Master Degree Project

UNIVERSITY OF SKÖVDE

A Scalable Recommender System for Automatic Playlist Continuation

Master Degree Project: Report

Master Degree Project in Data Science
One Year Level 60 ECTS
Spring Term

Author: Jack Bennett
Contact: a17jacbe@student.his.se

Supervisor: Dr. Alan Said
Examiner: Prof. Göran Falkman

2018-May-31

Abstract

As major companies like Spotify, Deezer and Tidal look to improve their music streaming products, they repeatedly opt for features that engage with users and lead to a more personalised user experience. Automatic playlist continuation enables these platforms to support their users with a seamless and smooth interface to enjoy music, own their experience, and discover new songs and artists.

This report details a recommender system that enables automatic playlist continuation; providing the recommendation of music tracks to users who are creating new playlists or curating existing ones. The recommendation framework given in this report is able to provide accurate and pertinent track recommendation, but also addresses issues of scalability, practical implementation and decision transparency, so that commercial enterprises can deploy such a system more easily and develop a winning strategy for their user experience. Furthermore, the recommender system does not require any rich and varied supply of user data, instead requiring only basic information as input such as the title of the playlist, the tracks currently in the playlist, and the artists associated with those tracks.

To accomplish these goals, the system relies on user-based collaborative filtering; a simple, well-established method of recommendation, supported by web-scraping and topic modelling algorithms that creatively use the supplied data to paint a more holistic image of what kind of playlist the user would like. This system was developed using data from the Million Playlist Dataset, released by Spotify in 2018 as part of the Recommender Systems Challenge, evaluated using R-precision, normalised discounted cumulative gain, and a proprietary evaluation metric called Recommended Song Clicks, that reflects the number of times a user would have to refresh the list of recommendations provided if the current Spotify user interface was used to communicate them. Over an 80:20 train-test split, the scores were: 0.343, 0.224, and 15.73.

Contents

1.0 Introduction

Accurately defining Data Science, as both a theoretical field and commercial industry, remains difficult. To consider it as a combination of Computer Science and Statistics can perhaps cover much of the necessary ground, but as technology continues to move forwards at rapid pace, this definition must continually evolve - as must the skills of a Data Scientist. Applications of machine learning and other Data Sciences are becoming more common in many industries, from healthcare, to finance, to defence, with the French politician and mathematician Cédric Villani claiming that soon, "*Artificial Intelligence will be everywhere, like electricity*" (Motoyama, 2018).

One example of such artificially intelligent (AI) architectures is the recommender system. These systems already permeate daily life for many people around the world, recommending useful products, services, and anything else, to users. Well-known examples of recommender systems include Amazon's personalised systems that recommend products to customers, which boost revenues for Amazon and improve the customer experience. Likewise, McDonald's customers in Japan spend an average of 35% more when they are targeted by a personalised item-recommender system (Marr, 2018).

As businesses have become more cognisant of this impact that artificial intelligence can have, the applications themselves have become more diverse. In the music industry, a rapidly growing area of recommender systems research is in automatic playlist continuation, where music tracks are recommended to users who are creating or curating music playlists. The research in this area so far is limited, but as music streaming and personalised music management services like Spotify, Tidal and Deezer grow in popularity, this area of research has become more and more pertinent from a commercial perspective (Schedl et al., 2018).

To focus further research into this area, Spotify have collaborated with the ACM Recommender Systems (RecSys) community and released a large dataset of real playlist data from their service, as part of the 2018 RecSys Challenge. The ACM RecSys Challenge is an annual competition associated with the ACM Conference on Recommender Systems, which in previous years has focused on other emerging trends within the recommender systems field, such as automated recruitment screening and targeted advertising. The explicit aim of anybody taking part in the RecSys Challenge this year will be to develop a recommender system that can recommend relevant music tracks for a user who is manually building a personal playlist. Such a project would naturally involve the use of data mining theory, statistical logic and computer programming tools.

In practice however, a good recommender system not only needs to be accurate and effective, but easily implementable, interpretable and scalable. Businesses are always constrained by budgets, processing capabilities and regulations; and their primary interest is not research, but servicing their customers. A complicated idea that works on paper but that can only be implemented by an expert team of engineers is unlikely to be used, black-box systems may

breach regulations and non-technical executives may be unwilling to embrace them, and most importantly, slow recommender systems or systems that require state-of-the-art infrastructures to run quickly are unlikely to deliver great customer experiences. Therefore, the aim of this project is not only to develop a system which provides accurate recommendations for automatic playlist continuation, but one which can scale to can handle industrial volumes of data on a simple architecture.

1.1 Recommender Systems - A Background

In large commercial environments, customers may find themselves faced with an overwhelming level of product choice, and manually sifting through the associated information to find the right purchase can be an intractable task. As noted above, using a recommender system to filter the information down to only that which is relevant can simplify the customer experience and drastically increase the likelihood of the customer spending money with that business (depending on the quality of the recommendation). These customer-centric systems are typically complex, and may need to present an ordered selection, or choice of relevant recommendations to the user, determined by perhaps limited data. Furthermore, most large businesses will have a variety of customers with their own tastes and personal preferences, and it is becoming essential to personalise the system output for each user in order to accommodate for this (Dodda, 2018).

Despite the inherent complexity of the problem, it is difficult to find an industry-leading business that does not have an effective solution. While online retailers like Amazon, ShopDirect and eBay represent excellent examples of businesses that use user-item recommendations to drive revenue, the same principles have been applied in other domains; Airbnb (McMillan, 2014), Tinder (Crook, 2015) and Spotify (Ciocca, 2017) have all developed powerful, personalised systems that improve the customer experience for their respective users - recommending travel destinations, romantic partners, and new music with great accuracy.

The logic behind systems like these has evolved as data mining techniques have advanced over the years, particularly since the formation of the GroupLens research group at the University of Minnesota in the early 1990s. Whilst a more detailed presentation of the known methodologies for effective recommendation will be presented in the next section, there is a general consensus that the field of recommender systems turned a significant corner in 1994, when Paul Resnick and John Riedl published their seminal paper on collaborative filtering (Said, 2017). This algorithm personalises recommendations by using user data; user profiles can be built and recommendations for these profiled users can be made based on what had been enjoyed by other users with similar profiles.

Whilst research into the field of recommender systems and the associated personalisation AI would continue to progress, the biggest advancements in the state-of-the-art arguably arose as

a result of *The Netflix Prize* a decade later. In 2006, Netflix (a DVD-rental and movie streaming service) released a dataset containing 100 million movie ratings made by their users. Netflix challenged the data mining community to develop a new recommender system that could outperform their current model, and offered a $1 million prize to anybody who could achieve results above a certain threshold. Such a large offering increased the focus within the data mining and machine learning community on the development of recommender systems, and many new and innovative approaches were developed as part of this competition (Amatriain and Basilico, 2012).

1.2 The 2018 Recommender Systems Challenge

The ACM Conference on Recommender Systems has been held annually since 2007, typically in either Europe or North America. It is an international forum for new research, ideas and collaboration within the broad field of recommendation and recommender systems. Since 2010, along with the workshops, tutorials and conference presentations that usually feature at such events, an annual academic competition has also taken place. This "RecSys Challenge" is in many ways the natural successor to *The Netflix Prize* - it is competitive, it uses real data and it is designed to sharpen the analytic focus and instigate new research into specific areas of recommender systems theory (Said, 2016). Previous challenges have been centred around topics such as movie recommendation, advertisement campaigns, user engagement, and candidate recommendation within professional recruitment. This year, the challenge is being carried out in coordination with Spotify AB, to focus on automatic music playlist continuation.

Founded in 2006 by Daniel Ek, Spotify AB is known as one of Sweden's 'Big Five' tech companies, and is the global leader in music streaming. The Spotify service is subscribed to by over 70 million users worldwide (Plaugic, 2018), who use the service to legally and easily stream music, listen offline or on mobile devices, and create their own custom playlists (with many more millions of users enjoying a free version).

Currently, Spotify users can create their own playlists by searching for songs within the service before manually adding them to their custom playlist. Users can also create a name for their playlist, write a description and even upload cover art. While the feature works well for many users, many others user struggle to develop new playlists effectively, as they do not have the extensive knowledge to find the music they are looking for. To therefore aid the 'new music discovery' process, Spotify already have a recommender system in place; as users build their playlist, Spotify will provide them with a list of ten other songs that the user can review and select from. The list of ten is refreshable, and the recommendations are based on the songs that are already in the playlist, so that the general playlist theme (for instance, 'happy music', 'country and western', 'going to the gym' and so on) can continue.

To allow researchers the means to carry out further research into this area, Spotify has released The Million Playlist Dataset (MPD). The MPD features one million real, user-generated playlists of varying nature, along with associated metadata. As the playlists are genuine, any models that are developed should be statistically robust enough to achieve the same level of performance if they were deployed within a real and dynamic production environment, especially since the dataset as a whole features a reasonably realistic level of variety, in terms of both playlist theme and playlist completeness. This latter characteristic refers to the existing size of the playlist and the quality of the available metadata; the task is not to generate new playlists for users but to recommend appropriate extensions of playlists that are already under construction. This is significantly harder when the only available data point is the playlist title, whereas it is easier when there are already 100+ songs in the playlist.

2.0 Literature Review

From a theoretical standpoint, recommender systems can be designed and implemented many different ways, and there is no specific 'go-to' design that will always work best. The best recommender systems are in fact those which achieve the desired results using the simplest and most transparent approach, selecting any algorithms in respect of both the nature of the available data and the exact type of solution that is being sought. This section documents the methods that are commonly used in recommender systems and highlights some of the common issues, both in general and within the music industry, before taking a look at some of the current state-of-the-art approaches.

2.1 Popular Methods

2.1.1 Non Personalised Methods

Some of the most straightforward (yet often surprisingly effective) systems are those that use non-personalised approaches (Said, 2017), such as recommending the item that is the most popular across the user-base. For instance, recommending the movie with the highest average rating in a DVD-rental/movie streaming context, or recommending the most-purchased piece of clothing on an e-Commerce website. This is common sense - if the item is known to be widely popular, then many users are likely to benefit from the recommendation. Variations of this strategy include recommending multiple popular items so that the user retains some element of choice, and also subsetting the item space into categories, and recommending the range of items that are each the most popular item within their category. For example, segmenting a movie database by genre, and then recommending a list of movies, featuring the most popular action movie, thriller, rom-com, sports movie and family movie. Other data aggregations are also common, as different metrics could be used to determine popularity. A simple movie recommender system might suggest the top-rated movies, along with the movies with the most

overall views, or movies that are 'trending'. 'Trending', is a more recently derived aggregation that typically relates to the items that have gained or surged in popularity in some recent time period, but there does not seem to be a consistent definition that is more precise than this.

The trade-off for this level of simplicity and ease of implementation is some level of naïvety in return. Such a basic recommender system cannot cater for particular interests, tastes, or user subpopulations given the implicit assumption that all users enjoy the same things. With this in mind, and given how easily such a system could be implemented, these methods can be used to benchmark other, more advanced approaches that seek to personalise recommendations.

2.1.2 Collaborative Filtering

Collaborative filtering (CF) is more advanced algorithm which does permit personalised recommendations, based on the similarity between either users or items. User-based CF (Resnick et al., 1994) assumes that users who behaved similarly in the past will agree again in the future. A common implementation of this idea uses the historic likes and dislikes that have been made by users to evaluate their level of similarity, before recommending to 'target' users the items which have been enjoyed by other system users to whom the target user is considered similar. In the example context of movie recommendation, the similarity between two users might be established using the ratings left on a movie that has been rated by them both. The more similar their ratings, then the more similar the two users are thought to be (and this is ideally carried out across a large set of mutually-rated movies). This approach is typically implemented using a *k*-nearest neighbours approach, where the preferences of the *k* most similar users are taken into account, ensuring that whilst the recommendation is still personalised, the system is less sensitive to the impact of outlier ratings.

Item-based CF (Sarwar et al., 2001) is similar, but focuses on the similarity between items instead. This similarity is established by looking at the co-ratings between items made by all the users in the database; and if it can be seen that two items were consistently ranked similarly by the various users who used both, then the two items are assumed to be similar. To make a recommendation for a user, reference is made to the existing set of items that the target user already likes, and items that are known to be similar to those that he or she already likes are recommended. The effects of item-similarity and user-preference are usually weighted and averaged together when determining the optimal recommendation.

While user-based CF is conceptually simple and reasonably logical, it can be very computationally expensive when the user or item base is very large. Item-based CF can overcome this to some degree, as in many applications the number of items is much lower than the number of users. Both approaches however are often victims of the data sparsity problem; wherein data matrices[1]

[1] Ratings matrices are the tabular data structures that these algorithms typically operate on. They contain the user-item relationships (e.g. like, dislike, no interaction, etc.) that are to be exploited by smart algorithms

are typically extremely sparse as users will have typically only interacted with a tiny subset of the available products, and so large overlaps in ratings (and statistically robust similarity calcuations between users and items) are rare. For similar reasons, CF systems often struggle to make accurate recommendations for new users or of new items, because they lack the historic data on which similarity is established and decisions are made. This is known as the 'cold start' problem.

Matrix factorisation approaches can also support another version of collaborative filtering. In large datasets, with many items and perhaps many, many users, carrying out nearest neighbour operations on the full ratings matrix becomes far too computationally intensive. An assumption can however be made, in that user preference will be based on some underlying features in the data and/or some underlying personal interests. For instance, consider a preference for action movies as an example of a latent feature in some hypothetical movie recommendation ratings matrix. A user must have their own level of 'interest in action movies', and a movie must have its own level of 'action movie-ness' about it, and by combining the two variables, an estimate of how much the user will like the movie can be obtained. Of course, an accurate estimation would need to take many of these latent factors into account.

Singular Value Decomposition (SVD) (Sarwar et al., 2000) can be used to find these latent factors. Consider a ratings matrix X, containing elements r_{ij} as the rating of each user i on each item j. Each r_{ij} can be calculated in terms of the latent factors by $u_i {v_j}^T$, where u_i is a vector that quantifies the association between all of the latent factors and user i, whilst v_j represents the same but in respect of the latent factors and item j. The dimensions of these vectors is equal to the rank of X as this is equivalent to the total number of latent factors that can influence any r_{ij}. Reconstructing each r_{ij} in X (where missing ratings are substituted with the mean item rating) from $u_i {v_j}^T$ and minimising the error (typically with either stochastic gradient descent or alternating least squares, and both typically with regularisation) will find the optimal parameters for all u_i and v_j. This gives $X = UV^T$, where U and V are matrices of all u_i and v_j.

Standardising the values in U and V to the same scale gives $X = USV^T$, where S is a diagonal matrix of *singular values*, the magnitudes of which quantify the importance of the corresponding factors. By removing the least important factors (those with the smallest corresponding singular values), a close approximation, $\hat{X}$, of X can be obtained. Crucially, $\hat{X}$ will have all of its elements populated and therefore rating predictions for the unrated items in X can be found from the corresponding entries in $\hat{X}$, at a greatly reduced computational cost compared to that of which a standard CF approach would require. These matrix decomposition techniques however are still iterative algorithms and so it may be necessary to carry out some initial segmentation prior to SVD when working with extremely large datasets to further reduce the computational overhead.

In this discussion around collaborative filtering, much reference has been made to 'ratings', 'likes' and so on as a way of understanding user preference. Typically, this kind of data is either explicit

or implicit. Explicit data is data that the user has consciously supplied with the intention of quantifying their sentiment towards an item, such as a five-star rating on a movie, whereas implicit data only suggests what the user preference might be. An example of implicit data might be the number of views on an item, where the assumption is that a greater number of views indicates a higher level of interest. Implicit data might be harder to obtain and make sense of, but this type of data is actually preferable to explicit data (Said, 2017). Explicit data tends to be inaccurate (for instant, good movies are usually rated very highly while other movies are just not rated at all) and due to unconscious biases, users tend to build somewhat idealistic profiles for themselves when given the opportunity to use these explicit ratings. It is much harder for users to obscure their true interests and preferences when they are modelled using indirect and implicit data. Sophisticated techniques do exist to model implicit feedback with great accuracy (Hu et al., 2008) but in many cases it is adequate to model implicit feedback broadly, as either 'positive' or 'negative' (or even more generally, as 'not positive') (Kirwin, 2016).

2.1.3 Content-Based Filtering

As an alternative to collaborative filtering, content-based (CB) filtering is another widely used method for designing recommender systems (Song et al, 2012). The 'content' of item refers to the actual descriptive characteristics of an item, such as (at a high level) the genre of a movie and the actors in its cast, or (at a low level) the pitch and tempo of a music track. In a CB filtering system, these characteristics are often broken down into 'tags' and items matched to user preferences (as modelled by the tags) accordingly. For example, a content-based movie recommender system would likely recommend the movies *Taken* and *The Commuter* to a user who liked movies with the tags 'action' and 'Liam Neeson'. Depending on availability of accurate tags for users and items, content-based filtering can help a recommender system overcome the cold-start problem. In some systems, new items can be tagged with descriptive tags by the users who use the item, and user preferences can be established either by their historic preferences, or by an up-front prompt for preferences when the user joins the system. The drawbacks of using descriptive tags is that they can be broad descriptors, only providing limited insight into the actual item.

Outside of matrix factorisation, it has been noted that quantitatively establishing 'similarity' between either users and items is often necessary. This can be done with several methods in both collaborative filtering and content-based filtering systems. In cases where the data vectors are numeric, (such as the historic movie ratings of two users who have rated all the same movies), then the similarity between the users can be established from the distance between the two data vectors. Measures such as the Euclidean distance, Cosine distance or Pearson's r correlation coefficient can all be used for this. In cases where the data is categorical, (such as 'like', 'dislike' ratings, or vectors of user-item tags) then the Jaccard index or Hamming distance can be used instead.

2.2 The Particular Challenges of Music Data

Working with music data and playlist data poses many challenges, specifically in respect of scalability, evaluation, recommendation serendipity and user modelling.

Large datasets are very common. In the case of this analysis for example, the MPD alone features one million playlists and over two million unique songs, and so a complete ratings matrix would contain over two trillion entries. This is before further metadata is even taken into account; data points such as the playlist title and description, and any derived information from those, as well as artist and song metadata. In cases like this, there are significant concerns from a computational perspective and it is not always feasible to directly apply standard methods. As a comparison, Netflix has only a few thousand movies and TV shows on its platform (Luckerson, 2016).

Another well documented problem in the recommender systems domain relates to their evaluation, as quantifying and measuring the end-value impact the system has on the user is not straightforward. In this music context, Schedl et al. (2018) go beyond the known difficulties in assessing the error and accuracy of song recommendation, to highlight a number of further considerations. These authors argue that a good music recommender system should be able to maintain a level of diversity and novelty in the recommendations delivered in order to prevent the user from getting bored with what turns out to could otherwise be a fairly monotonous playlist.

Furthermore, it is quite likely that the users of a service like Spotify will also be listening to music elsewhere, such as on the TV and on the radio; and whilst collaborative filtering algorithms generally work well, their logic often leads to the recommendation of music that is already widely popular. This means that systems which rely solely on collaborative filtering methods may not actually perform well in practice as the user may already be familiar with the songs recommended. There is no feeling that the system is helping them discover new music. From a technical perspective, items can be seen as being in either ‘the head’ or ‘the long tail’ (Said, 2017). The former category represents the small subset of the total items that are widely known and widely popular, whereas the vast majority of items are all otherwise found in the long tail. At best, they are niche, and at worst, they are rubbish - however a system that is highly nuanced and personalised should be able to retrieve the items from the long tail that the target user (unlike most other users) would actually really enjoy, whilst having little or no prior familiarity. Supporting the discovery of new music like this is highly desirable and would enormously boost the user experience, however it requires excellent user modelling in order to work successfully.

Good user modelling is a fundamental requirement of any personalised recommender system, as it provides the frame of reference from which recommendations are served. In regards to music however, user tastes and interests are very subjective; two users may listen to exactly the same songs but for very different reasons and this can impact the performance of a system

designed to support playlist creation or extension. The same songs may be in many different playlists for many different reasons, due to the subjectivity of musical taste and the different ways in which different users match their emotions and feelings to music. Understanding and modelling these aspects of the user's mood and motivation is very difficult (Schedl et al., 2018). These authors also note the social and contextual factors involved in music preference, noting that the time of day can influence listening behaviour (something that is currently incorporated into Spotify's own existing recommender systems). This is an example of *context*-based filtering.

2.3 State-of-the-Art Approaches

Most recommender systems in production are driven by the logic from either the collaborative filtering (user-based, item-based or SVD-based CF) or content-based filtering methods discussed in the sections above. This section details two further areas of recommender systems and machine learning theory: model hybridisation and topic modelling. These approaches seek to address some of the weaknesses that emerge when off-the-shelf CF or CB methods are applied to problems within the music industry.

2.3.1 Hybrid Models

Within machine learning generally, model hybridisation is seen as a reliable approach to overcome the various shortfalls of different models that each have their own respective strengths and weaknesses. In a combined or hybridised model, the model performance is typically more robust, especially when the strengths and weaknesses of the various component models complement one another; this allows the performance of the model to remain strong even under the various conditions where the different component models may independently perform poorly. For example, whilst collaborative filtering generally performs well, it is affected by the cold-start problem and popularity bias. Content-based filtering is not affected by these issues and so combining the two approaches could lead to an even better recommender system, since the ability to make content-based recommendations persists even when there is not enough data to make effective CF-based recommendations. There are many ways to combine or hybridise models, though the simplest option is just through linear combination.

Spotify already use a hybrid model in their *Discover Weekly* service (Ciocca, 2017). *Discover Weekly* is an automatically generated playlist, unique for each user and deeply personalised to their tastes. It is an extremely popular feature of the service, with tens of millions of users using it to stream billions of tracks (Popper, 2016) which are often by lesser-known artists. This kind of automatic playlist generation is very closely related to the current issue of automatic playlist continuation, and model hybridisation has allowed Spotify to connect the dots between artists and listeners very well for this purpose (Pasnick, 2015).

Bernhardsson (2013), Ciocca and Pasnick all provide excellent detail on how Spotify's hybridised *Discover Weekly* algorithm works; standard user-based collaborative filtering approach is used in tandem with a context-based model that is trained to learn the fine and granular relationships between artists and musical genres by analysing natural language descriptions, reviews and discussions scraped from the web. User taste is modelled accordingly and in real-time. A third component in the final system is a content-based model, which uses convolutional neural networks (a type of 'deep learning') to analyse visual representations of the raw audio data of the songs that the user already enjoys, so that songs from the long-tail can be recommended based on their similar timbre, pitch and key.

2.3.2 Topic Modelling

Of course, the *Discover Weekly* solution seeks to holistically model the end user to provide their perfect playlist. Schedl et al. (2018) and Song et al. (2012) consistently note in their work that since music preference has such strong roots in emotion and mood, it may be beneficial to model users psychologically. In this application, the users are anonymous in the data and the challenge is instead to recommend tracks to add to an existing playlist, however this does not represent an obstacle. Whilst automatic playlist generation requires a holistic understanding of who the target-user is, automatic playlist continuation only needs to model the mindset of the target user *in that moment*. This is because users will almost certainly have a range of playlists that they listen to at different times, and for different reasons, and it is underlying purpose of the playlist that matters from a recommendation perspective. Modelling the mood, emotion and underlying context of the playlist can then provide a logical basis from which to recommend tracks. The genre of many of the tracks would strongly relate to this as well.

In natural language processing (NLP), topic modelling is used to determine the general themes that written documents could be associated with. For example, if a corpus of several hundred newspaper articles were subjected to topic modelling, the words within them would be analysed and small number of latent topics would emerge, such as 'sport', 'business', and so on, with the articles each being classified accordingly. This project report might be classified as being about 'music', or 'data science', or both. The same NLP techniques can be applied in the context automatic playlist continuation, where the tracks and playlists are analogous to words and documents, and the latent topics reveal the latent 'mood' of the playlist.

Probabilistic latent semantic indexing (pLSI, also known as probabilistic latent semantic analysis) is one of these NLP topic modelling techniques. An adaptation of the SVD-based LSI approach (Deerwester et al., 1990), pLSI probabilistically maps terms and documents onto a latent topic space (Hofmann, 2001). In this method, the number of latent topics is determined *a priori* (Oneaţă, 2011), and the model is expressed as:

$$P(d_i, w_j) = P(d_i)P(w_j \mid d_i), \qquad P(w_j \mid d_i) = \sum_{K=1}^{K} P(w_j \mid z_k)P(z_k \mid d_i),$$

where $P(d_i, w_j)$ is the joint probability of a term i and document j. This follows since the likelihood of a term given a document depends on the distribution of topics (modelled by z_k) within that document combined with the likelihood of a word being used in regards to a certain topic. Parallels can be drawn here to the factorised matrices that result from SVD. The expectation maximisation (EM) algorithm (Dempster, 1977) is used to fit the model parameters and so the distributions of topics within documents and the distributions of words within topics are obtained. In the context of music recommendation, these distributions can be used to make recommendations, matching new tracks to playlists based on their modelled topical content.

An extension to (and generalisation of) pLSI is Latent Dirichlet Allocation (LDA). LDA (Blei et al., 2003) works in very much the same way as pLSI, however it is more popular since it takes a Bayesian approach of attaching prior probabilities to the document-topic and topic-word relationships which are then optimised through Gibbs sampling (Geman and Geman, 1984). This generally leads to better results than pLSI-based models do. The priors are obtained from a Dirichlet distribution, though when the volume of data is very large anyway, they can have a reduced impact. LDA is also in practice more easily implemented, as its popularity as a method has led to the development of a number of robust and open-sourced implementations.

Topic modelling is an unsupervised process. Exactly what defines a topic is subjective; there may be some hierarchical order; and so on, and so evaluating an obtained topic model is tricky for there is no hard 'ground truth' to compare the output against. Visual approaches for evaluation such as bubble and bar plots are commonly used, and more advanced visual analytics procedures have also been put forth by Chuang et al. (2012) - but visual analytics is difficult to automate. Mimno et al. (2011) discuss a quantitative 'coherence score' for topics, the average of which could be taken to evaluate a topic model. Their metric is given by:

$$C(t; V^t) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} log \frac{D\left(v_m^{(t)},\ v_l^{(t)}\right) + 1}{D(v_l^{(t)})},$$

where the coherence for a topic given its vocabulary $C(t; V^t)$ is obtained by dividing the number of documents containing both of a pair of words known to be very popular within the topic $D\left(v_m^{(t)},\ v_l^{(t)}\right)$, divided by the number of documents which contain just one of those words $D(v_l^{(t)})$. The log is taken of this value and for that of all other possible pairs from M, the set of the most popular words within the topic. The logic of this metric lies in the assumptions that the most robust, independent and coherent topics will be reliant on a small set of words that occur

very frequently in the context of that topic and very rarely elsewhere. This logic should extend to word-pairs, and if it doesn't, then there is the suggestion that the topic being evaluated actually contains a number of unrelated topics or sets of 'intruding' words, and that the words in M do not make sense together. Newman et al. (2010) also describe coherence metrics. Besides using heuristic evaluation metrics, different models can also be compared through A/B testing. If the topic model lies within a recommender system, changes to the topic model can be evaluated by observing positive or negative changes in the quality of recommendation. Of course, a good eye-balling of the top results in any of the word-topic distributions is a reliable, common-sense approach as well.

3.0 Implementation

This section details the main considerations that were made as part of the analytic approach to building a scalable recommender system, and also describes how the system was tested and evaluated. Predicting which tracks a user would like to add to their playlist is a difficult challenge, and so developing a solution that was sophisticated enough to capture the purpose of the playlist while remaining quick, scalable and transparent was not straightforward. The analysis and modelling was carried out using Python 2.7.14, and the final system was also implemented in this language. The computation was carried out using a 2011-edition Windows laptop computer.

The proposed framework draws from several of the areas of the literature review detailed above. Less complex methods of recommendation, such as collaborative filtering and popularity aggregators are well-documented, well-understood, and have been shown to be portable across various application scenarios, so it is therefore not unreasonable to assume that these methods can be applied to this emerging domain of automatic playlist continuation as well. While it is highly likely that as-yet unknown, specialised methods developed solely for this domain might outperform collaborative filtering, CF is easy to implement and requires no bespoke architecture.

In order to add scalability to the system, the dataset must be segmented prior to CF. Performing user-based collaborative filtering on a dataset of one million users (playlists) would lead to the most accurate results, however this would not be expedient, as one trillion total pairwise comparisons would need to be computed. A brute-force nearest neighbour search is of a complexity that is at least $\mathcal{O}(n^2)$. By first segmenting the dataset into pots of much smaller sizes and then performing collaborative filtering within each of these pots, the system will run much more quickly, and these processes are also then very easily parallelisable, providing even lower latency. The drawback to this segmentation is a less accurate collaborative filtering; highly similar neighbours are searched for among only a subset of the total data, meaning the closest neighbours may not be matched against at all. So, to preserve the accuracy of the CF approach, while retaining the scalability benefits of a reduced neighbourhood size, the partitioning should

be performed intelligently to ensure that similar playlists naturally find themselves in the same pots as one another.

At a granular level, the playlists contain highly varying numbers of tracks, and any potential feature vector would be of extremely high dimensionality and sparsity. Any clustering algorithm such as *k-means* clustering would encounter the same problems as a large scale CF, so simpler representations of the playlists are sought instead. Topic models, playlist names, and externally scraped metadata can support the development of such representations. In summary, this hybridised, content-based clustering system should work in tandem with the collaborative filtering approach to provide a simple and scalable recommender system for automatic playlist continuation.

3.1 External Metadata Acquisition

The MPD is a collection of 1,000 JSON files, totalling approximately 32GB. Each JSON file represents a 'slice' of 1,000 playlists. The main features of each playlist are the title and description, which are both text strings, and then the playlist itself. This part of the data notes the names of the tracks within it, the corresponding artists and albums, and all of their unique identifiers. Not all playlists (only approx. 10%) use the description field, and tracks can appear more than once in a playlist. Each playlist also has its own unique identifier, since the playlist name needs not to be unique. All of the playlists contain between five and 250 tracks.

There is a general lack of metadata, in that there are no content-based 'tags', nor is there raw audio data, like as has been used in other analyses. Upon inspection, most of the description fields that have actually been filled out by the users who curate the playlists contain only nondescript garbage which would require very advanced NLP techniques to reliably extract any information. Some of the playlist titles are similarly meaningless, though not all.

In order to extend the possibility of a content-based approach, further data was sought on Wikipedia. Wikipedia is a free encyclopaedia that anybody can edit; which suggests that the data within Wikipedia may not be reliable, especially within a context that has as much natural subjectivity as the music domain does. Studies have shown that within scientific domains, Wikipedia is a remarkably good resource, though due to the sometimes hearsay nature of pop-culture, this reliability may not always hold in respect of research applications like this one (Wolchover, 2011). Nevertheless, due to the enormous size of Wikipedia and its ease of accessibility, it was selected as a potential source of further data.

Wikipedia provides regular data dumps, however due to the practical limitations of data storage on a personal laptop, it was not feasible to download any of these data dumps for offline analysis. Wikipedia structures its online data in a machine readable way, and there exists a *Wikidata* API that enables the querying of the encyclopaedia in an SQL-like way. However, it was not clear how

this API could be used to automate large volumes of queries or how it could be integrated with a Python program. Therefore, a custom web-scraping approach was developed instead using Python, designed to scrape information from the Wikipedia webpages directly.

It would not be viable to search for data at the track level; there are 2.5 million unique tracks in the MPD and for a track to have its own Wikipedia page, it would need to hold significant cultural relevance. Searching at artist level however could be more fruitful - famous and successful individuals or groups are likely to have their own pages and there are only approx. 300,000 unique artists within the MPD. Wikipedia also has a semi-structured format, and relevant artist pages will often contain a specific 'infobox' containing basic information like their age, place of birth, and crucially, the genres that the artist is most associated with. It is this genre information that is most relevant, as it can be used as proxy for the genre of a track by the artist in question within a playlist.

The program was developed and optimised over several iterations, as various opportunities to improve the program were made clear by the presence of errors that were noticed upon the inspection of the collected metadata. First of all, the set of unique artists within the MPD was obtained. For each artist, their name was searched in the Wikipedia search box and the top five results returned. The titles of each of these results were tested to see if any of a range of substrings such as 'entertainer, 'band', 'singer' etc. were present. Any result which did contain these substrings was identified as the correct search result, otherwise the top hit was used.

Using the search function in this way had several advantages. In many cases, such as when searching for 'Elvis Presley', the correct page was simply at the top, under 'Elvis Presley'. However, in cases where search term ambiguity could be an issue, such as with the artists 'Madonna', 'Queen', or 'Oasis', the substring filtering step allowed the correct artists' pages to be selected ahead of the more popular pages (which in those cases, related to religion, monarchy, and desert geography). Furthermore, Spotify has an enormous collection of very niche artists who may not be well-known enough to have their own Wikipedia page, and using the search function could sometimes return a page which while not devoted to the artist, was still relevant. For example, Toronto-based indie band 'Magic Shoppe' have no Wikipedia page themselves, but their top search result returns a Wikipedia page for their record label, 'Optical Sounds', from which accurate genre metadata for 'Magic Shoppe' could be obtained. Of course, no metadata could be found for many niche artists.

If a relevant Wikipedia page was found, the HTML was parsed and any genre data from the 'infobox' was collected. Given Wikipedia is maintained by a large user audience, there was some variation in the formatting of this infobox and the data within. This variance was handled within the Python program and a set of genre tags was obtained in a standardised format. In total, tags were obtained for approximately 10% of the artists in the entire MPD. Many of the artists for whom no tags could be obtained were the niche artists who do not feature prominently in the

dataset, and so at playlist-level, metadata tags were practically always obtained. In fact, many of the artists in the dataset were so niche, that the first 60% of the total tags obtained could be obtained using only 1% of the MPD.

Other external data sources were also identified[2], such as *AllMusic*, *Discogs, Last.fm*, *MusicBrainz*, and *WhoSampled* but they were all disregarded since they were either not complete enough or were too difficult to access.

3.2 Experimental Design

The RecSys Challenge stipulates that exactly 500 tracks must be recommended for each playlist, with no duplicates, and details three recommendation evaluation criteria. R-precision (below) is equivalent to the number of relevant tracks that are retrieved, $R_{1:|G|}$, divided by the number of known relevant tracks, $|G|$.

$$\frac{\left|G \cap R_{1:|G|}\right|}{|G|}$$

NDCG (below) evaluates the ranking ability of the system, rewarding the system when highly relevant tracks are recommended earlier, and vice versa.

$$DCG = rel_1 + \sum_{i=2}^{|R|} \frac{rel_i}{log_2(i+1)}$$

$$IDCG = 1 + \sum_{i=2}^{|G \cap R|} \frac{1}{log_2(i+1)}$$

$$NDCG = \frac{DCG}{IDCG}$$

A third criterion is also specified; recommended song clicks. The recommender system currently used by Spotify for this application shows a selection of ten songs at a time in a list to the user as they are building their playlist. The list is refreshable in case the user does not want to choose any of the offered recommendations, and so this evaluation metric rewards and penalises systems based on how many refresh clicks are required until a relevant track is encountered. While Spotify's existing algorithm has not been shared, it is reasonable to assume that Spotify rank their recommendations best-first, and so this evaluation metric can also be coded up in the

[2] https://en.wikipedia.org/wiki/List_of_online_music_databases

analysis, rewarding in line with where the first relevant track is found; in the first ten results, or the second ten, and so on. If no relevant tracks are recommended, then 51 is used as the default score.

Therefore, the system can be evaluated according to these three metrics. Each of the playlists in the MDP can be split, with 80% of the tracks in each playlist being retained for system input, and the set of 500 recommendations being used to compute the evaluation scores in respect of the 'ground truth' 20% sample of tracks that were held back. This 80:20 split assumes that the user curating the playlist has at least made some kind of a start – which isn't unreasonable considering that when they decided to create the playlist, they presumably had at least a few songs in mind and would not be starting from a total cold-start.

3.3 Algorithms and Parameter Tuning

Two main algorithms drive the recommender system; a partitioning algorithm and a recommendation algorithm. The partitioning algorithm is parameterised by a maximum and minimum cluster size, and takes the obtained representations for each playlist as input. A maximum cluster size parameter prevents the neighbourhood from becoming too large, ensuring system scalability. The minimum size parameter prevents clusters from becoming too niche and so ensures that neighbourhoods remains large enough to permit effective collaborative filtering. The playlist representations themselves are short vectors of content tags, which should reflect the content and dominant themes of the playlist.

3.3.1 Data Partitioning

A recursive algorithm was developed to partition the clusters of playlists into appropriately sized partitions. On the first pass, the algorithm would partition the entire sample according to the dominant genres in each playlist. All predominantly 'pop' playlists would form one cluster, all predominantly 'rock' playlists would form another, and so on. In cases where the resultant clusters were still too large, they would be partitioned again based on the second-most dominant genre, and so on, forming clusters such as 'pop: r&b', 'pop: edm', or 'rock: folk rock'.

This means that playlists will be binned together on the basis that their first (and most important) representation elements match, and that therefore the playlists themselves will be similar. If any of the resultant partitions are too big, then the playlists within them are re-partitioned again based on the next most important characteristic. If all the characteristics in the representations are exhausted, the remaining playlists are partitioned by their index position.

```
i = 0
partition ALL_DATA by REPRESENTATIONS[i]
do until all partitions <= MAX SIZE:
  i + 1
  for each PARTITION:
    if PARTITIONED_DATA > MAX SIZE:
      partition PARTITIONED_DATA by REPRESENTATIONS[i]
      if REPRESENTATIONS[i] is out of range:
        partition PARTITIONED_DATA into sets of MAX SIZE according to index position
```

When different partitions are recursively split, similar bins will form. For example, one resultant partition could be described as A: B, and another described as C: B. Both partitions feature playlists with important themes of B, though with different hierarchies before that. Therefore, any resultant partitions that are smaller than the minimum size parameter can be re-aggregated together and fed back into the algorithm, with i = 1 instead of zero, so that the partitioning can be re-attempted to produce partitions of the right size. Here, the playlists from the two somewhat similar partitions would now be matched together.

2.3.2 Playlist Representation

The playlist representations themselves are composed of four elements, with the tags themselves drawn from the scraped Wikipedia data, topic model assignments and the playlist names. To obtain tags at the playlist level from the Wikipedia data, a playlist is first translated from a list of tracks to a list of the artists who feature within it. The genre tags that are associated with those artists can then be substituted in and counted over the entire playlist to infer the dominant genres.

Similarly, LDA can be used to develop a topic model from which the dominant topics within playlists could be obtained. In order to optimise the topic-playlist assignments, a large number of LDA models with varying parameters (from 3-topic models to 50-topic models) were tested and evaluated against the coherence measures described in Mimno et al. (2011) and Newman et al. (2010). The representations obtained from these models were also used to obtain recommendations for a 10% sub-sample of the MPD, and on the evaluation of these recommendations, the optimal model was found to be a 7-topic model. However, given the smaller size of the sub-sample, an 8-topic LDA model was selected instead for use in the larger analysis, since there was not an unreasonable likelihood that further topics would be present within the much larger data sample. Analytics showed that the optimal splitting order for the elements within the playlist representations was as follows:

```
if TITLE in 15_MOST_COMMON_PLAYLIST_TITLES:
   REPRESENTATION = [TITLE, WIKIPEDIA_1, WIKIPEDIA_2, TOPIC_MODEL1]
else:
   REPRESENTATION = [WIKIPEDIA_1, WIKIPEDIA_2, TOPIC_MODEL1, TOPIC_MODEL2]
```

Only four tags were used in each representation, to prevent the representations becoming too nuanced and potentially detached from what the playlist was really about. This conceptually not too dissimilar to model regularisation to prevent the fitting of noise. A series of A/B tests determined that the optimal maximum and minimum sizes for the partitions were 10,200 and 4,000 playlists respectively.

3.3.3 Recommendation

Once partitions are obtained, user-based collaborative filtering takes place. Each playlist is compared with all others until a neighbourhood of similar playlists is obtained, using the Jaccard Index (below) as a similarity metric. This metric is appropriate since each playlist consists of a set of tracks, and playlists with large numbers of common songs are clearly similar.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

The recommendation algorithm recommends all of the tracks from the nearest neighbouring playlist, before moving on and recommending the tracks of the next-nearest neighbour, until 500 recommendations are obtained. Tracks that were in the original playlist are not re-recommended, and in the event that a playlist has insufficient overlap with the other playlists to get 500 nearest-neighbour based recommendations, then most popular tracks within the partition are recommended as a fall-back.

```
for each PLAYLIST:
   calculate NEIGHBOURS
   i = 0
   while RECS < 500:
      RECS = de-duplicate(RECS + NEIGHBOURS[i]['TRACKS'])
      i + 1
      if NEIGHBOURS[i] is out of range:
         RECS = de-duplicate(RECS + MOST_POPULAR_LOCAL_TRACKS)
   RECS = RECS[:500]
```

To improve recommendation ranking, the tracks recommended from the first few nearest neighbours were shuffled. This, in theory, would reduce the sensitivity of the system, however in practice this did not improve the results, so this feature of the system was not retained.

3.3.4 Results

The system was benchmarked against a simpler partitioning strategy, which would split the data into a number of evenly sized partitions (of size: MAX SIZE) according to the index position of the playlists in the data. This indexing is random and therefore this benchmark approach makes no attempt to produce buckets containing like-for-like playlists.

Results	**R-precision**	**NDCG**	**Recommended Song Clicks**
Index-Based Partitioning	0.302	0.209	18.000
Content-Based Partitioning	0.343	0.224	15.730

The content-based approach to partitioning has outperformed the index-based approach on all three evaluation criteria, with improvements in R-precision, NDCG and Recommended Song Clicks of 13.6%, 7.2%, and 12.6%.

4.0 Discussion

4.1 Interpreting the Evaluation Results

The general quality of results is satisfactory, particularly with respect to R-precision. First and foremost, the system is functioning as designed, outputting track recommendations for users. In terms of rediscovering the ground truth, the R-precision score indicates that on average, over 34% of the original holdout set have been recommended to the user. Practically speaking, this means that in most of the sample cases, the user has received enough accurate recommendations to put together a playlist that fulfils their desired purpose and that is also long enough (in duration) to provide a realistic, continuous music experience. Furthermore, considering that R-precision is unable to measure how well the system is supporting new music discovery, it is likely that the recommendations are even better than the metric suggests. Given over a third of the original holdout set have been rediscovered, it is very reasonable to assume that some of the other recommendations being made are also relevant, and that the user behind the original playlist was simply unaware of the tracks and so they don't feature in the ground truth that is used to compute R-precision. Enabling new music discovery like this is one of Spotify's key strategies for delivering a good user experience.

Evaluating the ranking ability of the system is done using normalised discounted cumulative gain and 'Recommended Song Clicks'. The NDCG score was almost unchanged by the shuffling

approach that was trialled, wherein the ranking of recommendations made by the nearest two or three neighbours were randomly shuffled. The logic here was that by making the system less sensitive and 'averaging' the voices of several close neighbours, ranking results might be improved, but the absence of any change implies that this problem may be more difficult to solve. One limitation to solving the ranking problem revolves around understanding user preference; currently a song is either in a playlist, or not in a playlist, and preference is therefore modelled binarily and implicitly, as a 'like' or 'dislike'. If further metadata at the track level, such as the play count on a track was available, a greater depth of insight into how much a user likes a song could be obtained. Having a better understanding of what the user is most likely to enjoy is clearly beneficial when a system must rank its recommendations. It is also likely that due to the low holdout proportion (20%), versus the number of recommendations (500), that the Recommended Song Clicks score is exacerbated by the weaknesses in the system's ranking ability.

4.2 Further Considerations

The system has proved to be scalable. If no partitioning was completed, estimates suggest that providing recommendations for one million playlists using collaborative filtering would have taken around six months. The computation for this analysis on the other hand only took 48 hours, and had it been successfully parallelised, estimates indicate that results would have been available after just 30 minutes or so. Within a production environment, this system could be realised with a model-based implementation, rather than the memory-based implementation used here to decrease latency even further. The partitions themselves can also be automatically named after the content-tags from which they are derived, to provide a level of transparency and explainability to the recommendations that are made. From a commercial perspective, this not only enables the system to be audited, tested and potentially debugged more easily, but also provides a basis for user preferences to be taken into account, wherein a user might want to manually override the system and turn off recommendations from certain pots that they knew they weren't interested in.

Perhaps the most significant take-away from this analysis is how the recommender system and partitioning framework creatively use the data that is initially available. User-based collaborative filtering itself is an interesting and clever algorithm but it is certainly not new, either in the context of automatic playlist continuation or elsewhere. What are more interesting however, are the methods by which additional data is accumulated from non-trivial sources; online and via topic models. Using web-scraped data does not constitute a novel approach since other music recommender systems also implement this, though in most other applications of these techniques, the generated tags are then used to determine item-item similarity and directly fuel the recommendation algorithm, rather than for the pre-partitioning of data to permit scalable collaborative filtering as has been accomplished here. Re-formulating topic modelling approaches from their normal use within NLP for the purposes of playlist modelling makes a lot

of sense given the references to the need for psychological and mood-based profiling that are made by Song et al. (2012) and Schedl et al. (2018). Given topic modelling requires no third party data sources, and is free from the ethical issues that are associated with web scraping, it is perhaps the more commercially-friendly approach of the two.

4.3 Future Improvements

Several aspects of the system are not optimised. The web-scraping algorithm currently only looks at Wikipedia, only retrieving tags for 10% of the total artists in the dataset. As discussed above, this is largely because many of the artists are niche and almost totally unknown, and does not constitute a significant problem due to the severely unequal distribution of artists within the data. Playlists nearly always have tags. However, obtaining data on these niche artists would further optimise the truthfulness of these playlist level tags, thus leading to better representations of the key playlist themes. From a user's perspective, finding and recommending these niche artists would not only enhance the music discovery aspect of the system but make the system feel far more personalised and user-centric. There is also at least one error in the currently scraping program, since no tags are retrieved for a small subset of widely popular artists such as AC/DC and Britney Spears, despite them having Wikipedia pages in the expected format. The documentation behind some of the Python libraries is insufficient to adequately de-bug this issue and so the Wikipedia search engine itself remains a black box to some degree.

There is also the assumption that the parameters of the optimal topic model should remain the same regardless of the amount of data. This is almost certainly a naïve assumption; due to resource constraints the LDA model parameters were obtained by comparing several LDA models that were all derived from a 10% sample of the MPD. While the evaluation criteria, A/B test results and manual inspection all indicated that the models were good, featuring clearly distinct and easily identifiable topics such as Country music (perhaps unsurprisingly given the MDP was generated entirely from US-based accounts), Indie music, Hip-Hop, and Christmas music, the list of topics was by no mean exhaustive and didn't feature many of the psychological or mood-based topics that were theorised. As the data subsequently scaled up to include a further 900,000 playlists, it is assumed that more topics such as these may have emerged, though the exact number is not known. By not accounting for these additional topics, the LDA model that was ultimately used to generate the playlist representations was likely fitted with sub-optimal parameters.

The partitioning algorithm is also sub-optimal. Speed and simplicity were prioritised since the playlist representations that are taken as input are only *representations*, and it would be very easy to over-fit the data with a complex algorithm. However, in cases where some of the obtained partitions are outside the partition size criteria, they are simply aggregated together and repartitioned. This typically discounts a key component of each of the affected playlist representations, and while it not an illogical approach, the algorithm does lose potency over a

large number of passes. To overcome this, a smarter implementation might (at the cost of some additional computation) merge small partitions based on inter-cluster similarity. This would presumably improve the partitioning and also allow for increased automation, as currently some prior data analysis is necessary to determine what the optimal order of the elements in the playlist representations should be so as avoid a large number of partitioning iterations.

A next step for this project, aside from further A/B testing and parameter tuning, along with addressing the points noted above, should focus on exploring a range of recommendation approaches. Currently UBCF is used but SVD, item-based CF and various other simple, reliable methods are available. The framework for this system is quite modular, (obtaining metadata tags, generating playlist representations, partitioning the dataset, recommending within the resultant partitions...), and so where new methods can be easily tested and compared, they should be.

5.0 Conclusion

5.1 Conclusion

Despite its large size, the Million Playlist Dataset is a highly homogenous data source with almost no data beyond basic playlist content, and so modelling the user or labelling the purpose of the playlist in order to generate new track recommendations is a major challenge. Basic approaches that would use this level of data are slow, and the businesses looking for solutions in the automatic playlist continuation space need systems that are fast and scalable so as to best service their customers. The recommender system detailed in this report overcomes these challenges by creatively sourcing metadata tags via web scraping and topic modelling so that the drawbacks of simple mechanism can be addressed.

Scalability concerns are tackled by modelling the playlists with their newly-sourced metadata, before rapidly partitioning the playlists into according to the similarities between these modelled representations. Collaborative filtering can then be performed within these partitions, both at a greatly increased speed and yet with a comparable level of performance to a non-partitioned approach, since many of the playlists that are truly similar will find themselves within the same partition. This metadata-based partitioning approach outperforms a random-partitioning approach.

However, the system is not fully optimised and it is likely many of the system parameters could be better tuned, from the partition size to the choice of recommendation algorithm, and refining these parameters should form the body of any immediate future work. What is perhaps the most interesting however, is how this system and framework will compare to more advanced, complex architectures with respect to the trade-off between recommendation accuracy, and the ease of

implementation, maintenance and transparency considerations of which commercial organisations are becoming increasingly cognisant as they embrace big data.

References

Amatriain, X. and Basilico, J. (2012) *Netflix Recommendations: Beyond the 5 Stars*. [online] 20th of June, 2012
Available at: https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-2-d9b96aa399f5
[Accessed: 11th of April, 2018]

Bernhardsson, E. (2013) *Collaborative Filtering at Spotify*. [online PowerPoint]
Available at: https://www.slideshare.net/erikbern/collaborative-filtering-at-spotify-16182818/7-Aggregate_dataThrow_away_temporal_information
[Accessed: 27th of April, 2018]

Blei, D., Ng. A. and Jordan, M. (2003) Latent Dirichlet Allocation. *Journal of Machine Learning Research*, Volume 3, pp. 993-1022

Chuang, J., Manning. C. and Heer, J. (2012) Termite: Visualization Techniques for Assessing Textual Topic Models. *Proceedings of the 2012 Working Conference on Advanced Visual Interfaces.* Capri, Italy. pp. 74-77

Ciocca, S. (2017) *How Does Spotify Know You So Well?* [online] 10th of October, 2017
Available at: https://medium.com/s/story/spotifys-discover-weekly-how-machine-learning-finds-your-new-music-19a41ab76efe
[Accessed: 11th of April, 2018]

Crook, J. (2015) *Tinder Introduces a New Matching Algorithm*. [online] 11th of November, 2015
Available at: https://techcrunch.com/2015/11/11/tinder-matching-algorithm/
[Accessed: 11th of April, 2018]

Deerwester, S., Dumais, G. W., Furnas, S. T., Landauer, T. K. and Harshman, R. (1990) Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, Volume 41(6), pp. 391-407

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977) Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, Volume 39(1), pp. 1-38

Dodda, R. (2018) *Personalisation Tech: The Key to the Future of Marketing Automation*. [online] 11th of April, 2018
Available at: https://inc42.com/resources/personalisation-tech-the-key-to-the-future-of-marketing-automation/
[Accessed: 11th of April, 2018]

Geman, S. and Geman, D. (1984) Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 6(6), pp. 721-741

Hofmann, T. (2001) Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Machine Learning*, Volume 42(1), pp. 177-196

Hu. Y., Koren. Y and Volinsky, C. (2008) Collaborative Filtering for Implicit Feedback Datasets. *Proceedings of the 2008 IEEE International Conference on Data Mining*. Washington, DC, USA. pp. 263-272

Kerwin, W. (2016) *Implicit Recommender Systems: Biased Matrix Factorization*. [online] 11th of January, 2016
Available at: http://activisiongamescience.github.io/2016/01/11/Implicit-Recommender-Systems-Biased-Matrix-Factorization/
[Accessed: 25th of April, 2018]

Luckerson, V. (2016) *The Number of Movies on Netflix is Dropping Fast*. [online] 25th of March, 2016
Available at: http://time.com/4272360/the-number-of-movies-on-netflix-is-dropping-fast/
[Accessed: 26th of April, 2018]

Marr, B. (2018) *How McDonald's is Getting Ready for the 4th Industrial Revolution using AI, Big Data and Robotics.* [online] 4th of April, 2018
Available at: https://www.forbes.com/sites/bernardmarr/2018/04/04/how-mcdonalds-is-getting-ready-for-the-4th-industrial-revolution-using-ai-big-data-and-robotics/#5bf31f1c3d33
[Accessed: 5th of April, 2018]

McMillan, R. (2014) *Airbnb is Quietly Building the Smartest Travel Agent of All Time*. [online] 16th of July, 2014
Available at: https://www.wired.com/2014/07/airbnb-recommendations/
[Accessed: 11th of April, 2018]

Mimno D., Wallach, H., Talley, E., Leenders, M and McCallum, A. (2011) Optimizing Semantic Coherence in Topic Models. *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, UK. pp. 262-272

Motoyama, S. (2018) *Meet the 'Lady Gaga of Mathematics' Helming France's AI Task Force*. [online] 28th of March, 2018
Available at: https://www.theverge.com/2018/3/28/17170104/cedric-villani-french-mathematician-ai-report-interview
[Accessed: 4th of April, 2018]

Newman, D., Lau, J., H., Grieser, K. and Baldwin, T. (2010) Automatic Evaluation of Topic Coherence. *Proceedings of the 9th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Los Angeles, CA, USA. pp. 100-108

Oneaţă, D. (2011) *Probabilistic Latent Semantic Analysis* [online]
Available at: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV1011/oneata.pdf
[Accessed: 28th of April, 2018]

Pasnick, A. (2015) *The Magic that makes Spotify's Discover Weekly Playlists So Damn Good*. [online] 21st of December, 2015
Available at: https://qz.com/571007/the-magic-that-makes-spotifys-discover-weekly-playlists-so-damn-good/
[Accessed: 27th of April, 2018]

Plaugic, L. (2018) *Spotify now has 70 million Subscribers*. [online] 4th of January, 2018
Available at: https://www.theverge.com/2018/1/4/16850742/spotify-subscriber-count-70-million-users
[Accessed: 12th of April, 2018]

Popper, B. (2016) *Spotify's Discover Weekly Reaches 40 Million Users and 5 Billion Tracks Streamed*. [online] 25th of May, 2016
Available at: https://www.theverge.com/2016/5/25/11765472/spotifys-discover-weekly-40-million-users-5-billion-streams
[Accessed: 27th of April, 2018]

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. (1994) GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*. Chapel Hill, NC, USA. pp. 175-186

Said, A. (2016) A Short History of the RecSys Challenge. *AI Magazine*, Volume 37(4), pp. 102-104

Said, A. (2017) *Data Mining: Recommender Systems*. University of Skövde, 21st of November, 2017

Sarwar, B., Karypis, G., Konstan, J. and Riedl, J. (2000) Application of Dimensionality Reduction in Recommender System - A Case Study. *Proceedings of the ACM WEBKDD 2000 Web Mining for eCommerce Workshop*. Boston, MA, USA. pp. 421-428

Sarwar, B., Karypis, G., Konstan, J. and Riedl, J. (2001) Item-based Collaborative Filtering Recommendation Algorithms. *Proceedings of the 10th International Conference of World Wide Web*. Hong Kong, Hong Kong. pp. 285-295

Schedl, M., Zamani, H., Chen, C.W., Deldjoo, Y. and Elahi, M. (2018) Current Challenges and Visions in Music Recommender Systems. *International Journal of Multimedia Information Retrieval*

Song. Y., Dixon. S, and Pearce, M. (2012) A Survey of Music Recommendation Systems and Future Perspectives. *Proceedings of the 9th International Symposium on Computer Music Modelling*. London, United Kingdom. pp. 395-410

Wolchover, N. (2011) *How Accurate is Wikipedia?* [online] 24th of January, 2011 Available at: https://www.livescience.com/32950-how-accurate-is-wikipedia.html [Accessed: 10th of May, 2018]