their recommendation system falls under the category of content based filtering. There is no public information on whether Pandora uses collaborative filtering. If they are using collaborative filtering, it might hurt the brand to release information about it. Without collaborative filtering, it becomes a challenge to provide the user with new music. If a song falls just outside the neighborhood of suggestion for a particular station, should Pandora suggest it? The answer appears to be yes. Pandora will occasionally make a “mistake” to see if the user appreciates the song [13].

### Echonest

Though there is no formal information on the web about how Echo Nest does its recommendations, it would seem that it does its recommendations based on a clustering algorithm around the different attributes that it can detect about a song. It’s possible that they are using any of a large set of clustering algorithms, but many of them are prohibitively slow for the dataset that Echo Nest has. Solving the problem exactly is likely unfeasible when comparing vectors across thirty-five million songs. It’s probable that they are using an approximate clustering algorithm that is continuously improved upon, or there are long lived background jobs computing the exact solutions over the course of days. Two possible algorithms are the approximate nearest neighbor algorithm and the locality sensitive hashing algorithm.

At the time of this writing, according to their front page, Echo Nest has 33,000 data points for every song in their database. This data is based on analysis of the song as well as scraped data from the internet. The analysis data contains information about the tempo, key, acousticness, danceability, speechiness, and more. The data that is scraped from the internet is run through natural language processing for filtering later. Because Echo Nest does not have a way for users to rate songs, recommendations must be using content based filtering when api users request recommendations [14].

### Spotify

Spotify has the most diverse recommendation engine of the services covered in this text. They use all of the methods of Pandora and Echo Nest [15]. They are also leaders in much of the technology that they use for these purposes. Additionally, they are much more open about their policies and technologies than the other two services covered in this paper. Their approach is quite hybridized.

For content based filtering, Spotify uses fingerprinting for their music to offer recommendations based on content based filtering. Echonest is partnered with Spotify, so it is likely that they are using Echo Nest for at least some of the fingerprinting. It is not clear whether Spotify does their own analysis of songs, though presentations given by Spotify make it appear that they are [16]. This classification is likely coupled with the non-analysis data that they use.

Much of the data that feeds the filtering models is gathered by web scrapers. They scrape data from open, public databases, and also from social media. Scraping social media allows Spotify to potentially build a graph of similar users elsewhere on the web. This can then be used to create a suggestion graph for those users [17]. The data that Spotify gathers also contains social tags that can be analyzed and used for both collaborative filtering and content based filtering techniques [18].

The hundreds of billions of data points that must be parsed through at Spotify is big enough that they spend a considerable amount of resources optimizing them. One such way is via matrix factorization [16] which is a way of decomposing matrices into more useful vectors [19]. They use probabilistic models for finding appropriate decompositions of matrices. They also use both explicit and implicit ratings for the collaborative filtering models.

## Conclusion

The world of recommendation engines is rapidly expanding and becoming more necessary every year. In the context of music streaming services, there does not seem to be a *best* way to do recommendations. The most popular is collaborative filtering, though, content based filtering is also in wide use. For further research, it would be interesting to find out how different streaming services scale the processes. When databases can no longer fit on hard drives, analyzing them requires novel technologies. One of the largest bottlenecks in the industry right now is the technology for creating these real time systems.

The closer to real time the models can be, the more accurate suggestions will be. Twitter faced this issue, and solved it by creating Storm, and using Nathan Marz’s lambda architecture [21]. This architecture replaces a monolithic batch processing unit with a real-time streaming unit and a batch processing unit. Storm made a splash in the open source community because it was a powerful stream processing framework. Recently, LinkedIn has released Samza, which appears to be similar to storm [22]. If and when someone comes up with a reasonable algorithm for stream processing of collaborative and content based filtering, the world of recommendations will change dramatically. The paradigms for developing complex map reduce jobs in hadoop will transform to paradigms for creating processing nodes in a complex topology for stream processing.

In the music space, we are very close to being able to produce impressively pleasant music via computers. As stream processing becomes more powerful and the media datasets become larger, I predict that we will see the advent of music composition by computers. Our recommendation engines will become so good that they will be able to compose music themselves. An endless stream of unique music will be created by computers that are fed with songs and data about those songs. Echonest has already proven some of the concepts behind this [23]. The only question that remains to be answered is: will anyone care?

## Bibliography

[1] S. K. Lee, Y. H. Cho, and S. H. Kim, “Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations,” *Inf. Sci.*, vol. 180, no. 11, pp. 2142–2155, Jun. 2010.

[2] V. Formoso, F. Cacheda, and V. Carneiro, “Algorithms for Efficient Collaborative Filtering,” in *Efficiency Issues in Information Retrieval Workshop*, p. 17.

[3] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[4] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, “An optimal algorithm for approximate nearest neighbor searching fixed dimensions,” *J ACM*, vol. 45, no. 6, pp. 891–923, Nov. 1998.

[5] M. Slaney and M. Casey, “Locality-Sensitive Hashing for Finding Nearest Neighbors [Lecture Notes],” *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 128–131, Mar. 2008.

[6] G. Tsoumakas and I. Katakis, “Multi-label classification: An overview,” *Int. J. Data Warehous. Min. IJDWM*, vol. 3, no. 3, pp. 1–13, 2007.

[7] H. Zhang, “The optimality of naive Bayes,” *A A*, vol. 1, no. 2, p. 3, 2004.

[8] Netflix, “Netflix Prize: Home,” *Netflix Prize*. [Online]. Available: http://www.netflixprize.com/. [Accessed: 20-Oct-2013].

[9] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, “Recommender systems survey,” *Knowl.-Based Syst.*, vol. 46, pp. 109–132, Jul. 2013.

[10] D. T. Lee and A. Yamamoto, “Wavelet analysis: Theory and applications,” *Hewlett Packard J.*, vol. 45, pp. 44–44, 1994.

[11] G. D. Bergland, “A guided tour of the fast Fourier transform,” *IEEE Spectr.*, vol. 6, no. 7, pp. 41–52, 1969.

[12] S. Hardy, “Creative Generalist Q&A: Tim Westergren | Creative Generalist,” *Creative Generalist Q&A: Tim Westergren*, 31-Oct-2006. [Online]. Available: http://creativegeneralist.com/2006/10/creative-generalist-qa-tim-westergren/. [Accessed: 20-Oct-2013].

[13] L. Tischler, “Algorhythm and Blues,” *Fast Company*, 01-Dec-2005. [Online]. Available: http://www.fastcompany.com/54817/algorhythm-and-blues. [Accessed: 20-Oct-2013].

[14] B. Whitman, “How music recommendation works — and doesn’t work,” *Brian Whitman @ variogr.am*. [Online]. Available: http://notes.variogr.am/post/37675885491/how-music-recommendation-works-and-doesnt-work. [Accessed: 20-Oct-2013].

[16] “Recommendations at Spotify v4” [Presentation Slides]. Available: http://files.meetup.com/1516886/Recommendations%20at%20Spotify%20v4.pdf. [Accessed: 20-Oct-2013]

[16] E. Bernhardsson, *Music recommendations at Spotify*. 2013.

[17] S. Tan, J. Bu, C. Chen, and X. He, “Using Rich Social Media Information for Music Recommendation via Hypergraph Model,” in *Social Media Modeling and Computing*, S. C. H. Hoi, J. Luo, S. Boll, D. Xu, R. Jin, and I. King, Eds. Springer London, 2011, pp. 213–237.

[18] A. Nanopoulos, D. Rafailidis, P. Symeonidis, and Y. Manolopoulos, “MusicBox: Personalized Music Recommendation Based on Cubic Analysis of Social Tags,” *IEEE Trans. Audio Speech Lang. Process.*, vol. 18, no. 2, pp. 407–412, 2010.

[19] D. Seung and L. Lee, “Algorithms for non-negative matrix factorization,” *Adv. Neural Inf. Process. Syst.*, vol. 13, pp. 556–562, 2001.

[8] Olga Veksler, “CS434a/541a: Pattern Recognition - KNN”.

[21] *Runaway complexity in Big Data systems... and a plan to stop it - Nathan Marz*. 2013.

[22] “Apache Samza: LinkedIn&#039;s Real-time Stream Processing Framework.” [Online]. Available: http://engineering.linkedin.com/data-streams/apache-samza-linkedins-real-time-stream-processing-framework. [Accessed: 23-Oct-2013].

[23] Echo Nest, “Echo Nest Remix,” *Echo Nest Remix*. [Online]. Available: http://echonest.github.io/remix/examples.html. [Accessed: 23-Oct-2013].