

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

JONAS CALVI MEINERZ

Evaluating a prediction engine based on collaborative filtering and binary ratings

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Claudio Fernando Resin Geyer

Porto Alegre

2016

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Carlos Alexandre Netto

Vice-Reitor: Prof. Rui Vicente Oppermann

Pró-Reitor de Graduação: Prof. Vladimir Pinheiro do Nascimento

Diretor do Instituto de Informática: Prof. Luís da Cunha Lamb

Coordenador do Curso de Ciência da Computação: Prof. Carlos Arthur Lang Lisbôa

Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

Agradeço a todos os professores e colaboradores da Universidade Federal do Rio Grande do Sul, especialmente aos professores do Instituto de Informática e ao meu orientador, Claudio Geyer. Agradeço também à minha família e meus amigos pelo apoio. Obrigado.

Avaliação de um sistema de predição baseado em filtragem colaborativa e avaliações binárias

RESUMO

Existem alguns problemas com alguns dos mais famosos e utilizados sistemas de recomendação para filmes disponíveis nos dias de hoje na internet. Algumas das técnicas usadas por esses sistemas podem ter influências negativas no grau de sucesso de suas predições. As técnicas de filtragem baseada em conteúdo aplicadas pelos sistemas de recomendação podem introduzir tipos de falso-positivos que podem incomodar a alguns usuários desses sistemas. Vários dos sistemas disponíveis utilizam-se de certos elementos nas suas interfaces que podem confundir seus usuários e levá-los a interpretar erroneamente as escalas de avaliação de itens, o que pode invalidar uma parte importante dos dados que são coletados e utilizados para gerar predições para todos os usuários do sistema. Este trabalho questiona a efetividade de algumas das técnicas utilizadas por essas plataformas e reflete em maneiras mais simples e intuitivas para gerar predições e coletar avaliações sobre as experiências dos usuários com os itens do sistema. O sistema utiliza-se de uma escala simples com um sistema de avaliação binário (uma avaliação corresponde a “gostei” ou “não gostei”) para gerar predições probabilidade de usuários gostarem de filmes via filtragem colaborativa item-item. Uma implementação do modelo é apresentada e usada para avaliar a efetividade do modelo. Um experimento é conduzido de maneira a verificar a precisão do sistema de predições, assim como sua validade. Os resultados desse experimento para o sistema de avaliações e predições são descritos e analisados.

Palavras-chave: Sistemas de predição, Recomendação de filmes, Avaliações binárias, Filtragem colaborativa.

Movie recommendation engine powered by collaborative filtering and binary ratings

ABSTRACT

There are some problems with some of the most prominent recommendation systems for movies available today on the internet. Some of the techniques these systems use may lower the success rate of their predictions. The content-based filtering techniques used by said recommendation systems may introduce kinds of false positives that can be specially bothering to some users. Many of the systems available also employ user interface elements that can confuse users and lead them to misunderstanding the scale for rating items, which may taint one important part of the input that will be used to generate predictions for all the users of that given system. This work questions how effective this approach really is and reflects on more intuitive and simpler ways to generate predictions and gather input from users on their experiences with the items they are rating. Employing a simpler scale with a binary rating system and basing predictions solely on collaborative filtering, a prediction engine for movie recommendations is proposed, described, prototyped and analysed. The model hereby presented consumes binary user ratings (a rating can be either “liked” or “disliked”) to generate predictions of the likability of movies using item-based collaborative filtering for those users. An implementation of the model is presented and used in order to test the model's effectiveness. An experiment is conducted so that the accuracy of the prediction engine can be verified, as well as its validity. The results of an experiment for the rating system and prediction engine are laid out and analysed.

Keywords: Prediction engine. Movie recommendations. Binary ratings. Collaborative filtering.

LIST OF FIGURES

Equation 4.1: Similarity equation for items i, j	27
Equation 4.2: likability of item i to user u	28
Equation 4.3: Probability of user u liking item i	28
Algorithm 4.4: pseudocode representing computation of optimal recommendation for user	29
Algorithm 5.1: snippet of code representing User model	32
Algorithm 5.2: snippet of code representing Movie model	33
Algorithm 5.3: code that calculates the probability of a user liking a movie	34
Algorithm 5.4: method that yields optimal recommendation for user	34
Figure 6.1: Movies selected per release decade and movies selected per genre ...	38
Figure 6.2: Predictions vs Ratings - distribution	39

LIST OF TABLES

Table 2.1: Cat preferences catalogue.....	16
Table 2.2: Classification of cat activities	17
Table 2.3: Comparison of filtering algorithms	19
Table 6.1: Illustration of form used to gather data for validation experiment.....	35
Table 6.2: List of movie titles used to gather data for validation experiment.....	36

LIST OF ABBREVIATIONS AND ACRONYMS

IMDb	Internet Movie Database
CPO	Chief Product Officer
CF	Collaborative Filtering

SUMMARY

RESUMO.....	5
ABSTRACT.....	6
LIST OF FIGURES.....	7
LIST OF TABLES	8
LIST OF ABBREVIATIONS AND ACRONYMS.....	9
SUMMARY	10
1. INTRODUCTION.....	12
1.1. Structure of this text	14
2. RELEVANT CONCEPTS	15
2.1. The optimal recommendation problem	15
2.2. Information filtering system	15
2.3. Recommendation system	15
2.4. Collaborative filtering.....	16
2.4.1. Item-based collaborative filtering.....	16
2.5. Content-based filtering	17
2.6. Hybrid recommendation systems	18
2.7. Prediction engine	18
3. RECOMMENDATION SYSTEMS	19
3.1. Relevant movie recommendation engines in the web	20
3.1.1. Netflix.....	20
3.1.2. MovieLens	22
3.1.3. IMDb.....	22
3.2. Considerations about the state of the art in movie recommendations.....	22
4. MOVIE PREDICTION ENGINE FOR BINARY RATINGS WITH COLLABORATIVE FILTERING	24
4.1. User ratings.....	24

4.2. Information filtering	25
4.3. Prediction engine	26
4.3.1. Predicting likability.....	26
4.3.2. Item similarity computation.....	27
4.3.3. The likability of an item for a user.....	27
4.3.4. Complexity analysis.....	28
5. IMPLEMENTATION	30
5.1. Technologies used	30
5.1.1. Neo4j database	30
5.1.2. Google forms.....	31
5.1.3. Ruby programming language.....	31
5.2. Modelling the entities and their relationships	31
5.2.1. Users	32
5.2.2. Movies.....	32
5.3. Collecting and parsing data	33
5.4. Generating predictions	33
6. VALIDATION EXPERIMENT	35
6.1. Gathering Data	35
6.2. Results	39
7. CONCLUSION	41
REFERENCES	43

1. INTRODUCTION

One of the key ways through which industries try to keep their customers engaged is recommendations (MELVILLE; SINDWHANI, 2010). Even though recommendations have been playing a significant role in consumerism for a long time, the advent of the web has enhanced the presence and very likely increased the precision of recommendations a great deal when compared to mouth-to-mouth and conventional media recommendations (AAMIR; BHURSY, 2015). With the Internet, advertisers no longer had to bulk demographics into a loosely affiliated target audience, seldom hitting their mark, but could rather tailor their efforts to fit each individual singularly.

The Web itself is basically a web of recommendations, or so is how much of the money on the internet is made. Ads try to recommend products and services based on the habits of those browsing it. Streaming services aim to keep their subscribers engaged by looking at their streaming history and suggesting things their algorithms think their users will like. Social networks often recommend their users to “follow this profile, since you follow this other profile” or to “like this page, since you liked that other page”. Those examples illustrate the importance of intelligent recommendations, and how recommendations play a role in trying to captivate and further immerse the users of online platforms.

Some years ago, the streaming giant Netflix claimed that over 75% of the films watched through their platform were watched as a direct consequence of their recommendation system. In the past, they have also launched a contest of which the goal was to improve their recommendation algorithm (or, more precisely, lower the error rate of that algorithm by at least 10%) that awarded one million dollars to the contest's winners (AMATRIAN; BASILICO, 2012).

Evidently, a lot of effort is being put in recommendation system nowadays, but the most popular platforms for rating and streaming movies apply relatively similar techniques for filtering data. Hybrid recommendation systems are used by the leading platforms for streaming and recommending movies, but these systems have some drawbacks. These hybrid systems can comprise collaborative, content-based, demographic and other kinds of filtering. One of the problems with content-based filtering is that in many cases item classification cannot be automated, and some

items, such as works of art, can be quite complex and multi-layered and therefore hard to translate into a set of quantitative or objectively-describing tags. In the context of content-based filtering, when items are not thoroughly classified it is very likely that the number of wrong predictions will rise. The same applies to demographic filtering: if there is not information enough about the users, or each demographic is defined too broadly, the system may base its predictions on flukes. If however a set of items and users is accurately classified, research shows that it can improve the accuracy of predicted ratings as well when combined with collaborative filtering (ALI; PAZZANI, 1996).

But what if the ratings the users provide are not quite on par to their own experience? Then predicting how a user would rate an item is not at all that useful.

A suggested exercise to the reader of this text is to write down the name of 10 or so movies they have watched, and rate those movies on a scale from 1 (being the worst rating) and 5 (being the best rating). Let's say the first title on the reader's list is a *magnum opus*, a perfectly executed and one of the all time greatest works of art mankind has ever had the pleasure to come across. Let's say the reader rates that film as a 5. Continuing rating the films on that list, the reader may come across a decent and pleasant film, which they do not regret watching, but may catch themselves rating that film as a 2, thinking that it couldn't possibly be half as good as the *magnum opus* he just rated... But would the reader really watch a movie that was recommended to him at a rating of just 2 out of 5 stars?

The little fictional example is a simple example of how difficult it can be to rate subjective and complex things when there is room for the scale to be interpreted as a linear numeric interval scale, and that even if a recommendation engine tells someone that they will probably rate a given item 2 out of 5, that does not mean consuming that item or watching that film would necessarily be an unpleasant experience. That depends on how the user interpreted the scale to generate their assessments. Therefore, it presents an example on how users of a given recommendation system can be misguided by their own ratings due to the fact that a scale that can be perceived as numeric can be not trivial. In some cases, though, such as Yelp¹ and TripAdvisor², the platforms show a label indicating what their

¹ <http://www.yelp.com/>

² <https://www.tripadvisor.com/>

rating actually mean. In other cases, such as Netflix³, that indication is hidden somewhere in their information pages. MovieLens⁴, a movie recommendation platform by researchers from the University of Minnesota, does not have a particular meaning for ratings, or it is never explained.

The problems cited above, regarding inaccurate classification of items and demographics, as well as the potential misinterpretations that can happen by users when rating movies on a scale that can be misperceived depending on how it is presented on the user interface, are inspirations to seek a different and more intuitive approach when it comes to rating and recommending movies.

The approach hereby explained consists in an attempt to simplify ratings, trying to make them more meaningful and intuitive: a user either likes or dislikes a movie. This is not a new technique and is used by various recommendation systems, but it's different from the most frequently used by the mainstream movie platforms on the internet. From there on, collaborative filtering techniques are applied and the likelihood of a user liking a given movie is predicted. Besides proposing and presenting experiments on the collaborative-filtering-based prediction system for recommending movies, this work will also discuss relevant works in the field and prominent recommendation systems in the web, as well as basic concepts necessary to understand them.

1.1. Structure of this text

Chapter 2 will present relevant concepts for the development and understanding of this dissertation. Chapter 3 will go deeper into relevant information about mainstream recommendation systems for movies. Chapter 4 explains in detail the proposed approach to movie recommendations, as well as the motivation behind it. Chapter 5 discusses an implementation for the proposed algorithm, while Chapter 6 presents an experiment for validating the model described in Chapter 4 that uses Chapter 5's implementation. Chapter 7 summarises a conclusion for this work.

³ <https://www.netflix.com/>

⁴ <https://www.movielens.org/>

2. RELEVANT CONCEPTS

This chapter lays out concepts that are relevant for the understanding and reproduction of this work. Some basic understanding in mathematics, statistics, logics, computer science and programming are already required in order to fully understand the explanations and idioms employed in this dissertation. In order not to make this text overly verbose, this chapter will brush the surface of topics that are important to the development of this work. For further information and deeper explanations of the cited topic, the reader is encouraged to consult the titles listed in the reference sheet at the end of this text.

2.1. The optimal recommendation problem

The recommendation problem is to find the most useful (optimal) item belonging to a finite set of items for a given user. A user has their own utility function. The optimal recommendation recommends the item which yields the highest value when used as an input for a user's utility function (ADOMAVICIUS; TUZHILIN, 2005).

2.2. Information filtering system

An Information filtering system (HANANI, 2001) is an automated system that filters irrelevant or undesired information or data. Examples of usage: recommendation systems, spam filters, search engines.

2.3. Recommendation system

Recommendation systems (also sometimes referred as "recommender systems") are systems developed to try and predict preferences of a user (or subject) within a finite set of items (ADOMAVICIUS; TUZHILIN, 2005). The usage of this kind of system is widely spread around the Internet, from movie streaming services to e-commerce. This class of systems is a subclass of the system class denoted "information filtering" (MELVILLE; SINDHWANI, 2010).

2.4. Collaborative filtering

In the context of recommendation systems, collaborative filtering is a technique where multiple sources of data are used to place an educated guess about a particular data point that is missing from a dataset (TAKÁCS et al., 2009). For example, let us imagine that the following dataset was collected following a very serious and thorough research project (it was in fact made up to illustrate this example):

Table 2.1: Cat preferences catalogue

Cat name	Eating cat food	Sleeping belly up	Playing w/ thread ball	Jumping up and down
Furry	Likes	Dislikes	Likes	Likes
Black Paw	Likes	Dislikes	Likes	Likes
Wormtail	Dislikes	Likes	Dislikes	Dislikes
Mikhail	Likes	?	Likes	Likes

Source: this work's author

Let us now imagine that the researchers that comprise the fictitious very serious and thorough project want to use collaborative filtering to determine where Mikhail, the cat, likes sleeping belly up. All that they can infer from the dataset is that Mikhail shares all its known preferences with Furry and Black Paw, and its known preferences are the opposite of Wormtail's. If collaborative filtering is applied to try and predict the preference of Mikhail regarding sleeping belly up, the predicted preference would be that Mikhail dislikes sleeping belly up. The reason for that prediction is that the rest of the dataset implies that cats that like eating cat food and playing with thread ball do not like sleeping belly up.

2.4.1. Item-based collaborative filtering

There are multiple techniques that fall under the genre of collaborative filtering. Item-based collaborative filtering computes the similarity between an item

the user has not rated and the items the user has already rated. The prediction will be the outcome of the weighted average of that item's similarity with the items the user has rated (SARWAR et al., 2001). The similarity between two items can be computed via a similarity function. Examples of those are cosine-based similarity, correlation-based similarity and adjusted cosine similarity (SARWAR et al., 2001).

2.5. Content-based filtering

Content-based filtering is, very much like collaborative filtering, a technique used by recommendation systems. The difference between the two is that while collaborative filtering focuses solely on the subjects' relation to a given item (i.e. Wormtail dislikes jumping up and down), content-based filtering breaks down items into categories and judges the subjects experiences with an item based on that item's characteristics (PAZZANI, 1999). Let us bring back the fake cat experiment and suppose that the researchers classified the cat activities as follows:

Table 2.2: Classification of cat activities

Activity	Tags
Eating cat food	Moving a bit, Feeding.
Sleeping belly up	Staying still, Resting.
Playing w/ thread ball	Moving a lot, Fun.
Jumping up and down	Moving a lot, Fun.

Source: this work's author

Based on those tags, the researchers would then try to predict Mikhail's opinion on sleeping belly up, but this time they will only regard Mikhail's own experience with other activities. A content-based filtering algorithm when presented the cat preference catalogue might conclude that since Mikhail likes activities that involve moving, Mikhail might as well dislike activities that involve staying still, therefore disliking sleeping belly up. Note that even though the predicted outcomes were the same for both examples (collaborative filtering and content-based filtering), the reasoning behind the prediction was different, and these two techniques are not guaranteed to yield similar results.

2.6. Hybrid recommendation systems

Recommender systems that make use of more than one filtering technique are called “hybrid recommendation systems” (MELVILLE; SINDHWANI, 2010). There are several examples of information filtering techniques, such as collaborative filtering, content-based filtering, demographic filtering, and others. Hybrid recommendation systems are sometimes deemed by literature to be a more precise approach to recommendation systems when compared to simple filters (AAMIR; BHUSRY, 2015). Having said that, they usually require more on classifier agents to enhance the dataset. Those agents may be users chipping in with more data about themselves and their experience rather than a simple rating, or they may depend on people dedicated to classifying the dataset in some way.

2.7. Prediction engine

The prediction engine is the part of the recommendation system that is responsible of generating a prediction based on the existing data and models (DELGADO; ISHII, 2009). It is the part of recommendation systems that usually holds the definition of utility functions, graph theory and machine learning implementations (when applicable).

3. RECOMMENDATION SYSTEMS

Recommendation is a natural form of social interaction and discovery. There were already research fields tackling problems related to recommendation systems prior to 1992, when first commercial recommendation system, called *Tapestry* (GOLDBERG et al., 1992), coined the phrase “collaborative filtering” and sprung new life into this area of research.

With the interest from industry and academy on the subject growing hand in hand, much has been produced in the area during the past 25 years and some systems can achieve great accuracy in predicting tastes and ratings for its users.

There are multiple genres of filtering algorithms used by prediction engines, the most popular ones being collaborative filtering and content-based filtering, which are often applied together in hybrid recommendation systems (AAMIR; BHUSRY, 2015), and within those genres there are multiple variations and a variety of subgenres of algorithms (SU; KHOSHGOFTAAR, 2009). For a more detailed explanation about the three subcategories of recommendation systems, the reader is advised to go back to Chapter 2 of this text.

Note that each filtering genre is subdivided in many other different techniques that can bring different sets of perks and shortcomings. The table below presents and compares characteristics that are usually a part of the genres as a whole.

Table 2.3: Comparison of filtering algorithms

	Advantages	Disadvantages
Collaborative filtering	<ul style="list-style-type: none"> • Require little or no data from the recommendable items belonging to the data set. • Good scaling for co-rated items. • Extending the dataset is very simple. 	<ul style="list-style-type: none"> • Low performance when dataset is sparse. • If one user’s taste is radically different from most of the users in the system, predictions may be very inaccurate.

	Advantages	Disadvantages
Content-based filtering	<ul style="list-style-type: none"> • Performance does not depend on dataset sparsity. • Powerful for recommending items that can be easily and objectively classified. 	<ul style="list-style-type: none"> • Classifying complex items may require a lot of human effort. • Subjective and complex items may well be wrongly recommended.
Hybrid	<ul style="list-style-type: none"> • More accurate predictions • Can overcome some of the disadvantages of the simpler approaches 	<ul style="list-style-type: none"> • Expensive implementation • Usually requires a lot of human effort to enrich dataset.

Source: assembled by this work's author

Since the focus of this work lies on movie recommendations, it is worth mentioning that most recommendation engines around work in a similar way. The following section presents some of the most relevant or widely-used movie recommendation engines nowadays.

3.1. Relevant movie recommendation engines in the web

When it comes to classifying movies there are several limitations to what can be done using neural networks and machine learning without relying on extensive human input. Some nuances are very hard to be picked up by an artificial intelligence and one good example to that is Netflix's tagging system, which relies on data inserted by human hands in order to label and classify the movies in their database. MovieLens, on the other hand, relies on users doing the expert's work of classifying movies, which moves the labour from their own payroll to crowdsourcing.

3.1.1. Netflix

"Netflix" may likely be the first word to pop in anyone's mind if they were asked to engage on an exercise of free association between the words "internet" and "movies". Inarguably a powerhouse in the streaming industry, Netflix takes its recommendation engine very seriously, having claimed that over 75% of their content was accessed as a direct consequence of it (AMATRIAIN; BASILICO, 2012). They

also sponsored a competition of which the goal was to produce a recommendation engine with a success ratio superior to the one they had, and it rewarded the winners with the sum of one million dollars (AMATRAIN; BASILICO, 2012).

Neil Hunt, Netflix Chief Product Officer responds to a question posted on Quora⁵ (“How does Netflix tag their movies?”, 2011):

"Expert reviewers watch the movies and assign tags from a predefined list. Some are binary, some are quantitative (1-5 scale). A variety of rules utilize tag data to assemble the "micro genres" ("Foreign movies featuring a strong female lead"), the "like" lists of similar movies, and as an input to predicting interest, and hence assembling pages to present."

In January 2016, Hunt also told Business Insider that Netflix is studying more intuitive approaches to rating, such as a “like/dislike” rating system. He indicated that people were confused by the current rating system and were sometimes rating movies like critics instead of basing their ratings on how much they enjoyed watching that movie, which according to Hunt, is the goal of the rating system.

Also in Quora, an explanation of Xavier Amatrain can be found for the question “How does Netflix movie recommendation algorithm work?”. He answers:

“(…) there are many recommendation algorithms at Netflix. (…) there are two algorithms that are being used in production right now: Restricted Boltzman “ - he mispronounces Boltzmann’s name - “Machines (RBM) and a form of Matrix Factorization. (…) There are many other recommendation algorithms from personalized ranking to page optimization that make up the Netflix recommendation system. (…)”

Netflix also makes use of content-based filtering, as mentioned in the blog post “Netflix Recommendations: Beyond 5 Stars (Part 2)” (AMATRAIN; BASILICO, 2012), which when added to the use of Restricted Boltzmann Machines for collaborative filtering (SALAKHUTDINOV; MNIH; HINTON, 2007), classify their recommendation engine as a hybrid one.

For what it’s worth, other streaming services also apply similar methods to their recommendation engines. Netflix was chosen as a case study for the amount of knowledge out there about the platform (such as their tech blog⁶ and their staff

⁵ <https://www.quora.com/>

⁶ <http://techblog.netflix.com/>

answering to questions on Quora) and also for its huge share of the movie and tv series streaming market.

3.1.2. MovieLens

MovieLens is a movie recommendation web platform run by GroupLens, a research lab at the University of Minnesota. They use the data collected on the MovieLens platform to fuel further research on recommendation systems. Their web platform offers many different prediction algorithms for the user to choose from.

MovieLens applies a similar rating logic as Netflix, asking for ratings from 0.5 to 5, but it differs a bit from Netflix by asking that users weight what kinds of movies they like before they can rate the actual movies. This works by the users rating bulks of tags. One reason why this technique may be applied is to try and avoid the cold start problem (MELVILLE; SINDWHANI, 2010).

3.1.3. IMDb

Worth mentioning for its traffic and huge database, IMDb is one of the biggest forums on movies and tv shows on the internet. Their rating system is very similar to the ones in Netflix and MovieLens, but it ranges from 0 to 10 instead. They also apply collaborative filtering and content-based filtering in order to generate predictions, as mentioned in their “Personalized Recommendations Frequently Asked Questions”⁷.

3.2. Considerations about the state of the art in movie recommendations

Different to lifestyle and social recommendations, the efforts of research and industry seem hardly focus on a different paradigm of user interaction with recommendation systems when it comes to recommending movies. Twitter’s⁸ “Who to follow” algorithm (GUPTA et al., 2013) is a good example of a massively used and highly successful recommendation algorithm that does not require its users to rate

⁷ http://www.imdb.com/help/show_leaf?personalrecommendations

⁸ <https://twitter.com/>

pages with arbitrary values in order to receive good recommendations. Foursquare's⁹ rating system only requires users to "like" or "dislike" places. The examples cited are by no means proof that a simple "yes or no" rating is better than a scale represented by stars or numbers, but it gives room for one to wonder whether more literal ratings would lead to more meaningful predictions, and therefore more useful recommendations.

Another possible downside with the content-based filtering approach, which is used by the three recommendation engines mentioned in section 3.1, but more evidently so by MovieLens and Netflix, is the kind of false positives it brings. A good rating for "The Dark Knight" in MovieLens or Netflix will bring loads of recommendations of super hero comic-book adaptation films, even though "The Dark Knight" is inherently different than most super hero comic-book adaptation films. False positives are unavoidable, even more so for users who fall under the labels of *grey sheep* or *black sheep* (SU; KHOSHGOFTAAR, 2009), but perhaps different approaches could alleviate some of the frequency of those kinds of false positives.

⁹ <https://foursquare.com/>

4. MOVIE PREDICTION ENGINE FOR BINARY RATINGS WITH COLLABORATIVE FILTERING

As mentioned at the end of Chapter 3, there can be some downsides to the way some relevant movie recommendation platforms collect user ratings, make predictions, and therefore recommend items. This chapter brings suggestions to address some of the problems discussed in the previous one, discussing and presenting a model for a prediction engine to be used by movie recommendation systems.

4.1. User ratings

Stars have long been present in the movie rating world. It works quite well for movie critics to evaluate pictures, but when it comes to normal people in a streaming service it may become confusing depending on how it is presented. And the very reason it confuses people is that it makes them feel like critics and judge the works of art like critics, instead of rating according to their experience. Going back some years, movie-rating systems such as Netflix's and MovieLens at least offered a direct translation of what one-star and four-stars meant (they used to mean, "I didn't like it" or "I really liked it", respectively, or something close to that), but now the interface leaves users simply with the stars and no labels, which enhances the possibility of users getting confused.

In January 2016, Netflix's CPO Neil Hunt told Business Insider that this user behaviour, when they rate for "quality" instead of "enjoyment", is causing strange anomalies in their data (MCALONE, 2016). But if you're not supposed to judge it for "quality", why use the same metric and symbolism used for when critics do so? And why hide the meanings of ratings somewhere in the help pages? It's only natural that users would be confused. To avoid that kind of problem and provide a more intuitive, unambiguous experience for users, and expecting that meaningful ratings can be more easily converted into meaningful recommendations, this work adopts a very simple user rating system.

A user has 4 options when it comes to a movie, and she must pick only one. The four options presented to the user for evaluating her experience with a film are the following:

- A. I have not watched it
- B. I have watched it and neither like nor dislike it
- C. I have watched it and I like it
- D. I have watched it and I dislike it

The rating system described above is not at all ambiguous or subjective, but it instead allows users to encapsulate the subjectivity of complex works of art into simple statements that translate their experiences with those works of art. The like/dislike scale is used by several recommendation systems and has been present in literature for some time (PAZZANI, 1999). There should be the acknowledgement that if all interfaces could be perceived correctly by users, avoiding users to rate items like critics when they should be assessing their enjoyment, a more nuanced scale would potentially be a better metric due to its nature, providing more detail in how much a user has liked or disliked a given movie, possibly enhancing predictions by providing a richer way to calculate the correlation between the movies in the dataset and the users' experiences with those movies. Unfortunately, as mentioned a couple of paragraphs back, there is evidence that simply displaying a scale may lead users to assess the quality of the movies they have watched, which can cause anomalies to the dataset.

4.2. Information filtering

As a mean to develop the prediction engine powered by the rating system described above, it was decided that the information filtering technique used would be CF (collaborative filtering). In this case, only non-neutral ratings towards movies will be taken into account (likes and dislikes). This approach allows for quick implementation and does not require experts to analyse and classify entries on the dataset extensively, making it a very good candidate for a small experiment. There is also evidence that collaborative filtering can be more effective than content-based

filtering, as shown in various research experiments. Two experiments that exemplify that evidence are “Recommendation as Classification: Using Social and Content-Based Information in Recommendation” (BASU; HIRSH; COHEN, 1998) and “A Framework for Collaborative, Content-Based and Demographic Filtering” (PAZZANI, 1999). With that in mind, it’s worth mentioning that this filtering technique does not cover other important issues in recommendation systems such novelty (KAPOOR et al., 2015) and cold-start (SU; KHOSHGOFTAAR, 2009).

The CF approach chosen was item-based collaborative filtering. When compared to user-based CF, item-based is showed yield more precise predictions and therefore better recommendations (DESHPANDE; KARYPIS, 2004).

4.3. Prediction engine

In the previous sections (4.1 and 4.2), the basic features of the prediction engine were laid out. In section 4.1 it is specified how the data that will be used to train the prediction engine is modelled, while in section 4.2 it is specified the general approach to information filtering used to power predictions. In this section, the prediction engine will explained in more detail, and its computational complexity will be analysed.

4.3.1. Predicting likability

As previously mentioned, the type of collaborative filtering used to generate recommendations chosen is the so called “item-based collaborative filtering”. Item-based collaborative filtering works by recommending to the user items that are similar to the ones the user has already consumed and enjoyed (SARWAR et al., 2001).

In that context, instead of answering the question “How likely is it that *User U* would enjoy watching *Movie M*?” the prediction engine tries to answer a proxy question: “How similar is *Movie M* to the movies that *User U* has watched and liked, and how different is *Movie M* to the movies that *User U* has watched and disliked?”. This approach, like content-based filtering, focuses on the affinity of items among themselves, but note that item-based collaborative filtering looks into the similarities

of experiences users had with those items, while content-based filtering looks for similarities in the characteristics of the items.

4.3.2. Item similarity computation

The first step in order to determine how similar two items are through collaborative filtering is by isolating all users that had experiences with those two items (i.e., users that rated both items). Then there are many ways to compute the similarity between those two items. For this work, the chosen method is *conditional probability-based similarity*. The conditional probability-based similarity is essentially the probability of two users giving equivalent ratings to the same items (DESHPANDE; KARYPIS, 2004). *Agreeing* ratings are when multiple ratings for the same items converge. In other words, ratings from different users *agree* when they have the same value for a given item. Otherwise, those ratings are *disagreeing*.

Let i, j be two different items. Let U be the subset of users who have rated both items i and j . Let s be the number of similar (agreeing) ratings by all users in U for items i and j . Let d be the number of different (disagreeing) ratings by all users in U for items i and j . The similarity equation for i and j is described by:

Equation 4.1: Similarity equation for items i, j

$$sim(i, j) = \frac{s}{s + d}$$

Source: this work's author

4.3.3. The likability of an item for a user

Being able to compute the similarity of two items allows us to move on to the next step: determining the probability that a user will like a given item. In other words, computing the likability of an item for that user. We want to recommend items that are similar to the items that the user has liked and to not recommend items that are similar to items the user has disliked.

Let i be an item in the item set. Let u be a user that has not yet rated i , for which we are interested in predicting the likability of i . Let L be the set of items the user has liked and D be the set of items the user has disliked. The function below is used to calculate the likability of item i to user u :

Equation 4.2: likability of item i to user u

$$\sum_{l \in L} sim(i, l) - \sum_{d \in D} sim(i, d)$$

Source: this work's author

The function described above ranges from -1 (when i is utterly similar to all items the user has disliked) to 1 (when i is utterly similar to all items the user has liked). In order to read that as a probability of the user liking that item, we can do some simple operations. Let $P(u, i)$ be the probability that user u would like item i . The probability function can be described as follows:

Equation 4.3: Probability of user u liking item i .

$$P(u, i) = \frac{1 + \sum_{l \in L} sim(i, l) - \sum_{d \in D} sim(i, d)}{2}$$

Source: this work's author

And that concludes the prediction engine. In order to transform that into a recommendation system, all there is needed is to calculate the probability for users to like every single item they have not rated and return the n items with the highest probability of the user liking them.

4.3.4. Complexity analysis

The pseudocode below represents the extraction of the item with the highest likability from the prediction engine for a given user. This would be the equivalent of

an optimal recommendation of size 1 amongst the items not yet consumed by that user.

Algorithm 4.4: pseudocode representing computation of optimal recommendation for user

```

1. map unrated_item in user.unrated_items do
2.   map rated_item in user.rated_items do
3.     similarity = similarity(unrated_item, rated_item)
4.     if user.liked_items.includes?(rated_item)
5.       return similarity
6.     else # user has disliked rated_item
7.       return -similarity
8.     end
9.   end
10. end.max

```

Source: this work's author

Let us begin with the loop that ranges from line 2 to line 9. On line 3, we calculate the similarity between two items. In order to do that, we have to iterate through all users (except the user for which we are computing the prediction) and check whether they have rated those two items, and what was the rating. If there are n users in the database and i items, in the worst case we will iterate through $(n-1)*i$ ratings. Let $j = \max(n, i)$. That leaves the *similarity* operation with a complexity of $O(j^2)$. After that, we need to check if our subject has liked the current item. The complexity of that is $i-1$, therefore the complexity of the loop ranging from lines 2 to 9 is $O(j^2+i-1)$, which equals $O(j^2)$. The loop ranging from lines 1 to 10 repeats the loop ranging from lines 2 to 9 i times, which makes the worst case for that loop $O(j^3)$ in the occasion where $j = i$. The final operation costs $O(i)$ and therefore does not affect the complexity of the algorithm.

5. IMPLEMENTATION

All the code that comprise the implementation of the prediction engine for movie recommendations can be found at <<https://github.com/jmeinerz/siwi>>. In order to make this text easier to understand, some details will be abstracted from the dissertation and the code presented should be interpreted as pseudocode, but it will presented as close as possible to the original version without impairing the understandability of this text.

5.1. Technologies used

The following sections will list and justify the technologies used to implement a version of the recommendation engine. The choices were made aiming for a simple and understandable prototype.

5.1.1. Neo4j database

Since a graph was used to model and describe the universe containing movies, users and their relationships, the most intuitive choice for modelling the database would be elements in a graph.

Neo4j¹⁰ is described by their creators as "a highly scalable native graph database that leverages data relationships as first-class entities" ("Neo4j: The world's leading graph database", 2016). It offers an intuitive way to model the entities for this work, which allows for readability of the codebase and quick implementation.

Achieving high computing performance was not the immediate purpose of this work and therefore no measurements or comparisons were drawn between the use of the Neo4j graph database versus the use of tables or matrices to store data. In order to decide whether this choice is ideal for an eventual large-scale implementation, benchmarks to evaluate the feasibility of using Neo4j as the database would be a requirement.

¹⁰ <http://neo4j.com>

5.1.2. Google forms

Google forms¹¹ was chosen due to the ease to create and respond to forms. The forms can also be exported to a .csv file, which was very useful when it was time to parse the responses. The form created was open to the public, accessible for anyone in possession of the form URL.

5.1.3. Ruby programming language

Ruby¹² is very easy to read and be understood by anyone who knows how to program. Reading ruby code is almost like reading pseudocode, which makes it look good in stuff like an undergraduate thesis. It is also arguably fast to learn and comes with many built-in utilities for parsing files and text, which came in very handy for implementing the recommendation system prototype.

For what it's worth though, Ruby is not the fastest programming language around. This means that choosing this particular language for a large scale deployment of the recommendation engine may not be ideal.

5.2. Modelling the entities and their relationships

The engine operates based on two entities: users and movies. Those will be the nodes on our graph database. There are two types of edges connecting the nodes of our graph, the first one being a user-to-movie edge, weighted based on the referenced user opinion on that given movie and the second one being a movie-to-movie edge, representing how likely the connected movies are of receiving similar ratings from a given user.

In order to translate this universe into code and connect our entities with our database, the library Neo4jrb was used. It is an utility for translating Neo4j database models into Ruby objects. The tool is open sourced and available at Neo4jrb's GitHub repository¹³.

¹¹ <https://docs.google.com/forms/>

¹² <https://www.ruby-lang.org/en/>

¹³ <https://github.com/neo4jrb/neo4j>

The following sections will explain and exemplify the modelling of each one of our entities and relationships.

5.2.1. Users

As previously mentioned, users have relationships with movies. We represent that as an edge going from a user to a movie. That edge has one of two weights: “liked”, for when that user has liked the movie or “disliked”, for when the user disliked that movie. For readability and, we represent that on the code as users having two types of outbound edges: “liked”, and “disliked”, representing respectively the two weights just mentioned. The following snippet of code illustrates the way a user is modelled into code in the engine:

Algorithm 5.1: snippet of code representing User model

```
class User
  include Neo4j::ActiveNode # this is a node in the db

  has_many :out,           # has many outbound edges
           type: 'like',   # of type liked
           model_class: :Movie # pointing to Movie nodes

  has_many :out,           # has many outbound edges
           type: 'dislike', # of type disliked
           model_class: :Movie # pointing to Movie nodes

end
```

Source: this work's author

5.2.2. Movies

The modelling of movies is analogous to the one for users, except that we keep connections to other movies for when they are rated by the same user. One “alike” connection means they received the same rating from a user, one “unlike” means the opposite of that. A method that implements the similarity function described on Chapter 4 is also added. The following algorithm exemplifies the

modelling of the Movie entity, as well as its connections and algorithm to determine the similarity between two movies.

Algorithm 5.2: snippet of code representing Movie model

```
class Movie
  include Neo4j::ActiveNode

  property :title, type: String

  has_many :in, type: 'like', model_class: :User
  has_many :in, type: 'dislike', model_class: :User

  has_many :both, :alikes, rel_class: :Alike
  has_many :both, :unlikes, rel_class: :Unlike

  def similarity(other_movie)
    similar = alikes.where(title: other_movie.title).count
    different = unlikes.where(title: other_movie.title).count
    total = similar + different
    return 0.5 if total == 0
    similar.to_f / total
  end
end
```

Source: this work's author

5.3. Collecting and parsing data

An online form was set up using Google Forms to collect data from movie watchers. The data collection will be detail in Chapter 6, so that part will be ignored for now. In order to parse the responses and use them as inputs for the recommendation engine, a simple script was written using the Ruby programming language.

5.4. Generating predictions

The algorithm described below calculates the probability of a user liking a movie. As described by the model in the previous chapter, the return value of the function will be the difference between the similarity of that movie among the ones

the user has liked and the similarity between the ones the user has disliked. That value is shifted to fit between 0 and 1 before being returned.

Algorithm 5.3: code that calculates the probability of a user liking a movie

```
def like_probability(user, movie)
  avg_liked_similarity = user.liked.map do |liked_movie|
    movie.similarity(liked_movie)
  end.inject(:+) / liked.size

  avg_disliked_similarity = user.disliked.map do |disliked_movie|
    movie.similarity(disliked_movie)
  end.inject(:+) / disliked.size

  (1 + avg_liked_similarity - avg_disliked_similarity).to_f / 2
end
```

Source: this work's author

Finally, we can generate the optimal recommendations for a user, consisting of the n items the user will most likely enjoy watching.

Algorithm 5.4: method that yields optimal recommendation for user

```
def optimal_recommendation(user, n)
  options = Movie.all - user.liked - user.disliked
  options.map do |movie|
    { movie: movie, like_probability: like_probability(user, movie) }
  end.sort_by { |m| m[:like_probability] }.last(n)
end
```

Source: this work's author

We now have the tools to make recommendations. It's now time to verify how precise those recommendations are.

6. VALIDATION EXPERIMENT

In order to test the validity and precision of the prediction engine, an experiment was developed. In this chapter, the experiment and its results will be described.

Following the premise that data originated from sources such as Netflix, MovieLens and IMDb might not be totally trustworthy due to users misinterpreting the different versions of the Likert-type scale (LIKERT, 1932) presented by the interfaces of those systems, it was decided that their databases would not power this research experiment. Therefore, an online survey was set up in order to gather inputs formatted such as the input of this work's prediction engine requires. The next section explains in further detail how this survey was set up.

6.1. Gathering Data

A survey containing a table analogous to the one below was presented to respondents in the format of a form:

Table 6.1: Illustration of form used to gather data for validation experiment.

WHAT IS YOUR EXPERIENCE WITH THE FOLLOWING MOVIES?

Movie title	I haven't watched it / Neither like nor dislike it	I liked it	I didn't like it
Title 1			
Title 2			
...			
Title N			

Source: this work's author

That table would allow users to rate their experience with a list of movies (that list will be displayed further in this section). The respondents would then check one column per row. The actual set of movie titles included in the form is listed as follows:

Table 6.2: List of movie titles used to gather data for validation experiment

101 Dalmatians (1961)	Goldfinger (1964)	One Flew Over the Cuckoo's Nest (1975)	The Passion of the Christ (2004)
2012 (2009)	Gone With the Wind (1939)	Pinocchio (1940)	The Poseidon Adventure (1972)
Airport (1970)	Gravity (2013)	Pirates of the Caribbean: The Curse of the Black Pearl (2003)	The Robe (1953)
Aladdin (1992)	Grease (1978)	Raiders of the Lost Ark (1981)	The Rocky Horror Picture Show (1975)
Alice in Wonderland (2010)	Guardians of the Galaxy (2014)	Rocky (1976)	The Sixth Sense (1999)
American Graffiti (1973)	Hancock (2008)	Skyfall (2012)	The Sound of Music (1965)
Around the World in 80 Days (1956)	Harry Potter and the Sorcerer's Stone (2001)	Sleeping Beauty (1959)	The Sting (1973)
Avatar (2009)	Home Alone (1990)	Smokey and the Bandit (1977)	The Ten Commandments (1956)
Back to the Future (1985)	House of Wax (1953)	Snow White and the Seven Dwarfs (1937)	The Towering Inferno (1974)
Bambi (1942)	Inception (2010)	Spectre (2015)	Thor: The Dark World (2013)
Batman (1989)	Independence Day (1996)	Spider-Man (2002)	Thunderball (1965)
Ben-Hur (1959)	Indiana Jones and the Kingdom of the Crystal Skull (2008)	Star Wars: Episode I - The Phantom Menace (1999)	Titanic (1997)
Beverly Hills Cop (1984)	Indiana Jones and the Temple of Doom (1984)	Star Wars: Episode IV - A New Hope (1977)	Tootsie (1982)
Big Hero 6 (2014)	Inside Out (2015)	Star Wars: The Force Awakens (2015)	Toy Story 3 (2010)
Blazing Saddles (1974)	Interstellar (2014)	Superman (1978)	Transformers (2007)
Butch Cassidy and the Sundance Kid (1969)	Iron Man 3 (2013)	Swiss Family Robinson (1960)	Twister (1996)
Captain America: The Winter Soldier (2014)	It's a Mad, Mad, Mad, Mad World (1963)	The Amazing Spider-Man (2012)	Up (2009)
Cinderella (1950)	Jaws (1975)	The Bells of St. Mary's (1945)	West Side Story (1961)
Cleopatra (1963)	Jurassic Park (1993)	The Best Years of Our Lives (1946)	X-Men: Days of Future Past (2014)

Close Encounters of the Third Kind (1977/1980)	Jurassic World (2015)	The Bridge On The River Kwai (1957)	
Dawn of the Planet of the Apes (2014)	Kung Fu Panda (2008)	The Chronicles of Narnia: The Lion, the Witch and the Wardrobe (2005)	
Deadpool (2016)	Lady and the Tramp (1955)	The Da Vinci Code (2006)	
Despicable Me 2 (2013)	Lawrence of Arabia (1962)	The Dark Knight (2008)	
Doctor Zhivago (1965)	Love Story (1970)	The Dark Knight Rises (2012)	
Duel in the Sun (1946)	M*A*S*H (1970)	The Exorcist (1973)	
E. T. The Extra-Terrestrial (1982)	Madagascar 3: Europe's Most Wanted (2012)	The Godfather (1972)	
Fantasia (1940)	Maleficent (2014)	The Graduate (1967)	
Fast & Furious 6 (2013)	Man of Steel (2013)	The Greatest Show on Earth (1952)	
Fast Five (2011)	Marvel's The Avengers (2012)	The Hobbit: An Unexpected Journey (2012)	
Finding Nemo (2003)	Mary Poppins (1964)	The Hunger Games (2012)	
Forrest Gump (1994)	Men in Black (1997)	The Incredibles (2004)	
Frozen (2013)	Minions (2015)	The Jungle Book (1967)	
Furious 7 (2015)	Mrs. Doubtfire (1993)	The Lion King (1994)	
Ghost (1990)	My Fair Lady (1964)	The Lord of the Rings: The Fellowship of the Ring (2001)	
Ghostbusters (1984)	National Lampoon's Animal House (1978)	The Martian (2015)	

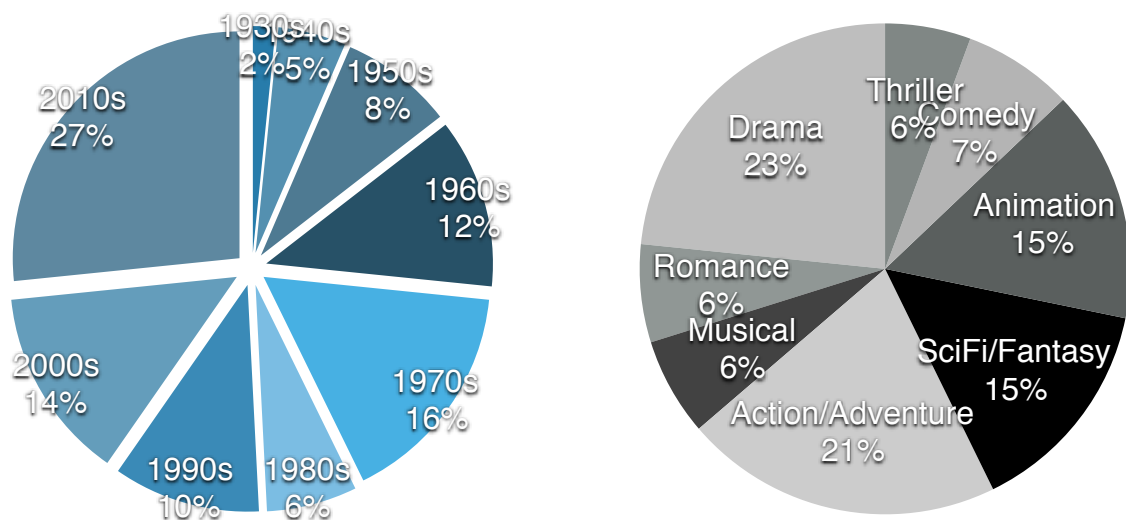
Source: assembled by this work's author

The list shown above was built based on two other lists: Top 100 Worldwide Grosses for a Movie¹⁴ and Top 100 USA Domestic Grosses for a Movie, Adjusted for

¹⁴ <http://www.boxofficemojo.com/alltime/world/>

Inflation¹⁵ according to Box Office Mojo¹⁶. From those two lists, 124 titles were picked semi-arbitrarily, aiming for a very diverse set of movies. The resulting list presents various kinds of movies, ranging from musicals to action films, covering animations and science fiction. The list also brings titles with release dates spanning from 1937 to 2016. The list was assembled in such a way in order to be appealing to various demographics with varied preferences and movie watching behaviours, from the most casual movie watchers to children to aficionados. In the graphs below, respectively, are described the percentage of movies divided per decade of release and the percentage of movies divided by their main genre:

Figure 6.1: Movies selected per release decade and movies selected per genre



Source: this work's author

The survey previously described was then posted online on the IMDb forums and Reddit, besides being shared through social media. There were 148 responses to the survey. Titles were said to be liked 4036 times, disliked 3570 times and they were said to be neither liked nor disliked or not watched 10894 times. In average, respondents watched 52 out of the 124 titles.

¹⁵ <http://www.boxofficemojo.com/alltime/adjusted.htm>

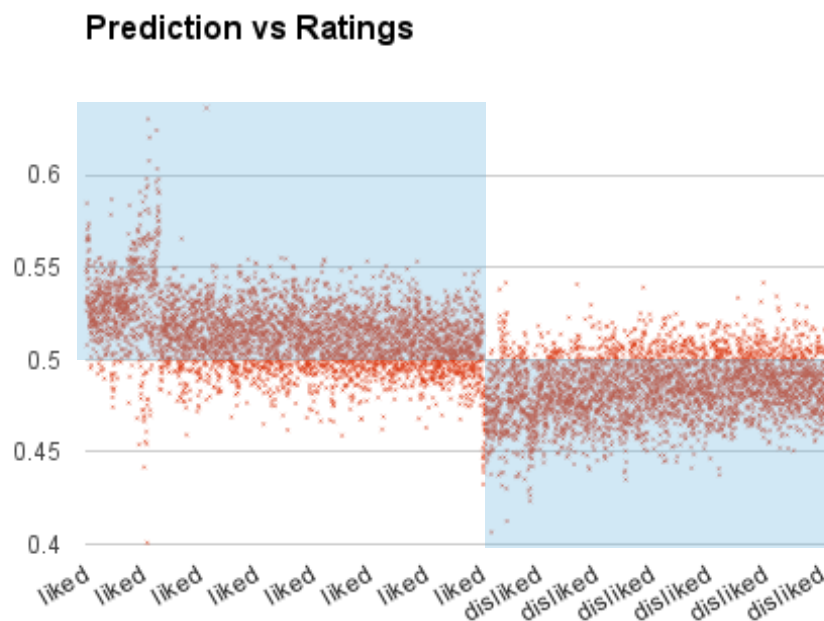
¹⁶ <http://www.boxofficemojo.com/>

6.2. Results

Leave-one-out cross validation (GEISSER, 1993) was applied in order to predict ratings. That allowed us to predict 7606 ratings with a training database of 7605 ratings at a time. A user is assumed to like an item when the probability of that happening is over 50% and to dislike the item when the probability is lower than 50%. In the unlikely event of the prediction engine predicting that a given user has exactly 50% of change of liking a given user, we treat that as inconclusive. Luckily, this did not happen in the experiment.

The x axis in the graph below represents the actual rating given by the users, while the y axis represents the predicted probability of a user liking a given film:

Figure 6.2: Predictions vs Ratings - distribution



Source: this work's author

Each one of the red dots represent a prediction by the system for a pair of movie and user versus the actual user rating for that movie. The shaded areas of the graph show accurate predictions, while the dots outside those areas are wrong ones. In fact, approximately 80.64% of the predictions were good, while approximately

19.36% of the predictions were bad. The success ratio (i.e., precision) was 100% for predictions that yielded over 62% or under 38% of likelihood that the user would like that movie. For predictions ranging from 45% to 55%, the success ratio was approximately 80.10%.

One other example of academic work that uses binary ratings as an input and tries to predict the user rating as a like/dislike in the context of movies is “Recommendation as Classification: Using Social and Content-Based Information in Recommendation” (BASU; HIRSH; COHEN, 1998). In their 1998 paper, the authors evaluate that *Recommender*, a user-based collaborative filtering recommendation system (HILL et al., 1995), has a success ratio of 78% percent, while their recommendation system, achieves a success ratio of 77% with a user-based collaborative filtering prediction engine. They also indicate that their success ratio drops to 73% when they replace collaborative filtering for content-based filtering, but is boosted to 83% combining collaborative and content-based filtering. Let there be known that study serves as a simple reference, but since the dataset used is different it is not possible to draw a strong correlation between those results and this work's. In that case, the authors also applied a function to translate user ratings from a 1 to 10 scale to binary.

In order to fully grasp the effectiveness of the rating scale applied and the accuracy of the prediction engine and how useful the predictions are to users, a more detailed evaluation of the system would have been required, but that was not possible for this version of the experiment. It is worth to mention that in a complete recommendation system there are several factors for success that may or may not depend on the prediction accuracy. Factors such as the novelty of recommendations (KAPOOR et al., 2015), the prioritisation over determined aspects of items other than the pure likelihood of a user enjoying it, trust (FORSATI et al., 2015) notions, and even risk factors (ZHAO; ZHANG; WANG, 2015), among others, are important features within a recommendation system.

7. CONCLUSION

This work introduced a prediction engine for movie recommendation systems. The engine uses a simpler rating system and pure collaborative filtering in order to try and tackle some of the issues found in prominent platforms for movie recommendation. A version of the like/dislike rating scale was used to assess the enjoyment users had with movies and item-based collaborative filtering was used to predict how much users would enjoy movies they have not yet watched.

The main objective of this work was to try and tackle two problems that exist in famous recommendation systems: the possible misunderstanding of the rating by its users and the kind of false positives caused by content-based filtering. To address the first problem, the usual rating scale was replaced with a version of the like/dislike scale, but it was not possible to conduct an experiment proving that users were more likely to understand this type of scale on an interface. In order to address the false positives caused by content-based filtering, the use of item-based collaborative filtering was proposed.

A prototype was implemented and used to extract predictions using ratings that were collected via an open survey destined to movie watchers. It was verified via leave-one-out cross-validation that the prediction engine had a precision of over 80% in saying whether the user had a bigger chance of liking or disliking a movie. The data used to train the algorithm comprised the binary ratings given by users who had taken part in responding to the survey.

It is hard to compare the results of this work with other works in the field or state of the art tools due to the choice of using a different dataset from the rest of the works. This means that no strong correlations between the results could be drawn, and the comparisons made are to be interpreted as a basic reference rather than measurement of whether this system is more or less efficient than others. On that note, it would be interesting to also refer to the accuracy for prediction systems such as Foursquare's, which employ binary rating rather than casting from another scale to binary, but unfortunately such data could not be found. Having said that, an eventual implementation of this recommendation system inside a movie recommendation platform may achieve over 80% of precision, based in ratings that are potentially meaningful, and direct. While in the case of other platforms that use a more complex

scale for their rating systems, some evidence show that the ratings they base their predictions on can be inaccurate depending on how the rating scale is presented, which may taint their predictions.

It is safe to say that there is room for improvement in the model for predicting user enjoyment with movies presented by this work. When contemplating the hypothesis of using the hereby described prediction engine to power a potential recommendation service, it's worth having in mind that the pure item-based collaborative filtering technique does not work too well when there is little or no data in the dataset, which would cause the prediction engine to suffer heavily from the cold-start problem (SU; KHOSHGOFTAAR, 2009). In addition to that, research in general shows that recommendation systems with hybrid filtering perform better than even the ones with the best collaborative filtering techniques alone (SU; KHOSHGOFTAAR, 2009).

As a future work, it would be interesting to use the prediction engine described here to power a complete movie recommendation platform in a way so that users could have the full experience of rating movies and receiving recommendations. Measurements of user trust and ease of use of the platform would be interesting metrics to compare the effectivity of the platform in terms of user experience with other systems available on the web. It would also be valuable to use the dataset collected here to compare the proposed prediction engine with engines powered by other types of information filtering, such as hybrid prediction engines, or use open datasets to train the hereby proposed prediction engine so that it could be compared against other works in the field.

REFERENCES

AAMIR, M.; BHUSRY, M. Recommendation system: State of the Art Approach. **International Journal of Computer Applications**, v. 120, n. 12, p. 25–32, 2015.

ADOMAVICIUS, G.; TUZHILIN, A. **Toward the next generation of Recommender systems: A survey of the state-of-the-art and possible extensions**. IEEE Transactions on Knowledge and Data Engineering, 25 abril 2005

GOLDBERG, D. et al. **Using collaborative filtering to weave an information tapestry**. Communications of the ACM, v. 35, n. 12, p. 61–70, 12 jan. 1992.

BELL, R. M.; KOREN, Y.; VOLINSKY, C. **The BellKor solution to the Netflix prize neighborhood-based model (k-nN)**. [s.l.: s.n.]. Disponível em: <http://www.netflixprize.com/assets/ProgressPrize2007_KorBell.pdf>. Acesso em: 21 maio. 2016.

SALAKHUTDINOV, R.; MNIH, A.; HINTON, G. **Restricted Boltzmann machines for collaborative filtering**. Proceedings of the 24th international conference on Machine learning - ICML '07, 2007.

AMATRIAIN, X.; BASILICO, J. **Netflix Recommendations: Beyond the 5 stars (Part 1)**. Netflix tech Blog. Disponível em: <<http://techblog.netflix.com/2012/06/netflix-recommendations-beyond-5-stars.html>>. Acesso em: 21 maio. 2016.

AMATRIAIN, X.; BASILICO, J. **Netflix Recommendations: Beyond the 5 stars (Part 2)**. Netflix tech Blog. Disponível em: <<http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html>>. Acesso em: 21 maio. 2016.

MovieLens. Disponível em: <<https://movielens.org>>. Acesso em: 28 maio. 2016.

GUPTA, P. et al. **WTF: the who to follow service at Twitter**. WWW '13 Proceedings of the 22nd international conference on World Wide Web, p. 505–514, 13 maio 2013.

HILL, W. et al. **Recommending and evaluating choices in a virtual community of use**. CHI '95 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, p. 194–201, 5 jan. 1995.

ZHAO, X.; ZHANG, W.; WANG, J. **Risk-hedged venture capital investment recommendation**. Proceedings of the 9th ACM Conference on Recommender Systems, p. 75–82, 16 set. 2015.

KOREN, Y. **The BellKor solution to the Netflix grand prize**. [s.l: s.n.]. Disponível em: <http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf>. Acesso em: 21 maio. 2016.

FORSATI, R. et al. **PushTrust: An Efficient Recommendation Algorithm by Leveraging Trust and Distrust Relations**. RecSys '15 Proceedings of the 9th ACM Conference on Recommender System, p. 51–58, 16 set. 2015.

MELVILLE, P.; SINDHWANI, V. **Recommender systems**. Encyclopedia of Machine Learning, 22 abril 2010.

LIKERT, R. **A Technique for the Measurement of Attitudes**. Archives of Psychology 140: 1–55, 1932.

BASU, C.; HIRSH, H.; COHEN, W. **Recommendation as classification: Using social and content-based information in recommendation**. Association for the Advancement of Artificial Intelligence '98 Proceedings. 1998.

SU, X.; KHOSHGOFTAAR, T. M. **A survey of collaborative filtering techniques**. Advances in Artificial Intelligence, v. 2009, 27 out. 2009.

TAKÁCS, G. et al. **Scalable collaborative filtering approaches for large Recommender systems**. Journal of Machine Learning Research, v. 10, p. 623–656, 2009.

HANANI, U. et al. **Information filtering: Overview of issues, research and systems**. User Modeling and User-Adapted Interaction, v. 11, p. 203-259, 2001.

PAZZANI, M. J. **A framework for collaborative, content-based and demographic filtering**. Artificial Intelligence Review, v. 13, p. 393–408, 1999.

GEISSER, S. **Predictive inference**. Monographs on Statistics and Applied Probability. Chapman & Hall, 1993.

DELGADO, J.; ISHII, N. **Memory-Based Weighted-Majority Prediction for Recommender systems**. Proceedings ACM SIGIR 1999 Workshop Recommender Systems: Algorithms and Evaluations, 1999.

SARWAR, B. et al. **Item-based collaborative filtering recommendation algorithms**. WWW '01 Proceedings of the 10th international conference on World Wide Web, p. 285–295, 4 jan. 2001.

DESHPANDE, M.; KARYPIS, G. **Item-based top-n recommendation algorithms**. ACM Transactions on Information Systems (TOIS), v. 22, n. 1, p. 143–177, 1 jan. 2004.

ALI, K. M.; PAZZANI, M. J. **Error reduction through learning multiple descriptions**. Machine Learning, v. 24, n. 3, p. 173–202, 1996.

How does Netflix tag their movies? Disponível em: <<https://www.quora.com/How-does-Netflix-tag-their-movies>>. Acesso em: 28 maio. 2016.

Personalized Recommendations Frequently Asked Questions. Disponível em: <http://www.imdb.com/help/show_leaf?personalrecommendations>, Acesso em: 28 maio. 2016.

Neo4j: The world's leading graph database. Disponível em: <<http://neo4j.com>>. Acesso em: 4 jun. 2016.

KAPOOR, K. et al. **“I like to explore sometimes”**: Adapting to Dynamic User Novelty Preferences. RecSys '15 Proceedings of the 9th ACM Conference on Recommender Systems, p. 19–26, 16 set. 2015.

MCALONE, N. **Netflix wants to ditch its 5-star rating system**. Disponível em: <<http://www.businessinsider.com/netflix-wants-to-ditch-5-star-ratings-2016-1>>. Acesso em: 29 maio. 2016.