ENHANCING CONTENT DISCOVERY: AN EVALUATION AND CLASSIFICATION PERSPECTIVE

HUMBERTO JESÚS CORONA PAMPÍN
UCD student number: 11213109

This thesis submitted to University College Dublin in fulfilment of the requirements for the degree of Research Masters

School of Computer Science and Informatics
University College Dublin

Head of School: Prof. Pádraig Cunningham

Supervisor: Dr. Michael P. O'Mahony

September 2014

Without music, life would be a mistake.

— Friedrich Nietzsche.

A mis padres,
por el amor y apoyo incondicional que he recibido todos estos años.

ABSTRACT

In an age where information is one click away, users find themselves overloaded with choice. Thus, there is a need for intelligent systems to help users make decisions by organising content in smarts ways. Moreover, the business model for entertainment content has dramatically changed in the last decade. Nowadays users pay a monthly subscription to services that provide all-you-can-eat access to large catalogs of movies or songs. Thus, in this scenario content discovery is critical both for content producers and consumers. For this reason, we analyse the problem of content discovery from two different perspectives. First, we study recommender systems from a general perspective, focusing on the evaluation of collaborative filtering algorithms. Then, we focus on the problem of music classification for content discovery, studying different ways to represent and classify songs.

Recommender systems attempt to solve the information overload problem by presenting the user with those items which are relevant for her. However, there is evidence showing that some algorithms may contribute to the filter bubble problem, making it harder for the user to discover new items. In this thesis we perform a comprehensive offline evaluation of three collaborative filtering algorithms; *User K-Nearest Neighbour*, *Item K-Nearest Neighbour*, and *Weighted Matrix Factorisation*. We present a performance analysis studied in terms of different recommender systems properties, such as accuracy, popularity, diversity or coverage. The results obtained show that the performance of an algorithm can not be completely defined by measuring one facet, and in order to enhance content discovery, a number of metrics have to be considered together, studying their relations and tradeoffs.

Music is a domain where content discovery has become critical. Here, automatic classification is used to help users navigate through large music collections by enabling automatic playlist creation or personalised music recommendations. In this thesis, we address the problem of music mood and genre classification by exploiting information extracted from lyrics alone. In contrast to previous work, in which experiments were carried out using different datasets and evaluation methodologies, here we use a large freely-available dataset to compare the performance of classifiers trained on a number of different feature sets. The experiments show that the vector space model approach outperforms the low-dimensional feature representation based on arousal, valence and dominace features in this particular domain.

DECLARATION

I herby certify that the submitted work is my own work, was completed while registered as a candidate for the degree stated on the Title Page, and I have not obtained a degree elsewhere on the basis of the research presented in this submitted work.

Humberto Jesús Corona Pampín,
September 2, 2014

PUBLICATIONS

Some ideas and figures have appeared previously in the following thesis publications:

Humberto Jesús Corona Pampín , Houssem Jerbi, and Michael P. O'Mahony. "Evaluating the Relative Performance of Neighbourhood-Based Recommender Systems." In Proceedings of the 3rd Spanish Conference on Information Retrieval, pages 25-36, 2014.

Humberto Jesús Corona Pampín and Michael P. O'Mahony. "A Mood-based Genre Classification of Television Content". In proceedings of the first ACM Workshop on Recommendation Systems for Television and Online Video (to be published in October 2014).

This work is supported by Science Foundation Ireland through the CLARITY Centre for Sensor Web Technologies under grant number 07/CE/I1147.

ACKNOWLEDGEMENTS

First of all I would like to thank Dr. Michael P. O'Mahony for his guidance and support during this two years. I am very lucky and proud to have you as my supervisor. Thank you for teaching me so much, for always finding the time to meet, discuss, correct and provide feedback to my work. Thank you for listening and answering my questions, but specially for making many more questions. I would also like to thank Dr. Rachael Rafter and Dr. Houssem Jerbi, for providing different perspectives to my work in our meetings.

I gratefully acknowledge the support of LyricFind, who kindly provided the access to their lyrics API used in part of this work. Also, thanks to Peter Olwell for your support in the process.

Thanks to my colleagues and friends in CLARITY, specially John, Kevin, Owen, Ruihai, Sandra and Steven. Thank you for making me feel as part of the group and at home since day one. Thanks for all your good advice and the the great times we shared.

Hace casi diez años conocí a las cuatro mejores amigas alguien pueda pedir. Ana, Antía, Majo y Sara; muchísimas gracias por todos los buenísimos momentos que he vivido con vosotras, FAFS, vacaciones, visitas y demás aventuras pasadas y futuras!. Ana, muchísimas gracias por ser un gran apoyo académico y personal durante estos dos años, por escuchar, entender y compartir mis research y non-research related issues 24/7. Por otro lado, declaro la pirámide, cosa del pasado.

Gracias Martiño por entenderme, apoyarme y quererme. Muchísimas gracias por tus incontables visitas. Has sido un apoyo, fuente de felicidad e inspiración inmensa durante estos tres años. He disfrutado y aprendido muchísimo en cada momento que compartí contigo y he sido muy feliz compartiendo mi vida contigo. Te quiero muchísimo.

Gracias a mis padres, por apoyarme en esta (y todas) etapa de mi vida, por preocuparos e interesaros en mi trabajo y sobretodo gracias por ayudarme a ser quien soy ahora, os quiero muchísimo!. Daniela y María José, gracias por ayudarme a desconectar, y por compartir vuestras inquietudes conmigo. Vuestra dedicación y actitud hacia el trabajo y la vida son una gran fuente de inspiración para mi. Gracias a mi tía María Jesús por escucharme y contarme muchísimas cosas durante horas al teléfono, haciéndome sentir como si estuviese en casa. Gracias a mi Abuelo Jesús por siempre darme muy buenos y sabios consejos y gracias a mis abuelas por siempre ser tan cariñosas.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The Internet can be defined as *a global system of interconnected computer networks*, and since the appearance of the World Wide Web (WWW) in 1989, it has been constantly changing and growing, becoming a vital part of our lives in contemporary society [120]. Before the ubiquitous presence of the Internet and the WWW, people had access to books only in libraries, films in cinemas or rental shops, and records were sold in physical format. Thus, access was limited and difficult. As physical storage was needed, it was impossible to find every book (film or record) in the same physical space at once. Nowadays the access problem is solved, and people have access to these and many other products and services through smart devices connected to the Internet. However, the solution of the information access problem led to an information overload problem. For example, in the 80's a common user had access to hundreds of songs, while nowadays streaming services like Spotify[1] provide access to dozens of millions of songs. Moreover, traditional ways of content discovery such as the so-called *word of mouth* on physical social networks (neighbours, family or friends) have been translated into their online counterparts (Facebook[2], Twitter[3], etc.), changing the way content is discovered and consumed.

As access to media becomes easier, and the amount of user generated content builds up, there is an increasing difficulty when dealing with the vast amount of information which is available. Finding a new movie to watch, discovering new bands to listen to, or deciding whom to follow in an online social network is becoming increasingly difficult for the user. In fact, we often find ourselves overloaded with information and choice. Therefore, there is a need for a new class of intelligent systems that help us to navigate through all this information, by automatically identifying relevant content and filtering that which is irrelevant. Recommender systems [1, 146, 96, 145, 154] perform this task by learning the products and services we like, the news articles we find interesting, or the music we listen to. All this knowledge is then applied to personalise the user experience in different services; from advertising to clothes recommender systems, from movie recommender systems to music discovery. Thus, much of the

1 http://www.spotify.com
2 http://facebook.com
3 http://twitter.com

content that users access on the internet can be filtered according to their needs in a personalised fashion.

Recommender systems are a class of information filtering systems which aim to present information in a personalised fashion, aiding the user to browse large collection of items, i.e., scientific articles [169] or movies [175, 53] . These type of systems have been widely exploited in the last decade, and are often found on internet sites dealing with large amounts of information such as Netflix[4] (video streaming platform) or Spotify (music streaming platform). Moreover, the concept of recommender system is very broad. For example, they have been used in collaborative search using reputation systems [110], to get recommendations of people to follow in social networks [57], to classify news feeds [133, 46] or to help users navigate through large collections of products reviews in an e-commerce scenario [40], among many other applications.

Figure 1 shows the Netflix web interface, which is completely personalised. For example, the movies in the *top picks* row are shown based on the user's past behaviour, while the second row shows similar items to a previously watched (and liked) television show.

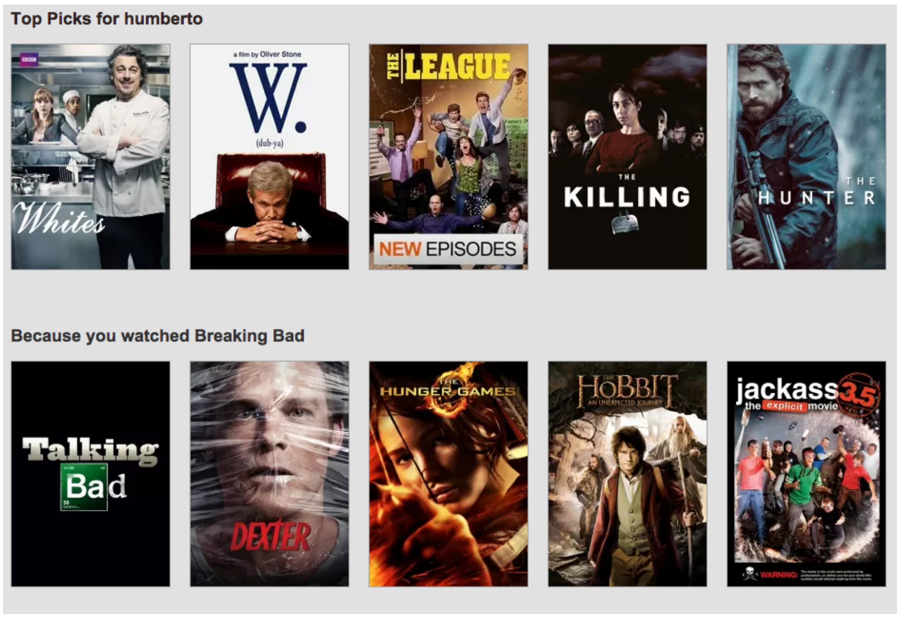


Figure 1: The Netflix user interface shows different recommendations streams.

Figure 2 shows the non-personalised recommendations generated by the popular e-commerce site Amazon[5], based on the "people who bought this also bought that" model of recommendation.

4 http://www.netflix.com
5 http://www.amazon.com

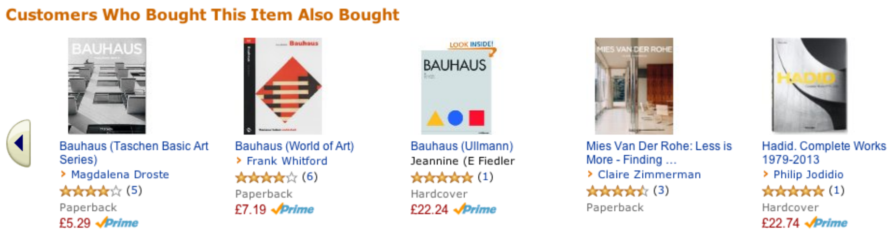


Figure 2: Amazon present users with non-personalised recommendations of similar items calculated by shopping co-ocurrence.

Recommender systems can be classified according to the information they use when generating recommendations. *Collaborative recommender systems (CF)* [153, 86, 7] use the wisdom of the crowd identify trends on how users interact with the system. They are able to find like-minded users and use that information to make recommendations, relying only on preference data from the users in the system. These type of algorithms are widely present in commercial systems such as the e-commerce platform Amazon [99], which uses item-to-item collaborative recommendations. Moreover, they have proven to be accurate, specially thanks to advances developed in the context of the Netflix Prize [12]. *Content-based Filtering (CBF) recommender systems* [114, 100, 130] utilise user feedback provided by users to generate recommendations of similar items to those users have liked in the past, by analysing the content metadata. These type of systems are particularly useful in cases where the access to other user's preferences is limited but rich metadata is available, for example in the case of television recommendations [16]. *Hybrid systems* [24, 18] usually combine content-based with collaborative filtering approaches, with the aim of obtaining a better performance than with a single approach, while reducing the drawbacks of each approach. The most commonly used hybrid approaches are *weighted hybrid recommenders*, *switching hybrid recommenders* and *mixed hybrid recommenders*, as described in [24].

An interesting and widely studied area of recommender systems is evaluation [60, 158]. Traditionally, recommender systems evaluation focused on measuring accuracy. However, in recent times the focus have extended to measure recommender systems properties beyond accuracy [85, 111] such as diversity, popularity, novelty or serendipity. Performing an evaluation that considers all these facets is necessary to better understand all the factors that affect the performance of this type of systems. Moreover, there is a growing interest in studying the evaluation of recommender systems considering not only *offline evaluation* experiment, but also *user studies* and *A/B testing* [158].

A particularly interesting field where recommender systems have been widely applied to enable content discovery is music. This industry has dramatically changed in the last decades and has become an example of a scenario where information overload is a critical problem. The formats and devices in which music is played have changed several times; from vinyl, to cassettes, CDs, mp3 players and finally to smartphones. Products like the iPod[6] (first released in 2001) or services like Napster[7] (a P2P music sharing service first released in 2000) challenged the traditional music distribution channels such as record stores and radios. More recently, services like Last.FM[8] (which allows users to track and tag all the music listened to), Spotify or rdio[9] (which give unlimited access to a catalog of dozens of millions of songs) are changing the way music is consumed and distributed again. Traditionally, the music model was based on the purchase of either albums or songs, while online services provide a (sometimes free) streaming subscription to their musical libraries. In this context, music discovery, classification and recommendation has become a more important part of each of these products, driving research both in industry and academia [96, 94, 1, 29].

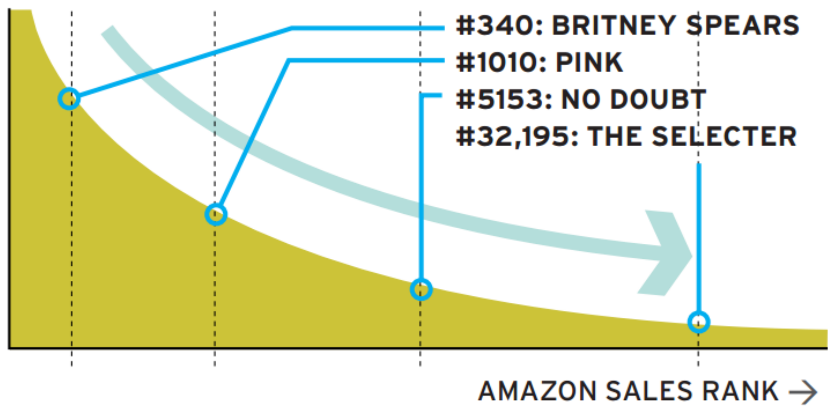


Figure 3: Sales per artist on the e-commerce website Amazon.com, from the 2010 Wired Magazine article *The Long Tail*, by Chris Anderson [3].

Even when effort has been put into enabling music discovery, overall listening behaviour has, however, not changed, and still follows a long tail distribution. Thus, a very reduced amount of artists are extremely popular, while there is a long list of artists and songs that rarely get sold or listened to. An example is pictured in Figure 3, which shows total sales per artist on the e-commerce site Amazon (vertical axis) sorted by popularity (horizontal axis). Here, the total number of records sold by artists like *Britney Spears* are several orders of magnitude bigger than other less popular artists such as *The Selecter*. Thus, the long tail of artists needs to be exploited when

6 https://www.apple.com/ipod/
7 http://ie.napster.com/start
8 http://www.last.fm/
9 http://www.rdio.com/

generating recommendations to increase sales, artist visibility, and provide novel recommendations to the consumer.

The problem of music discovery has been widely studied in recent times and several reviews of the field have been published [163, 29, 94]. One of the approaches to solve the information overload in this field is by automatically classifying music, which allows sorting large music collections or automatically generating playlist and recommendations. This type of approach has been successfully used to generate music recommendations based on mood [56] and genre [64]. For example, Figure 4 shows the Spotify desktop application interface, where playlists are classified according to their moods and genres.

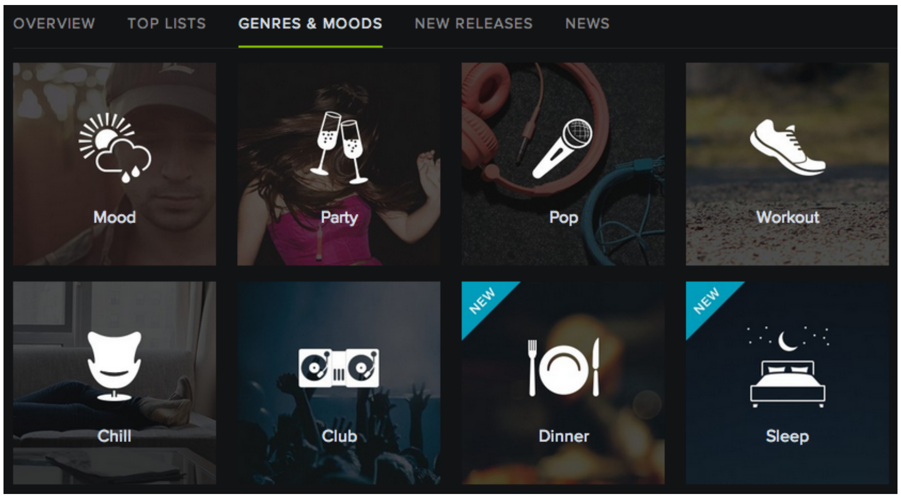


Figure 4: The Spotify interface, showing different mood and genre playlists.

1.1 MOTIVATIONS

Here, we present a description of the problems within content discovery that motivate and define the lines of work addressed in this thesis. The first part deals with general issues relating to recommender systems, focusing on recommender systems evaluation, studying the performance of collaborative filtering algorithms using accuracy and beyond accuracy metrics. In the second part of this thesis we focus on a specific application domain (music), and address a key step involved in music discovery, mood and genre classification, which, as we have previously mentioned, can be used as an input to music recommendations.

The evaluation of recommender systems has been widely studied and several reviews on the topic have been published [60, 158], describing the most common evaluation methodologies and metrics. Most of those metrics, such as precision and recall [23] or discounted cumulative gain [143] are adapted from the field of information retrieval;

a field which is closely related to recommender systems. However, recent work [85, 111] highlights that studying accuracy metrics alone is not sufficient to evaluate the performance of a recommender algorithm. Moreover, evaluation is one of the key challenges in recommender systems.

Thus, in recent times, focus has also been put on the analysis of recommender systems performance measured in terms that go beyond accuracy. Metrics like diversity, coverage or novelty have been studied [49, 172]. However, these metrics are often discussed in isolation, and the relation and tradeoffs between these and accuracy metrics on different algorithms has not been widely studied. Moreover, some of these metrics (such as serendipity or novelty) are difficult to evaluate in an offline scenario, since there are not freely-available datasets on which these metrics can be quantified.

New recommender systems arise together with new information needs, platforms and scenarios. However, to build better recommender systems, effort has to be put in studying the evaluation of recommender systems from from a wider perspective that captures the overall quality of the system. Thus, in this work we will present a comprehensive evaluation of collaborative filtering algorithms. We will focus on understanding and explaining the relations and tradeoffs between the different evaluated metrics, using standard evaluation procedures and well known publicly available datasets.

In the above, we have presented the motivations for the first part of this thesis, which studies recommender systems and content discovery from a general perspective. We now describe the main motivations for the second part, which focuses on the specific application domain of music classification and discovery.

Music recommender systems often rely on classification tools to generate personalised recommendations or create automatic playlists [56, 64], with *genre* [105, 107] and *mood* [116, 76] being two of the most popular ways to classify music. Most of the proposed sentiment or mood ontologies rely on models developed by research in the psychology field [148, 168], which have been later adapted to the music classification task [43]. However, moods in music are complex to infer and people perceive them differently [165]. Genres are also widely used for music classification in several scenarios [125, 4]. For example, genres such as *pop, metal, rock* or *electronic* are the predominant genres into which users sort music, record stores classify their items and people catalog their music taste. However, music genres can also be complex, and new derivative genres are constantly being created.

Music mood and genre classification has been often explored from an audio-based representation perspective [171, 62]. However, lyrics have proven to be an important part of mood perception, as it provides information which is processed independently from audio by the brain [15]. However, there are still many open questions in this area of research. For example, there is still no consensus over the mood and genre granularity model to be used or how to derive mood and genres from social tags, and different authors take different approaches [180, 61].

Moreover, it is difficult to compare previous findings [42, 180] since previous works have reached contradictory conclusions based on experiments carried out on different datasets using different evaluation methodologies. As the audio, lyrics and even artworks for songs and albums are copyrighted, previous work has mostly relied on personal collections to build datasets for music classification, retrieval and recommendation. This presents a problem for repeatability of experiments and comparability of results. Even when there has been a strong effort by the community to provide a standard for evaluation (the Music Information Retrieval Evaluation Exchange, MIREX[10]) and large freely-available datasets (the Million Song Dataset [14]), these have not been widely used.

In our work we present an evaluation of two different approaches to music mood and genre classification for discovery. We evaluate the results using standard tools, such as the machine learning library Weka [55], and a large free-available dataset, the Million Song Dataset, with the aim of better reproducibility and comparability with future work. Moreover, when needed, we obtain the full lyrics for songs through the LyricFind API[11], which kindly provides access to their lyrics catalog for research proposes.

1.2 CONTRIBUTIONS

We have presented the main problems that motivate this thesis: the need for a comprehensive evaluation of recommender systems algorithms, and the use of automatic classification for music content discovery and recommendation. Here, we present the main contributions of this thesis in each of the areas considered.

1.2.1 *A Comprehensive Evaluation of Collaborative Filtering Recommender Systems Algorithms*

In the first part of this thesis, we analyse the general problem of recommendation algorithms performance, studying the evaluation

10 http://www.music-ir.org/
11 http://lyricfind.com

of different widely used collaborative filtering recommender system algorithms.

- In Chapter 2, we present an extensive study on the current trends in recommender system evaluation. We focus on both accuracy (top-N accuracy metrics and error-based metrics) and metrics beyond accuracy (diversity, novelty, serendipity, reach, coverage, uniqueness and popularity bias), presenting a discussion on the relevant related work.

- We perform a comprehensive evaluation of three state-of-the-art collaborative filtering algorithms; *user-based collaborative filtering(UKNN), item-based collaborative filtering (IKNN)* and *weighted Matrix Factorisation (WMF)* in two novel experimental settings. The results are presented using several datasets and the previously studied metrics. We analyse the relationships and tradeoffs between those metrics, with the aim to better understand the performance of the evaluated algorithms. Moreover, we explain why certain algorithms might perform better according to certain metrics. For example, are the recommendations generated by a particular algorithm more accurate because it presents users with the all the popular items?

- We study how the algorithm parameters, in particular the number of neighbours in the *UKNN* algorithm, affect the performance of the recommendations. We perform a comparative evaluation of the recommendations generated by the *UKNN* and *IKNN* algorithms using two different datasets from the movie domain. We measure the performance in terms of precision, diversity and popularity and present the results analysing the relationships and tradeoffs between the different metrics as the algorithm parameters change.

1.2.2 *Music Mood and Genre Classification*

In the second part of this thesis we analyse the particular scenario of music discovery, studying the problem of music mood and genre classification.

- We present an extensive discussion on the current trends for music mood and genre classification, focusing on the approaches that use lyrics or combinations of audio and lyrics features. From the related work it is clear that works presenting similar approaches lead to contradictory results because, despite community efforts, music mood and genre classification has not been widely evaluated using large freely-available datasets.

- We present an approach for music mood and genre classification which uses a supervised single label classifier and features extracted from lyrics. Our approach is based on features inferred from the ANEW dataset [19] mapped to Russell's model of affect [148]. In contrast with pervious work, we evaluate this approach using a large freely-available dataset of one million contemporary popular tracks, and we obtain the lyrics using legal sources through the LyricFind API. Moreover, we compare the results with the widely-used vector space model approach.
- We experiment with different term-weighting approaches in the vector space model, using the evaluation methodology previously described, and comparing the results with several state-of-the-art techniques. In addition, we propose a modification to the *inverse class frequency* term-weighting approach [180], which measures the number of classes in which a particular term appears. This novel approach (specifically developed for the classification task), known as *normalised inverse class frequency*, takes in account the proportion of songs within each class where a particular term appears.

1.3 THESIS OVERVIEW

The remainder of this thesis is organised in two parts. In the first part we study the general problem of recommender systems evaluation. In the second part we focus on the particular problem of music discovery, from the perspective of music mood and genre classification.

In the first part, Chapter 2 provides an overview of recommender systems, focusing on collaborative filtering algorithms and the different techniques and metrics used for evaluating them. Chapter 3 presents two experiments on collaborative filtering algorithms evaluation. The first is a comprehensive analysis of collaborative filtering algorithms in different scenarios, where the performance of the algorithms is measured in terms of top-N accuracy metrics, and metrics beyond accuracy, such as popularity, diversity, reach or coverage. The second experiment presents a comparative analysis of the two main neighbourhood-based collaborative filtering algorithms. Here, we study how the number of neighbours in the *UKNN* algorithm affects its performance in terms of precision, diversity and popularity.

In the second part of this thesis, Chapter 4 deals with the problem of classification. It describes the different algorithms, the vector space model, and presents a discussion on the related work for music mood and genre classification. Chapter 5 presents two experiments that evaluate our proposed approach for music mood and genre classification using a supervised classification strategy. We study a classification approach based on ANEW-derived metafeatures, and we compare the results with the standard vector space model approach, considering also different term weightings. Moreover, the proposed approaches are evaluated using a large freely-available dataset.

Finally, Chapter 6 presents the conclusions derived from the work carried out in this thesis. Here, we highlight the key findings from each of the experiments presented, and several lines of future work related to the thesis are further discussed.

Part I

A COMPREHENSIVE EVALUATION OF COLLABORATIVE FILTERING RECOMMENDER SYSTEMS ALGORITHMS

Recommender systems help users cope with the large amount of information available on the Internet by filtering and presenting information in a personalised fashion. From personalised search to music recommendations, these type of systems have nowadays become ubiquitous. However, there is evidence showing that some widely used collaborative filtering algorithms may contribute to the filter bubble problem, generate non-diverse recommendation lists or perform poorly in terms of catalog coverage.

Here, we present an overview on the field of recommender systems, introducing the most popular collaborative filtering algorithm families and describing in detail the methodologies and metrics used for evaluation. We perform a comprehensive evaluation of several collaborative filtering algorithms. The analysis of performance is studied in terms of accuracy and other metrics such as popularity, diversity or coverage, with the with the aim to better understand the recommendations generated, and the relations and tradeoffs between the different metrics used for evaluation. Moreover, we perform experiments in different datasets from several domains, with the aim of results generalisation.

2 BACKGROUND AND RELATED WORK

2.1 INTRODUCTION

As access to different types of media becomes easier and the amount of user-generated content increases, it has become very difficult to deal with the large catalogs of information available on the Internet. For example, finding a new movie to watch, discovering new bands to listen to, or deciding whom to follow in online social networks is no longer a straightforward task. This problem is commonly referred to as *information overload* [17]. Thus, there is a need for a new class of intelligent systems that help users to navigate through all this information, by automatically selecting the most relevant content and filtering that considered to be irrelevant. Recommender systems [145, 154, 146] address this problem by learning the preferences of users. The gathered knowledge is then applied to personalise the user experience in different services; from advertising to clothes, movies or music personalised recommendations.

Several types of recommender systems have been proposed, according to different information needs or recommendation scenarios. *Collaborative filtering* algorithms exploit the wisdom of the crowd by finding similarities across users [59] or items [78, 153], and more advanced techniques exploit latent features to make recommendations [87]. *Content-based filtering* algorithms [100] use the unstructured item metadata to generate recommendations by creating a user profile and finding items which are similar to it. *Case-based filtering* approaches [161] use structured metadata to generate recommendations in a similar fashion. In this chapter some well known collaborative filtering algorithms are explained, followed by a detailed description of the evaluation process, including methodologies and metrics to evaluate various properties such as popularity, diversity or accuracy.

The remainder of the chapter is organised as follows. First the basic concepts to understand recommender systems are presented in Section 2.2, including the notation used in this thesis and a taxonomy of recommender systems. In Section 2.3 the most popular algorithms for neighbourhood-based collaborative filtering are introduced. Later, Section 2.4 introduces the matrix factorisation approach. Section 2.5 presents an overview of recommender systems evaluation, describing the experimental methodology and the key properties of recommender systems, discussing the related work in the field.

2.2 BASIC CONCEPTS

In this section we introduce the notation used in this thesis, together with a taxonomy that classifies recommender systems according to the different recommendation scenarios. We also describe the different types of user feedback that recommender systems use as a source of information to generate the recommendations.

2.2.1 *Notation*

The following notation is used in this thesis:

- A set of m users $U = \{u_1, u_2, \cdots, u_m\}$.
- A set of n items $I = \{i_1, i_2, \cdots, u_n\}$.
- A user-item matrix $M_{m \times n}$, which captures the preferences of each of the m users for each of the n items.
- A set of items for which a user has expressed her preference or feedback: I_u.
- A set of candidates items to be considered for recommendation $C = \{c_1, c_2, \cdots, c_k\}$.
- The rating of user u for item i: $r_{u,i}$.
- A prediction for the rating of user u for item i: $R_{u,i}$.
- A set of recommendations made for a user u is denoted by L_u.
- The train and test sets for user u are denoted by T_{r_u} and T_{e_u} respectively.

2.2.2 *Recommendation Tasks*

Herlocker et al. [60] proposes a taxonomy for recommender systems where the various recommendation tasks are divided according to the end-user goals. *Annotation in context* and *finding good items* are the two most common tasks.

- In the *annotation in context* task, items are separated into at least two categories, e.g., relevant/irrelevant items. This task is very common in e-mail mailboxes (separate priority/non priority emails) or message boards (relevant/irrelevant messages).

- In the *finding good items* task, user are presented with a list of recommendations. This can be divided into two different subtasks, according to the way recommendations are presented to the user. In the *generate a prediction* task, the goal of the recommender system is to assign to each item a numeric score $R_{u,i}$, which represents the predicted rating that a certain user u would give to a particular item i. In the *Recommend a list of items* task the items are presented as a ranked list according to the predicted user preference. This task is very common in systems such as Netflix[1], where the *top picks* (the most relevant movies) are presented to users.

2.2.3 *Types of User Feeedback*

User feedback is a key component in a recommender system; it is needed to measure the level of interaction between a particular user and an item or set of items. Moreover, it is necessary to build user profiles, which the recommender system exploits to make personalised recommendations. User feedback can be classified into two types: *implicit feedback* and *explicit feedback*.

2.2.3.1 *Implicit Feedback*

Implicit Feedback [123] Is the result of storing basic user-item interaction information without the need for explicit activity on the part of the user, e.g., click rates, number of played or skipped songs, query logs, etc. This kind of data is very valuable and is unobtrusive, since the user is not asked to give any type of information, therefore it does not impose extra effort for the user when using the system. However, even when this data is plentiful, it is also very noisy. Various recommender system algorithms specially designed for implicit feedback have been proposed [66, 123, 82], obtaining good performance.

2.2.3.2 *Explicit Feedback*

Explicit Feedback [73] Is the type of feedback data that the system asks the user to manually input. This type of data is considered to capture user preference more accurately than implicit feedback, as users are requested to directly specify their preferences. However, the quantity of available explicit feedback is often limited, as users tend to find it time consuming to provide explicit ratings about many items. The most common types of explicit feedback data are described below.

1 http://www.netflix.com

- **Unary ratings** is the most simple type of explicit feedback, where the only information available is whether the user likes a certain item, and negative feedback is not considered. This type of feedback is popular in online social networks like Facebook[2] or google+[3].
- **Binary ratings** express feedback over an item on a binary rating scale (like/dislike, +1/-1, yes/no). This type of feedback is used in content portals such as youtube[4].
- **Numerical ratings** indicate the user's interest expressed for an item using a numerical scale, usually [1-5] or [1-10]. This type of explicit feedback is very popular, and used in several movie recommendation sites such as IMDB[5] or the video streaming platform Netflix.
- **Tagging** allows the user to classify items using predefined text (keywords) and also expresses the user's interest for an item. For example, tagging has successfully been used in the music portal Last.fm[6].
- **Text reviews** allow the user to express her opinion on items using free text. This type of feedback is widely exploited in the field of e-commerce, in websites like Amazon[7] or hotel reviews portals such as tripadvisor[8].

Most systems make use of either implicit or explicit feedback [66, 123, 87, 54] , yet recent work in the area of music recommendation, for example, [73, 128] has focused on the comparison of implicit and explicit feedback. The results show that implicit and explicit feedback can be used as complementary in the same system, and that the performance of both approaches is comparable in terms of accuracy.

2.3 NEIGHBOURHOOD BASED MODELS

Neighbourhood-Based algorithms [58] dominated the field of recommender systems until very recently. Prior to the Netflix prize [12, 8], most of the approaches relied on these algorithms to generate recommendations. Two algorithms are prominent in this family: user-based collaborative filtering (*UKNN*) [159, 59] and item-based collaborative filtering *(IKNN)* [59, 78, 153]. Both approaches exploit the nearest neighbourhood model to find similar users (or items) to the ones

2 https://facebook.com
3 https://plus.google.com
4 http://www.youtube.com
5 http://www.imdb.com
6 http://www.last.fm
7 http://www.amazon.com
8 http://www.tripadvisor.com

they are making recommendations for, and can be used to either generate a rating prediction for a particular user-item pair (*generate a rating prediction* task), or to present a ranked list of recommendations (*recommend a list of items* task).

Neighbourhood models are very popular and widely present in e-comerce applications including commercial sites such as Amazon [99]. However, they have some drawbacks (shared with other collaborative filtering approaches) worth mentioning. When a new user logs into the system, she has to rate some items before getting recommendations *(the cold start problem, [155])*. Similarly, an item will not be recommended before being rated *(first rater problem, [113])*. Finally, users with very rare taste are not able to get good recommendations as they do not have many (or any) close neighbours.

Figure 5 presents a conceptual representation of a collaborative recommender system, where the ratings are stored in a user-item matrix $M_{m \times n}$ with m users and n items. Each user is represented by a unique row in the matrix and each item is represented by a unique column. In this example there are 4 users and 9 items. Empty boxes indicate that users have not rated the corresponding items. In the majority of real-world scenarios, the user-item matrix is significantly more sparse than the one depicted here. For example, most of the entries in the user-item matrix are typically empty, indicating that users only interacted with a small fraction of all available items.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9
u_1	2	5			4		1	2	3
u_2	4	3	4				2	5	
u_3	1	2		2		4		1	
u_4	5		2		3		5		4

Figure 5: An example of a user-item matrix, in which each user is represented by a unique row in the matrix and each item is represented by a unique column.

In the representation shown in Figure 5 each element in the matrix $M_{n \times m}$ represents the preference (explicit) of an user for an item. For example, in this case user u_2 has assigned a rating of 4 out of 5 to time i_1.

2.3.1 *User Neighbourhood Models: UKNN*

User-based collaborative filtering *(UKNN)*, first introduced in [59], operates by finding users who have similar taste to the active user

u_a, for whom rating predictions or recommendations are being made. *UKNN* algorithms exploit the rating matrix and the active user's list of rated items I_{u_a}, finding a set of *neighbours* for the active user for which recommendations are made. In a way, this model automates the idea of the word of mouth [159] in which recommendations are spread in the society.

In what follows, the main components of the *UKNN* algorithms are described. First, the metrics used to identify similar users are presented. Then, the main approaches to neighbourhood formation are explained, followed by the techniques used to generate predictions and top-N recommendations for the active user.

2.3.1.1 *Similarity Computation*

The objective of this step is to construct a *similarity matrix* S, where each entry $S_{a,u}$ represents the similarity between users a and u. The similarity metric used can be a simple overlap between the number of rated items, which is the most simple approach, yet recent work has found it is socially accurate [11]. Other metrics include cosine similarity [20], and Pearson's correlation [144, 84], which is widely used and is defined in Equation 1.

$$S_{a,u} = \frac{\sum_{j \in I_a \bigcap I_u} (r_{a,j} - \overline{r_a})(r_{u,j} - \overline{r_j})}{\sqrt{\sum_{j \in I_a \bigcap I_u} (r_{a,j} - \overline{r_a})^2 \cdot \sum_{j \in I_a \bigcap I_u} (r_{u,j} - \overline{r_u})^2}}, \tag{1}$$

where $\overline{r_a}$ is the mean of all ratings assigned by user a, and $r_{a,j}$ is the rating of user a for item j.

In the Pearson correlation the summations are calculated only over those items that have been rated by both users u and a. Pearson correlation results in a value between -1 and 1. A correlation of +1 indicates perfect agreement on all co-rated items rated by each user, and a correlation of -1 indicates total disagreement on co-rated items. Note that, if there are no co-rated items between two users, the Pearson correlation similarity value will be zero.

2.3.1.2 *Neighbourhood Selection*

Once the similarities between user u_a and all other users in the system are calculated, the top K most similar users are selected as the user's neighbourhood. The value of K is usually calculated empirically [33].

2.3.1.3 *Rating Prediction and Recommendation Generation*

The predicted rating of a certain item for the active user ($R_{a,i}$) is computed by calculating a normalised weighted average over the ratings for the item from each of the active user's neighbours, for example using Resnisk's formula, described in Equation 2. Finally, the items are ranked according to their predicted ratings, and the top-N items with the highest predicted ratings are returned as a recommendation to the target user u_a.

$$R_{a,i} = \overline{r_a} + \frac{\sum_{u=1}^{K} S_{a,u} \cdot (r_{u,i} - \overline{r_u})}{\sum_{u=1}^{K} |S_{a,u}|}, \qquad (2)$$

where K is the neighbourhood size, $\overline{r_a}$ is the mean of all ratings assigned by user a and $S_{a,u}$ is the similarity between users a and u.

2.3.2 *Item Neighbourhood Models: IKNN*

In the item-based collaborative filtering approach (*IKNN*) [78, 153], the models rely on the analysis of the rating matrix to identify similar items to those for which the user expressed her interest. This approach was developed as an answer to the scalability issue of user neighbourhood models [151]. In a typical e-commerce scenario, the number of items is significantly smaller than the number of users, and new items are introduced at a slower pace. Thus, computing similarities between items is a less expensive task in terms of computation. Also, as the average number of ratings per item is usually greater than for users, a new rating over a particular item i will generally not substantially change the item-item similarity, making the *IKNN* algorithm scalable as similarities do not need to be updated in real-time.

In essence, this algorithm recommends items that are similar to items that users have previously rated (purchased or transacted with in some manner). In what follows, the main components of the *IKNN* algorithms are presented. First, the metrics used to identify similar items are described. Then, the main approaches to neighbourhood formation are presented, followed by the techniques used to generate predictions and top-N recommendations for the active user.

2.3.2.1 *Similarity Computation.*

Computing similarities between items is a key feature of the item-based collaborative filtering algorithm. In terms of the user-item matrix representation shown in Figure 5, item similarities are computed across the *columns* (i.e., items) of the matrix. The objective of this step is to construct a *similarity matrix* S, where each entry $S_{i,j}$ represents the similarity between items i and j. A number of metrics have been proposed to compute the similarity between items; for example, Pearson correlation [151], conditional-probability based similarity [78] and cosine similarity [78], shown in Equation 3.

$$S_{i,j} = \frac{\sum_{u \in U_i \cap U_j} r_{u,i} \cdot r_{u,j}}{\sqrt{\sum_{u \in U_i} r_{u,i}^2} \sqrt{\sum_{u \in U_j} r_{u,j}^2}}, \tag{3}$$

Where $r_{u,i}$ is the rating assigned to item i by user u, U_i is the set of users that have rated item i and $U_i \cap U_j$ is the set of users who have rated both items i and j.

2.3.2.2 *Candidate Recommendation Generation.*

In this step a candidate set of recommendations is generated for the active user u_a by taking the union of the K most similar items for each item $i \in I_u$, i.e., the items that user u has previously rated. Depending on the application domain, any items in the candidate set that user u has previously rated can be removed from it to avoid recommending items that the user already knows.

2.3.2.3 *Ranking Candidate Recommendations.*

For each item in the candidate set $c \in C$, its overall similarity to the set I_u (i. e., the items previously rated by the target user) is computed as the sum of the similarities between all items $i \in I_u$ and c. Finally, the top-N most similar items in C are returned as recommendations to the target user.

2.4 MATRIX FACTORISATION

Matrix factorisation [88, 87] is a class of latent factor models where the latent features are induced from the user-item rating matrix M. Thus, in general terms, it characterises the items and users in a f-dimensional space, where the factors are calculated from implicit or explicit feedback, and where a high correspondence between item and user factors leads to a recommendation.

Matrix factorisation models have been gaining popularity, performing very well in terms of accuracy [26]. Moreover, since the dimension of the new space is much smaller than the original one ($f << n$, where f is the number of factors and the n is the number of items) this type of approach is less computationally expensive and more scalable [82].

In a matrix factorisation approach, the user-item interactions are modelled as a inner product in the f-dimensional space. The ratings for each item i are modelled as a vector $q_i \in \Re^f$ (the item's latent factors) , while each user is associated with a vector $p_u \in \Re^f$ (user's latent factors). Thus, the rating of user u for a given item i can be approximated as $R_{u,i} = q_i^T p_u$. However, even when making a rating prediction seems to be a straightforward process, in the new space the meaning of the induced latent factors is usually unknown and domain dependent. For example, in the movie domain the factors could measure the amount of comedy or drama a movie possesses or a user enjoys.

Matrix factorisation models are known to outperform the traditional neighbourhood based models, previously described in Section 2.3, in terms of accuracy and scalability. Moreover, matrix factorisation algorithms can model specific constrains or user needs, such as temporal dynamics [88], or take in account context by adding dimensions to the original rating matrix [5, 77].

Matrix factorisation algorithms often rely on singular value decomposition (SVD), which has been successfully applied in the field of information retrieval to identify the underlying latent semantic factors in documents [37, 13]. However, when this approach is applied in a collaborative filtering recommender system scenario, the sparseness of the user-item matrix represents a problem, since missing ratings need to be dealt with in the traditional matrix factorisation models. Thus, even when early approaches tackled this problem by estimating the missing values [174, 152], an issue remains as to how best estimate those.

More recent matrix factorisation approaches [88, 169, 9], rely solely on the feedback data. These methods learn the user and item factors by minimising a loss function like the one shown in Equation 4 and adding a regularisation function to avoid the overfitting of the learned parameters.

$$\min_{q,p} \sum_{(u,i) \in T_r} L(r_{u,i}, q_i, p_u) + \alpha(|q_i|^2 + |p_u|^2)\,, \tag{4}$$

where q and p represent the factors for all items and users respectively, T_r is the train set, L is the loss function to be minimised (which depends only on the user ratings seen in the train set), and α is the regularisation parameter which is usually set empirically.

$$L(r_{u,i}, q_i, p_u) = (r_{u,i} - q_i^T p_u)^2 \tag{5}$$

One of the most widely used loss functions is *squared error loss* [77], shown in Equation 5. However, several others, such as, *Hubert loss* [67] or a multi objective loss function that includes diversity [69] have been proposed.

2.4.1 *Learning Algorithns*

In order to minimise Equation 4, several approaches have been proposed. Simon Funk[9] first introduced the use of a *stochastic gradiant descend methods* (SGD) [184] in the context of the Netflix prize [12]. Another commonly used algorithm for minimisation is *alternating least squares*.

2.4.1.1 *Stochastic Gradient Descend*

The *stochastic gradient descend* algorithm is often used in matrix factorisation recommender systems [86, 129]. It works by looping through all the ratings in the train set, calculating a rating prediction for each of the user-item pairs $R_{u,i} = q_i^T p_u$. Thus, if we consider the Loss function as in Equation 5, for each rating in the train set, the parameters are modified in the direction of the gradient, as shown in equation 6.

$$\begin{aligned} q_i &\leftarrow q_i + \gamma \cdot (L \cdot p_u - \alpha \cdot q_i) \\ p_u &\leftarrow p_u + \gamma \cdot (L \cdot q_i - \alpha \cdot p_u), \end{aligned} \tag{6}$$

where γ controls the weight in which the function moves towards the gradient, and α controls the regularisation part of the function.

2.4.1.2 *Alternating Least Squares*

The *alternating least squares* algorithm [167, 9] works by fixing the q and p vectors in a rotational matter. When one of these parameters is fixed the problem becomes a quadratic optimisation problem that

9 Netflix Update: Try This at Home. http://sifter.org/~simon/journal/20061211.html

can be solved optimally. Thus, on each step, one of the two free parameters (q or p) is set, while the other is recomputed by solving a least squares problems. This process is repeated until the function in Equation 4 converges [9].

This algorithm is particularly useful for scalable recommender systems, as it can be parallelized [183]. Moreover, it is also more efficient than gradiant descend methods when applied in implicit feedback data, as shown in [66].

2.5 RECOMMENDER SYSTEMS EVALUATION

The evaluation of recommender systems is a core part in the development of new and better techniques. It has been widely studied and several reviews on the topic have been published, describing the most common evaluation methodologies and metrics [60, 158]. Most of those metrics, such as precision and recall [23], or discounted cumulative gain [143] are adapted from the field of information retrieval; a field which is closely related to recommender systems.

In the following an overview of the methods and metrics to evaluate recommender systems is presented. First, a general description of the evaluation methodology followed in the field is presented in Section 2.5.1. The most popular prediction accuracy metrics, such as, *RMSE* or *MAE*, which are widely used in recommender systems evaluation are described in Section 2.5.2.1, and widely used top-N accuracy metrics, such as precision, recall, or F-measure are described in 2.5.2.2. A discussion of the relevant related work on accuracy evaluation is presented in Section 2.5.2.3. Finally, the properties beyond accuracy, such as diversity, or popularity are described in Section 2.5.3, together with the related work that has used these properties for evaluation.

2.5.1 *Evaluation Methodology*

The development and evaluation of a recommender system is often a continuous cyclical process. It begins when the requirements of the system are described, and all possible candidate algorithms are considered and evaluated using offline experiments. Once the system is built, user studies are carried out to test the hypothesis and correlate the results of the offline evaluation. Finally, once the system is online, in form of an application or portal, new approaches or variations are continuously evaluated using A/B tests to validate the solution. Every time a new algorithm is introduced, it is typically evaluated following the same approach (offline evaluation, user studies and online evaluation) to ensure that the best performance is achieved [158]. A detailed description of each evaluation step follows.

2.5.1.1 *Offline Experiments*

In an offline experiment [60] the main goal is to filter out inadequate solutions at a very early stage, leaving a small subset of algorithms for further evaluation. The evaluation is done using an existing dataset, which represents the interaction for a set of users over the set of items. Frequently used datasets include the MovieLens dataset [59], the NetFlix prize dataset [12] or the Million Songs dataset [14], which are *natural datasets* based on historical live-user data. Other datasets such as BigFlix [156] (a *synthetic* extended version of the original NetFlix prize dataset) include artificial data created with specific properties.

Offline experimentation is the first part of a recommender system evaluation, as it presents a closed and controlled scenario in which to simulate user behaviour. Moreover, it is the most inexpensive approach in terms of economic resources, human effort and computational costs, facilitating the evaluation of a large number of alternatives. In order to evaluate the system, the dataset is usually divided into a train set T_t, which is used to train the system, and a test set T_e, where the system is evaluated.

With the aim of better representing the user behaviour when performing an offline evaluation, several techniques have been proposed to create the train/test splits [2]. *Repeated random sub-sampling* randomly splits the dataset into train and test several times. *K-fold cross-validation* randomly partitions the dataset into K subsamples. Each of the K times, one of the subsamples is used as the test set, while the remanding K-1 subsamples are used as the train set. The *leave-one-out* technique selects a single observation from the dataset as the test set, while the rest is used as the train set.

2.5.1.2 *User Studies*

User studies [158] are the first step in online evaluation, allowing to create small scale experiments to test the system for bugs or evident malfunctions before doing extensive online testing. User studies are designed to elicit the users opinion on the system, or evaluate the users interaction with it. There are different possibilities when performing user studies, such as questionaries, eye-tracking studies, usage studies, etc.

The user study is conducted by recruiting a set of test subjects, and asking them to perform several tasks requiring an interaction with the recommender system. The interactions of the users with the system are recorded and later analysed to evaluate its performance.

The main drawback of this approach is its cost. Conducting extensive user studies is expensive both in terms of time and economical cost. Moreover, gathering enough users willing to participate can be a difficult task, and usually economic compensation is given to the users who participate, possibly introducing bias into the process. As such, it is important to sample the test subjects in a way that they represent the population of users that will use the actual system.

2.5.1.3 *A/B Testing*

A/B testing [158, 83] is an evaluation technique widely used in all kind of real-world systems, from e-commerce sites to search engines. It consist in randomly redirecting a small part of the traffic to each of the alternatives being evaluated, so part of the users will see the current system and others will see the new system (or part of the system) which is being evaluated.

This methodology allows to measure the overall system goal (for example, monthly user retention or click-through rate), understand the relation between this goal and other recommender system properties such as accuracy or diversity, and finally decide which is the best approach to implement. However, the user behaviour depends on many factors, making it difficult to fully understand the reasons behind certain behavioural patterns. Therefore, when performing an A/B test only one factor should be tested at a time. For example, if the user interface changes the recommendation algorithm should not and vice versa.

2.5.2 *Evaluation Metrics — Accuracy*

The ability of a recommender system to generate relevant recommendations for users is obviously a key factor on its performance, and hence, accuracy metrics are a key consideration in any recommender system evaluation. Thus, accuracy metrics have been extensively used in recommender systems evaluation and different metrics are applied depending on the recommendation tasks previously described. The prediction accuracy metrics described in Section 2.5.2.1 are better suited for the *generate a prediction* task, and top-N accuracy metrics presented in Section 2.5.2.2 are often used to measure the system's performance in the *recommend a list of items* task.

2.5.2.1 *Prediction Accuracy Metrics*

Prediction accuracy metrics calculate the error between a series of rating predictions made for the users by the system and the known ratings for these items. Thus, these metrics are suited for the *generate a prediction* task. However, it is known that prediction accuracy metrics are limited in the sense that they do not capture how users interact with items when a list of recommendations is generated (the most common output of a recommender system). Nevertheless, this type of metrics have been widely applied in recommender systems evaluation, specially since the very popular Netflix Prize [12] took place.

MEAN ABSOLUTE ERROR (MAE) is a very popular metric, which measures the average absolute deviation between a set of predicted ratings for which the true rating is known. It is computed as per Equation 7. Typically, the ratings are known because they are part of the test set in an offline experiment, or because they were obtained through an user study in an online experiment.

$$\mathrm{MAE} = \frac{\sum\limits_{u,i \in T_e} |R_{u,i} - r_{u,i}|}{|T_e|}, \tag{7}$$

where $R_{u,i}$ is the predicted rating of user u for item i, $r_{u,i}$ is the known rating of user u for item i, and T_e represents all the user-item pairs in the test set.

ROOT MEAN SQUARE ERROR (RMSE) is an error prediction metric similar to *MAE*, but where the error between the individual predictions and the known ratings is squared. This has an effect of penalising larger errors [60], i.e., the errors that are indicative of a poor prediction performance. *RMSE* is calculated as per Equation 8.

$$\mathrm{RMSE} = \sqrt{\frac{\sum\limits_{u,i \in T_e} (R_{u,i} - r_{u,i})^2}{|T_e|}} \tag{8}$$

Accuracy metrics are useful in certain tasks, and extremely popular for recommender system evaluation. However, there are some considerations that need to be taken in account when using them. For example, [109] agues that prediction accuracy metrics have several flaws and limitations that might hurt the user experience in a recommender system.

1. If the evaluated user u has not rated a particular item i, it is impossible to calculate the error for the prediction $R_{u,i}$, hence it is impossible to know if the recommendation is good or not.
2. An error of size ϵ has the same impact on *MAE* regardless of where that error places the item in a top-N ranking. While this might not be an issue in some cases, it is a problem in the *recommend a list of items* task, where a small subset of items are recommended to the user. Thus, an error when predicting an item in the top positions of the recommendation list will be usually more important than an error made further in the list.
3. Highly accurate predictions on many mediocre or bad items (items which are not relevant for the user) and a poor performance on highly ranked items can result on a poor user experience, yet the system's overall accuracy will still be good.
4. Prediction accuracy metrics cannot be used to measure the accuracy of recommendation techniques that do not produce a predicted rating for each item, but only show a list of top-N recommended items.

2.5.2.2 *Top-N Accuracy Metrics*

Top-N accuracy metrics are used in the *recommend a list of items* task, where a list of top-N recommendations are presented to users. These metrics measure the ability of a recommender system to produce a list of relevant recommendations. Here, we describe the top-N accuracy metrics from the field of information retrieval which are commonly used to evaluate recommender systems.

RECALL (RCL) measures the proportion of relevant items which are recommended to the user, as shown in Equation 9.

$$RCL = \frac{|recommended| \cap |relevant|}{|relevant|}, \tag{9}$$

where $|recommended|$ represents the number of items which are recommended to a user, and $|relevant|$ the number of items for which the user expressed her positive (implicit or explicit) feedback.

PRECISION (PRC) measures the ability of the system to suggest useful items to the user. It is calculated as the proportion of relevant items out of those which are recommended, as shown in Equation 10.

$$\mathsf{PRC} = \frac{|recommended| \cap |relevant|}{|recommended|} \tag{10}$$

F-MEASURE Precision and recall are often conflicting properties. For example, as the number of items included in the recommended list increases, precisions tends to decrease, while recall improves. Thus, the F-measure [102], which combines precision and recall, is introduced to provide a single score for the relevance of items recommended in the list. F-1 is calculated as per Equation 11.

$$\mathsf{F\text{-}1} = 2\frac{\mathsf{PRC} \cdot \mathsf{RCL}}{\mathsf{PRC} + \mathsf{RCL}} \tag{11}$$

Both precision and recall are widely used in recommender systems (and information retrieval) evaluation. However, as mentioned before, they have one major drawback: a system that retrieves all the items will perform perfectly in recall, while having a poor performance in precision. Another issue is how to evaluate the non-rated items, i.e., those items for which relevance is not known. McLaughlin et al. [109] proposes that the non-rated items should be treated as non-relevant, since not doing so would lead to inaccurate results in the system's performance. However, counting non-rated items as non-relevant will produce and underestimate of the precision on the system.

2.5.2.3 *Discussion*

Recent work in the area of recommender systems has begun to focus more on the *recommending a list of items* task, and how to best evaluate the relevance of the recommendations generated [54, 166]. For example, research on evaluation methodology has been described in [10]. The authors analyse different experimental configurations, focusing on train and test set creation, and using top-N accuracy metrics, such as precision or recall [158] for evaluation. They highlight the fact that top-N accuracy metrics are often applied in different experimental configurations; which makes it difficult to compare works, even when the same metrics and datasets are used.

The relationship between prediction accuracy metrics (such as *RMSE*) and top-N accuracy metrics (precision and recall) is studied in detail in [35] . The authors explain how prediction accuracy metrics are not appropriated to evaluate algorithms that present the user with a list recommendations without an associated prediction value, which is also highlighted in [54, 166]. They also explain that improvements in algorithms measured by *RMSE* do not translate to improvements in other metrics.

The top-N recommendation task is also studied in [135], where an analysis of the popularity effect and the rating distribution of missing ratings is considered. The study shows that, just as with known ratings, missing ratings can be modelled as being non-random. The study concludes that ignoring non-rated items biases the evaluation showing very positive results, while considering the missing ratings as negative biases the evaluation in favour of algorithms that exploit popularity [35, 10].

An evaluation of neighbourhood models is presented in [141], which focuses on the influence of neighbour selection on the performance of *UKNN* and *IKNN* algorithms. The authors explain how the performance of the algorithm is biased by the evaluation methodology. They show that neighbourhood selection has in fact a limited effect on performance, and that often the contributions of neighbours are disguised or misleading.

Meyer et al. [115] propose different evaluation metrics (e.g., combining new metrics such as the average measure of impact with metrics like *RMSE* or precision) depending on the recommender systems tasks described in Section 2.2.2. The work concludes that there is not a clear correlation between prediction accuracy metrics and the quality of the recommendations. Moreover, it also analyses the effect on recommendations based on a user segmentation that takes in account the long tail distribution of items and user ratings. The results show that the performance is closely tight to these segments.

The related work described above clearly highlights that different metrics to evaluate recommender systems accuracy have been proposed. As such, there is much work to be done in order to define a clear and consistent approach to tackle the evaluation of recommender systems. In the next chapter we will return to this question, presenting an evaluation approach to compare and contrast the performance of multiple recommender systems algorithms, measuring the performance in terms of top-N accuracy metrics, and studying the relations and tradeoffs between those and properties beyond accuracy.

2.5.3 *Evaluation Metrics — Beyond Accuracy*

The metrics described in the previous section do not evaluate certain aspects involved in the performance of a recommender system and the user's perceived quality of it. Even thought accuracy is one of the most important properties to evaluate in a recommender system, other properties also need to be taken in consideration. Previous work [60, 158] has described such properties, and how they can be evaluated in a recommender system scenario. A discussion of these properties is presented below.

2.5.3.1 *Diversity*

Diversity is a property that can be evaluated in the *recommend a list of items* task. It tries to capture how different the recommended items in a list are from each other [161, 185, 70]. It is calculated by doing a pairwise comparison of the items recommended in the list, as per Equation 12.

Generating diverse recommendations is important to enhance the user experience in a recommender system and enable content discovery [33]. For example, recommending all five *Harry Potter* movies for a user who liked *Harry Potter and the Philosopher's Stone* might be accurate, however not relevant due to the lack of diversity.

$$\mathrm{Diversity}(r_1, r_2, \cdots, r_N) = \frac{2}{|N||N-1|} \cdot \sum_{i=1}^{N} \sum_{j=i+1}^{N} (1 - \mathrm{Sim}(ci, cj)), \tag{12}$$

where the diversity is computed as a function of the top-N recommendations $r_1, r_2, \cdots, r_N$ and *Sim* is a similarity function such as Pearson Correlation or cosine similarity which is computed using the item co-ratings information.

2.5.3.2 *Per-user Item Coverage (UIC)*

This metric measures the proportion of items out of the total that are potential recommendation candidates for a particular user, as defined in Equation 13. For example, only items rated by the active user's neighbourhood can be recommended to the user by the *UKNN* algorithm. This property is very important in the case of recommender systems applied to e-commerce, where ideally every item should be recommendable to users with the aim of increasing the percentage of items from the stock being sold.

$$\mathrm{UIC} = \frac{\sum_{\forall u \in U} \frac{|C_u|}{n}}{m}, \tag{13}$$

where $|C_u|$ is the number of items which are recommendation candidates for user u, and m and n represent the number of users and items in the system respectively.

2.5.3.3 *Item Reach (IRE)*

Item reach, also known as *catalog coverage* [52], is calculated as the proportion of items that get recommended from all items in the system (Equation 14). Although a recommender system can have perfect per-user item coverage, in an scenario where a recommender system generates a ranked list of the top-N recommendations, only a small subset of items will be recommended to users; this metric measures the size of that subset over all users. Thus, this property is very important for e-commerce providers, as it is important to maximise the amount of products recommended and subsequently purchased.

$$IRE = \frac{|\cup_{\forall u \in U} L_u|}{n}, \tag{14}$$

where L_u is the set of items which are recommended for to user u, and n is the number of items in the system.

2.5.3.4 *Popularity (POP)*

Popularity can have several meanings depending on the context of the recommendations. For example, in the music domain artist popularity is measured by the number of records sold, while Twitter identifies trending (i.e., popular) topics using text analysis approaches. We use the number of available ratings per item in the train set to calculate item popularity. The popularity metric $POP_i \in [0, 1]$ for item $i \in I$ is defined in Equation 15, which is the total number of ratings for item i, $\aleph_i$, divided by the total number of ratings overall in the system, $\aleph_I$.

$$POP_i = \frac{\aleph_i}{\aleph_I} \tag{15}$$

Popularity plays a major role in e-commerce purchases, where around 80% of the purchases come from 20% of the items, while the reminding 80% of the items are in the so-called long tail [3]. A recommender system should be able to exploit the long tail to enhance content discovery. Thus, this property should be measured to better understand the performance of the system.

2.5.3.5 *Uniqueness*

We propose this metric to measure the ability of an algorithm to generate recommendations which are not generated by other algorithms, defined as per Equation 16. Thus, if we consider two recommendation sets, R^a and R^b, which are generated by different algorithms, the uniqueness of the recommendation set R^a with respect to R^bcan be calculated as the number of items in the set R^a which are not found in R^b.

$$\mathrm{Uniqueness}(R^a, R^b) = R^a \setminus R^b \tag{16}$$

Measuring the uniqueness of an algorithm is important to predict the change in user behaviour when recommendations from two different recommender system algorithms are presented to the user. If an algorithm generates a high proportion of unique recommendations the user experience will change drastically. However, if only a small proportion of the recommended items are unique, most of the users might not even note the difference.

2.5.3.6 *Other Metrics*

NOVELTY measures is the ability of a system to recommend items that the user was not already aware of [28, 70]. While non-novel recommendations are still valuable, for many applications novelty is a key property. A system that exploits this property will not make recommendations for known items, i.e, if the user rated the movie *Harry Potter and the Philosopher's Stone*, recommending other movies from the *Harry Potter* saga will not result in novel recommendations.

SERENDIPITY measures the system ability to generate surprising (and successful) recommendations, by generating recommendations of unknown items which are not obviously related to the user's profile [121]. The key difference between novelty and serendipity in recommendations is that novel recommendations are recommendations that the user would likely discovery without the aid of a recommender system, while serendipitous recommendations are those the user would not be likely to discover alone.

LEARNING RATE measures how fast a system becomes an effective predictor of taste, as the amount of training data grows. Learning rates can be computed per-user, per-item, or measuring the total number of ratings in the system that lead to a particular point in the system's performance curve.

SCALABILITY measures the behaviour of the system in terms of computation time as the amount of available data (users, items and ratings) grows. Scalability is a key issue in commercial systems [156], where recommendations are often computed in real time and amount of data is usually large, e.g., Netflix had more than 44 Million subscribers at the end of 2013.

ROBUSTNESS measures the stability of the system when new information is input to the system [124, 25]. When the system is robust, a single new rating should not result in major changes in the system's recommendations. Also when fake profiles are injected during a profile-injection attack the system should be able to distinguish between real and fake information.

2.5.3.7 *Discussion*

Recent work [85] highlights the key challenges in recommender systems nowadays, focusing on the evaluation of the user experience as a whole, rather than treating the algorithm as a separate entity. Problems like scalability, better exploitation of user-generated content, developing a common research infrastructure for evaluation, and the fact that accuracy alone is not sufficient to evaluate the overall performance of the system are already well stablished [112, 48]. In this context, new metrics to measure different properties of the recommender system have been proposed, together with experimental methodologies to evaluate them.

In [172], a new framework for evaluating novelty and diversity in recommender systems is introduced. The work is motivated by the lack of agreement when evaluating these properties - different studies use different metrics, the relationship between them is not studied and some of them have flaws. The work studies novelty from two different perspectives: popularity and distance, and considers ranking and relevance when measuring novelty in recommendations, giving a more complete picture of the system.

Vargas et al. [173] do an analysis of diversity, adapting the definitions, evaluation metrics and procedures from ad-hoc information retrieval into a recommender systems task, by introducing the concept of user profile as an analogy to a query. The authors evaluate several state-of-the-art algorithms using four different diversity metrics. Jawhaheer et al. [73] also analyse the diversity in a music recommendation service, arguing that optimising recommendations for prediction accuracy metrics such as *RMSE* might lead to less diverse recommendations.

Said et al. [150] use a furthest neighbour model to increase diversity, generating almost orthogonal recommendation lists, when compared to the traditional *UKNN* approach. Hurley [69] introduces diversity in the optimisation function of a matrix factorisation algorithm. The results show more diverse recommendations, without having to post-filter the recommendation lists to increase diversity.

Diversity is also studied in [181], which analyses the diversity problem of top-N recommendations from a binary retrieval perspective. This work tries to maximise both diversity and similarity to the query (i.e., the user). A new metric for evaluating the proposed diversity approach is also presented. The results show improvements in both accuracy and diversity.

An evaluation of recommender systems coverage and serendipity is explored in [52], which focuses on the contribution of those metrics to the quality of a system. The paper proposes metrics to evaluate the two properties that include the usefulness of a recommended item. The authors argue that the proposed metrics correlate better with real user experience than previous metrics.

Work presented in [30] analyses the novelty and popularity of recommendations. The authors conclude that content-based recommender systems (applied to music) are not biased towards making recommendations of popular items, while a standard collaborative approach is perceived as providing higher quality recommendations even when the recommendations are not as novel.

From previous work we conclude that there is no agreement on how to best evaluate a recommender system algorithm in an offline setting. Often, different evaluation frameworks are used, and these are not consistent in the metrics they use or the experimental methodology proposed. Moreover, the properties that define the performance of a recommender system are often studied in isolation. In this work, together with an analysis of top-N accuracy metrics, properties such as diversity, popularity or coverage are also evaluated and a study on the uniqueness of the recommendation list generated by different algorithms is also presented in the next chapter. Moreover, these properties are studied as a whole, trying to understand their relations and tradeoffs.

2.6 CONCLUSIONS

This chapter introduced the basic concepts of recommender systems, the main properties that define a system's performance, and the different tasks for which a recommender system can be designed. It also described several collaborative filtering algorithm approaches, focusing on the ones that will be evaluated in the next chapter.

The most important accuracy metrics were presented, together with a discussion on the related work in the area. Moreover, the need for evaluating metrics beyond accuracy was also explained, several properties were introduced and advances in the evaluation using these properties were discussed. From the related work it is clear that there is a need for consensus in the metrics and methods used for the offline evaluation of recommender systems, which would allow for better comparison and reproducibility of the work.

The following chapter presents an extensive evaluation of different properties that are important for the performance of the system and the perceived quality of the recommendations. We study the *UKNN* and *IKNN* algorithms presented in this chapter, together with a matrix factorisation approach. An offline evaluation as described in Section 2.5.1.1 is carried out using various well-known real-world dataset, measuring the performance in terms of both accuracy metrics (presented in Section 2.5.2.1), and metrics beyond accuracy (presented in Section 2.5.3). We first perform a comprehensive analysis of the diversity, popularity, coverage and accuracy for several collaborative filtering algorithms. We then focus on the two neighbourhood models, doing an analysis of some of these properties in relation to the algorithm parameters (number of neighbours). The results are presented with an analysis of the relation and tradeoffs between the different recommender systems properties studied, paying special attention to accuracy and its relation with properties beyond accuracy such as diversity, popularity and uniqueness.

3 AN EVALUATION PERSPECTIVE

3.1 INTRODUCTION

In the previous chapter, the basic concepts of recommender systems were described, together with some well known algorithms, and the metrics and methodologies used to evaluate them. Also, the recent findings in the field of recommender systems evaluation were presented in the context of this work.

Despite many recent efforts for proposing an exhaustive and comprehensive evaluation in recommender systems [158], it is yet not clear how different factors such as the popularity bias, diversity or novelty of recommendations affects the performance of a recommender system and the user experience. In this chapter, these factors are studied with detail, presenting an analysis that focuses in the relationships and tradeoffs between the different metrics presented in the previous chapter. Two experiments are presented in this chapter, with the aim to better understand the recommendations generated in a *recommend a list of items* task, to to highlight what are the weakness and strengths of the different algorithms evaluated, and to understand how can these metrics and algorithms be used to enhance content discovery.

The rest of this chapter is organised as follows. First, in Section 3.2 we present an exhaustive evaluation of two neighbourhood based algorithms and one matrix factorisation algorithm, using three different datasets from different domains, and with different statistical properties. Thus, we study properties beyond accuracy, such as popularity bias, diversity, reach and coverage, and we present an analysis that focuses on the relationship between those properties and accuracy. Later, in Section 3.3 we extend this approach, focusing on the evaluation of the relative performance of the two neighbourhood based algorithms described in the previous chapter; *IKNN* and *UKNN*. We study the performance in relation to the neighbourhood size and the uniqueness of the recommendations, presenting and evaluation of the most prominent properties of those studied in Section 3.2 in a different experimental scenario. Moreover, we present an analysis of the results which focuses not only in the relationship between accuracy metrics and properties beyond accuracy, but also on the accuracy on the unique and common recommended items as the neighbourhood size grows.

3.2 A BIAS ANALYSIS

Most of the evaluation in recommender systems has focused on accuracy, with the hypothesis that producing more accurate recommendations leads to a better user experience. However, properties like popularity, coverage or diversity are also very important, and must be measured to better estimate the performance of a recommender system. In this section we present a comprehensive evaluation of three collaborative filtering algorithms, *UKNN*, *IKNN* and *WMF*, performed using three datasets from different domains. Moreover, the results are presented focusing on the relationship between the different evaluated properties, the datasets, and the evaluated algorithms, paying special attention to the popularity bias evaluation of the different algorithms studied.

3.2.1 *Algorithms*

In the proposed approach three different collaborative filtering algorithms are evaluated, together with a non-personalised benchmark algorithm which presents each user with the most popular items unknown for her. All the algorithms described below are evaluated in the *recommend a list of items* task described in Section 2.2.2.

We evaluate the *UKNN* and *IKNN* algorithms described in Sections 2.3.1 and 2.3.2 respectively, and cosine is used as the similarity metric. The MyMediaLite [51] implementation of the *Weighted Matrix Factorisation (WMF)* algorithm is also evaluated. This particular matrix factorisation algorithm uses the fast learning method proposed by Hu et al. [66, 127] and a global parameter optimisation to give observed values higher weights.

3.2.2 *Datasets*

To perform the proposed evaluation, three datasets are used. Two of them are publicly available, while the last one is built specifically for this work using the Facebook graph API[1]. The datasets are presented bellow, alongside a description of the preprocessing that was performed.

THE FACEBOOK DATASET (FB) was built for the purpose of this study. For each user that logged into our application (and all of her friends), we stored all her liked items. This dataset contains

1 https://developers.facebook.com/

1,428 users, 5,846 items and 64,612 unary explicit ratings (like actions of Facebook *musician/band* pages).

LASTFM-HETREC DATASET (LastFM) is a music dataset released for the *Hetrec2011 Workshop* [27]. The original dataset contains 92,834 user-artists listening interactions (unary ratings) for 1,892 users and 17,632 artists.

MOVIELENS-HETREC DATASET (ML) is the sampled version of the MovieLens dataset [59], which was expanded with additional metadata and also released for the *Hetrec2011 Workshop* [27]. The original dataset contains ratings, social tags and metadata information (directors, actors, genres, etc.) for 10,197 items and 2,113 users, with a total of 855,598 explicit numerical ratings expressed in a 1 to 5 scale.

In this experiment, only users with at least 12 items in their profiles were included in the evaluation. Furthermore, the ratings in the MovieLens dataset were transformed to unary, as in the Facebook and LastFM datasets. All items which received a rating of at least 4 out of 5 are considered as positive. This particular rating threshold has been used in previous work with similar datasets [10, 72]. The statistics of the dataset obtained after preprocessing are shown in Table 1.

Statistics	Datasets		
	FB	LastFM	ML
Number of users	1,428	1,864	2,040
Number of items	5,846	6,945	7,459
Number of ratings	64,612	82,037	374,352
Ratings per user (mean)	45.25	44.01	183.31
Ratings per user (standard deviation)	49.47	7.09	186.72
Ratings per item (mean)	11.05	11.81	50.19
Ratings per item (standard deviation)	26.28	31.70	110.38
Density	0.0077	0.0063	0.0246

Table 1: Statistics of the FB, LastFM and ML datasets, after preprocessing.

3.2.3 *Methodology*

Three different recommender systems algorithms are evaluated in this experiment: *UKNN*, *IKNN* and *WMF*, as previously described in Section 3.2.1. The neighbourhood size parameter is set to $k = 300$ for *IKNN* and $k = 80$ for *UKNN*, the number of features set to 20 and the number of iterations set to 100 for the *WMF* algorithm[2]. Each algorithm presents a user with a list of 10 recommendations. Therefore, prediction accuracy metrics are not used in the evaluation of the algorithms, as the related work [60] highlighted that they are not adequate in this scenario.

In the following, the results of a comprehensive offline evaluation of the three proposed algorithms in the various datasets previously described is presented. Here, the train and test sets are built by randomly selecting the 80% of the ratings for the train set and the remaining 20% for the test set.

3.2.4 *Results*

In this section the results of evaluating the approach proposed in Section 3.2.3 are presented. Recommendations made using the different algorithms are based on train set data only, and are evaluated in three different datasets. Each user is presented with 10 recommended items which are evaluated based on that user's test set. The results of the generated recommendations are evaluated both in terms of top-N accuracy metrics (described in Section 2.5.2.2) and metrics beyond accuracy (described in Section 2.5.3). The purpose of this experiment is to explain the relationships and tradeoffs between the evaluated properties, algorithms, and datasets.

3.2.4.1 *Popularity bias*

One of the premises of a recommender system is to help users find relevant items that are unknown to them. However, if the recommendations are biased toward popular items, for example, recommending artists like *The Beatles* or *Lady Gaga*, the user experience will be poor and the user will not be able to discover new music.

Figure 6 shows a histogram of the recommendations made for the top-100 most popular items on each dataset, i. e., how often the 100 most popular items in the dataset are found in the top-10 recommendations made by each algorithm for each user. The vertical axis represents the frequency of recommendation (normalised by the total number of users in the system)[3], and the horizontal axis shows the

2 Fixing neighbourhood sizes and the parameters of the *WMF* algorithms allows to reduce the number of free parameters in the experiment. Moreover, preliminary analysis showed that these values lead to a good performance of the algorithms in the different datasets evaluated. A more detailed analysis of the number of neighbours for the *UKNN* algorithm is presented in the next experiment.

3 To avoid the noise and highlight the general trends, the popularity functions for the *UKNN*, *IKNN* and *WMF* algorithms are approximated by a 5-degree polynomial function

100 most popular items in each dataset. For example, item 1 is the most frequently rated item in the train set.

The results show that the popularity of the *Most Popular* approach is (as expected) significantly higher than for the rest of algorithms. The popularity of this algorithm increases up to the 10^{th} most popular item, and then dramatically decreases. This effect is caused by experimental methodology, in which each user is presented with a list of ten recommendations, and users are never recommended items which are contained in their train set.

In general terms, the most popular items are most frequently recommended by the algorithms. The results show that the *UKNN* algorithm is the most biased toward making popular recommendations, while the popularity trends of the *IKNN* and *WMF* algorithms are different across datasets. However, both are biased towards making recommendations of popular items.

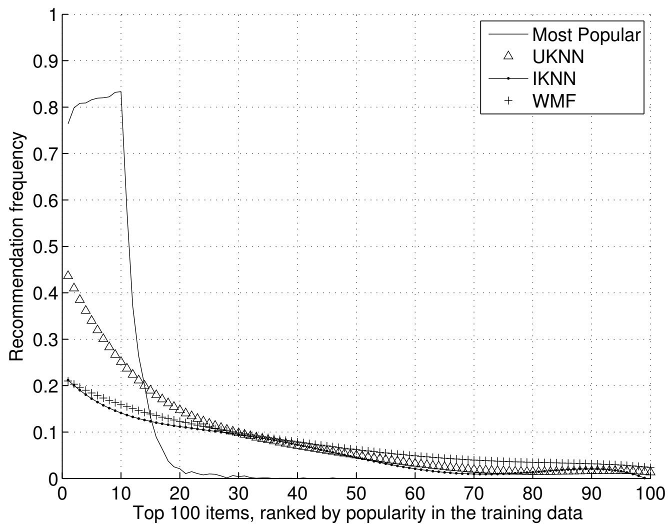
Table 2 shows the average popularity values per algorithm on each of the datasets, as per Equation 15. Here, the popularity of recommendations made by each user are calculated as the average of the popularity for the top-10 recommendations made. The results correlate with those shown in Figure 6, highlighting that the *UKNN* algorithm is the most biased toward making popular recommendations, while the the *WMF* and *IKNN* are less biased and their performances are comparable. For example, the the average popularity for the *WMF* algorithm is circa 20% less than for the *UKNN* algorithm.

Dataset	Algorithm			
	Most Popular	UKNN	IKNN	WMF
FB	0.500	0.326	0.244	0.287
LastFM	0.537	0.319	0.253	0.249
ML	0.284	0.252	0.248	0.202

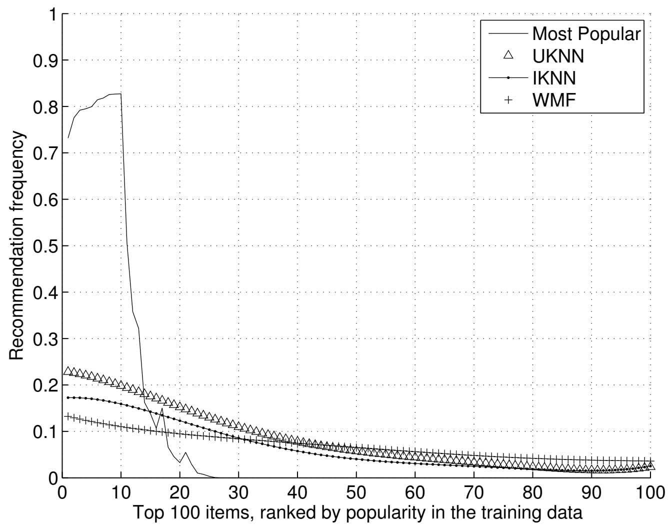
Table 2: Average popularity for the top-10 recommendations.

Slightly different trends are found for the MovieLens dataset, which is is three times more dense than the Facebook and LastFM dataset. Here, the average popularity for the *UKNN* and *IKNN* algorithms are very similar, and around 23% higher than for the *WMF* algorithm.

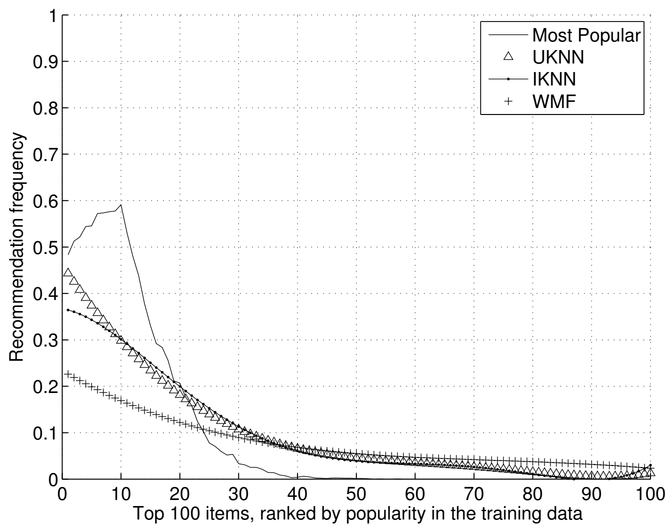
In summary, it can be seen that the *WMF* algorithm performed best in two of the three datasets evaluated, and in all cases it was less biased towards making popular recommendations than the *UKNN* algorithm. Hence, for this evaluation criteria, *WMF* appears to be the best performing approach.



(a) Facebook dataset.



(b) LastFM dataset



(c) MovieLens dataset

Figure 6: Recommendation frequency on the top-100 most popular items per algorithm.

3.2.4.2 *Other Properties*

Tables 3, 4, and 5 show the result of four of the properties evaluated on each dataset. Two types of metrics can be distinguished. First, precision (PRC), recall (RCL), F-1 and diversity are related to the list of recommendations presented to each user. However, item reach (IRE) relates to the recommendation candidates (e. g., in the *UKNN* algorithm, only items rated by the user neighbours are considered as recommendation candidates), and per-user item coverage (UIC) relates to all the items recommended by the system.

Algorithm	IRE (%)	UIC (%)	DIV	PRC	RCL	F-1
Most Popular	0.684	100.000	0.706	0.066	0.089	0.078
UKNN	4.191	19.683	0.709	0.136	0.180	0.158
IKNN	27.386	40.478	0.672	0.133	0.182	0.157
WMF	7.065	98.957	0.748	0.155	0.202	0.178

Table 3: Accuracy, diversity, coverage and item reach in the FB dataset.

Algorithm	IRE (%)	UIC (%)	DIV	PRC	RCL	F-1
Most popular	0.374	100.000	0.654	0.068	0.073	0.070
UKNN	6.177	13.453	0.722	0.162	0.177	0.170
IKNN	30.194	38.815	0.714	0.180	0.201	0.191
WMF	5.385	98.675	0.788	0.180	0.197	0.189

Table 4: Accuracy, diversity, coverage and item reach in the LastFM dataset.

Algorithm	IRE (%)	UIC (%)	DIV	PRC	RCL	F-1
Most popular	0.724	100.000	0.490	0.221	0.082	0.120
UKNN	2.386	36.149	0.529	0.296	0.112	0.162
IKNN	3.365	50.611	0.527	0.284	0.106	0.154
WMF	8.031	99.464	0.596	0.343	0.132	0.190

Table 5: Accuracy, diversity, coverage and item reach in the MovieLens dataset.

Per-user item coverage (UIC, Equation 13), measures the potential of an algorithm to select recommendation candidates. Here, the results show that the *WMF* algorithm considers almost every item as a candidate (UIC > 98%). We also see that the *UKNN* algorithm performs poorly in for this property, as by definition, only the items which are in the user's neighbourhood can be considered as recommendation candidates. *IKNN* was also seen to outperform *UKNN* in all cases.

Item reach (IRE, Equation 14) measures the proportion of items that are recommended out of the total set of items in the catalog. Here, the *IKNN* algorithm, which exploits item similarity, performs significantly better than the other algorithms (in the Facebook and LastFM datasets), covering up to 30% of the item catalog in its recommendations across users and up to 6 times more items than the *UKNN* and *WMF* algorithms.

In terms of *diversity* (Equation 12), the *WMF* algorithm performs clearly better than the rest, with its performance around 9% higher on average than the best neighbourhood-based approach, showing a gap in performance between this approach and the neighbourhood based algorithms.

The results for *top-N accuracy metrics* — precision (Equation 10), recall (Equation 9) and F-1 (Equation 11) — show that overall the *WMF* algorithm performs slightly better than the rest (for the Facebook and MovieLens datasets), while the accuracy of the *UKNN* and *IKNN* algorithms is very similar (in the Facebook and MovieLens datasets). For example, the *WMF* algorithm shows an average accuracy improvement (measured in terms of F-1) with respect to the *UKNN* algorithm of circa 13%.

As previously highlighted, some of the results are not consistent through the different datasets considered. All the properties (with the exemption of diversity and per-user item coverage) show a different order in the best performing approach in one of the three datasets evaluated. This highlights the necessity of performing the evaluation in different datasets with different statistical properties. For example, in the MovieLens dataset, which is three times more dense than the Facebook and LastFM datasets, the item reach of the *IKNN* algorithm is ~ 10 times smaller than for the MovieLens and Facebook datasets.

Hence, an analysis of the kind presented in this section should be performed for each particular domain under consideration, and the algorithm that delivers optimal performance across the metrics of interest, should be selected.

3.2.5 *Conclusions*

In this experiment an extensive evaluation of three different recommender system algorithms has been performed. We have presented the analysis of the evaluation results in terms of top-N accuracy metrics, diversity, per-user item coverage, item reach and popularity. Furthermore, the experiments were performed using three different datasets with different statistical properties (number of users, items and sparsity).

First, the analysis of popularity shows that the *UKNN* approach is inherently biased toward making popular recommendations, recreating the distribution of items seen in the train set. The results also show that the *WMF* approach is less biased toward making popular recommendations, while still generating more diverse and accurate recommendations than the rest of algorithms evaluated (in two of the three datasets evaluated).

It is also noteworthy that the *IKNN* algorithm has much higher item reach than all the other evaluated alternatives, recommending a much larger subset of items, at least for two of the three datasets evaluated. As expected, the item reach and per-user item coverage of the *UKNN* algorithm is poor in all the datasets evaluated, as the recommendations are limited to the items seen in the user's neighbourhood.

This experiment lead to interesting findings in terms of the correlation between the evaluated metrics, the algorithms and the datasets. Particularly, diversity and popularity will be further studied in the next experiment, which focuses on the two neighbourhood based algorithms presented, studying the relationships between these metrics when the number of neighbours change. Furthermore, the results will also be analysed in terms of uniqueness of the recommendation lists.

3.3 A COMPARATIVE ANALYSIS

Neighbourhood-based models, introduced in Section 2.3, are widely used and have been extensively evaluated. However, it is not completely clear how different factors such as neighbourhood selection or item popularity bias affects the overall performance of these algorithms. Here, we study the effect of neighbourhood selection for the *UKNN* algorithm, which relies on finding similar users to generate recommendations. More specifically, we experiment with different values for the number of neighbours (k) and compare the results with the *IKNN* approach, which makes recommendations based on items similar to those that the user has liked in the past.

To explore how neighbourhood size affects the performance of these algorithms, an study of recommendation accuracy, diversity and popularity is performed, comparing the results for different values of k for two different datasets. Moreover, we study the uniqueness of a recommendation list (introduced in Section 2.5.3.5), and evaluate the top-N accuracy of the common and unique set of items recommended. Performing this analysis allows us to understand what is the optimal value for k, and what are the tradeoffs between the different properties evaluated an the values of this parameter.

3.3.1 *Algorithms*

To evaluate the proposed approaches we use the *UKNN* and *IKNN* collaborative filtering algorithms, as described in the previous experiment (Section 3.2.1). Both algorithms are used in the *generate a list of recommendations* task, generating a list of top-10 recommendations per each user.

3.3.2 *Dataset*

To evaluate the proposed approach, two of the MovieLens [59] datasets are selected: the *MovieLens-100k* (ml-100k), and the *MovieLens-1M* (ml-1M), containing 100 thousand and 1 million ratings, respectively. Only users with at least 10 liked items are considered in the evaluation. Table 6 shows the statistics for both datasets, once the users that do not meet this requirement have been discarded.

Statistics	Datasets	
	ml-100k	ml-1M
Number of users	495	4,335
Number of items	1,486	3,561
Number of ratings	76,982	918,139
Ratings per user (mean)	155	211
Ratings per user (standard deviation)	108	208
Ratings per item (mean)	51	257
Ratings per item (standard deviation)	61	347
Density	0.105	0.060

Table 6: MovieLens datasets statistics after preprocessing.

3.3.3 *Methodology*

The previous section presented an exhaustive evaluation of several recommender systems algorithms, presenting an analysis of the bias of popularity, diversity, item-reach and per-user item coverage in three different datasets. In this section, two different approaches to recommendation are compared: the standard *IKNN* and *UKNN* algorithms, as described in Section 2.3. The former is evaluated with a fixed number of neighbours ($k = 300$)[4], while for the latter, neighbourhood sizes between 10 and 200, in increments of 10, are consid-

4 Fixing neighbourhood size at $k = 300$ for *IKNN* allows us to reduce the number of free parameters in the experiment. Moreover, preliminary analysis of the effect of

ered ($k \in [10, 200]$). The goal of the evaluation is to understand the effects of neighbourhood size on the performance of the user-based approach, and perform a relative comparison of the *UKNN* and *IKNN* algorithms.

Both algorithms are evaluated in the context of the *recommend a list of items* task (as described in Section 2.2.2), generating a list of top-10 recommendations per user. Therefore, prediction accuracy metrics are not used in this evaluation, since the related work has highlighted that they are not adequate for this task. To perform the offline evaluation, the test set for each user contains 10 liked items, and top-10 recommendations are made for each user. Hence, the values of *precision*, *recall* and *F-1* will be the equal in this setting.

The details of the train and test set construction are as follows:

- Both of the datasets considered in this work include users with at least 20 rated items in their profile. Only users with at least 10 liked items are included in our evaluation. A rating threshold of 4 is used to decide whether an item is liked or not; i.e. all items which received a rating of $\geqslant 4$ on a 5-point rating scale are considered liked. This particular rating threshold has been used in previous work with similar datasets [10, 72].
- The test set for each user, Te_u, is created using 10 randomly selected liked items. In this way it is guaranteed that each user's test set will have the same number of items, all of which are liked. The train set for each user, Tr_u, is created using all items which are not included in the test set.

3.3.4 *Results*

In this section, the results of the approach proposed in this experiments are presented. Recommendations made using both algorithms are based on train set data only. Each user is presented with 10 recommended items which are evaluated based on that user's test set items. The results of the generated recommendations are evaluated in terms of precision, popularity, diversity and uniqueness, and the performance is compared as the neighbourhood size, k, in the *UKNN* approach is increased.

With this experimental methodology, we expect to better understand the relationship between the evaluated properties and their dependency on neighbourhood size. Moreover, an analysis of the differences between the recommendation lists generated by the two algo-

neighbourhood on the *IKNN* algorithm did not show trends as pronounced as for the *UKNN* approach.

rithms is performed, showing how much they differ from each other by measuring their average uniqueness.

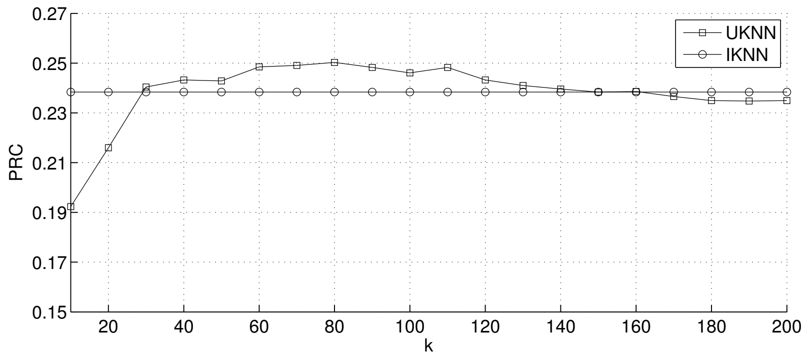
Figures 7 and 8 show the results of the four properties evaluated for both datasets. Overall, precision, popularity, diversity and uniqueness show very similar trends for both datasets. To begin, precision results are first discussed, followed by the remaining properties, focusing on the relationship between each metric and precision.

The results for *precision* (Equation 10) are shown in Figures 7a and 8a. They show that the overall precision of UKNN increases up to $k \sim 90$, and thereafter declines for the ml-100k dataset but remains relatively constant for the ml-1M dataset. However, the precision of *UKNN* exceeds that of *IKNN* at much smaller neighbourhood sizes — at $k = 30$ and at $k = 10$ for the ml-100k and ml-1M datasets, respectively.

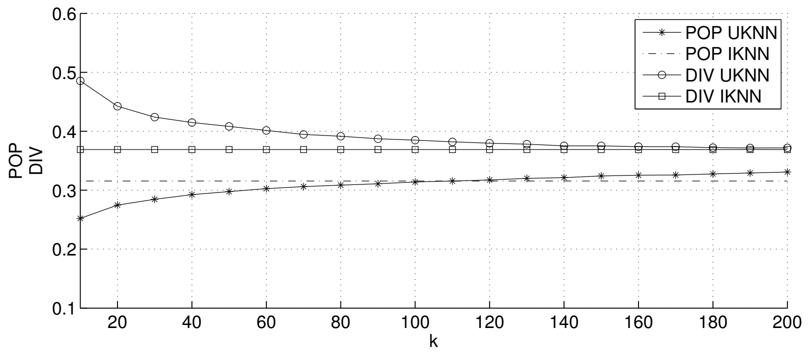
The results for recommendation *popularity* (Equation 15) and *diversity* (Equation 12) are shown in Figures 7b and 8b. For *UKNN* the average popularity of recommended items increases with k. Thus, as expected, there is a high bias toward popularity at larger sizes of k, and a corresponding decrease in recommendation diversity. In the limit, if every user was considered a neighbour, the recommendations would resemble the distribution of item ratings in the dataset, and thus only the most popular items would be recommended.

The average recommendation diversity decreases as k increases for the *UKNN* algorithm, and approaches that of the *IKNN* algorithm at $k = 200$. However, this reduction in diversity for *UKNN* is not compensated by a corresponding gain in precision at larger neighbourhood sizes. As previously described, the *UKNN* algorithm attains its maximum accuracy at $k \sim 90$, yet the loss in diversity is a monotonically decreasing function. We can conclude that there is a high inverse correlation between popularity and diversity, and both suffer at larger values of k; i.e. recommendation lists contain mainly popular items which have a high degree of similarity to each other.

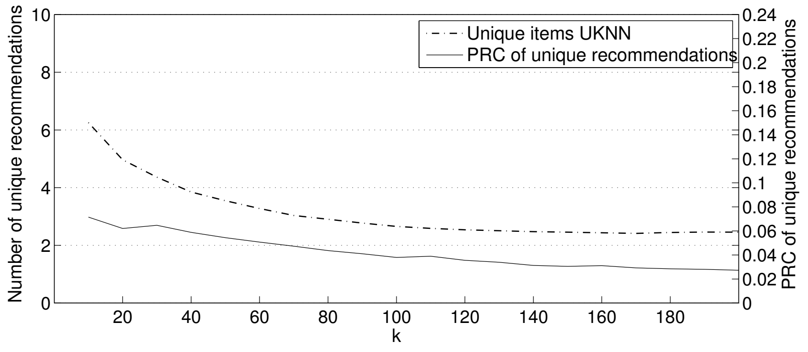
Finally, Figures 7c, 7d, 8c and 8d show the average numbers of recommendations which are *unique* to the *UKNN* algorithm and those which are *common* to both algorithms, and the corresponding precision results over the unique and common recommended items. The results show that number of unique recommendations decreases significantly as k increases. For example, at $k = 100$, on average more than 7 of the 10 recommended items produced by both algorithms are the same in the case of the ml-100k dataset. It is also apparent that recommendation accuracy is largely due to the common items which are recommended by both algorithms. Consider again the ml-100k dataset at $k = 100$, where the precision of unique and common items is approximately 0.03 and 0.21, respectively. However, it should be



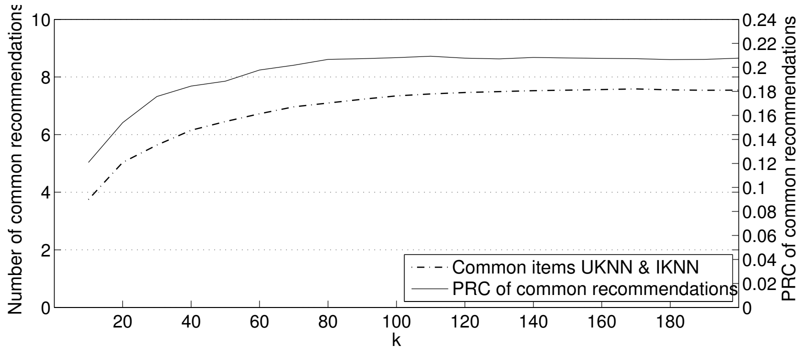
(a) Precision vs neighbourhood size.



(b) Popularity and diversity vs neighbourhood size.

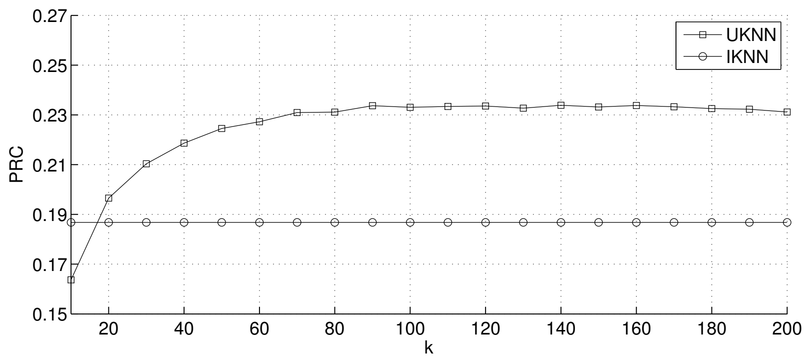


(c) Precision of unique *UKNN* recommendations vs neighbourhood size.

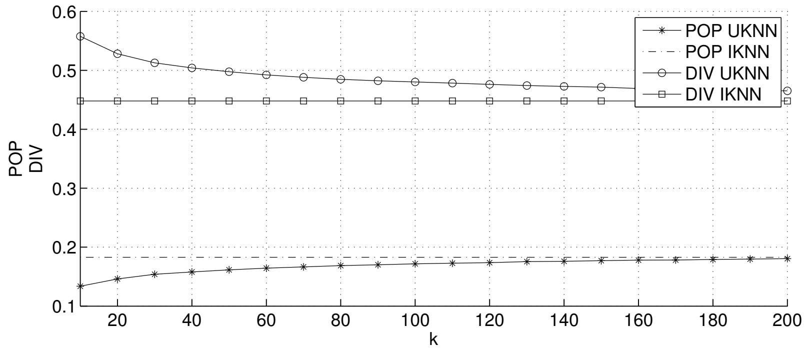


(d) Precision of common recommended items vs neighbourhood size.

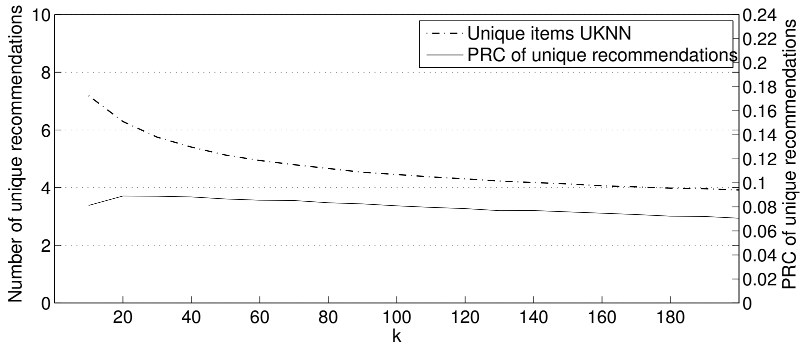
Figure 7: Evolution with neighbourhood size, ml-100k dataset.



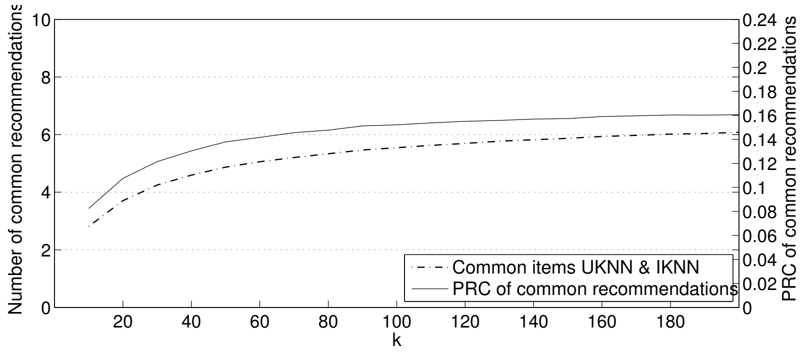
(a) Precision vs neighbourhood size.



(b) Popularity and diversity vs neighbourhood size.



(c) Precision of unique *UKNN* recommendations vs neighbourhood size.



(d) Precision of common recommended items vs neighbourhood size.

Figure 8: Evolution with neighbourhood size, ml-1M dataset.

noted that the unique items recommended by the *UKNN* algorithm are not necessarily irrelevant — although these items are not present in the test set (which is used as the ground truth), they may still represent useful recommendations to users. Moreover, at smaller values of k, items recommended by *UKNN* are less popular, and such items – by definition – are less likely to appear in test sets. This is a well-known limitation of the standard approach to the 'offline' precision evaluation of recommender systems as performed in this work, and it can only be addressed by live user trials, which is left to future work.

In summary, the experimental results indicate the bias inherent in the *UKNN* algorithm toward popular recommendations at larger neighbourhood sizes, which is more evident in the ml-100k dataset. However, the loss of diversity and bias toward popular recommendations at larger values of k is not compensated by a gain in precision. From the results, we can also see that the uniqueness of the recommendation algorithms also depends on k, and that smaller neighbourhood sizes lead to more unique, less popular, and more diverse recommendations.

3.3.5 *Conclusions*

This Section presented an evaluation of two very popular collaborative recommender systems, the *UKNN* and *IKNN* algorithms. We have described the analysis of the evaluation results, which focuses on the effect of neighbourhood size on the relative performance of the algorithms. The results are explained according to four main properties: precision, diversity, popularity, and uniqueness of the recommendations.

Experiments performed using two different datasets show that optimising for accuracy in the user-based approach leads to a poor performance in terms of diversity, a higher bias toward popularity, and less unique recommendations. Moreover, choosing smaller numbers of neighbours for *UKNN* leads to more diverse and less popular recommendations, at the cost of a relatively small decrease in accuracy. The differences between the two approaches have also been analysed in terms of unique recommendation capabilities, showing that when the number of neighbours in the *UKNN* approach is large, both algorithms tend to produce very similar recommendations (up to 70%).

Recommender systems have proven to be an indispensable tool for guiding the user in the information finding process. Many websites rely on these kind of systems to help the user find relevant products and increase sales. To understand and predict the behaviour and performance of these systems, it is important to evaluate the different properties which contribute to overall system performance. Notwithstanding that the evaluation process is key in order to determine system performance, there is no common agreement on how best the evaluation process should be approached, which properties should be evaluated and which metrics should be used.

In the first experiment (Section 3.2) an evaluation of several collaborative filtering algorithms was carried out using three different datasets. Popularity, diversity, item reach, per-user item coverage and uniqueness were evaluated together with top-N accuracy metrics. The results show that to understand the performance of a recommender system, all its properties have to be evaluated, and not only its accuracy.

This experiment showed that the evaluated matrix factorisation approach *(WMF)* leads to more accurate and diverse results while being less biased toward popular recommendations. Also, the results show that the *IKNN* has a better item reach than the other evaluated approaches (in two of the three datasets evaluated), potentially being able to recommend more items.

In Section 3.3, an evaluation comparing two very popular collaborative recommender systems was studied, focusing on the effect of neighbourhood size on the relative performance of the algorithms. The results show that for smaller neighbourhood sizes, the *UKNN* algorithm produces more diverse and less popular recommendations with a small tradeoff in accuracy.

The results presented in this chapter show interesting findings in terms of the correlation between accuracy, popularity, diversity and coverage. A further study (left for future work) on how the neighbour selection affects these properties can be performed by, for example, taking in account different types of profiles (heavy users, light users).

Other properties which are important for recommender system performance can also be considered. For example, a further study could consider the introduction of new metrics to evaluate novelty and serendipity, and to perform a user study to investigate the correlation between the various properties and real user behaviour.

The first part of this thesis presented a general approach to recommender systems and content discovery, focusing on the evaluation of a family of recommender systems (collaborative filtering recommender algorithms). The second part of this thesis presents a different perspective on content discovery. The particular problem of classification applied to the music domain is presented in Chapter 4. Later the proposed approach for music mood and genre classification is evaluated in Chapter 5.

Part II

MUSIC MOOD AND GENRE CLASSIFICATION

Music classification helps users navigate and organise the large collections of music available on the Internet by enabling automatic playlist creation and personalised recommendations. For example, by analysing song's lyrics and classifying its mood or genre, recommendations for songs which talk about similar topics or express similar opinions can be obtained, facilitating more serendipitous and novel recommendations.

In this thesis, the problem of music classification is addressed by exploiting information extracted from lyrics alone. In contrast to previous work in this area, in which contradictory conclusions were reached based on experiments carried out using different datasets and evaluation methodologies, a large freely-available dataset is used to compare the performance of classifiers trained on a number of different feature sets.

4 BACKGROUND ON CLASSIFICATION

4.1 INTRODUCTION

Large collections of different types of products are constantly created and accessed on the Internet. For example, Spotify[1] gives users access to dozens of millions of songs[2], while the Netflix[3] catalog includes thousands of movies. Thus, there is a need for tools that can effectively organise data, and aid browsing through these collections. In the physical counterpart, the sorting and browsing of collections is usually done manually, i.e., sorting books in a library by genre, topic or author is usually done by the librarian. However, online collections are often much larger, making it impossible to manually organise. Therefore, automatic classification approaches have become popular when trying to solve this particular problem.

The classification problem can be tackled in two different ways according to the information needs or constraints. In unsupervised learning [36], the concepts to be learned are derived from the un-labeled data in the learning process, while in supervised learning [90], a model learned on available training labeled data is used to classify unseen examples. These type of approaches are very popular for a wide variety of problems, such as classification of reviews for product recommendation [41, 40], or short text categorisation on platforms like twitter [46].

This work focuses on music *genre* and *mood* classification. In this area, most of the work uses a supervised approach in which the features can be based either on audio (harmonics, dynamics or rhythm) as in [50, 97], lyrics (where features are usually represented using a vector space model) [42, 177] or on a combination of both [105, 95].

The rest of this chapter is organised as follows. First, Section 4.2 presents an overview of the classification problem, including the basic concepts and approaches. Then, Section 4.3 describes the main algorithms for supervised classification. Section 4.4 introduces the vector space model used for document classification, describing different term weighting approaches. Section 4.5 introduces the problem of music classification, including the related work on representation of

1 http://www.spotify.com
2 http://press.spotify.com/no/information/
3 http://www.netflix.com

genre (Section 4.5.1) and mood (Section 4.5.2). An overview of music classification using audio features is described in Section 4.5.3 and finally an overview of music classification using lyrics features is presented in Section 4.5.4.

4.2 OVERVIEW ON CLASSIFICATION

As the amount of content in the Internet grows, classification has become a very important part of the machine learning field, helping to organise large amounts of information and navigate through large collections of data. Some examples of recent works include categorisation of user-generated content, such as tweets [46], or social tags associated with music [45]. There are two main approaches to classification: *supervised learning*, which uses labeled data in the classification process, and *unsupervised learning* where the data does not have to be labeled. In this type of approaches, the instances are usually represented in a large dimensional space defined by a number of features. In the case of text documents, each separate term is usually considered as a feature. For example, when classifying images ,the features are usually values describing the image, such as colour histogram, contrast, brightness, texture, etc. The following elements, described in [71], are part of every classification scheme:

- A *concept*, noted as c, is defined as the object to be learned. In this case, a class, or a cluster into which an instance is classified.
- An *instance* or *document* is each one of the individual examples of the concepts to be classified. Each document is noted as d and a set of m documents (collection) is denoted as D. Thus $D = \{d_1, d_2, \cdots, d_m\}$.
- Each document is represented by a fixed number of y features, denoted as t. Thus, a document $d = \{t_1, t_2, \cdots, t_y\}$.
- Each of the k clusters or classes are denoted as $S = \{S_1, S_2, \cdots, S_k\}$, and $n_1, n_2, \cdots n_K$ is the size of clusters.

4.2.1 *Unsupervised Learning*

In unsupervised learning or clustering [36, 6], the data is partitioned without using a predefined class label. Thus, it is the algorithm which decides how to best partition the data in a (often) predefined number of clusters. The algorithm receives an input of $d_1, d_2, \cdots, d_m$ instances, and builds a representation of the input that is used to predict future inputs by finding patterns in the data. In this approach the learning is not supervised by labeled training examples, and consequently, building a model from the data is more difficult. Moreover, as there are no predefined labels into which classify the instances, the evaluation of this approach is known to be more complicated than supervised learning.

Several clustering approaches have been proposed, those are usually divided according to the way the clusters are generated. A brief description of some of the most popular approaches is presented below.

4.2.1.1 *Partitional Clustering*

Partitional clustering methods decompose the data into a set of k flat disjoint clusters $S = \{S_1, S_2, \cdots, S_k\}$. The most popular partitional clustering algorithm is *standard k-means* [2], which employs an iterative process to produce *k* clusters by locally minimising the distortion between the data objects and a set of *k* cluster centroids, measured in terms of euclidean distance.

4.2.1.2 *Fuzzy Clustering*

Fuzzy clustering techniques [178] allow instances to belong to different clusters, and map the membership of each instance by a probabilistic weight. The Fuzzy *c-means* algorithm [44] is a generalisation of the *standard k-means* algorithm described above. Here, each instance can belong to different clusters, and the membership is measured by a probabilistic weight, usually stored in a confusion matrix. Another clustering algorithm, the *expectation maximisation algorithm* [119] uses a model-based approach to identify groups in data.

4.2.1.3 *Hierarchical Clustering*

Hierarchical clustering methods [75, 147] build a hierarchy of concepts. They rely on building a set of nested clusters that is often arranged in a form of a tree structure. These methods are often used in document clustering, since they mimic the nature of documents [36], which often contain more than one topic. For example, an article can be classified in the "tennis" cluster, within the "sports" cluster.

There are two main types of hierarchical clustering approaches. *Agglomerative Clustering* [182] applies a bottom-up strategy where on each step the most similar pair of clusters are merged. *Divisive clustering* [38] begins with a single cluster, and on each step a cluster is divided in two sub-clusters.

4.2.2 *Supervised Learning*

Supervised learning [179, 106, 157, 71] relies on annotated data to partition documents into a predefined number of classes. A supervised classification algorithm learns a model based on labeled training examples, and then uses this model to predict the output for unseen data. The classification task aims to assign each document to one (single label) or more (multi label) predefined classes. The typical scenario includes a set of m documents which are annotated according to k classes. Those documents are then fed into a supervised classification algorithm, that automatically builds classification rules by exploiting the characteristics of each class.

In this thesis, the focus on supervised learning for document classification. Thus, in the next section, several supervised learning algorithms are described in detail.

4.3 ALGORITHMS FOR SUPERVISED CLASSIFICATION

In this section we describe a number of supervised learning classification algorithms which have been shown to perform well in several scenarios, including text document classification.

4.3.1 *Naïve-Bayes*

Naïve-Bayes [71] is a fast and accurate technique to perform classification based on the features that occurs in a document. It is often used for text classification [106], where a document is represented by a vector in the y-dimensional term space. The Naïve-Bayes algorithm calculates the probability for a document to belong to a class, based on the distribution of each feature across all the classes, selecting the class which is more likely to have generated the document.

This technique does not take into account the weight that the feature has in each document (for example, number of times a term occurs in a document). It assumes independence between all the features within each class (what is known as the Naïve-Bayes assumption) and also assumes that the distribution of features across documents follows a normal distribution. Even when the independence assumption is not true for most of the cases, Naïve-Bayes classifiers perform very well, and because of the independence property, the learning process is simple even when the number of documents in the collection is large.

The *Naïve-Bayes multinomial* algorithm [79] exploits the feature weights on each document when estimating the probabilities for each class, something particularly interesting for the text classification problem. This variation still preserves the assumptions of independence over the terms, but its performance is better than the original Naïve-Bayes algorithm, specially for larger collections [106].

4.3.2 *Decision Trees*

Decision trees [71, 136, 137] tackle the classification problem by doing a non-incremental learning from examples. The algorithm is presented with a set of cases relevant to the task, and develops a decision tree. Each node in the tree will test a particular feature, and following a top-down approach each instance will be classified.

In this approach, all instances are considered independent. To classify a new unknown instance, the instance is routed in the tree according to its features, which are tested on each node. When a leaf is reached, the instance is classified with the class assigned to that leaf.

The ID3 algorithm [138] and C4.5 algorithm [139] are very popular examples of decision trees that choose the order in which features are considered, by measuring its information gain. In the ID3 algorithm, once a feature is selected for a node, the data is split according to the selected feature to produce subsets of the data creating several branches. This process is repeated for each feature until the tree is fully built. Thus, each non-terminal node represents the feature into which the data is to be split, and each terminal node (leaf) represents a class.

4.3.3 *Support Vector Machines*

Support Vector Machines (SVM) [34, 74, 71] work by finding a special type of linear model (the maximum margin hyperplane), which is the one that separates all the classes with the smallest true error. The instances which are closest to the maximum margin hyperplane are called support vectors. The algorithm uses linear models to implement nonlinear class boundaries, by transforming the linear input into a high dimensional non-linear space in which a linear decision surface is constructed. The resulting linear space can represent a non-linear decision boundary in the original space.

Support Vector Machines learn a threshold function. Thus, by changing the learning function (kernel) they can learn a linear function, or learn polynomial classifiers, radial basic function networks or sigmoid neural nets. One of the main properties of SVMs is they learn the complexity of the hypothesis based on the margin of the separation, and not in the number of features in the data. Thus, it can be applied to high dimensionality data maintaining a low computational cost.

4.3.4 *Ensamble Classifiers*

Sometimes using a single classifier can not increase the performance of a classification system above a certain threshold, as it has exploited all the knowledge available in its instances. In those cases, the use of *ensemble classifiers* [71] can help to improve the classification performance over a single model. Algorithms such as bagging with costs [132] or boosting [140] have been recently developed, showing very good performances. However, the gain in performance of this algorithms comes at a cost of not being able to easily explain the reasons for the performance increase when dozens or even hundreds of results are combined.

4.3.4.1 *Early Fusion*

In an *early fusion* approach [162], the features are combined into a single representation, often by performing concatenation of different feature groups or subset of features. Once the features are combined, the classification is done in the same way as with the original instances. This approach has been widely used to merge different kind of features in a common feature space, for example combining audio and lyrics features in music classification [63, 122].

4.3.4.2 *Late Fusion*

In *late fusion*, the outputs (decisions) of different models are combined producing a single prediction. The most common approach is to calculate the output by majority voting [149]. However, more sophisticated approaches have been proposed. In *bagging* [21] each model receives the same weight. *Boosting* [140] is an evolution of *bagging* where each model receives a different weight. Another possibility is to use the categeories' probabilities estimates generated by each classifier [92], and calculate the average between the possible outcomes, selecting the maximum value.

Random Forest [22] is a particular example of ensemble classifiers, which combines random decision trees with bagging, achieving very high accuracy values. Each tree decides the class in which an instance should be classified, by using a random selection of features to split each node, and the final decision is made by consensus. This type of approach is known to perform well when the number of trees in the collection is large. The classification power depends on the accuracy of each tree and the correlation between them.

4.4 THE VECTOR SPACE MODEL

The most common approach to represent a document is by a vector in an y-dimensional term space, defined by the terms that occur in the document $d = \{t_1, t_2, \cdots, t_y\}$. The vector can either be a binary vector, or take in account the number of times each word occurs in the instance. The order in which terms occur is not relevant for this type of representation, and only the presence and frequency for each particular term is usually recored. This representation model is usually known as the *Vector Space Model (VSM)*.

As the order of the terms in the documents is lost, the semantic information is also lost. For example, in the a VSM representation, the document d_1 =*"Anna likes watching action films but Bob does not"* will have the same representation as the document d_2 =*"Bob likes watching action films but Ana does not"*, even when their meanings are opposite.

This model is widely used in the Information Retrieval (IR) field [31], where the goal is to retrieve the best possible subset of documents for a given query. To calculate the best matches, the similarity between the query and the documents is computed using a similarity metric such as cosine similarity (Equation 17).

$$\cos(d_i, d_j) = \frac{d_i \cdot d_j}{|d_i||d_j|} \tag{17}$$

Equation 17 shows the cosine similarity between two documents, d_i and d_j. Here the similarity is calculated as the cosine of the angle between vectors, as shown in Figure 9. Thus, if the vectors (documents) are orthogonal, they do not share any common terms, and the cosine value is 0. If the angle between the two vectors is 0^o, the documents are equivalent and the cosine value is 1.

Figure 9 shows the representation of two documents in a two-dimensional space, i.e., each document is represented by two terms. Here, $\theta_2 - \theta_1$ represents the angle between documents d_1 and d_2.

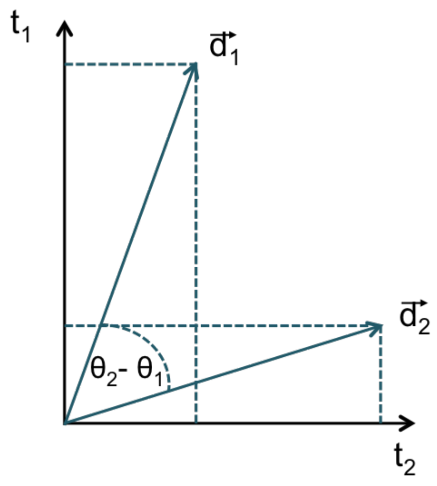


Figure 9: Cosine similarity between two documents.

4.4.1 *Preprocessing*

Once the collection of documents is obtained, a preprocessing stage is carried out. This process usually includes the selection of a language, stop-word removal, and sometimes term stemming.

- *Language selection* is usually the first preprocessing step, where the decision about the language (monolingual) or languages (multilingual) used is taken. The most common and simple approach is to build a monolingual system.
- *Stop-word removal* [160, 176] is the process in which all the words which have been agreed on not being useful are deleted from the documents. The most common approach is to use a stop-word dictionary, which contains mainly function words which occur very frequently, and have no information value for the task.
- *Stemming* [126, 68] is the process of reducing terms into their lexical root. This process has the advantage of reducing the dimensionality of the space, while unifying very similar terms. For example, if the terms *love, loved, beloved* and *loves* are stemmed using the SnowBall porter stemmer algorithm [134], the stemmed representation of all of them will be the root term *love*, which will increase the number of matches for a query that uses any of these terms in a retrieval system. However, its main disadvantage is that when stemming is performed, the number of terms is reduced, and part of the information is lost.

  According to the system needs, this stage can also include the conversion to lowercase of all words, removal of punctuation signs, removal of non alphabetical characters, etc.

4.4.2 *Term Weighting Approaches*

The basic term representation in the vector space model considers a binary vector to represent each document $d = \{t_1, t_2, \cdots, t_y\}$, indicating the presence or absence for each of the y terms. However, several term weighting approaches have been developed to extract more and meaningful information from the document collection [103], and to better understand how particular terms in documents are related to queries. In what follows, we will introduce several term weighing schemes, alongside an example document collection with toy documents preprocessed as described in Section 4.4.1. However, for purposes of illustration, stemming is not considered.

d_1 = *"car, car, best, insurance, insurance, insurance, risk, risk"*.

d_2 = *"auto, engine, race, race, race, risk, pilot, pilot"*.

d_3 = *"auto, best, car, car, race, risk, pilot"*.

4.4.2.1 *Binary weighting*

This is the most simple approach, in which the document is described solely by the presence or absence of the terms composing it. Table 7 shows a term-document matrix representation for the example presented above. A value of 1 represents words which are present and a value of 0 means the term is not present in that document.

	auto	best	car	engine	insurance	race	risk	pilot
d_1	0	1	1	0	1	0	1	0
d_2	1	0	0	1	0	1	1	1
d_3	1	1	1	0	0	1	1	1

Table 7: Term-document matrix for the binary representation of documents. Each document is characterised by the presence or absence of a term.

4.4.2.2 *Term Frequency*

Term Frequency (tf) accounts for the number of times a term t occurs in the document d. The main idea behind this weighting is the fact that when a term occurs more frequently in a document, it is more relevant than a term which occurs very few times. For example, the term-document matrix presented in Table 8 shows how the term "insurance" occurs 3 times in d_1, while the term "best" occurs only once. To take into account the document length (since relevant terms are likely to occur more in longer documents), several normalisation metrics have been proposed. The *maximum tf normalisation* [31], shown in Equation 18, introduces a smoothing factor, which usually takes the value $a = 0.4$.

$$ntf_{t,d} = a + (1-a)\frac{tf_{t,d}}{tf_{max,d}}, \tag{18}$$

where $tf_{t,d}$ is the term frequency for term t in document d, $tf_{max,d}$ is the maximum term frequency for all terms in document d, and a is the smoothing factor.

	auto	best	car	engine	insurance	race	risk	pilot
d_1	0	1	2	0	3	0	2	0
d_2	1	0	0	1	0	3	1	2
d_3	1	1	2	0	0	1	1	1

Table 8: Term-document matrix for the *tf* (not normalised) representation of the documents. In this case the number of times a term occurs in the document is taken in account.

4.4.2.3 *Term Frequency - Inverse Document Frequency*

Inverse document frequency (idf) is a measure of whether the term is common or rare across all documents. It is defined as the logarithm of the total number of documents (collection size) m, divided by the document frequency.

$$idf_{t,D} = \log\frac{m}{df_{t,D}}, \tag{19}$$

where the document frequency, $df_{t,D}$, is the number of documents in the collection D that contains the term t, and m is the number of documents in the collection.

	auto	best	car	engine	insurance	race	risk	pilot
idf	0.17	0.17	0.17	0.47	0.47	0.17	0	0.17

Table 9: *idf* values of terms in the example.

In Table 9, the inverse document frequency (*idf*) weighting was applied to each term in the collection. A term which occurs in all the documents of the collection will have an *idf* value of 0. Also, in this scenario a term which occurs in two documents has an *idf* value of $\log\frac{3}{2}$, while a term which occurs in a single document will have an *idf* value of $\log \frac{3}{1}$. Thus, more weight is given to terms which are rare in the collection.

Term frequency - inverse document frequency (tf-idf) [142] is a combination of term frequency and inverse document frequency. The *tf-idf* weight for a term t in document d is calculated as the product of $tf_{t,d}$ and $idf_{t,D}$. Thus, a document d is represented as in Equation 20. This metric was engineered to fit a classic information retrieval scenario, giving higher weights to terms which are rare in the collection, but which occur frequently in certain documents.

$$d = \{tf_{1,d} \log \frac{m}{df_{1,D}}, tf_{2,d} \log \frac{m}{df_{2,D}}, \cdots, tf_{y,d} \log \frac{m}{df_{y,D}}\}, \tag{20}$$

where $tf_{i,d}$ is the frequency of the i_{th} term in the document d and $df_{i,D}$ is the number of documents that contain the i_{th} term.

	auto	best	car	engine	insurance	race	risk	pilot
d_1	0	0.17	0.34	0	1.41	0	0	0
d_2	0.17	0	0	0.47	0	0.51	0	0.34
d_3	0.17	0.17	0.34	0	0	0.17	0	0.17

Table 10: Term document matrix for the *tf-idf* representation of the documents in the example collection.

Table 10 shows a term-document matrix for the example considered. Here *tf-idf* weighting is applied, by multiplying the *tf* values shown in Table 8 by the *idf* values shown in Table 9. For example, the term "insurance" occurs frequently in document d_1 but rarely in the collection (it occurs in a single document). Thus, this term is important in that particular document, as its *tf-idf* weight reflects.

4.4.2.4 *Inverse Class Frequency*

Inverse class frequency (icf) [180][4] is a modified version of the *idf* metric designed for the classification task. It measures how common (or rare) a term is across the different classes. A term which only occurs in a small subset of classes will be valuable for classifying instances of those classes, while a term that occurs in every class will not be helpful. The *icf* value is obtained by dividing the total number

4 In [180], this metric is referred to as *tf*. However, it is calculated following the description presented here.

of classes by the number of classes containing the term, and then computing the logarithm, as in Equation 21.

$$\mathrm{icf}_{t,C} = \log \frac{k}{\mathrm{cf}_{t,D}} + 1\,, \tag{21}$$

where $\mathrm{cf}_{t,D}$ is defined as the number of classes in the collection D that contain the term t, and k is the number of classes in the collection.

Normalised inverse class frequency (nicf), defined in Equation 22, is our proposed modification to the *icf* weight. This new metric takes in account the proportion of documents within each class in which a term occurs. A term might occur in a large number of classes, but the distribution within classes may not be the same. For example, a term which occurs in many documents in a particular class, and in a very small number of documents in other classes will have a low *icf* weight, while intuitively, its weight for documents in the class in which it occurs frequently should be high. Thus, *nicf* solves this by measuring the proportion of documents in each class in which a term occurs. Similarly to *idf*, this metric can be applied in conjunction with *tf*, as in Equation 23.

$$\mathrm{nicf}_{t,c_i,C} = \mathrm{icf}_{t,C} \cdot \frac{\mathrm{kf}_{t,c_i}}{n_i}\,, \tag{22}$$

where kf_{t,c_i} is defined as the number of documents in the class c_i that contain the term t, and n_i is the number of documents in the class c_i.

$$\text{tf-nicf} = \mathrm{tf}_{t,D} \cdot \mathrm{nicf}_{t,c_i,C} \tag{23}$$

4.5 MUSIC CLASSIFICATION

The growth of different streaming platforms such as Spotify or Rdio[5] allows easy access to dozens of millions of songs. In this scenario the user often finds herself overwhelmed with choice. To help the user cope with the large amount of available information, several solutions have been proposed, mostly in terms of personalised recommender systems [94, 93, 1] that guide the user when navigating through large collections of songs, generate automatic playlist, etc. Recommender systems often rely on music classification to find new and interesting items for the user to enjoy. For example, Figure 10 shows several

5 http://www.rd.io

playlists on the Spotify desktop user interface, classified by genres and mood.

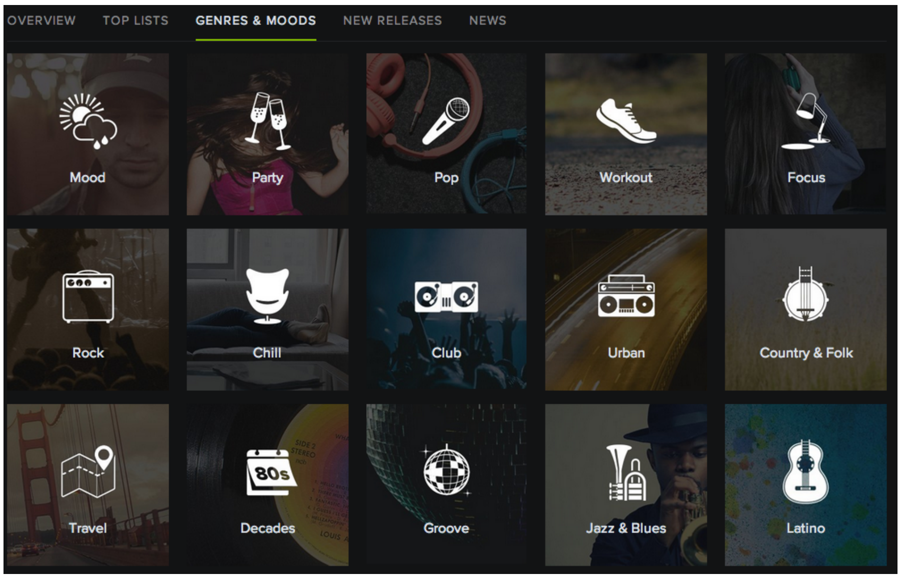


Figure 10: Spotify lets the user browse for music using genres and moods such as *party, latino or chill*.

Two popular ways to classify music is by *genre* [105, 107] and *mood* [116, 76]. However, moods in music are complex to infer and people perceive them differently [165]. Moreover some songs, like *Bohemian Rhapsody, by Queen*[6], express different moods in different parts of the song. Most of the proposed sentiment or mood ontologies rely on models developed by research in the psychology field [148, 168], which have been later adapted to the music classification task [43]. In contrary, genres are more extended and widely used for music classification in several scenarios [125, 4], they are the natural way into which users sort music, and even record stores classify their items according to genres. However, there is no agreement on the genre taxonomy to use, and often the same words are used to describe different genres in different taxonomies [125].

4.5.1 *Genre Representation*

Genre is one of the most popular and distinctive ways to describe music, both by producers and consumers. Aucouturier et al. [4] present a review paper where they highlight that genre is a crucial part of the metadata, and that the definition of genre is not trivial. The authors study three different approaches to extract musical genres (*manual*, *prescriptive* and *emerging* approaches), discussing the strengths and weakness of each approach. *Manual* approaches rely on music

6 http://open.spotify.com/track/1fNo4jzUtg9EC0yyHcZY5j

experts to represent the music, *prescriptive* approaches model new music based on a existing genre model, and *emerging* approaches use objective similarity measures to build a genre taxonomy. The authors conclude that *prescriptive* approaches (those based on audio-features) can only distinguish genres when a small taxonomy and very different genres are considered. They also highlight that each approach uses a different conception of genre.

Pacet et al. [125] study three different genre taxonomies from Internet sites: allmusic.com, amazon.com and mp3.com. They find no consensus in the genre names or structures of the taxonomies studied, making them semantically inconsistent.

In this work we the consider classification in the context of six distinctive and popular music genres based on features extracted from lyrics only. Moreover, we use manually generated labels from LastFM[7] as the ground truth for genres.

4.5.2 *Mood Representation*

Music moods are difficult to infer: people perceive them differently [165] and they are culture dependent [89]. Most of the proposed mood ontologies rely on models developed in the psychology field — *Russells model of affect* [148] being one of the most widely used. This model, represented in Figure 11, is based on the evidence that the affective dimensions are built in a *highly systematic fashion*, instead of being independent dimensions. Each mood $m \in M$ is mapped onto a two-dimensional space defined by valence v (which measures the good–bad dimension of sentiment) and arousal a (which measures the active–passive dimension of sentiment). Therefore, each mood $m \in M = \{v, a\}$ can be represented by a vector in the two-dimensional valence-arousal space.

In Figure 11, the horizontal axis represents the valence dimension, and the vertical axis represents the arousal dimension. For example, in this representation the mood "calm" has a low value in the arousal dimension and a high value in the valence dimension. The mood "enthusiastic" has a high value in both dimensions.

Recently, Russell's theoretical model has been adapted to the music classification problem [61] using social tags to infer the classes. However, it is also important to highlight the lack of consensus on the names for concepts to be learned; some authors refer to *moods* while others refer to *sentiment* to define the same concept. There is also little consensus when deciding the mood classes to consider, which makes comparing research output in this area problematic.

7 http://www.last.fm

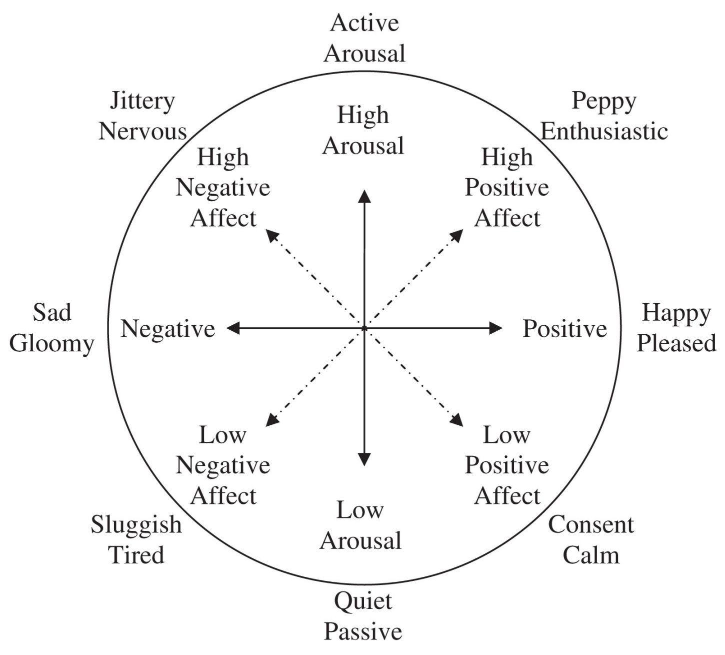


Figure 11: A representation of moods in Russell's circumplex model.

Some research has been carried out in the field of human perception of music. In [15], a study of the music perception shows that information coming from lyrics (semantic), and from the tunes (harmonic), are processed independently by the brain, even when those two types of information are presented together and are very related to each other. The authors claim that the results are consistent with the modular organisation of the human cognitive system, and raise the question on wether the classification rules learned by using lyrics features and those learned by using audio features should be complementary when classifying music moods.

In this work, moods are infered from social tags, as proposed in [61, 43]. Then, those moods are grouped into four classes following Russell's model of affect. This facilitates the creation of a large dataset in which the mood groups are clearly defined.

4.5.3 *Classification Using Audio Features*

This work focuses on an analysis of lyrics for music classification, yet far more research has been carried out classifying music using features extracted from the audio. Here, some well known approaches that use audio features for music classification are described, together with the different feature representations used in music mood and genre classification.

Feature group	Features
Dynamics	RMS energy
Timbre	MFCCs, spectral shape, spectral contrast
Harmony	Roughness, harmonic change, key clarity, majorness
Register	Chromagram, chroma centroid and deviation
Rhythm	Rhythm strength, regularity, tempo, beat histograms
Articulation	Event density, attack slope, attack time

Table 11: A taxonomy of the most popular audio features for music classification.

Table 11, described in [81], shows the most popular features for music mood and genre classification.

Genre classification has been studied in [97], where Daubechies Waevelet Coefficients Histograms (DWCHs) are used as a feature extraction method, which significantly improves the accuracy of genre classification. This approach uses timbral texture features (MFCCs, spectral centroid, rolox and flux, etc.), rhythmic content features, and pitch content features. The results show that the performance of timbre features is better than pitch or beat features, and the use of DWCHs further improves the accuracy on all the evaluated methods, specially for SVM classifiers.

Lidy et al. [98] study different psycho-acoustic transformations for the task of genre classification. They introduce two new rhythm features (rhythm histogram features and statistical spectrum descriptors) which improve the original results using rhythm features by up to 16.4%.

Multi label mood classification using acoustic features is described by Thohidis et al. [170]. Here, the authors explore rhythm features (beats per minute, beats histograms), timbre features (MFCCs, FFT) to classify music into moods following the Tellegen-Watson-Clark model [168].

A study on spectral features (spectrum centroid, brightness, skewness, spectral entropy, MFCC, among others) for music mood classification [164] shows that when these features are used in conjunction with an SVM classifier with a polynomial kernel, the results outperform those in which features based on rhythm, dynamics (RMS energy, slope, attack, low energy) and harmony are used.

4.5.4 *Classification Using Lyrics Features*

The classification of music mood and genre using lyrics has not been explored as much as other approaches which use cultural features [81], or audio features, as described in Section 4.5.3. However, some interesting work in the topic has been published in recent times, sometimes obtaining similar or even better classification results than with audio features by using lyrics features alone, or in combination with other sources of information.

Hu et al. [65] propose a method for detecting moods for 500 manually labeled Chinese songs using lyrics. The approach maps moods into a two-dimensional space of valence and arousal following Russell's model of affect, and uses a translated and expanded version of the ANEW (Affective Norms for English Words) dataset [19]. Then a fuzzy clustering method is used to group the lyrics' sentences according to their moods, and to extract one prominent mood from each song. The results show that lyrics moods are more correlated to valence than to the arousal dimension.

Downie et al. [43] propose a lyrics-based approach to mood classification using a binary SVM classifier. The authors propose using a vector space model representation, combined with other statistical textual features such as part-of-speech tags. The proposed approach is evaluated using a private dataset with 5,585 songs and using the 18 mood classes presented in [62]. The results show that the combination of both audio and lyrics features can improve the classification performance. In later work [63], the authors explore different lyrics features and modifiers, such as stylistic features or features obtained from the ANEW dataset. The results show that a lyrics-based classifier can outperform an audio-based classifier for some mood classes.

Kim et al. [80] also explore lyrics-based mood classification. The approach uses partial syntactic analysis to extract emotions or moods from songs, achieving an accuracy of around 60% when evaluated in a manually labeled dataset of 500 Korean songs. This paper proposes an approach which includes novel features such as negation detection, time of emotion and change of emotion.

A different approach is adopted by Kumar et al. [91], who uses Sentiwordnet [47] to extract mood features from 185 lyrics labeled with one of four mood classes: *happy, angry, love* and *sad*. This work compared three classifiers: KNN, SVM, and Naïve-Bayes; the latter classifier performed best, achieving a classification accuracy of up to 81%.

A comparison of audio features, lyrics features and cultural features for the task of genre classification is studied in [107]. The lyrics feature approach relies on stylistic features extracted from 250 songs.

Cultural features include Yahoo music[8] co-occurrence page counts and user generated tags from LastFM. The results show that, in some cases, combining the different feature groups results in a decrease in performance, and that lyrics features are less effective than other feature types.

Dodds et al. [39] use features extracted from text and the ANEW dataset to measure the sentiment of songs, blogs and State of the Union presidential speeches. The aim of the work is to quantify the evolution of the overall happiness in the different contexts. The approach calculates the average valence of each instance (song, blog post or speech) as a measure of happiness. The results show that, for example, valence can help distinguish between genres, when a large number of songs are considered.

Laurier et al. [95] explore the use of audio and lyrics features to classify songs into 4 different moods *(angry, happy, sad, relaxed)* following Russell's model of affect. The results indicate that using *latent semantic analysis (LSA)* and standard distance-based metrics, the classification using lyrics is better than random, but it does not outperforms the audio-based approaches. This work also explores an early fusion of the audio and lyrics features. The results improve the audio classification baseline, indicating that audio and lyrics information can be complementary.

Chi et al. [32] perform a study where several moods are extracted from songs by dividing each song into fragments, following the idea that a song can contain different moods over its length. The data is manually generated in a user trial where 246 users expressed their mood rating over 600 pop songs. The audio-based classification approach uses MFCC features, while for the lyrics approach a unigram and bigrams representation is used. The moods are obtained dividing Russell's model in four different quadrants with positive and negative valence and arousal values. The classification results using a SVM classifier show that that lyrics are a good data source to improve audio-based systems, plus lyrics can be a better estimate when music and lyrics models conflict.

Natural language processing is used in [101] for different tasks such as language identification, structure extraction, similarity search and topic categorisation, distinguishing between four topics in songs: *love*, *violent*, *protest*, *christian* and *drugs*. The work is most effective in the task of language identification, while structure extraction is not reliable, but the authors consider that it could be used for bootstrapping audio segmentation algorithms.

From the related work it is clear that mood and genre classification of music using lyrics is an established and interesting problem.

8 https://music.yahoo.com/

However, it is difficult to compare previous findings, since different works have reached contradictory conclusions based on experiments carried out on different datasets, using different mood and genre taxonomies and evaluation methodologies. For these reasons, in this work we consider lyrics-based classification using a representation based on the vector space model described in Section 4.4, incorporating and features based on the valence, arousal and dominance dimension derived from the ANEW dataset, in similar fashion to previous work [63]. Moreover we expand on previous work [39], exploring the dataset properties in terms of moods and genres and their relation to the valence and arousal dimensions, and evaluate the proposed approaches using a large publicly-available dataset.

4.6 CONCLUSIONS

This chapter covered the basic concepts related to the different classification approaches, describing some popular algorithms and different feature representation approaches. Moreover, we presented an overview of music classification, including the representation of mood and genre and how audio and lyrics features can be used to classify songs.

Music classification is a very important task, and from the literature it is clear that there is room for further advances, especially since not much work has been done to exploit the lyrics of the songs, which have proven to be an important part of mood perception.

Much of the existing work studies a single label classification, using well known classifiers such as SVM and Naïve-Bayes. However, there is no consensus over the mood granularity model, and each work proposes its own taxonomy for genre or mood classification.

It is noteworthy that the results obtained by different works are not consistent in their methodology or outcomes, and often show contradicting results. Previous work has found that lyrics features can outperform audio features in music classification [42, 104] in certain cases. However, [108] suggests that lyrics perform worse than other features. Moreover, comparing results from different approaches is difficult, as the classes selected for classification are not consistent, and the datasets used are different are not publicly-available.

The next chapter will introduce the approach that this work follows (based on Russell's model of affect and features extracted from the ANEW dataset). The evaluation is carried out using a publicly-available dataset and lyrics obtained in legally using the lyricFind API. Moreover, the results of the proposed approach are compared with the standard vector space model approach.

5 EVALUATING MUSIC CLASSIFICATION

5.1 INTRODUCTION

In this chapter we present an analysis of mood and genre music classification using information obtained from song lyrics. We evaluate the performance of a meta-feature based representation of songs using the valence, arousal and dominance dimensions, in a similar fashion to Russell's model of affect. We follow the approach proposed in [63] to derive features from the ANEW dataset [19], and expand it with new features. Moreover, we compare the results with the vector space model approach, studying several term weighting approaches.

Contrary to previous approaches based on different relatively small or private datasets [65, 117], we use a large freely-available dataset to facilitate the reproduction and comparison of findings.

With this common representation of song lyrics' using both the vector space model and meta-features, it is possible map songs into limited number of classes. Thus, an application of this approach can be to recommend songs which are similar in semantic content. For example, songs that talk about similar topics or express similar opinions, i.e., sad songs about break-ups, or protests songs against the establishment. The application of music mood and genre classification techniques for recommendation is left to future work.

The remainder of this chapter is organised as follows. First, Section 5.2 presents the datasets used in this work. Then, Section 5.3 describes an experiment which studies the use of the different approaches presented in Section 4.4.2 for music mood and genre classification. Section 5.4 studies the use of meta-features for music mood and genre classification. Here, a preliminary analysis of meta-feature based representation of songs is also presented, motivating the use of this approach for the mood and genre classification task. Section 5.5 describes a further experiments on music classification, focusing on the analysis of the learning capabilities of the different approaches studied in the previous experiments. Finally, Section 5.6 summarises the main findings and presents some lines of future work.

5.2 DATASETS

The experimental studies proposed in this work use the *Million Song Dataset* [14]. It is a large freely-available, dataset which contains rich metadata and audio features for one million contemporary songs. Two other complimentary datasets were released together with it; the *LastFM* and *MusixMatch* datasets.

- The **Million Song Dataset (MSD)** [14] contains metadata information for 1 million contemporary popular music tracks from 44745 unique artists. It contains tag information provided by the Echo Nest[1] and musicbrainz[2], more than two million asymmetric similarity relationships between tracks and 55 fields of metadata information per each track, such as, key, tempo, loudness, hotness or release year, among others.
- The **LastFM dataset**[3] contains song-level tags for more than 500,000 songs, including a total of 522,366 unique tags and 8,598,630 (track - tag) pairs. The dataset also includes song-level similarities, with a total of 56,506,688 (track - similar track) pairs. The moods and genres class labels used for classification are derived using the social tags found in this dataset, following the approach proposed in [61].
- The **MusixMatch dataset**[4] provides lyrics for 237,662 songs from circa 46,000 unique artists. Each song is described by word-counts of the top 5,000 words across the set, which allows for the data to be distributed without infringing copyright.

In this work, we only use the songs from the MSD for which social tags (obtained from the LastFM dataset) are available, and for which lyrics from either the MusixMatch dataset or the LyricFind API[5] can be obtained. Furthermore, we only consider English language lyrics in this study.

The **ANEW** dataset [19] is a collection of 2,476 words annotated with emotional ratings in three dimensions: *valence*, *arousal* and *dominance*. The dataset was created using human assessment, and it aims to provide a *set of normative emotional ratings* for the words included. To take into account the large and rich vocabulary of music lyrics, the ANEW dataset is expanded using WordNet [118] synonyms, as suggested in previous work [63, 65]. The extended ANEW dataset contains approximately 5,000 words.

1 http://the.echonest.com/
2 https://musicbrainz.org/
3 http://labrosa.ee.columbia.edu/millionsong/LastFM
4 http://labrosa.ee.columbia.edu/millionsong/musixmatch
5 Access to the LyricFind API (http://www.lyricfind.com) was obtained upon request, for non-commercial research purposes only.

word	Valence		Arousal		Dominance	
	mean	std.	mean	std.	mean	std.
Abduction	2.76	2.06	5.53	2.43	3.49	2.38
Able	6.74	2.00	4.30	2.17	6.83	2.04
Abortion	3.50	2.3	5.39	2.8	4.59	2.54
Absent	3.69	1.72	4.73	1.76	4.35	1.87
Absurd	4.26	1.82	4.36	2.2	4.73	1.72
Abundance	6.59	2.01	5.51	2.63	5.80	2.16
Abuse	1.80	1.23	6.83	2.7	3.69	2.94
Accept	6.80	2.11	5.53	1.96	5.41	1.92
Acceptance	7.98	1.42	5.40	2.7	6.64	1.91
Access	6.14	1.62	5.07	1.68	6.25	1.53

Table 12: The first ten words of the ANEW dataset.

Table 12 shows an snapshot of the ANEW dataset. Each word is represented by three properties: valence, arousal and dominance, each one containing the mean value and the standard deviation (std). For example, the word *abuse* in the ANEW dataset has a very low valence value (it is a negative word) with very little deviation between the ratings.

5.2.1 *Dataset Preprocessing*

The mood tag groups are inferred as described in [43], obtaining the mood groups described in Table 13. Then, songs are selected using the same criteria as used for the *MIREX 2009 mood multi-tag dataset*[6]; a song has to be tagged at least twice with one term from a tag group, or with at least two terms from a tag group, each at least once. Moreover, in order for a song to be included in the dataset we add further considerations; if a song has two or more moods (or genres), the mood (or genre) used more times by users to tag the song is selected. This is done to avoid repeated instances with different class labels.

To obtain the desired mood classes described in Table 13, the space is divided into four quadrants, as proposed in previous work [65]. Each mood class corresponds to a quadrant, (v^+a^+, v^+a^-, v^-a^+, v^-a^-), which represents a positive or negative value for valence and arousal.

Six genres are also considered in this work; *blues, hip-hop, metal, soul, punk* and *country*. Those are inferred directly using social tags from

6 http://goo.gl/8qmnwG

Group	Tags	Mood
G29	aggression, aggressive	$v^- a^+$
G25	angst, anxiety, anxious, jumpy, nervous, angsty	
G28	anger, angry, choleric, fury, outraged, rage, angry music	
G1	excitement, exciting, threill	$v^+ a^+$
G2	upbeat, gleeful, high spirits, zest, enthusiastic, buoyancy, elation, mood: upbeat	
G6	cheerful, cheer up, festive, jolly, jovial, merry, cheer, cheering, cheery, get happy, rejoice, songs that are cheerful, sunny	
G5	happy, happiness, happy songs, happy music, glad, mood: happy	
G16	depressed, blue, dark, depressive, dreary, gloom, darkness, depress, depression, depressing, gloomy	$v^- a^-$
G15	sad, sadness, unhappy, melancholic, melancholy, feeling sad, mood: sad - slightly, sad song	
G17	grief, heartbreak, sorrow,	
G8	brooding, contemplative, meditative, reflective, broody, pensive, pondering, wistful	$v^+ a^-$
G12	calm, comfort, quiet, serene, mellow, chill out, calm down, calming,chillout, comforting, content, cool down, mellow music, mellow rock, peace of mind, quietness, relaxation, serenity, solace, soothe, still, tranquil,tranquility, tranquility	

Table 13: Mood groups extracted from the LastFM data and clustered into four classes.

the LastFM dataset. In [39], a similar genre selection is studied, while we eliminate *pop* and *rock*, two very popular and noisy classes in the MSD. Thus, we select very distinctive genres, for which a large number of tags and lyrics are available in the MSD.

To illustrate the description above, consider the song *Orchestra of Wolves*[7] by *Gallows*. This song is frequently tagged as *aggressive* and therefore it is included in the mood group $v^- a^+$, as it has a low valence and high arousal mood. The same song is also tagged as *punk* and therefore it is included in the punk genre class.

7 http://open.spotify.com/track/5BorbORef4VQUlNOjAjoDT

In the experiments described in this chapter, the data is preprocessed following the process described in Section 4.4.1, and each document is then defined by its lyrics and a class (mood or genre) label. Below, we present a detailed description of the preprocessing steps involved.

1. First, all songs whose lyrics are not in English are discarded.
2. The lyrics are parsed. Here, we only consider unigrams and remove all non-alphabetical characters (punctuation signs, numerical characters, etc.)
3. All stop-words are removed, using a standard stop-word dictionary.
4. Each word is stemmed, using the SnowBall Porter stemmer algorithm [134], transforming each word into a term.
5. For classification efficiency, we only consider terms that occur at least twice. This condition is enforced on a per class basis.
6. For computation efficiency, a subset of 2,000 words are kept on each class (when possible) for the classification task using Weka's *StringToWordVector* filter.
7. When ANEW-based meta-features are considered, the steps 5 and 6 are skipped. Thus, all terms that occur in both the ANEW dataset and the song lyrics are used to compute the metafeatures.

5.2.2 *The Classification Dataset*

Once the MSD dataset is preprocessed as previously described, the resulting dataset used for classification contains 1,000 randomly selected songs for each of the four moods and the six genres (except for the v^-a^+ class, for which only 784 songs with full lyrics were found). Thus, the mood dataset contains a total 3,784 songs, and the genre dataset a total of 6,000 songs.

To build this dataset, we use the full lyrics for each song, obtained through the LyricFind API. It is important to remark that previous work has depended on private music collections [65, 117], as legally obtaining full lyrics for songs is often problematic (for reasons of copyright and cost).

5.3 EXPLORING TERM WEIGHTING APPROACHES IN CLASSIFICATION

In this section, we analyse the performance of a single label supervised classification approach for music mood and genre, using a vector space model representation of instances. We performa a preliminary feature analysis of the lyrics, studying the differences of vocabulary and the term distribution across the mood and genre classes. Moreover, we experiment with the different state-of-the-art term weightings approaches described in Section 5.3.3. We compare their performance with the new metric proposed in this work: *normalised inverse class frequency* (*nicf*, Equation 22).

5.3.1 *Evaluation Methodology*

The aim of this experiment is to classify music mood and genre, using a supervised approach and a Naïve-Bayes multinomial classifier[8]. The results are evaluated using the dataset described in Section 5.2.2.

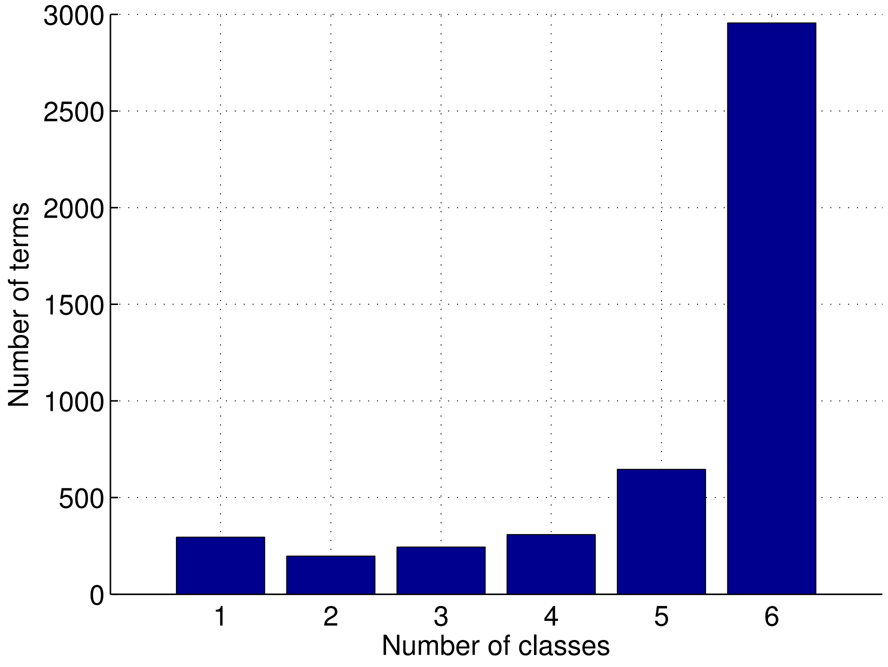
The classification was performed using the Weka machine learning framework [55]. On each case, a standard 5-fold cross validation approach was used to evaluate performance, expressed in terms of average true positive (TP) and false positive (FP) rates over all classes.

5.3.2 *Feature Analysis*

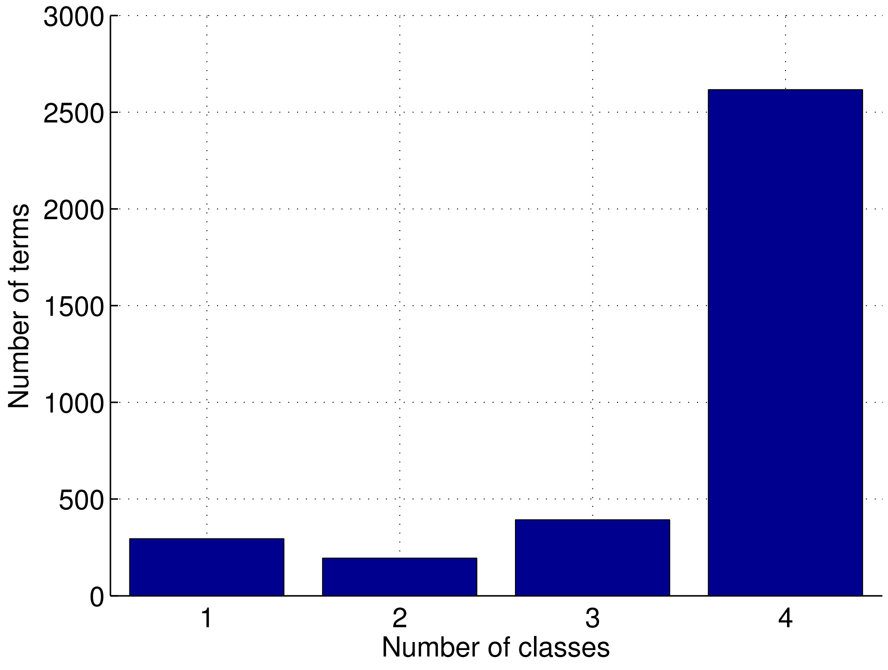
To understand how the different term weightings can affect the classification performance for the specific problem being tackled in this work, it is important to perform an analysis of the term distribution across documents and classes. The total number of distinct terms (vocabulary size) seen in the genre dataset is circa 4,700 terms, while the the mood dataset contains circa 3,600 distinct terms.

Figures 12a and 12b show histograms of the number of terms that occur in a certain number of classes for genres and moods respectively. For example, in the genre dataset $\sim 61\%$ of terms are common to all classes, but only $\sim 6\%$ of terms occur in a single class. Similar trends are seen for the moods dataset, where $\sim 72\%$ for terms are common to all classes and $\sim 8\%$ of terms occur on a single class.

8 The Naïve-Bayes multinomial classifier (described in Section 4.3.1) was chosen after preliminary analysis where several other algorithms (of those presented in Section 4.3), were also evaluated. The results indicated that the Naïve-Bayes multinomial approach was best suited for this task.



(a) Distribution of terms across genre classes.



(b) Distribution of terms across mood classes.

Figure 12: Distribution of terms in the mood and genre dataset.

Some term statistics are presented in Table 14. The number of distinct terms in each dataset is similar. For example between 3763 and 4023 in the genre dataset, and between 3048 and 3085 in the moods dataset. Then, moreover, it can be seen that a small number of unique terms is seen on each class (between 41 and 51 in the genre dataset), indicating that the vocabulary of lyrics has a great degree of commonality between classes.

Genre	**Number of distinct terms**	**Number of terms unique to the class**
Blues	3975	41
Hip-Hop	3983	47
Metal	4023	51
Soul	3920	53
Punk	3763	59
Country	3953	44

Table 14: Term statistics for the genre dataset.

Mood	**Number of distinct terms**	**Number of terms unique to the class**
v^-a^+	3048	63
v^+a^+	3084	72
v^-a^-	3085	75
v^+a^-	3111	85

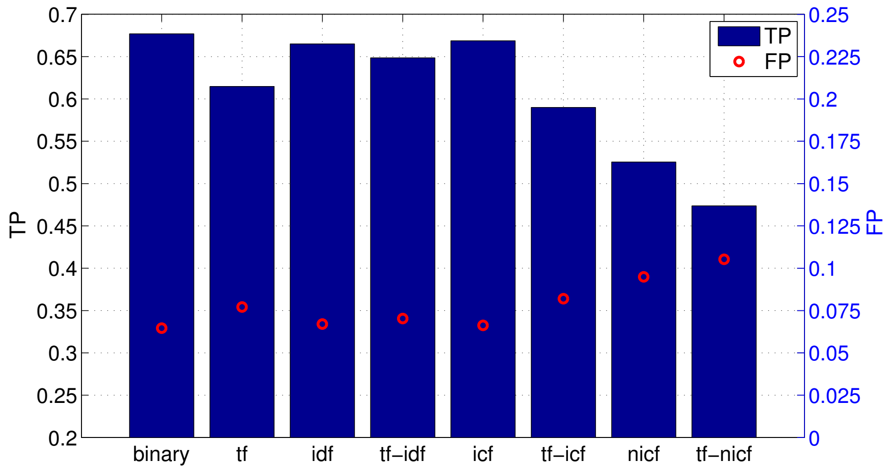
Table 15: Term statistics for the mood dataset.

Ideally, for best classification performance, the vocabulary should be very different between the different classes. If many of the same terms occur in all classes, it will be difficult to classify those songs into the mood and genre classes. Thus, to tackle this problem we propose the use of the *inverse class frequency (icf)* term weighting approach, introduced in Section 4.4. However, in the datasets considered, the vocabulary of different classes is very similar (all classes share circa 66% of terms). Thus, the effectiveness of the *icf* metric is likely to be limited from a classification perspective.

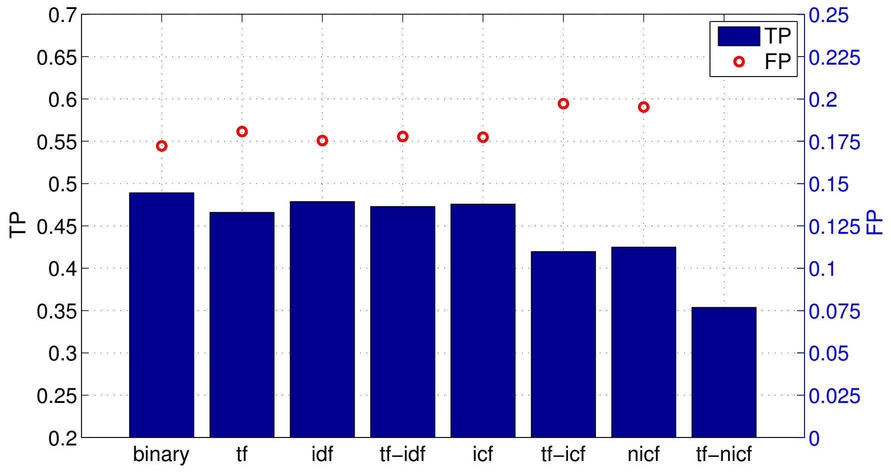
As such, the *normalised inverse class frequency (nicf*, Equation 22) metric is presented. This more sophisticated metric weights higher terms which occur in many documents within a class and in a small number of classes.

5.3.3 *Results*

Figures 13a and 13b show the classification results for different term weighing approaches for the mood and genre datasets respectively, in terms of average true positive rate (TP) and average false positive rate (FP). Here, we consider the various term weighting approaches described in Section 4.4.2, including state-of-the-art approaches (*tf, idf, tf-idf, icf*), as well as the term weighting approach proposed in this work (*nicf*) and combination of *tf* with *icf* and *nicf*.



(a) Genre dataset.



(b) Mood dataset.

Figure 13: Classification results for different term weightings.

In general terms, the performance of the vector space model classification approach is better in the genre dataset, where the maximum average TP rate is ~ 0.7, while the maximum TP in the mood dataset is ~ 0.5. These results align with previous findings using audio-based features in a binary classification problem [131]. However, the dataset used in that work is not publicly available, and to the best of our knowledge, no comparison between mood and genre classification using the same dataset and a lyrics-based approach has been previously studied. Moreover, as we described in Section 5.3.2, the genre dataset has more distinct terms per class compare to the mood dataset, which may have an influence on the classification performance observed.

The most simple approach, binary term weighting, outperforms all other term weighting approaches in both datasets. This results align with previous work [63], where this approach was compared with the *tf-idf* weighting approach. Moreover, the results show that the combination of *tf* with other metrics, such as *idf* or *icf*, results on a decrease of the performance. For example, the accuracy decrease of the *tf-icf* approach compared to the *icf* approach is $\sim 13\%$.

In both datasets, the performance of the *icf* and *idf* term weighting approaches is comparable, highlighting the limitations of the *icf* approach, as described above.

It is noteworthy that our proposed approach, *nicf*, does not performs better than the original *icf* approach. As previously discussed in Section 5.3.2, most of the terms that occur in a small subset of classes are only present in a small subset of documents in those classes. For example, in the genre dataset only circa 8% of terms occur on a single class, and those terms occur in only 1.22 documents on average. Thus, this metric can not exploit the differences between the distribution of terms that occur in a small number of classes. However, this result is likely to be domain dependent, and hence, the *nicf* metric might provide a better performance in other settings.

5.4 META-FEATURE BASED CLASSIFICATION

In the previous experiment a vector space model representation of songs was consider for classification. In this experiment, we consider a different approach based on an instance-based representation of each song using (1) statistical features derived from the valence, arousal and dominance dimensions described above; (2) features based on lyric sentiment derived from a sentiment lexicon and (3) stylistic features. First, we present a preliminary analysis of the features used in this experiment and its adequacy in the task of music mood and genre classification. We then analyse the performance of a single-label supervised classification approach for music mood and genre classification, considering also an early fusion ensemble approach [162] in which various features are combined prior to the classification. Finally, we compare these representations against a the vector space model approach evaluated in Section 5.3.3.

5.4.1 *Preliminary Meta-feature Analysis*

Here, we present an analysis of the moods and genres of songs and their relationship with the valence and arousal dimensions described in Russell's model [148]. To this end, we use the approach proposed in [39] to calculate the average valence values of songs, and expand it considering the arousal values, using the dataset described in Section 5.2. This approach is designed to better understand if and how the valence and arousal dimensions can be used to distinguish songs from different genres and moods. Moreover, we wish to determine how moods and genres are correlated to these dimensions.

5.4.1.1 *Approach*

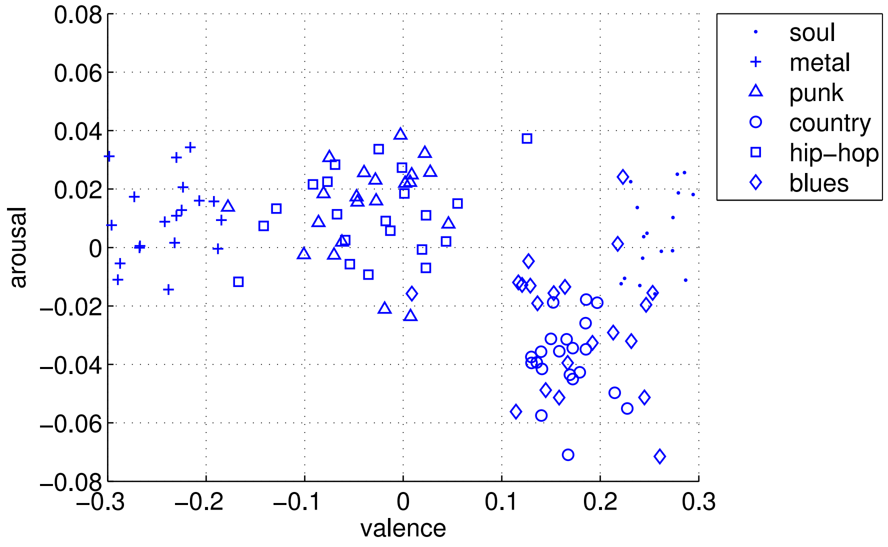
The classes are obtained as described in Section 5.2.1. Thus, we use four moods (v^+a^+, v^+a^-, v^-a^+, v^-a^-) and six genres (*blues, hip-hop, metal, soul, punk* and *country*). In the evaluation we select 44,707 songs tagged with one of the genre classes and 56,779 songs tagged with one of the mood groups described in Table 13. This are all the songs from the MSD that have all the following metadata fields: social tags from one of the selected classes, release year information and lyrics from the MusixMatch dataset (as it provides relatively easy and fast access to a very large number of lyrics).

The approach proposed by Dodds et al. [39] to calculate the average valence per document (song) is followed. Moreover, we use the same approach to calculate the average arousal value per document, something not considered in the original work. For each song, its valence (or arousal) is calculated as the sum of the valence (or arousal) of each individual term included in the ANEW dataset, divided by the sum of term frequencies of all terms.

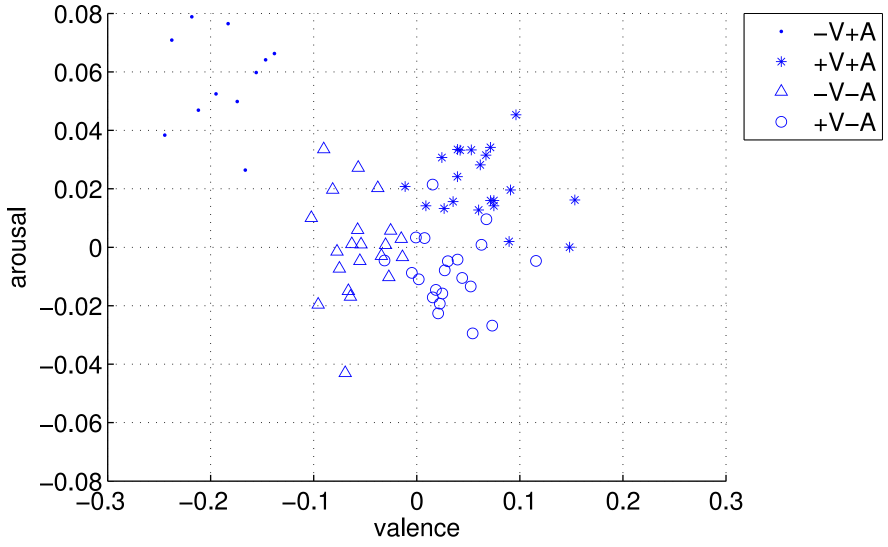
5.4.1.2 *Results*

In this section we explore the relationship between moods, genres and the dimensions of Rusell's model (valence and arousal). Figures 14a and 14b show the values of valence and arousal where each point in the figures represents the average value of valence and arousal for all the songs in a particular genre (or mood) for a given year.

From Figure 14a, we can infer that valence can potentially help to classify music genres. For example, the genre *soul* has on average a much higher valence value than the rest of the genres (~ 0.25), while the genre *metal* has the lowest average valence value (~ -0.25). However, the average valence values for some genres, such as *country* and *blues*, are similar. Nevertheless, this result is interesting and it



(a) Valence and arousal for different genres.



(b) Valence and arousal for different moods.

Figure 14: Valence and arousal values per class per year.

proves the hypothesis that this dimension is correlated with genres and can help to distinguish between them, providing evidence that features based on valence values may be useful from a classification perspective.

Figure 14b shows similar results for moods, where it can be seen that the positive and negative values of valence are well separated, though less so than for genres. This finding indicates that the classification accuracy of music genre will be, on average, higher than that for mood using features based on the valence dimension.

It is also clear from the results that arousal is a much less predictive dimension than valence in this domain. In the case of genres (Figure 14a), for example, only the *blues* and *country* classes have a lower average arousal value, distinguishing them from the rest of the genres. For moods, the arousal dimension should be distinctive for different classes as it maps directly to the mood taxonomy. However, the distinction between positive and negative arousal clusters is not as clear when compared to the valence dimension, as the results in Figure 14b indicate.

Finally, it is important to analyse the deviation over the mean values, which are not plotted in the figures for clarity. For example, for both mood and genre classes, the average standard deviation in the valence dimension is 0.4. This high variation in the data implies limitations when classifying moods and genres based on this data alone, an analysis of which is considered in the next section.

5.4.2 *Classification Approach*

The ANEW-based meta-features are based on those proposed in [63]. For each song, we extract the mean valence value (likewise mean arousal and dominance values) of all lyric words contained in the ANEW dataset and compute overall valance (likewise arousal and dominance) statistics for the song as per Table 16. The sentiment features described in Table 16 are calculated following the method described in [41], where a lexicon of positive and negative terms are used to calculate the average sentiment of each song, along with the number of positive and negative terms in the song. Finally, stylistic features based on word counts (frequently used in text classification) are also considered.

5.4.3 *Evaluation Methodology*

In this Section, we use the same methodologies proposed in Section 5.3 for the vector space model approach, and in Section 5.4.1 for the

Feature Group	Feature
ANEW features (valence, arousal, dominance)	Minimum value
	Maximum value
	Mean value
	Standard deviation
	Median value
Sentiment features	Num. positive words
	Num. negative words
	Average sentiment
Stylistic features	Num. words
	Num. unique words
	Num. unique ANEW words
	Max. word frequency

Table 16: ANEW, sentiment and stylistic feature groups.

meta-features based approach. We select the same songs detailed in Section 5.3.1. The classification was performed using the Weka machine learning framework [55]. In each case, a standard 5-fold cross validation approach was used to evaluate performance, expressed in terms of weighted true positive (TP) and false positive rate (FP) over all classes.

For each of the meta-feature groups (ANEW, sentiment and stylistic groups) shown in Table 16, we used a *random forest classifier*[9] with 100 trees, with the number of features per tree proportional to the square root of the number of features in each group. In preliminar experiments, this configuration proved to make the best use of the very restricted feature space available. For the vector space model approach, we used a Naïve Bayes classifier as in Section 5.3.3.

5.4.4 *Results*

We begin by examining the classification performance provided by each of the individual meta-feature groups. The results are shown in Table 17. For moods, the valence and arousal feature groups performed best; this result was expected as the classes into which each song is classified are represented by these dimensions. The remaining meta-feature groups all provided very similar performance. For

9 The selection of the random forest classifier for the meta-features based approach was done after preliminary analysis where this approach outperformed other studied classification algorithms such as Naïve-Bayes.

example, a TP (FP) rate of 0.347 (0.219) was achieved by the valence features, compared to 0.293 (0.237) for stylistic features.

Algorithm	**Moods**		**Genres**	
	TP	**FP**	**TP**	**FP**
Valence	0.347	0.219	0.324	0.135
Arousal	0.306	0.233	0.312	0.137
Dominance	0.296	0.236	0.299	0.140
Sentiment	0.294	0.236	0.302	0.140
Stylistic	0.293	0.237	0.373	0.125
Ensemble (early fusion)	0.414	0.196	0.483	0.103
VSM (binary)	0.489	0.172	0.677	0.065

Table 17: Meta-features accuracy for Classification.

For both moods and genres, as indicated by the findings in Section 5.4.1, the valence meta-features performed better than the arousal features, in the same way that valence was seen to better distinguish between classes. Interestingly, the stylistic features performed best for genre classification, resulting in a TP (FP) rate of 0.373 (0.125), while the valence features achieved 0.324 (0.135).

Overall, the vector space model approaches achieved the best results, particularly in the case of genres. However, when the different meta-feature groups were combined in the ensemble approach, the overall accuracy showed an increase over the best performing single meta-feature group. For example, the increase in TP rate was 0.11 for genres and 0.7 for moods. Even though this approach does not outperform the vector space model approach, it is reasonably efficient. This finding is important given the difficulty of sourcing complete lyrics for songs (for copyright reasons), upon which the strong performance of the vector space model approaches depends. However, a meta-feature representation based on, for example, valence and arousal features could potentially be used to classify music using different sources of information, such as reviews or comments for songs, now widely available on many online sites such as Twitter.

5.5 STUDYING LEARNING RATES IN CLASSIFICATION

In the previous experiments we studied two different approaches for music mood and genre classification. In the first experiment (Section 5.3) we studied different term weighting approaches for the vector space model, while in the second experiment (Section 5.4) we studied a meta-features based approach.

In this experiment, we further study the learning rates for the approaches shown in Table 17, with the aim to gain a better understanding of the role of training data in the classification performance. To this end, we build a set of learning curves that describe how fast the learning process is, and how the classifier performance changes as the amount of training data increases.

5.5.1 *Classification Approach*

In this experiment, we use the dataset described in Section 5.2.2. To study the classification learning rates, we vary the amount of training data from 50 instances per class to 600 instances per class. Moreover, to guarantee a consistent evaluation, all the classification results are evaluated using the same test set which contains 200 randomly selected instances per class which are not used in the training stage[10].

For the vector space model approach, we evaluate the best performing approach (*binary* weighting) using a Naïve-Bayes multinomial classifier, as in the experiment described in Section 5.3. For the meta-feature approach, the same feature groups and classifier as in the previous experiment (described in Section 5.4) are used.

5.5.2 *Results*

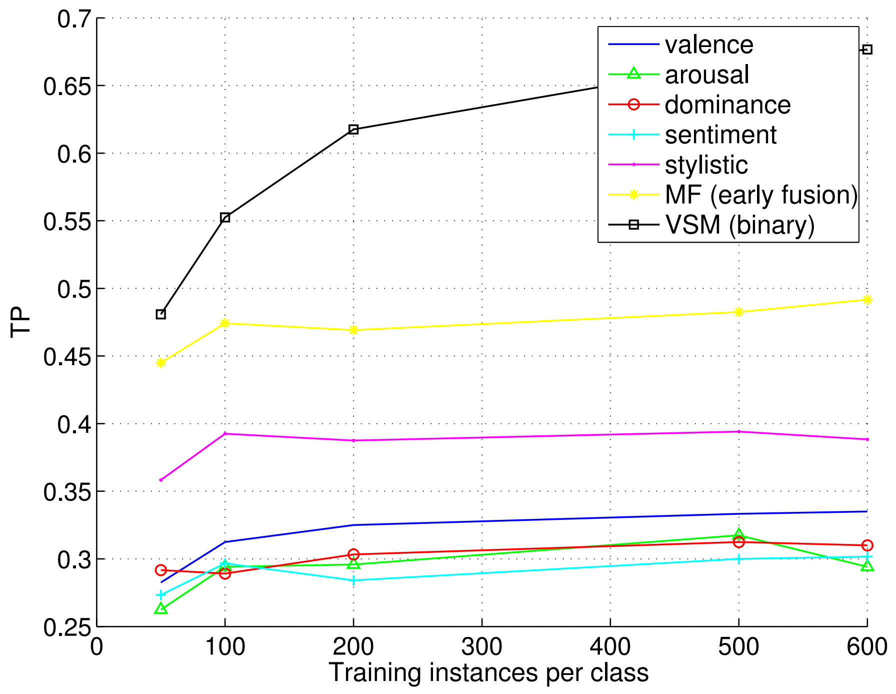
As the classification results obtained for the mood and genre datasets show very similar trends in this experiment, we focus on the analysis of the genre classification results.

Figure 15 shows the accuracy for each of the evaluated approaches. The horizontal axis represents the number of training instances per class (from 50 to 600), and the vertical axis represents the average true positive rate (Figure 15a) and false positive rate (Figure 15b).

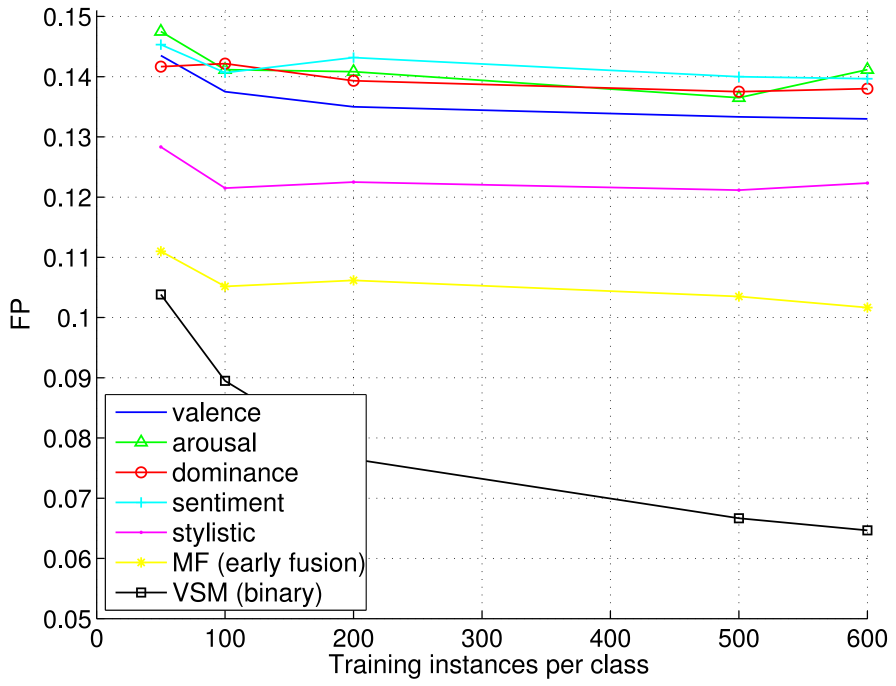
Two main trends arise from the analysis of the results; (1) the vector space model approach (binary term weighting) requires more training instances to learn, and (2) all the meta-features based approaches require small amount of data to achieve their optimal performance.

For the vector space model approach, the learning improves steadily until 200 training instances (TP ~ 0.62), thereafter the learning continues at a lower rate. Indeed, at 600 training instances (TP ~ 0.68) the classifier is still learning, indicating that a large number of training

10 Except for the $v^{-}a^{+}$ class, for which only 155 songs with lyrics are used as a test set.



(a) Learning curves - TP rates.



(b) Learning curves - FP rates.

Figure 15: Learning curves for the genre classification task.

examples is needed due to the high-dimensional representation of songs (the number of features is circa 4,700) used in this approach.

The meta-features based approach requires a relatively small number of training instances to achieve its optimal performance. For example, in the early fusion approach, the gain in accuracy between 50 and 600 training instances (measured in terms of TP) is close to 8%. This is likely due to the reduced dimensionality of the feature space used in this approach. It is also noteworthy that the approach based on valence features outperforms the approach based on arousal features throughout the learning curve. This result correlates with the findings described in the previous experiment. Moreover, the vector space model approach outperformed all the meta-features based approaches independently of the number of training instances used in truing, again highlighting the superior performance achieved by this approach.

5.6 CONCLUSIONS AND FUTURE WORK

In this chapter, we have presented a comprehensive evaluation of music mood and genre classification, relying solely on lyrics as a source of information. Moreover, our approaches have been evaluated using a large freely-available dataset, with the aim of future reproducibility and comparability of the results.

In the first experiment we studied a vector space model approach for classification where different term weighting approaches were evaluated. Our proposed approach (*nicf*) does not perform as expected. As previously highlighted, the distribution of terms across classes and documents is such that this metric was not effective. However, this might not be the case in other domains, and such the performance of this metric should be further explored.

In the the second experiment we derived meta-features from the valence, arousal and dominance dimensions, presenting a rationale for using them in the classification task and comparing the results with those obtained for the vector space model approach. From the feature analysis it is clear that not only moods but also genres can be mapped onto a two-dimensional space as that proposed by Russell. However, one of the variables in the space (valence) has proven to be more accurate when predicting different mood and genres classes.

When an ensemble of sentiment, stylistic and ANEW-based meta-features are used in the supervised classification approach, the results are promising, showing that the combination of the proposed meta-features using an early fusion approach leads to an increase in classification accuracy. However, it does not outperforms the vector space model approach in terms of accuracy. This result does not align with findings in previous work [42], where a similar combination of ANEW based and stylistic meta-features outperformed a more complex vector space model approach (which considered unigrams bigrams and trigrams). However, the previous work evaluated a binary classification approach using a dataset which is not publicly available for experimentation, making the comparison of results difficult.

Altought the performance of the meta-features based approach did not perform as good as the vector space model in the task of mood and genre music classification, it might be interesting to apply this model in other domains. For example, the lyrics are short in length by nature, and in cases where a larger number of terms per item is available (e. g., caption text from television shows or news articles), the meta-features based approach described in this work may enjoy better performance.

6 CONCLUSIONS AND FUTURE WORK

The work in this thesis is motivated by the exponentially growing amount of information available on the Internet (especially entertainment content), and the need for intelligent systems that enhance content discovery. For example, if we analyse the music industry, its business model has changed from physical copies to streaming services. This has caused users not to buy albums or songs, but rather pay a monthly subscription to have access to content in an all-you-can-eat fashion. However, even when users have access to catalogs of dozens of millions of songs, it is known that the average user listening behaviour has not changed, and users still listen to a very small portion of the music available. This creates is a long tail of content which is not being exploited.

In this scenario, where thousands of movies and millions of songs are available within one click, users are much more dependent on intelligent systems wich guide them, personalising the user experience to avoid information overload and enhance content discovery. Recommender systems tackle the problem of information overload by filtering the information in a personalised fashion, guiding the user when browsing through the large collections of content available.

Music is a particular domain where recommender systems have been widely studied, both in commercial systems and academia. New ways to find similarities between items are being studied, with the aim of enhancing content discovery, and consequently, the user experience. One example is music classification, which allows, for example, to create mood and genre specific playlist or recommendations.

In this thesis, we have tackled problem of enhancing content discovery from two different perspectives: first, we performed an offline evaluation of collaborative filtering recommender systems algorithms, measuring the performance in terms of accuracy, diversity, popularity, reach, uniqueness and coverage. We performed two experiments in a variety of datasets and algorithms obtaining interesting and novel results. Secondly, we studied a supervised classification approach and applied it to the music mood and genre classification task, exploiting a low-dimensional meta-feature based space. We compared its performance with the classical vector space model in a large freely-available dataset.

The rest of this chapter is organised as follows. The key findings of the first part of this thesis are presented in Section 6.1. Here, we study the problem of evaluating collaborative filtering recommender systems algorithms. Then, Section 6.2 presents the key findings of the second part of this thesis, which tackles the problem of music classification. Section 6.3 presents several lines of future work and Section 6.4 presents a high-level summary of this thesis.

6.1 A COMPREHENSIVE EVALUATION OF COLLABORATIVE FILTERING RECOMMENDER SYSTEMS ALGORITHMS

The evaluation of recommender systems is a key part for the development of new and better algorithms. In Chapter 2 we presented an overview of the current trends in recommender systems, focusing on how these systems are evaluated. We described the most popular accuracy metrics and discussed the relevant recent findings in the area. We also introduced several key properties that go beyond accuracy, together with a discussion on the related work, showing the main advances and open problems in the area. In Chapter 3 we performed two experiments on evaluation of recommender systems.

First, we performed a comprehensive evaluation of the *UKNN*, *IKNN* and *WRMF* algorithms, presenting interesting findings when properties such as popularity bias, diversity and accuracy were studied. Then, we performed a comparative study of the *UKNN* and *IKNN* algorithms, showing the importance of the algorithm parameters (i. e., number of neighbours) in its overall performance. The most relevant findings obtained from the experiments carried out in this part of the thesis are discussed below.

- Matrix factorisation algorithms are known to perform very well in terms of accuracy. Our results align with previous findings, also highlighting that the evaluated matrix factorisation algorithm (*WMF*) generally produces more accurate (in two of the datasets evaluated) , diverse and less-popular recommendations than the neighbourhood-based approaches evaluated.
- From the results, we can conclude that the *IKNN* algorithm has much higher item reach than all the other evaluated alternatives (in two of the three datasets studied), recommending a much larger subset of items. Moreover, the item reach behaviour of this algorithm highly depends on statistical properties of the dataset such as sparseness.
- The findings of this work indicate that the *UKNN* algorithm is inherently biased towards making recommendations of popular items, specially at larger neighbourhood sizes. Moreover, there

is a loss of diversity at larger neighbourhood sizes which is not compensated by a gain in precision.

- When comparing the *UKNN* and *IKNN* algorithms we found that the uniqueness of the recommendations generated depends on the number of neighbours, and that smaller neighbourhood sizes lead to more unique, less popular and more diverse recommendations.

These findings were based on the experimental methodology described in this work. However, it is noteworthy that, in general terms, an offline evaluation is always limited, and should not be considered as a complete evaluation process, but as one part of the many necessary to understand the overall quality of a recommender system. Several well known limitations of this evaluation methodology have been discussed in this work. The main one is the fact that items which are recommended to a user, but do not appear in the test set in the evaluation are considered bad recommendations, even when that might not be the case in the real-world scenario. Thus, there is a need for further online evaluation, as for instance, user studies and A/B testing, to comprehensively evaluate recommender systems.

6.2 MUSIC MOOD AND GENRE CLASSIFICATION

Music classification is an important and interesting area of content discovery. It allows users to automatically sort the large collections of music available on the Internet, and browse through them.

In Chapter 4 we presented an overview of the classification problem, focusing on music mood and genre classification. We presented the main trends for music mood and genre representation, together with the relevant work on classification using lyrics. We also described in detail the vector space model, introducing a novel term weighting approach. In Chapter 5 we presented two experiments on music mood and genre classification.

First, we performed an evaluation of the different term weighting approaches described in Chapter 4, comparing the newly proposed one with state-of-the-art approaches. We then evaluated a classification approach based on meta-features extracted from the ANEW dataset and mapped into Russell's model of affect. The most relevant findings obtained from the experiments carried out in this part of the thesis are discussed below.

- One of the evaluated term weighting approaches for the vector space model *(inverse class frequency)* shows promising results, being its performance comparable to the binary weighting approach (best performing approach in this scenario). However, the modification proposed in this thesis (*normalised inverse class frequency)* shows limited prediction power, leading to poor classification results due to the distribution of terms within classes. As we previously highlighted in Chapter 4, most of the terms which appear in a small number of classes, only appear in a very limited number of documents from those classes, which limits the performance of the proposed metric.

- From the results obtained, we can conclude that not only moods but also genres can be mapped into the proposed representation of valence and arousal which follows Russell's model of affect. Moreover, the valence dimension proved to have more predictive power in both scenarios.
- The proposed approach, which is based on an ensemble of stylistic, sentiment and ANEW-based meta-featues shows promising results. However, this approach did not outperform the standard vector space model approach when a binary term weighting is used. This, at least in part, can be explained by the nature of lyrics, which show a complicated ever evolving language, which is hard to capture using a static and limited dataset like ANEW. Thus, this classification of lyrics using this type of approach is limited in that sense.

The approach presented in this thesis for music classification has two main limitations. First, it depends on the selected genres and mood granularity for the classification task. However, as previously explained, moods are difficult to infer and there is evidence that they are culturally dependent. Moreover, the genre taxonomy is not fixed, and new genres often appear as the music evolves. Moreover, lyrics contain a limited vocabulary, which makes the classification based on lyrics alone is limited. Nevertheless, the approached described in this thesis may perform better in domains where larger documents are found, for example in the news domain or in the television domain. An analysis of this is left for future work.

6.3 FUTURE WORK

The work of this thesis has lead to interesting and results in the area of recommender systems evaluation. For example, uncovering the role that the neighbourhood size plays in the overall performance of the *UKNN* algorithm, or presenting interesting findings in terms of correlation between the algorithms studied and the recommender system properties evaluated. Thus, there are different resulting lines of research that can be explored. However, these are outside the scope of the thesis, and are left for future work.

- We analysed the performance of neighbourhood-based algorithms in relation with the number of neighbours. However, it would be interesting to analyse the performance along the user dimension, by comparing recommendation performances for users which are different number of items.

- The fact that offline evaluation is limited has already been discussed in this thesis. Thus, this type of evaluation has to be considered as one of the several parts in the overall evaluation. For future work, online user studies could focus on the correlation between the metrics, the algorithms studied and user behaviour. For example, as previously stated, we expect that when the *UKNN* and *IKNN* algorithms are compared, if the uniqueness between the two approaches is not large, the impact on the user experience would be limited. However, this hypothesis still needs to be evaluated.

- The properties that define a recommender system's performance evaluated in this work is broad. However, two properties that are very related to content discovery (novelty and serendipity) were not studied, as the offline evaluation of those is often difficult due to limitations of the data available. Thus, these properties should be studied in the context of an online evaluation

In the second part of this thesis, we focused on the particular problem of music classification with the aim to enhance content discovery. The performance of the proposed model, based on a low-dimensionality feature space for mood and genre classification, has proven to be limited in the music domain. Thus, we propose several lines of research to further explore the different facets of this model in different scenarios.

- We are currently studying the use of this model to analyse and classify television content, paying special attention to mood of news channels, trying to correlate the mood derived from text transcripts with the top news of the year 2013. Promising preliminary results have been obtained, and we are currently trying to find a wider source of information to evaluate the approach. The data for this work is obtained trough the Boxfish API[1], which has recently been open for research purposes for the *First Workshop on Recommender Systems for Television and Online Video in conjunction with the ACM RecSys 2014*[2].

- We also understand the limitations of the evaluation approach for music classification carried out in this work. It has been proven that moods are difficult to infer, they are perceived in

1 http://boxfish.com/login
2 http://boxfish.com/recsys

different ways by people, and they are culturally dependent. Thus, we propose to perform user studies and A/B testing to investigate the user perception of our mood and genre representation.

- In order to better understand the suitability of the proposed meta-feature based approach model in a recommendation scenario, a content-based recommender system that uses this representation could be investigated

6.4 SUMMARY

In this thesis we have studied the problem of content discovery from two different perspectives. First, we performed an evaluation of the most common collaborative algorithms in terms of accuracy, diversity, uniqueness, reach, popularity and coverage. The results showed that some algorithms are inherently biased towards making popular recommendations (*UKNN*), that diversity and popularity are correlated, and that the performance of neighbourhood based algorithms (measured in terms of accuracy, uniqueness, diversity and popularity) depends on the number of neighbours. We then tackled the problem of music classification for discovery. We evaluated two different feature representations of songs, showing that the low-dimensionality meta-features based feature representation approach can help when representing and classifying both moods and genres, while not outperforming the vector space model approach. Moreover, we have also proposed several possible extensions of this work for future research.

BIBLIOGRAPHY

[1] Pirkka Å man and Lassi A. Liikkanen. A Survey of Music Recommendation Aids. *WOMRAD 2010 Workshop on Music Recommendation and Discovery*, 2010:3–6, 2010.

[2] Xavier Amatriain, Alejandro Jaimes, Nuria Oliver, and Josep M. Pujol. Data mining methods for recommender systems. In *Recommender systems handbook*, pages 1–44. 2011.

[3] Chris Anderson. The Long Tail. *Wired Magazine*, 12, 2004.

[4] Jean-Julien Aucouturier and François Pachet. Representing Musical Genre: A State of the Art. *Journal of New Music Research*, 32(1):83–93, March 2003.

[5] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. Matrix Factorization Techniques for Context Aware Recommendations. *In Proceedings of the fifth ACM conference on Recommender systems*, pages 301–304, 2011.

[6] H.B. Barlow. Unsupervised Learning. *Neural Computation*, 1:295–311, 1989.

[7] Robert M. Bell and Yehuda Koren. Improved neighborhood-based collaborative filtering. *In KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 7–14, 2007.

[8] Robert M. Bell and Yehuda Koren. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[9] Robert M. Bell and Yehuda Koren. Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights. *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 43–52, October 2007.

[10] Alejandro Bellogin, Pablo Castells, and Iván Cantador. Precision-oriented Evaluation of Recommender Systems: An Algorithmic Comparison. *In Proceedings of the fifth ACM conference on Recommender systems - RecSys'11*, pages 333–336, 2011.

[11] Alejandro Bellogín, Pablo Castells, and Iván Cantador. Improving memory-based collaborative filtering by neighbour

selection based on user preference overlap. *In Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, (2):145–148, 2013.

[12] James Bennett and Stan Lanning. The netflix prize. *In Proceedings of KDD cup and workshop*, pages 3–6, 2007.

[13] Michael W Berry, Susan T Dumais, and Gavin W O'Brien. Using linear algebra for intelligent information retrieval. *SIAM review*, 37(4):573–595, 1995.

[14] Thierry Bertin-mahieux, Daniel P W Ellis, Brian Whitman, and Paul Lamere. The million song dataset. *Proceedings of the 12th International Society for Music Information Retrieval Conference*, (Ismir):591–596, 2011.

[15] M Besson, F Faita, and I Peretz. Singing in the Brain: Independence of Lyrics and Tunes. *Psychological Science*, 9:494–498, 1998.

[16] Yolanda Blanco-Fernández, José J. Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrer, and Martin López-Nores. Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems. *IEEE Transactions on Consumer Electronics*, 54:727–735, 2008.

[17] Al Borchers, Jon Herlocker, Joseph Konstan, and John Riedl. Ganging up on Information Overload. *Computer*, 31(4):106–108, 1998.

[18] Svetlin Bostandjiev, J O'Donovan, and Tobias Höllerer. Tasteweights: a visual interactive hybrid recommender system. *In Proceedings of the sixth ACM conference on Recommender systems, RecSys'12.*, pages 35–42, 2012.

[19] MM Bradley and PJ Lang. Affective norms for English words (ANEW): Instruction manual and affective ratings. 1999.

[20] JS Breese, D Heckerman, and C Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 45–52, 1998.

[21] L Breiman. Bagging predictors. *Machine learning*, 140:123–140, 1996.

[22] L Breiman. Random forests. *Machine learning*, pages 1–35, 2001.

[23] Michael Buckland and Fredric Gey. The relationship between Recall and Precision. *Journal of the American Society for Information Science*, 45(1):12–19, January 1994.

[24] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[25] Robin Burke, Michael P. O'Mahony, and Neil J. Hurley. Robust collaborative recommendation. In *Recommender Systems Handbook*, pages 805–835. 2011.

[26] Pedro G. Campos, Fernando Díez, and Manuel Sánchez-Montañés. Towards a more realistic evaluation: testing the ability to predict future tastes of matrix factorization-based recommenders. *In Proceedings of the fifth ACM conference on Recommender systems, RecSys'11*, pages 309–312, 2011.

[27] Iván Cantador, Peter Brusilovsky, and T Kuflik. Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011). *RecSys*, (HetRec):387–388, 2011.

[28] Pablo Castells, Jun Wang, Rubén Lara, and Dell Zhang. Workshop on novelty and diversity in recommender systems - DiveRS 2011. *Proceedings of the fifth ACM conference on Recommender systems - RecSys '11*, (DiveRS):393, 2011.

[29] Oscar Celma. *Music recommendation and discovery in the long tail.* PhD thesis, 2008.

[30] Òscar Celma and Perfecto Herrera. A New Approach to Evaluating Novel Recommendations. *In Proceedings of the 2008 ACM conference on Recommender systems - RecSys '08*, page 179, 2008.

[31] S Ceri, A Bozzon, and M Brambilla. *An Introduction to Information Retrieval*. Springer Berlin Heidelberg, 2013.

[32] Chung-Yi Chi, Ying-Shian Wu, Wei-rong Chu, Daniel C Wu, Jane Yung-jen Hsu, and Richard Tzong-Han Tsai. The power of words: Enhancing music mood estimation with textual input of lyrics. *2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*, pages 1–6, September 2009.

[33] Humberto Jesús Corona Pampín, Houssem Jerbi, and Michael P. O'Mahony. Evaluating the Relative Performance of Neighbourhood-Based Recommender Systems. *In Proceedings of the 3rd Spanish Conference on Information Retrieval*, pages 25–36, 2014.

[34] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[35] Roberto Cremonesi, Paolo, Koren, Yehuda, Turrin. Performance of Recommender Algorithms on Top-N Recommendation Tasks. *In Proceedings of the fourth ACM conference on Recommender systems - RecSys'10*, pages 39–46, 2010.

[36] Rudolf Cunningham, Prádraig and Green, Derek and Mayer. Unsupervised learning and clustering. *Machine learning techniques for multimedia*, pages 51–90, 2008.

[37] Scott Deerwester, ST Dumais, and TK Landauer. Indexing by latent semantic analysis. *JASIS*, 1990.

[38] Chris Ding and Xiaofeng He. Cluster merging and splitting in hierarchical clustering algorithms. *Data Mining, 2002. ICDM 2003. Proceedings.*, pages 139–146., 2002.

[39] Peter Sheridan Dodds and Christopher M. Danforth. Measuring the Happiness of Large-Scale Written Expression: Songs, Blogs, and Presidents. *Journal of Happiness Studies*, 11(4):441–456, July 2009.

[40] Ruihai Dong, Kevin McCarthy, and Michael P. O'Mahony. Towards an Intelligent Reviewers Assistant : Recommending Topics to Help Users to Write Better Product Reviews. *n Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, page 159.168, 2012.

[41] Ruihai Dong, Markus Schaal, Michael P. O'Mahony, and Barry Smyth. Topic Extraction from Online Reviews for Classification and Recommendation. *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 310–1316, 2013.

[42] J Stephen Downie. When Lyrics Outperform Audio for Music Mood Classification: A Feature Analysis. (Ismir):619–624, 2010.

[43] J Stephen Downie and Andreas F Ehmann. Lyric text mining in music mood classification. (Ismir):411–416, 2009.

[44] J. C. Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters, 1973.

[45] Douglas Eck and Paul Lamere. Automatic generation of social tags for music recommendation. *In Advances in neural information processing systems*, pages 385–392, 2008.

[46] Sandra Garcia Esparza, Michael P. O'Mahony, and Barry Smyth. CatStream : Categorising Tweets for User Profiling and Stream Filtering. *In Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 25–36, 2013.

[47] Andrea Esuli and Fabrizio Sebastiani. SENTIWORDNET : A Publicly Available Lexical Resource for Opinion Mining. *Proceedings of LREC*, pages 417–422, 2006.

[48] Daniel Fleder and Kartik Hosanagar. Blockbuster Culture's Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity. *Management Science*, 55(5):697–712, May 2009.

[49] Daniel M. Fleder and Kartik Hosanagar. Recommender systems and their impact on sales diversity. *Proceedings of the 8th ACM conference on Electronic commerce - EC '07*, page 192, 2007.

[50] R Foucard and Slim Essid. Exploring new features for music classification. *In Image Analysis for Multimedia Interactive Services (WIAMIS), 2013 14th International Workshop on (pp. 1-4)*, (1), 2013.

[51] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmid-Thieme. MyMediaLite: A free recommender system library. *In Proceedings of the fifth ACM conference on Recommender systems - RecSys'11*, pages 305–308, 2011.

[52] Mouzhi Ge, Carla Delgado-Battenfeld, and Jannach Diettmar. Beyond accuracy: Evaluating Recommender Systems by Coverage and Serendipity. *In Proceedings of the fourth ACM conference on Recommender systems*, pages 257–260, 2010.

[53] Jennifer Golbeck and James Hendler. Filmtrust: Movie recommendations using trust in web-based social networks. *In Proceedings of the IEEE Consumer communications and networking conference*, 96, 2006.

[54] Asela Gunawardana. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *The Journal of Machine Learning Research, 2009*, 10:2935–2962, 2009.

[55] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software, 2009.

[56] Byeong-jun Han, Seungmin Rho, Sanghoon Jun, and Eenjun Hwang. Music emotion classification and context-based music recommendation. *Multimedia Tools and Applications*, 47(3):433–460, August 2009.

[57] John Hannon, Mike Bennett, and Barry Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. *Proceedings of the fourth ACM conference on Recommender systems - RecSys '10*, page 199, 2010.

[58] Jon Herlocker, Joseph A. Konstan, and John Riedl. An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. *Information retrieval*, pages 287–310, 2002.

[59] Jon L. Herlocker and Joseph A. Konstan. An Algorithmic Framework for Performing Collaborative Filtering. *Proceedings of the 22nd 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, 1999.

[60] Jon L. Herlocker and Joseph A. Konstan. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[61] Xiao Hu. Music and mood: Where theory and reality meet. *Proceedings of iConference*, pages 1–8, 2010.

[62] Xiao Hu, J. Setphen Downie, Cyril Laurier, Mert Bay, and Andreas F. Ehmann. The 2007 MIREX Audio Mood Classification Task: Lessons Learned. 2008.

[63] Xiao Hu and J. Stephen Downie. Improving mood classification in music digital libraries by combining lyrics and audio. *Proceedings of the 10th annual joint conference on Digital libraries - JCDL '10*, page 159, 2010.

[64] Yajie Hu. A Music Recommendation System Based on User Behaviors and Genre Classification. 2012.

[65] Yajie Hu, Xiaoou Chen, and Deshun Yang. Lyric-based Song Emotion Detection with Affective Lexicon and Fuzzy Clustering Method. (Ismir):123–128, 2009.

[66] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. *In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272, 2008.

[67] P J Huber. Robust Statistics. *Statistics*, 60:1–11, 2004.

[68] David A Hull. Stemming Algorithms - A Case Study for Detailed Evaluation. *JASIS*, 47(1):70–84, 1995.

[69] Neil Hurley. Personalised Ranking with Diversity. *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, (1):379–382, 2013.

[70] Neil Hurley and Mi Zhang. Novelty and Diversity in Top-N Recommendation – Analysis and Evaluation. *ACM Transactions on Internet Technology*, 10(4):1–30, March 2011.

[71] Eibe Frank ian H. Witten. *Data Mining: Practical machine learning tools and techniques*. 2005.

[72] Tamas Jambor and Jun Wang. Goal-Driven Collaborative Filtering: a Directional Error Based Approach. *Advances in Information Retrieval*, pages 407–419, 2010.

[73] Gawesh Jawaheer, Martin Szomszor, and Patty Kostkova. Comparison of implicit and explicit feedback from an online music recommendation service. *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems - HetRec '10*, pages 47–51, 2010.

[74] Thorsten Joachims. *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. Springer Berlin Heidelberg, 1998.

[75] Stephen C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32:241–254, 1967.

[76] Pieter Kanters. *Automatic Mood Classification for Music*. PhD thesis, 2009.

[77] Alexandros Karatzoglou and Xavier Amatriain. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. *In Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86, 2010.

[78] George Karypis. Evaluation of Item-Based Top-N Recommendation Algorithms. *Proceedings of the tenth international conference on World Wide Web - WWW '01*, 2001.

[79] Ashraf M. Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. Multinomial naive bayes for text categorization revisited. *In AI 2004: Advances in Artificial Intelligence*, pages 488–499, 2005.

[80] Minho Kim and Hyuk-Chul Kwon. Lyrics-Based Emotion Classification Using Feature Selection by Partial Syntactic Analysis. *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, pages 960–964, November 2011.

[81] Youngmoo E Kim, Erik M Schmidt, Raymond Migneco, Brandon G Morton, Patrick Richardson, Jeffrey Scott, Jacquelin A Speck, and Douglas Turnbull. Music Emotion Recognition: A State of the Art Review. *ISMIR*, pages 255–266, 2010.

[82] Noam Koenigstein, Nir Nice, Ulrich Paquet, and Nir Schleyen. The Xbox recommender system. *Proceedings of the sixth ACM conference on Recommender systems - RecSys '12*, page 281, 2012.

[83] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M. Henne. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*, 18(1):140–181, July 2008.

[84] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonahtan L. Herlocker, Lee R. Gordon, and John Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, 40(3):77–87, 1997.

[85] Joseph a. Konstan and John Riedl. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22(1-2):101–123, March 2012.

[86] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. *In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008.

[87] Yehuda Koren and Robert Bell. Advances in Collaborative Filtering. In *In Recommender Systems Handbook*, pages 146–186. Springer US, 2011.

[88] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, pages 42–49, 2009.

[89] Katerina Kosta, Yading Song, György Fazekas, and Mark B. Sandler. A Study of Cultural Dependence of Perceived Mood in Greek Music. *ISMIR*, pages 317–322, 2013.

[90] SB Kotsiantis. Supervised machine learning: a review of classification techniques. *Informatica (03505596)*, 31:249–268, 2007.

[91] Vipin Kumar and Sonajharia Minz. Mood classifiaction of lyrics using SentiWordNet. *2013 International Conference on Computer Communication and Informatics*, pages 1–5, January 2013.

[92] L.I. Kuncheva. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281–286, 2002.

[93] Paul Lamere. Tutorial on music recommendation. *ISMIR*, 2007.

[94] Paul Lamere and Oscar Celma. Music recommendation and discovery revisited. *Proceedings of the fifth ACM conference on Recommender systems - RecSys '11*, page 7, 2011.

[95] Cyril Laurier, Jens Grivolla, and Perfecto Herrera. Multimodal Music Mood Classification Using Audio and Lyrics. *2008 Seventh International Conference on Machine Learning and Applications*, pages 688–693, 2008.

[96] Mark Levy and Klaas Bosteels. Music recommendation and the long tail. *1st Workshop On Music Recommendation And Discovery (WOMRAD), ACM RecSys.*, 2010.

[97] Tao Li, Mitsunori Ogihara, and Qi Li. A comparative study on content-based music genre classification. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval - SIGIR '03*, page 282, 2003.

[98] Thomas Lidy and A Rauber. Evaluation of Feature Extractors and Psycho-Acoustic Transformations for Music Genre Classification. *ISMIR*, 2005.

[99] Greg Linden, Brent Smith, and Jeremy York. Amazon.com Recommendations Item-to-Item Collaborative Filtering. *Internet Computing, IEEE*, (February), 2003.

[100] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based Recommender Systems: State of the Art and Trends. pages 73–105, 2011.

[101] Jose P. G. Mahedero, Álvaro MartÍnez, Pedro Cano, Markus Koppenberger, and Fabien Gouyon. Natural language processing of lyrics. *Proceedings of the 13th annual ACM international conference on Multimedia - MULTIMEDIA '05*, page 475, 2005.

[102] John Makhoul and Francis Kubala. Performance measures for information extraction. *In Proceedings of DARPA broadcast news workshop*, pages 249–252, 1999.

[103] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schutze. *An Introduction to Information Retrieval.* 2009.

[104] Rudolf Mayer, Robert Neumayer, and Andreas Rauber. Combination of Audio and Lyrics Features for Genre Classification in Digital Audio Collections Categories and Subject Descriptors. *In Proceedings of the 16th ACM international conference on Multimedia*, pages 159–168, 2008.

[105] Rudolf Mayer and Andreas Rauber. Musical genre classification by ensembles of audio and lyrics features. *In Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, (Ismir):675–680, 2011.

[106] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. *In AAAI-98 workshop on learning for text categorization .*, 752:41–48, 1998.

[107] Cory McKay, John Ashley Burgoyne, Jason Hockman, Jordan B.L. Smith, Grabiel Vigliensoni, and Ichiro Fujinaga. Evaluating the Genre Classification Performance of Lyrical Features Relative to Audio, Symbolic and Cultural Features. *ISMIR*, (Ismir):213–218, 2010.

[108] Cory McKay and Ichiro Fujinaga. Improving automatic music classification performance by extracting features from different types of data. *Proceedings of the international conference on Multimedia information retrieval - MIR '10*, page 257, 2010.

[109] Matthew R. McLaughlin and Jonathan L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. *In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–336, 2004.

[110] Kevin McNally, Michael P. O'Mahony, and Barry Smyth. Towards a Reputation-based Model of Social Web Search. *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 179–188, 2010.

[111] Sean McNee, John Riedl, and Joseph Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. *CHI'06 extended abstracts on Human factors in computing systems*, page 1101, 2006.

[112] Sean M. McNee, John Riedl, and Joseph A. Konstan. Accurate is not always good: How Accuracy Metrics have hurt Recommender Systems. *In CHI'06 extended abstracts on Human factors in computing systems*, pages 1097–1101, 2006.

[113] Prem Melville, Raymond J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. *AAAI/IAAI*, (July):187–192, 2002.

[114] Robin Van Meteren and Maarten Van Someren. Using content-based filtering for recommendation. *In Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop.*, 2000.

[115] Frank Meyer, Françoise Fessant, Fabrice Clérot, and Eric Gaussier. Toward a new protocol to evaluate recommender systems. *Workshop on Recommendation Utility Evaluation: Beyond RMSE (RUE 2012)*, 2012.

[116] Owen Craigie Meyers. *A Mood-Based Music Classification and Exploration System.* PhD thesis, Massachusetts Institute of Technology, 2007.

[117] Rada Mihalcea and Carlo Strapparava. Lyrics, music, and emotions. *In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, (July):590–599, 2012.

[118] George A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[119] Todd K. Moon. The expectation-maximization algorithm. *Signal processing magazine, IEEE*, 1996.

[120] Merrill Morris and Christine Ogan. The Internet as Mass Medium. *Journal of Communication*, 46:39–50, 1996.

[121] Tomoko Murakami, Koichiro Mori, and Ryohei Orihara. Metrics for Evaluating the Serendipity of recommendation lists. *New frontiers in artificial intelligence*, pages 40–46, 2008.

[122] Robert Neumayer and Andreas Rauber. *Integration of Text and Audio Features for Genre Classification in Music Information Retrieval*. Springer Berlin Heidelberg, 2007.

[123] Douglas W. Oard and Jinmook Kim. Implicit feedback for recommender systems. *In Proceedings of the AAAI workshop on recommender systems*, pages 81–83, 1998.

[124] Michael P. O'Mahony and Neil Hurley. Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology (TOIT)*, 4(4):344–377, 2004.

[125] François Pachet and Daniel Cazaly. A taxonomy of musical genres. *RIAO*, (April), 2000.

[126] Chris D. Paice. An Evaluation Method for Stemming Algorithms. *In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–50, 1994.

[127] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-Class Collaborative Filtering. *2008 Eighth IEEE International Conference on Data Mining*, pages 502–511, December 2008.

[128] Denis Parra and Xavier Amatriain. Walk the Talk Analyzing the Relation between Implicit and Explicit Feedback for Preference Elicitation. *In: User Modeling, Adaptation & Personalization*, pages 255–268, 2011.

[129] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering Categories and Subject Descriptors. *In Proceedings of KDD cup and workshop.*, pages 5–8, 2007.

[130] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. 2007.

[131] Geoffroy Peeters. A generic training and classification system for MIREX08 classification tasks: audio music mood, audio genre, audio artist and audio tag. *In Proceedings of the International Symposium on Music Information Retrieval (ISMIR'08).*, 2008.

[132] Bernhard Pfahringer. Winning the KDD99 classification cup: bagged boosting. *ACM SIGKDD Explorations Newsletter*, pages 4–5, 2000.

[133] Owen Phelan, K McCarthy, and Barry Smyth. Yokie: explorations in curated real-time search & discovery using twitter. *In Proceedings of the sixth ACM conference on Recommender systems*, pages 307–308, 2012.

[134] Martin Porter. Snowball: A language for stemming algorithms, 2001.

[135] Bruno Pradel, N Usunier, and P Gallinari. Ranking with non-random missing ratings: influence of popularity and positivity on evaluation metrics. *In Proceedings of the sixth ACM conference on Recommender systems*, pages 147–154, 2012.

[136] J.R. Quinlan. Induction of decision trees. *Machine learning*, pages 81–106, 1986.

[137] J.R. Quinlan. Simplifying Decision Trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[138] J.R. Quinlan. Decision trees and decision-making. *Systems, Man and Cybernetics, IEEE Transactions.*, 20(2):339–346, 1990.

[139] J.R. Quinlan. *C4.5: Programs for Machine Learning*, volume 1. 1993.

[140] J.R. Quinlan. Bagging, Boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 725–730, 1996.

[141] Rachael Rafter, Michael Paul O'Mahony, Neil Hurley, and Barry Smyth. What Have the Neighbours Ever Done for Us? A Collaborative Filtering Perspective. *User Modeling, Adaptation, and Personalization*, pages 355–360, 2009.

[142] Juan Ramos. Using tf-idf to determine word relevance in document queries. *In Proceedings of the First Instructional Conference on Machine Learning.*, 2003.

[143] Yang Eunho Ravikumar, Pradeep, Tewari, Ambuj. On NDCG consistency of listwise ranking methods. *In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval.*, 15:603–610, 2011.

[144] Paul Resnick, Neophytos Iacovou, and Mitesh Suchak. GroupLens: an open architecture for collaborative filtering of netnews. *In Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[145] Paul Resnick, Hal R Varian, and Guest Editors. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[146] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B Kantor. *Recommender Systems Handbook*. Springer US, Boston, MA, 2011.

[147] F James Rohlf. Adaptive Hierarchical Clustering Schemes. *Systematic Zoology*, 19:58–82, 1970.

[148] James A. Russell. A circumplex model of affect., 1980.

[149] Dymitr Ruta and Bogdan Gabrys. Classifier selection for majority voting. *Information Fusion*, 6(1):63–81, March 2005.

[150] Alan Said, Benjamin Kille, BJ Jain, and Sahin Albayrak. Increasing Diversity Through Furthest Neighbor-Based Recommendation. *Proceedings of the WSDM*, 2012.

[151] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-Based Collaborative Filtering Recommendation Algorithms. *Proceedings of the tenth international conference on World Wide Web - WWW '01*, pages 285–295, 2001.

[152] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of Dimensionality Reduction in Recommender System - A Case Study, 2000.

[153] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. *in Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[154] J. Ben Schafer, Joseph Konstan, and John Riedl. Recommender Systems in e-commerce. *In Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166, 1999.

[155] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '02*, page 253, 2002.

[156] Sebastian Schelter, Christoph Boden, Martin Schenck, Alexander Alexandrov, and Volker Markl. Distributed matrix factorization with mapreduce using a series of broadcast-joins. *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, (1):281–284, 2013.

[157] Fabrizio Sebastiani. Machine Learning in Automated Text Categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[158] Guy Shani and Asela Gunawardana. Evaluating Recommendation Systems. In *Recommender systems handbook*, chapter Evaluating, pages 257–297. Springer US, 2011.

[159] Upendra Shardanand and Pattie Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 210–217, 1995.

[160] C. Silva and B. Ribeiro. The importance of stop word removal on recall values in text categorization. *Proceedings of the International Joint Conference on Neural Networks, 2003.*, 3, 2003.

[161] Barry Smyth. Case-based recommendation. In *The adaptive web*, pages 342–376. Springer Berlin Heidelberg., 2007.

[162] Cees G. M. Snoek, Marcel Worring, and Arnold W. M. Smeulders. Early versus late fusion in semantic video analysis. *Proceedings of the 13th annual ACM international conference on Multimedia - MULTIMEDIA '05*, page 399, 2005.

[163] Yading Song, Simon Dixon, and Marcus Pearce. A survey of music recommendation systems and future perspectives. *9th International Symposium on Computer Music Modelling and Retrieval (CMMR 2012)*, (June):19–22, 2012.

[164] Yading Song, Simon Dixon, and Marcus Pearce. Evaluation of Musical Features for Emotion Classification. *ISMIR*, 2012.

[165] Yading Song, Simon Dixon, Marcus Pearce, and Andrea Halpern. Do Online Social Tags Predict Percieved or Induced Emotial Responses to Music? *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, 2013.

[166] Harald Steck. Training and Testing of Recommender Systems on Data Missing not at Random. *In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–722, 2010.

[167] Gábor Takács and Domonkos Tikk. Alternating least squares for personalized ranking. *Proceedings of the sixth ACM conference on Recommender systems - RecSys '12*, page 83, 2012.

[168] Auke Tellegen, David Watson, and Lee Anna Clark. On the Dimensional and Hierachical Structure of Affect. *Psychological Science*, 10(4):297–303, 1999.

[169] Andreas Toscher and Michael Jahrer. The bigchaos solution to the netflix grand prize. *Netflix prize documentation*, pages 1–52, 2009.

[170] Konstantinos Trohidis, G Tsoumakas, George Kalliris, and IP Vlahavas. Multi-Label Classification of Music into Emotions. *ISMIR*, pages 325–330, 2008.

[171] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *Speech and Audio Processing, IEEE transactions*, 10(5):293–302, 2002.

[172] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. *Proceedings of the fifth ACM conference on Recommender systems - RecSys '11*, page 109, 2011.

[173] Saúl Saul Vargas, Pablo Castells, and David Vallet. Intent-oriented diversity in recommender systems. *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*, (Sigir):1211, 2011.

[174] Manolis G. Vozalis and Konstantinos G. Margaritis. Applying SVD on item-based filtering. *5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*, pages 464–469, 2005.

[175] Chong Wang and David M. Blei. Collaborative Topic Modeling for Recommending Scientific Articles. *In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.*, pages 448–456, 2011.

[176] W. J. Wilbur and K. Sirotkin. The automatic identification of stop words, 1992.

[177] Dan Yang and Won-Sook Lee. Music Emotion Identification from Lyrics. *2009 11th IEEE International Symposium on Multimedia*, pages 624–629, 2009.

[178] M.-S. Yang. A survey of fuzzy clustering. *Mathematical and Computer Modelling*, 18(11):1–16, December 1993.

[179] Bei Yu. An evaluation of text classification methods for literary study. *Literary and Linguistic Computing*, 23(3):327–343, September 2008.

[180] Menno Van Zaanen and Pieter Kanters. Automatic Mood Classification Using TF* IDF Based on Lyrics. *ISMIR*, 2010.

[181] M Zhang and Neil Hurley. Avoiding monotony: Improving the Diversity of Recommendation Lists. *Proceedings of the 2008 ACM conference on Recommender systems - RecSys '08*, pages 123–130, 2008.

[182] T Zhang, R Ramakrishnan, and M Livny. BIRCH: An Efficient Data Clustering Databases Method for Very Large Databases. *ACM SIGMOD Record*, 25:103–114, 1996.

[183] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. *In Algorithmic Aspects in Information and Management.*, pages 337–348, 2008.

[184] Yong Zhuang, Wei-sheng Chin, Yu-chin Juan, and Chih-jen Lin. A Fast Parallel SGD for Matrix Factorization in Shared Memory Systems. (2):249–256.

[185] Cai-Nicolas Ziegler, Sean M. McNee, Joseph a. Konstan, and Georg Lausen. Improving Recommendation Lists Through Topic Diversification. *Proceedings of the 14th international conference on World Wide Web - WWW '05*, page 22, 2005.