POLITECNICO DI MILANO

School of Industrial and Information Engineering
Master of Science in Computer Science and Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria

Item Rating or Feature Rating?
Which is the Key for Building a User Profile?

Supervisor: Prof. Paolo Cremonesi

Co-Supervisor: Dr. Yashar Deldjoo

Master Thesis by:

Luca Luciano Costanzo, 862725

Academic Year 2018 - 2019

To my family, my friends
and my special person,
who have always supported me...

When something is important
enough, you do it
even if the odds are not
in your favour

— Elon Musk

Ringraziamenti

Dopo un percorso accademico lungo e tortuoso, dopo aver incontrato decine di persone nuove, dopo aver studiato centinaia di formule e risolto migliaia di equazioni, sono felice e orgoglioso di presentare questo elaborato di Tesi di Laurea Magistrale in Computer Science and Engineering.

Desidero ringraziare le persone che in questo percorso, con il loro aiuto, le loro parole o soltanto con la loro presenza, mi hanno permesso di andare avanti e di raggiungere questo arduo, ma bellissimo traguardo:

LA MIA FAMIGLIA, che con pazienza ed entusiasmo mi ha, in primis, permesso di frequentare un istituto prestigioso quale il Politecnico di Milano, e in seguito, sostenuto, pungolato e consolato durante i miei studi;

AI MIEI AMICI, M., F., A., J. protagonisti di serate divertenti, presenti da tempo immemore, sia durante le fatiche e lo sconforto, sia nei momenti di gioia e soddisfazione.

ALLA MIA PERSONA SPECIALE, Teresa la mia Stella Polare, sempre dietro alle mie spalle, pronta a darmi il suggerimento perfetto, l'occhiataccia giusta, o a farmi sorridere con una battuta. Grazie di essere e di essere stata parte di tutto questo.

PROF. PAOLO CREMONESI, il mio relatore per i suoi preziosi consigli, per aver ideato e promosso questo progetto di ricerca.

DOTT. YASHAR DELDJOO, il mio correlatore per avermi fornito tutti gli strumenti di cui avevo bisogno per intraprendere la strada giusta e portare a compimento la mia tesi; per avermi seguito con attenzione e dedizione, anche a distanza; per avermi dedicato tempo e conoscenze, volte a migliorare ed impreziosire questo mio progetto. Grazie.

DOTT. MAURIZIO FERRARI DA CREMA per essere stato l'intermediario tra me e il Professore, riproponendo i dubbi e i problemi sorti.

A TUTTI GLI ALTRI, ai miei colleghi e a tutte le persone che sono state parte di questa fase della mia vita sia provenienti dall'ambito accademico che dal mio nuovo mondo di lavoro.

Vi ringrazio, sinceramente, dal profondo del cuore.

Milano, 2019 L. L. C.

Abstract

In order to improve the accuracy of recommendations, many recommender systems (RSs) nowadays use side information beyond the user rating matrix, such as item content. These systems build user profiles as estimates of users' interest on content (e.g., movie genre, director or cast) and then evaluate the performance of the recommender system as a whole e.g., by their ability to recommend relevant and novel items to the target user. The user profile modelling stage, which is a key stage in hybrid content-driven RSs, is barely properly evaluated. One of the main reason for this is the lack of publicly available datasets that contain user preferences on content *features* of items.

With respect to the state of the art, our work advances user profile modelling for RS in several dimensions: (i) first, we investigate differences between *explicit* user preferences and *implicit* user profiles built via a suite of user profiling models consisting of 4 models, while previous work by Nasery et al. [34] only applied a basic method (comparing the number of times each feature is explicitly rated and the number of times it appears in the content of all rated movies); (ii) second, we evaluate the quality of user profiling models by integrating them into a recommender system, similar to the previous work of Nasery et al. [33]; (iii) lastly, we publicly release the collected dataset which includes ratings on movie content features.

Our results show a maximum average pairwise cosine similarity of 58.07% between the explicit feature preferences and the implicit user profiles modelled by the best investigated profiling method and considering movies' genres only. For actors and directors, this maximum similarity is only 9.13% and 17.24%, respectively. This low similarity between explicit and implicit preference models encourages a more in-depth study to investigate and improve this important user profile modelling step, which will eventually translate into better recommendations.

To further investigate this study in a real recommendation setting, we tested the integration and evaluation of user profiles by using 3 recommendation models, namely: (i) item-based collaborative filtering, considering only movie ratings, (ii) explicit user-user hybrid RS, embedding explicit feature ratings, and (iii) implicit user-user hybrid RS, embedding implicit user profiles built with the best-investigated method. The two hybrid RS integrating feature ratings present higher accuracy scores than the CF model, which consider only items ratings. Furthermore, the explicit hybrid RS performed better than the implicit one, hence more efforts should be done to improve user profiling methods in order to reach the final RS accuracy provided by the integration of true feature ratings.

Keywords: Recommender System, User Profiling, Features, Dataset, PoliMi, Master Thesis

Sommario

Al fine di aumentare il livello di accuratezza delle raccomandazioni, molti sistemi di raccomandazioni (RSs) oggigiorno sfruttano alcune informazioni secondarie oltre alla matrice di valutazioni degli utenti, come ad esempio il contenuto degli oggetti. Questi sistemi modellano i profili utente come stima degli interessi degli utenti riguardo il contenuto degli oggetti (ad esempio, i generi cinematografici, i registi o il cast) e successivamente valutano le prestazioni del sistema di raccomandazione nel suo insieme, ad esempio dall'abilità di suggerire oggetti rilevanti e nuovi per l'utente desiderato. La fase di modellazione del profilo utente, che rappresenta un elemento chiave nei RSs ibridi basati sui contenuti, viene valutata raramente. Uno dei motivi principali è la mancanza di un set di dati disponibili al pubblico che contengano le preferenze degli utenti sulle features (ovvero attributi/caratteristiche) degli oggetti.

Rispetto allo stato dell'arte, il nostro lavoro prosegue la modellazione del profilo utente per i RS in diversi modi: (i) prima, indaghiamo le differenze tra le preferenze *esplicite* degli utenti e i loro profili *impliciti* creati tramite un insieme di modelli di profilazione utente, composta da 4 metodi, mentre nel precedente lavoro di Nasery et al. [34] è stato applicato solo un metodo di base (confrontando il numero di volte in cui ciascuna feature viene votata esplicitamente e il numero di volte in cui appare nel contenuto di tutti i film votati); (ii) in secondo luogo, valutiamo la qualità dei modelli di profilazione degli utenti integrandoli in un sistema di raccomandazione, similmente al precedente lavoro di Nasery et al. [33]; (iii) infine, distribuiamo pubblicamente il set di dati raccolto che include le valutazioni sulle feature dei contenuti dei film. I nostri risultati mostrano una media di cosine similarity a coppie del 58,07% tra le preferenze esplicite riguardo le feature e i profili utente impliciti modellati dal miglior metodo di profilazione analizzato, considerando solo i generi come feature dei film. Considerando attori e registi, questa similarità massima è solo del 9,13% e del 17,24%, rispettivamente. Questa bassa somiglianza tra i modelli espliciti e quelli impliciti incoraggia uno studio più approfondito per indagare e migliorare questa importante fase di modellazione del profilo utente, che eventualmente si tradurrà in raccomandazioni migliori.

Per approfondire ulteriormente questo studio in una situazione reale di raccomandazioni, abbiamo collaudato l'integrazione e la valutazione dei profili utente tramite 3 modelli di racommandazione, i seguenti: (i) item-based collaborative filtering, che considera solo le valutazioni dei film, (ii) user-user hybrid RS esplicito, che incorpora le valutazioni riguardo le feature, e (iii) user-user hybrid RS implicito, che incorpora i profili utente impliciti modellati tramite il miglior metodo preso in esame. I due RS ibridi, che integrano le valutazioni delle feature, presentano punteggi di accuratezza più elevati rispetto al modello di CF, il quale considera solamente le valutazioni dei film. Inoltre, il RS ibrido esplicito ha performato meglio di quello implicito, dunque maggior impegno dovrebbe essere applicato per migliorare i metodi di profilazione utente al fine di raggiungere l'accuratezza finale fornita dal RS che integra le valutazioni reali delle feature.

Parole chiave: Recommender System, Profilazione utente, Feature, Dataset, PoliMi, Tesi

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Across the web, users are able to provide feedbacks with the simplicity of a mouse click, for instance a five-star rating assigned to a product on Amazon, which will represent a user's taste regarding that item, [1, 39]. The increasing size of this information and the interest on how to use them led to the birth of recommender systems (RSs). Recommender systems leverage users' interactions on items to understand their tastes and to recommend new relevant items that they would probably like [1, 39]. RSs play a fundamental role in research and particularly in business corporates interested into many decision making processes, in order to obtain several advantages [1, 39]. For instance, Amazon uses RS to recommend new products to users, while Netflix uses them to propose new movies or series to watch, improving customers' experience and retention.

The main paradigms of recommendation models include: (i) content-based filtering (CBF), which analyzes content features of items favoured by the target user in order to build a user profile, representing her taste and preferences on the content features [1, 36], (ii) collaborative filtering (CF), which uses the correlations between preferences of target users and other like-minded users [1, 36, 43] and, (iii) hybrid systems, which combine CBF and CF techniques to improve the effectiveness of both [1].

Over the last years, the performance of collaborative filtering (CF) recommendation models have reached a remarkable level of maturity. These models are now widely adopted in real-world recommendation engines because of their state-of-the-art recommendation accuracy. To further improve the quality of recommendation, new trend of recommendation models consider using various additional information sources (aka side information) beyond the user rating matrix [47]. A prominent example—and the one we focus on—is item content.

In the movie domain, for instance, a variety of content features have been

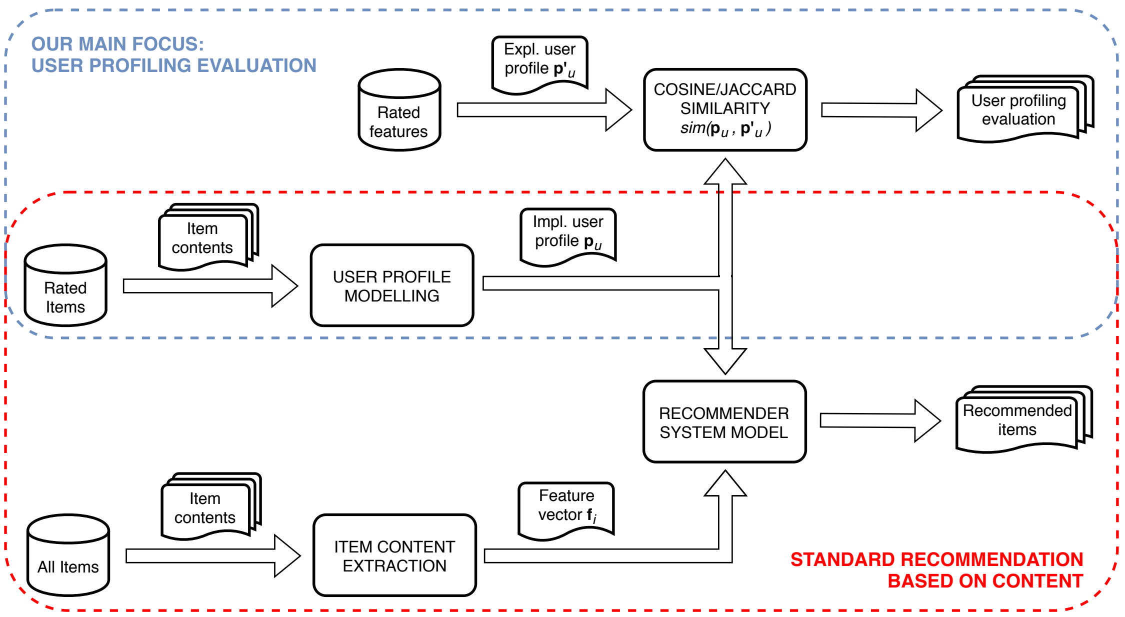

OUR MAIN FOCUS:
USER PROFILING EVALUATION
Rated features
Expl. user profile $\mathbf{p}'_u$
COSINE/JACCARD SIMILARITY $sim(\mathbf{p}_u, \mathbf{p}'_u)$
User profiling evaluation
Rated Items
Item contents
USER PROFILE MODELLING
Impl. user profile $\mathbf{p}_u$
RECOMMENDER SYSTEM MODEL
Recommended items
All Items
Item contents
ITEM CONTENT EXTRACTION
Feature vector $\mathbf{f}_i$
STANDARD RECOMMENDATION
BASED ON CONTENT

Figure 1.1: Main steps involved in a recommendation system leveraging content information, highlighting our contributions.

considered, such as metadata or features extracted directly from the core audio-visual signals. Metadata-based movie recommender systems typically use genre [16, 17, 23, 48] or user-generated tags [30, 53, 60] over which user profiles are built, assuming that these aspects represent the semantic content of movies [12]. In contrast, audio-visual signals represent the low-level content (e.g., color, lighting, spoken dialogues, music, etc.) [11, 14–17]. Some approaches try to infer semantic concepts from low-level representations, e.g., via word2vec embeddings [4], deep neural networks [52, 57], fuzzy logic [50], or genetic algorithms [32]. For these reasons, it is evident that item content plays a key role in building hybrid or content-based filtering (CBF) models and, furthermore, it is important to correctly distinguish and weight the item features by their estimated relevance for a target user, to better model his or her tastes.

A shortcoming of typical RS evaluation is that the user profiling stage, which is a key part of the RS, is barely evaluated. Usually, only the performance of the entire RS, which is composed of several components, is assessed and how effectively the user profiling step functions remains an open question. We hypothesize that correct modelling and evaluation of the user profiling stage can have a substantial impact on the rightfulness of the user profile and improvement of the final recommendation quality. In Figure 1.1 we illustrate a simplified diagram of user profile modeling in a CBF recommendation model (the lower part of the figure) and the contribution of our work (the upper part of the figure). In the upper part of the figure we depict how we evaluate the effectiveness of user profiling methods, by computing the similarity $sim(\mathbf{p}_u, \mathbf{p}'_u)$ between the modeled user profile $\mathbf{p}_u$ and the explicit user profile $\mathbf{p'}_u$, built from rated features.

The goal of this work is therefore to investigate the difference between explicit user ratings on individual movie content features (e.g., genre, actors, or directors) and implicit models inferred via state-of-the-art user modelling techniques from explicit ratings of the whole movies. To this end, (i) first, we evaluate several user profiling methods by comparing the modeled *implicit* preferences about movies' features against the true *explicit* ones, (ii) second, we integrate and evaluate the best user profiling method tested in a recommender system and (iii) lastly, we publicly release the dataset containing movies and features ratings which we collected and used for our analysis.

With respect to previous research, to the best of our knowledge, the only work that evaluates implicit user profiles against true ratings on content features is [34]. Nasery et al. compare actually rated features with the ones implicitly derived from rated movies, but no concrete user profiling methods are investigated. Instead, the number of times each feature is explicitly rated and the number of times it appears in the content of all rated movies is counted, and these counts are compared. The authors create a dataset of movies' feature ratings (genres, actors/cast, and directors), dubbed PoliMovie,[1] through a survey web application they built. Their approach, using limited survey questions and a fixed reduced dataset of top popular movies and features, extracted from IMDb,[2] tends to push users to limited and convergent preferences. In contrast, we systematically investigate 4 methods to model implicit user profiles and we compare them with explicit user profiles obtained by feature ratings.

To further investigate user profiling methods in a concrete application we evaluate them also via Recommender System, comparing several models to embed item ratings and feature ratings. Also Nasery et al. [33] incorporate known feature preferences in a recommender system using an adapted matrix factorization model.

Another contribution of the work at hand is the creation of a dataset[3] that includes ratings on movie content features. Other datasets commonly used in movie recommender systems research, but which do not contain such feature ratings, include MovieLens 20M (ML-20M) [21], IMDB Movies Dataset [27], The Movies Dataset [5], MMTF-14K and MVCD-7K[4] [10, 13] and the Netflix Prize dataset [35].

The structure of this thesis is organized as follows: (i) In chapter 2 we present

[1]PoliMovie: `http://bit.ly/polimovie`
[2]Internet Movie Database (IMDB): `www.imdb.com`
[3]The collected dataset is available at `http://bit.ly/2XzIYyL`
[4]MMTF-14K and MVCD-7K provide metadata plus audio and visual descriptors for thousands of movie trailers and movie clips.

the foundations in recommender systems and the state of the art in user profile modelling, in particular by presenting 4 methods proposed in literature, plus a basic TF-IDF; (ii) in chapter 3 we show the system used to collect the dataset of binary ratings on movies and their attributes; (iii) in chapter 4 we present the methodology used for the evaluation of user profiling stage, including the adaptation of user profile methods described in chapter 2, in order to apply them to our binary dataset; (iiii) in chapter 5 we present the experimental results of the analysis of user profile modelling methods proposed in the literature, compared with the set of true feature ratings included in our collected dataset.

Chapter 2

Foundations and State of the art

The goal of this chapter is to provide a tutorial on the fundamental concepts in recommender systems (RS) and provide an overview of the start-of-the-art research in user profile modeling. Toward this goal, this chapter is divided into two main sections: (i) foundation in recommendation system (section 2.1) and (ii) state of the art in user profile modeling (section 2.4).

2.1 Foundations in recommender systems

Recommender systems (RSs) are a sub-family of *information retrieval* (IR) techniques, they both aim to give user access to information or products [39]. RSs can be applied to various decision-making processes, such as what items to buy, what music to listen to [39], or, like in our considered case, what movies to watch. "Item" is the general term used to identify the element recommended to users [39].

Nowadays recommender systems are used in many application domains and the interest on them is growing because they have a strong impact on customer satisfaction and on profit for corporations. Our case of interest is the domain of movie recommendation systems (MRSs), hence the "item" is the movie to be recommended to a user. Famous services in which MRSs are used, are for instance Netflix[1]®, TMDb[2]® and MovieLens[3]®.

2.1.1 Basic Concepts

The use of recommender systems and their algorithms is related to different types of information. This information can be categorized in the following dimensions [31]:

[1]www.netflix.com
[2]The Movie Database (TMDb): www.themoviedb.org
[3]MovieLens: movielens.org

Table 2.1: List of abbreviations used throughout the thesis.

Abbreviation	**Term**
RS	Recommender system
IR	Information retrieval
MRS	Movie recommender system
URM	User rating matrix
ICM	Item content matrix
UCM	User content matrix
CS	Cold start
WS	Warm start
CBF	Content-based filtering
CF	Collaborative filtering
VSM	Vector space model
TF-IDF	Term Frequency-Inverse Document Frequency

- *User content*: information related to the user and given in two ways, (i) explicit : information given directly by the user and related to a specific item, for instance, the rating given to a movie by the user, (ii) implicit: information inferred from user behaviour, for instance, a movie details page viewed by the user.

- *Item content*: information strictly related to the item itself and not to the user, i.e. the characteristics of the item, called also attributes or, more commonly, *features*. For instance, the features of a movie can be its genre, year of production, country, cast and crew.

The above information are commonly used in recommender systems in several data structures, like bipartite graphs [29] or in the following matrix formats, where $\mathcal{I}$ and $\mathcal{U}$ are a set of items and users and $\mathcal{F}$ is the set of features:

- *User rating matrix (URM)*: is a $\mathcal{U} \times \mathcal{I}$ matrix used to represent explicit or implicit ratings given by users to items. The values for each cell of the matrix usually are binary (0 or 1) if we want to model only the presence or absence of interaction between user and item, or real numbers if we want to represent a rating (e.g. from 1 to 5) given by a user to an item.

- *Item content matrix (ICM)*: is a $\mathcal{I} \times \mathcal{F}$ matrix used to represent the *profiles* of the items, that are the set of *features* defining their *content*. The values of a feature can be of different types: real valued, categorical, binary and so on. This diversity in data representation can create difficulties in calculating the similarity between item profiles, so, usually, the different types of feature values are encoded in a binary profile. This means that a cell at position (i, j) of ICM will have value 1 if item identified by row index i contains the feature identified by the column index j, otherwise it will have value 0.

- *User content matrix (UCM)*: is a $\mathcal{U} \times \mathcal{F}$ matrix used to represent the profiles of the users, i.e. their *content*. In this case we call $\mathcal{F}$ the set of features attributed to users. This matrix has the same meaning of ICM matrix and is encoded in a binary profile in the same way.

2.2 Standard recommendation models

In this section we describe two main paradigms of recommendation models: (i) content-based filtering (CBF) [36] and (ii) collaborative filtering (CF) [43].

2.2.1 Content-Based Filtering

Content-based filtering (CBF) techniques analyze item descriptions to identify items that are of particular interest to the user [36]. The standard recommendation process based on content (CBF or hybrid) is structured in three main steps: (i) first, the content of each item i is extracted, to build a *feature vector* $\mathbf{f}_i$; (ii) second, the *profile of the target user* $\mathbf{p}_u$ is modeled, i.e., a structured representation of the user's preference over item content features; (iii) finally, the user profile $\mathbf{p}_u$ is matched against the feature vector $\mathbf{f}_i$ of each item to output the list of recommended items most similar to the target user's tastes.

Let us consider, for instance, two movies with some of their features, in table 2.2, and a user A. If A rated only the movie "Matrix" its profile $\mathbf{u}_A$ would be composed of all the features of that movie. Considering $\mathbf{f}_F$ the feature vector of "Fury" and $\mathbf{f}_M$ the feature vector of "Matrix", a CBF algorithm would match the attributes of $\mathbf{u}_A$ with the ones of $\mathbf{f}_F$ to estimate if A would like Fury.

In section 2.4 we will present the state of art in modelling *user profile* with some techniques to estimate the relevance of a feature for users.

2.2.2 Collaborative Filtering

Collaborative filtering (CF) is the process of filtering or evaluating items using the opinions of other people. CF takes its roots from something humans have been

Table 2.2: Example of movies with some of their features

Movies	Genres	Actors	Director
Fury	Action, Drama, War	Brad Pitt, Jon Bernthal	David Ayer
Matrix	Action, Science fiction	Keanu Reeves, Laurence Fishburne	Lana Wachowski, Lili Wachowski

Table 2.3: Example of user rating matrix

	V for Vendetta	Matrix	Fury
Alan	4	5	-
Bob	-	5	4

doing for centuries, sharing opinions with others [43]. Let us take, for instance, two users, Alan and Bob, with similar preferences. If Alan gives a positive rating to an item unknown to Bob, also the latter one will probably like that item, no matter of its features (unlike CBF), this is the assumption of CF algorithms. To be more formal, a *rating* consists of the association of two things, *user* and *item*, often by means of some value [43]. There are several ways of representing ratings in a CF algorithm, one of them is a matrix called user rating matrix (URM) (e.g. table 2.3), consisting of a table where each row represents a user, each column represents a specific movie, and the number at the intersection of a row and a column represents the user's rating value. The absence of a rating score at this intersection indicates that user has not yet rated the item [43]. Another representation of user ratings in CF could be a bipartite graph [29], for instance, the one shown in Figure 2.1

Ratings in a collaborative filtering system can take on a variety of forms [43] :

- *Scalar ratings*: can consist of either numerical ratings, such as the 1-5 stars

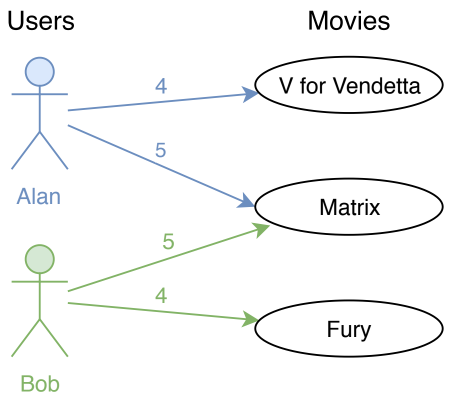


Figure 2.1: Example of user ratings represented as a bipartite graph

provided in MovieLens or ordinal ratings such as strongly agree, agree, neutral, disagree, strongly disagree.

- *Binary ratings*: model choices between agree/disagree or good/bad.
- *Unary ratings*: can indicate that a user has observed or purchased an item, or otherwise rated the item positively. The absence of a rating indicates that we have no information relating the user to the item (perhaps they purchased the item somewhere else).

Ratings may be gathered through explicit means, implicit means, or both. *Explicit ratings* require the user to actively provide his/her opinion on an item. *Implicit ratings* are those inferred from a user's actions [43]. For instance, adding a movie to favorites could be considered an explicit rating, while an implicit rating could be the visit to a movie details page.

CF algorithms can be (i) *memory-based* or (ii) *model-based* [42].

Memory-based techniques use the stored ratings directly in the prediction and they are usually easier to implement [46]. The ratings are based on user neighborhoods, that can be defined in two ways:

- *User-based*: user-based recommendation methods employ statistical techniques to find a set of users, known as *neighbors*, that have implicit or explicit ratings similar to the target user u. Once a neighborhood of users is formed, these systems use different algorithms to combine the preferences of neighbors to produce a prediction or top-N recommendation for the active user [42]. So we have to define a preference similarity $sim(u, v)$, that should summarize in a singular value how much two users u and v have similar tastes. This similarity value, then, will be used to weight the contribution of the ratings of user v to the prediction of the ratings of user u. The predicted rating $\hat{r_{u,i}}$ for user u and item i can be estimated as:

$$\hat{r_{u,i}} = \frac{\sum_{v \in U} r_{v,i} \cdot sim(u, v)}{\sum_{v \in U} |sim(u, v)|} \tag{2.1}$$

- *Item-based*: item-based methods make rating predictions for an item by the target user based on neighborhood of that item. The idea is that two items are similar if many users have rated rated them both [42]. We have to define a similarity $sim(i, j)$ between the target item i and another item j in order to compute the predicted rating $\hat{r_{u,i}}$ for user u and item i as:

$$\hat{r_{u,i}} = \frac{\sum_{j \in I} r_{uj} \cdot sim(i, j)}{\sum_{j \in I} sim(i, j)} \tag{2.2}$$

Model-based collaborative filtering algorithms instead provide item recommendation by first developing a *model* of user ratings. Algorithms in this category take a probabilistic approach and envision the collaborative filtering process as computing the expected value of a user prediction, given his/her ratings on the other items [42]. The *item-based* approach looks into the set of items the target user has rated and computes how similar they are to the target item i and then selects k most similar items $\{i_1, i_2, ..., i_k\}$. At the same time their corresponding similarities $\{s_1, s_2, ..., s_k\}$ are also computed. Once the most similar items are found, the prediction is then computed by taking a weighted average of the target user's ratings on these similar items [42].

As the Content-Based algorithm, Collaborative Filtering suffers from the *Cold-start problem* [9, 45], that can be of two types, *cold-user* or *cold-item*. The first problem occurs when a new user is inserted into the system, because there is no way to know his/her preferences a-priori [2]. A way to solve this problem is to invite new users to select some preferences in order to compute their recommendations. The second problem occurs when an item is rated by few users [59], hence its hard to build a relationship between these items and the others via ratings. A solution to this problem could be the use of *hybrid systems* that embed the information about items content to compute their similarity.

2.3 Evaluation of recommender systems

In the literature, many methods have been used to analyze the *accuracy* of a RS output, the list of recommended items. The challenge faced by researchers is to select the best metric to measure the accuracy of their RS algorithm in a specific context [22]. Recommendation accuracy metrics can be classified in: (i) *predictive accuracy* metrics (section 2.3.1), (ii) *classification accuracy* metrics (section 2.3.2), (iii) *ranking accuracy* metrics (section 2.3.3) [20, 22, 44] and (iiii) *beyond-accuracy* measures [44] (section 2.3.4).

2.3.1 Predictive accuracy metrics

Predictive accuracy metric measures how close the RS's predicted ratings are to the true users ratings [20, 22, 44].

Mean absolute error (MAE) is one of the most used predictive metrics. It measures how close the rating predictions generated by a RS are to the real user ratings by computing the average absolute deviation between these two vectors [22, 44]:

$$MAE = \frac{1}{|T|} \sum_{r_{u,i} \in T} |r_{u,i} - \hat{r_{u,i}}| \tag{2.3}$$

where $r_{u,i}$ and $\hat{r_{u,i}}$, respectively, denote the actual and the predicted ratings of item i for user u. MAE sums over the absolute prediction errors for all ratings in a test set T [44].

Root-mean-square error (RMSE) is similar to MAE and is computed as:

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{r_{u,i} \in T} (r_{u,i} - \hat{r_{u,i}})^2} \tag{2.4}$$

2.3.2 Classification accuracy metrics

Classification metrics measure how often a RS makes correct or incorrect decisions about whether an item is good [22].

Precision at K recommendations (P@K) measures the accuracy of a recommender system in predicting relevant items in the top K recommended. For each user u, $P_u@K$ is computed as:

$$P_u@K = \frac{\left|L_u \cap \hat{L_u}\right|}{\left|\hat{L_u}\right|} \tag{2.5}$$

where L_u is the set of relevant items for user u in the test set T and $\hat{L_u}$ denotes the recommended set containing the K items in T with the highest predicted ratings for the user u [44].

Recall at top K recommendations (R@K) is defined as:

$$R_u@K = \frac{\left|L_u \cap \hat{L_u}\right|}{|L_u|} \tag{2.6}$$

where L_u is the set of relevant items for user u in the test set T and $\hat{L_u}$ denotes the recommended set containing the K items in T with the highest predicted ratings for the user u [44].

2.3.3 Ranking accuracy metrics

Ranking accuracy metrics measure the ability of a RS algorithm to produce a recommended ordering of items that matches how the user would have ordered the same items. Unlike classification metrics, ranking metrics are more appropriate particularly to evaluate algorithms that will be used to present ranked recommendation lists to the user, in domains where the user's preferences in recommendations are non-binary [22].

Mean average precision at K (MAP@K) computes the average precision of the system at different sizes of recommendation lists [22, 44]. MAP@K is the arithmetic mean of the average precision at K (AP@K) over the entire set of users in the test set T. AP@K is computed as:

$$AP@K = \frac{1}{N}\sum_{i=1}^{K} P@i \cdot rel(i) \tag{2.7}$$

where $rel(i)$ is an indicator of relevance of i^{th} recommended item, it can assume value 1 if relevant or 0 otherwise [44].

2.3.4 Beyond-accuracy measures

The quality of a recommender system can also be evaluated according to some properties beyond-accuracy. Some of them are business-related because they are mainly interesting from a point of view of business and revenue [39, 44].

Coverage can be split into (i) *prediction coverage* and (ii) *catalogue coverage* [18]. The first one represents the percentage of things (items, users, ratings) that the recommender system is able to recommend. Not being able to predict a particular set of items is usually caused by an insufficient number of ratings, and is generally known as the *cold start problem* [18, 44]. *Prediction coverage* is computed as:

$$Prediction\ coverage = \frac{\left|\hat{T}\right|}{|T|} \tag{2.8}$$

where $|T|$ is the size of the test set and $\left|\hat{T}\right|$ is the number of ratings in T for which the system can predict a value. *Catalogue coverage* instead is usually measured on a set of recommendation sessions, for example by examining for a determined period of time the recommendations returned to users [18].

Diversity measures how much the recommended items are different from each other, where difference can relate to various aspects [44]. Diversity can be defined in several ways, the most common is the *list diversity*, computed as the pairwise distance between all items in the recommendation set, either averaged or summed [44]:

$$diversity(L) = \frac{\sum_{i \in L} \sum_{j \in L \setminus i} dist_{i,j}}{|L| \cdot (|L| - 1)} \tag{2.9}$$

where $dist_{i,j}$ is some distance function defined between items i and j.

Novelty determines how unknown recommended items are to a user. To measure the surprise of an item, its probability as a function of its rank for all users is determined [58]. A user's average novelty over the n items in their recommended list L can then be defined as [44]:

$$novelty = \frac{1}{|U|} \sum_{u \in U} \sum_{i \in L_u} \frac{-\log_2 pop_i}{N} \tag{2.10}$$

where pop_i is the popularity of item i measured as percentage of users who rated i, L_u is the recommendation list of the top N recommendations for user u.

Serendipity is the measure of how surprising the successful or relevant recommendations are. On a general level, serendipity of a recommendation list L_u provided to a user u can be defined as [44]:

$$serendipity(L_u) = \frac{\left| L_u^{unexp} \cap L_u^{useful} \right|}{|L_u|} \tag{2.11}$$

where L_u^{unexp} and L_u^{useful} denote subsets of L containing, respectively, recommendations unexpected to and useful for the user.

2.4 State of the art in user profile modelling

As described in section 2.2.1, CBF exploits the content of data items to predict its relevance based on the user's profile [31]. In order to build a user profile the features of his rated items are matched against the ones of all the other items. We can categorize the features as audio, visual and textual, but in most CBF systems, item descriptions are textual features extracted from Web pages, emails, news articles or products descriptions [31]. The wide area of features related to an item,

Table 2.4: Common symbols for user profile modelling methods

Symbol	Description
$\mathcal{U}$	Set of users
$\mathcal{I}$	Set of items
$\mathcal{F}$	Set of features
$h_{u,f}$	Weight of relevance of feature f implied for user u
$r_{u,i}$	Rating that user u assigned to the item i
r_{τ}	Threshold rating value

implies that even if a user rates it, he may be impressed by some of its features and not impressed by others [31, 56]. The relevance of a feature has to be derived in some way from the rating assigned by user to items. The issue is how to properly derive this *implicit* weight of feature relevance. In the literature, this problem has been addressed many times and several methods to compute feature weights in user profiles have been proposed [26, 28, 31, 38, 49, 56].

We analyze 4 state-of-the-art methods from literature to model user profiles and we refer to them according to the first author of the corresponding publication, for simplicity: (i) *Zhang method* [56] (section 2.4.1), (ii) *Li method* [28] (section 2.4.2), (iii) *Symeonidis method* [49] (section 2.4.3) and (iiii) *Kim method* [26] (section 2.4.4). In addition, we investigate a 4$^{\text{th}}$ method (section 2.4.5) which applies the TF-IDF (term frequency–inverse document frequency) term weighting idea, which is widely used in CBF and, in general, in information retrieval [31, 41, 51].

We will present the adaptation of the following methods in case of binary movie ratings, in chapter 4, then in chapter 5 we will show the results obtained by comparing the implicit user profiles produced by these methods, with the explicit user profiles collected by our web application. We will also integrate the best-investigated method in a RS to eveluation the accuracy of final recommendations.

We will use common symbols described in table 2.4.

2.4.1 Zhang Method

Table 2.5: Pros and cons of Zhang method

Pros	Cons
It can be applied both in tag-based RSs that support explicit feature ratings and in classic CBF that imply feature relevance from rated items	It assigns the same weight to all the unrated features belonging to a rated item for a user

Table 2.6: Symbols used for Zhang method

Symbol	Description
$t_{u,f}$	Rating that user u explicitly assigned to the feature f
$\{h_{u,f}(i)\}$	Bag of defined ratings $h_{u,f}(i)$ for all items $i \in \mathcal{I}$

Zhang et al. [56] build the user profile based on item ratings or explicit feature ratings.

Let $\mathcal{U}$ and $\mathcal{I}$ denote the set users and items, respectively, and $\mathcal{F}$ the set of all features of the items. Zhang method builds the profile of each user u in $\mathcal{U}$ by assigning weight $h_{u,f}$ to each feature f in $\mathcal{F}$, based on ratings explicitly given by u to the feature f or to the items containing f. Hence, depending on ratings, $h_{u,f}$ can assumes the following values [56]:

$$h_{u,f} = \begin{cases} t_{u,f}, & \text{if user rated feature } f \\ r_{u,i}, & \text{if user rated item } i \text{ but not its feature } f \\ undefined, & \text{otherwise} \end{cases} \tag{2.12}$$

where $t_{u,f}$ is the explicit rating given by user u to feature f.

For user u and feature f, $\{h_{u,f}(i)\}$ denotes the bag of defined ratings $h_{u,f}(i)$ for all items $i \in \mathcal{I}$.

2.4.2 Li Method

Table 2.7: Pros and cons of Li method

Pros	Cons
It reduces the number of items with low rating and improves the recommendation quality, by using a threshold value	The threshold rating value is unique and fixed a-priori for all the user profiles

Table 2.8: Symbols used for Li method

Symbol	Description
$N_{u,f,r_{tau}}$	Number of items containing feature f with rating greater than r_τ, by given user u
$M_{u,r_{tau}}$	Number of items with rating greater than r_τ, by given user u

Li et al. [28] ignore items with low rating and improve the recommendation quality by using a threshold value.

Let $\mathcal{U}$ and $\mathcal{I}$ denote the set of users and items, respectively, the relevance weight $h_{u,f}$ of each feature f for a user u is computed as [25, 28, 37, 38]:

$$h_{u,f} = \frac{N_{u,f,r_{tau}}}{M_{u,r_{tau}}} \tag{2.13}$$

where:

- $N_{u,f,r_{tau}}$ is the number of items containing feature f with rating greater than r_τ, by given user u:

$$N_{u,f,r_{tau}} = |\{\forall i \in \mathcal{I} : r_{u,i} > r_{tau} \text{ and } f \in i\}| \tag{2.14}$$

- $M_{u,r_{tau}}$ is the number of items with rating greater than r_τ, by given user u:

$$M_{u,r_{tau}} = |\{\forall i \in \mathcal{I} : r_{u,i} > r_{tau}\}| \tag{2.15}$$

2.4.3 Symeonidis Method

Table 2.9: Pros and cons of Symeonidis method

Pros	Cons
It weights more properly the relevance of rare features contained in less user profiles, by using a weighting factor conceptually similar to TF-IDF	A threshold rating value is unique and fixed a-priori for all the user profiles, like Li method

Table 2.10: Symbols used for Symeonidis method

Symbol	Description
$FF(u,f)$	Is the feature frequency of feature f in the profile of user u
$N_{u,f,r_{tau}}$	Number of items containing feature f with rating greater than r_τ, by given user u
$\|\mathcal{U}\|$	Total number of users
$UF(f)$	User frequency of feature f, i.e. the number of user profiles containing feature f
$IUF(f)$	Inverse user frequency

Symeonidis et al. [49] adopt an approach similar to TF-IDF to compute feature relevance weights, but define them in the vector space of user profiles. The rationale of using TF-IDF is to increase the relevance of rare features contained in less user profiles. Symeonidis et al. also use a fixed rating threshold to consider only the most relevant items.

Let $\mathcal{U}$ and $\mathcal{I}$ denote the set of users and items, respectively, the user profiles with this method are built in 3 steps:

1. The feature frequency $FF(u, f)$ is defined for feature f in the profile of user u [49]:

$$FF(u, f) = N_{u,f,r_{tau}} = |\{\forall i \in \mathcal{I} : r_{u,i} > r_{tau} \text{ and } f \in i\}| \tag{2.16}$$

   where $N_{u,f,r_{tau}}$ is the number of items containing feature f with rating greater than r_τ, by given user u.

2. Then the *Inverse user frequency* $IUF(f)$ of feature f is computed as:

$$IUF(f) = \log \frac{|\mathcal{U}|}{UF(f)} \tag{2.17}$$

   where $UF(f)$ is the *user frequency* of feature f, i.e. the number of users whose profile contains feature f at least once.

3. Finally the relevance weight $h_{u,f}$ of feature f for user u is computed as [24, 49]:

$$h_{u,f} = FF(u, f) \cdot IUF(f) \tag{2.18}$$

2.4.4 Kim Method

Kim et al. [26] for each user separate the items in two sets, the ones with high rating versus the ones with low ratings, in order to separate the relevance of the features contained in this sets of items.

The two sets of items, for each user u, are [26]:

- *positive items* $\mathcal{I}_u^{pos}$, the items whose rating is greater or equal than the average of all ratings given by u,
- *negative or irrelevant items* $\mathcal{I}_u^{neg}$, the items whose rating is less than the average of all ratings given by u

Table 2.11: Pros and cons of Kim method

Pros	Cons
It separates the items rated by a user and its belonging features into two sets depending on rating	It cannot be used in case of binary ratings because the the two sets of items would collapse into one unique set, obtaining the same results of Zhang method (except for a normalization factor)
The threshold rating, used to separate the items in this method, is not fixed like in Li and Symeonidis methods, but is computed as the average of all ratings for each user	

Table 2.12: Symbols used for Kim method

Formula	Description
$w_{u,i}(f)$	associated to each item i containing the feature f by user u
$\sqrt{\sqrt{\sum_{k \in \mathcal{I}} r_{u,k}^2}}$	L^2-*norm* of all ratings given to items by user u
$\mathcal{I}_u^{pos}$	Set of positive items rated by user u containing feature f
$\mathcal{I}_u^{pos}$	Set of negative items rated by user u containing feature f
$h_{u,f}^{pos}$	Relevance weight of feature f belonging to $\mathcal{I}_u^{pos}$
$h_{u,f}^{neg}$	Relevance weight of feature f belonging to $\mathcal{I}_u^{neg}$

Let $\mathcal{U}$ and $\mathcal{I}$ denote the set of users and items, respectively, first this methods computes the weight $w_{u,i}(f)$ associated to each item i containing the feature f [26]:

$$w_{u,i}(f) = \frac{r_{u,i}}{\sqrt{\sum_{k \in \mathcal{I}} r_{u,k}^2}} \tag{2.19}$$

where $r_{u,i}$ is the rating given by user u to item i and $\sqrt{\sum_{k \in \mathcal{I}} r_{u,k}^2}$ is the L^2-*norm* of all ratings given to items by user u.

Therefore, for each feature belonging to the set of positive and negative items, the average weight of the feature is computed, because it may appear in multiple items, both positive and negative [26]:

$$h_{u,f}^{pos} = \frac{1}{|\mathcal{I}_u^{pos}(f)|} \cdot \sum_{i \in \mathcal{I}_u^{pos}(f)} w_{u,i} \tag{2.20}$$

$$h_{u,f}^{neg} = \frac{1}{|\mathcal{I}_u^{neg}(f)|} \cdot \sum_{i \in \mathcal{I}_u^{neg}(f)} \tag{2.21}$$

where:

- $\mathcal{I}_u^{pos}(f)$ is the set of *positive items* rated by the user u containing the feature f,
- $\mathcal{I}_u^{neg}(f)$ is the set of *negative items* rated by the user u containing the feature f.

Finally the relevance weight $h_{u,f}$ of feature f for user u is computed as [7, 19, 26]:

$$h_{u,f} = \begin{cases} \frac{h_{u,f}^{pos} + h_{u,f}^{neg}}{2}, & \text{if } f \in \mathcal{I}_u^{pos}(f) \text{ and } f \in \mathcal{I}_u^{neg}(f) \\ h_{u,f}^{pos}, & \text{if } f \in \mathcal{I}_u^{pos}(f) \text{ and } f \notin \mathcal{I}_u^{neg}(f) \\ h_{u,f}^{neg}, & \text{if } f \notin \mathcal{I}_u^{pos}(f) \text{ and } f \in \mathcal{I}_u^{neg}(f) \end{cases} \tag{2.22}$$

2.4.5 TF-IDF Method

Most CBF systems use *Vector Space Model (VSM)* with basic *TF-IDF* weighting [31]. In VSM the item is considered as a text *document* represented by a vector in n-dimensional space, where each dimension corresponds to a term from the overall vocabulary of a given document collection, or to a feature of the item [31]. Let $\mathcal{I}$ and $\mathcal{F}$ the set of items and all features, respectively, each item i_j is represented in a n-dimensional space, so $i_j = \{w_{1j}, w2j..., w_{nj}\}$, where w_{kj} is the weight of feature f_k in item i_j.

With TF-IDF weighting, features that occur frequently in one document (TF = term-frequency), but rarely in the rest of items (IDF = inverse-document-frequency), are more likely to be relevant to the content of the item [31]. The TF-IDF value for each feature in each item is computed as:

$$TF\text{-}IDF(f_k, i_j) = TF(f_k, i_j) \cdot IDF(f_k) = TF(f_k, i_j) \cdot \log \frac{|\mathcal{I}|}{n_k} \tag{2.23}$$

where n_k denotes the number of items in $\mathcal{I}$ in which the feature f_k occurs at least once.

2.5 Similarity Indices

In order to compare two profiles, composed by different units, various statistical techniques are used.

In general all the statistical functions that compute the similarity between two statistical units are called similarity indices. Some of the most popular and used similarity indices are *Cosine similarity* (2.5.1) and *Jaccard similarity* (2.5.2).

2.5.1 Cosine Similarity

Cosine similarity is a similarity measure that applies to the vector space approach. It computes the normalized proximity between two vectors by using the cosine of the angle they form (eq. 2.24). This measure has been applied to several text classification applications [3, 8, 54]:

$$sim(A, B) = cos(\vartheta) = \frac{A \cdot B}{\|A\| \, \|B\|} = \frac{\sum_{k=1}^{n} A_k B_k}{\sqrt{\sum_{k=1}^{n} (A_k)^2} * \sqrt{\sum_{k=1}^{n} (B_k)^2}} \tag{2.24}$$

The cosine domain is defined between -1 and 1, but as a similarity measure is particularly used in positive space, so the completely opposite non-correlation between two vector is indicated with value “0” and completely similarity between two vector is indicated with value “1”.

2.5.2 Jaccard Similarity

Jaccard similarity, also known as Jaccard Index, measures the similarity between finite sets or between binary attributes of two vectors, by taking the intersection of both and dividing it by their union [55]. In other words, in case of binary attributes, given two sets, A and B, with n binary attributes, the Jaccard similarity measures the overlap of attributes of A with attributes of B.

In terms of the above definitions this gives (eq. 2.25):

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{M_{11}}{M_{01} + M_{10} + M_{11}} \tag{2.25}$$

where:

- M_{11} is the total number of binary values where both vectors have the value 1,
- M_{01} is the total number of binary values where first vector has value 1 and second has value 0,
- M_{10} is the total number of binary values where first vector has value 0 and second has value 1,
- M_{00} is the total number of binary values where both vectors have the value 0.

Chapter 3

Data Collection System

In this chapter, we present which data we used for the analysis of user profiling methods (described in chapter 4) and how we collected them.

First, in section 3.1, we describe the web application built for acquiring data from users and their ratings on movies and features. Then, in section 3, we present the phases of data collection and the structure of the dataset.

3.1 Movie Interests (mints)

Movie Interests (mints) is the *responsive* web application that we built for collecting the dataset of true feature ratings. It is still online, hosted on Heroku[1], at movieinterests.herokuapp.com . The high-level architecture of the system main components is illustrated in figure 3.1.

The main component of our system is the web application itself, represented at the centre of figure 3.1. We developed *mints* by using `Node.js` (Node)[2] as back-end framework and some of the best technologies nowadays for the front-end of web applications: `HTML5`, `JavaScript` (JS) and `Bootstrap`[3]. *mints* is designed as a *responsive* web application that can be easily navigated by any device (desktop, mobile, tablet) with immediate and simple user experience. The key idea of our system is to let users be free to search and select any movie or feature they like, without annoying them and pushing them to selections similar to each other user with survey-like questions. On *mints* the features are unrelated to movies, in the sense that users can search and add features alone as favorites in dedicated pages, even without searching them inside movies' contents.

[1]Heroku: `www.heroku.com`
[2]Node.js: nodejs.org
[3]Bootstrap: getbootstrap.com

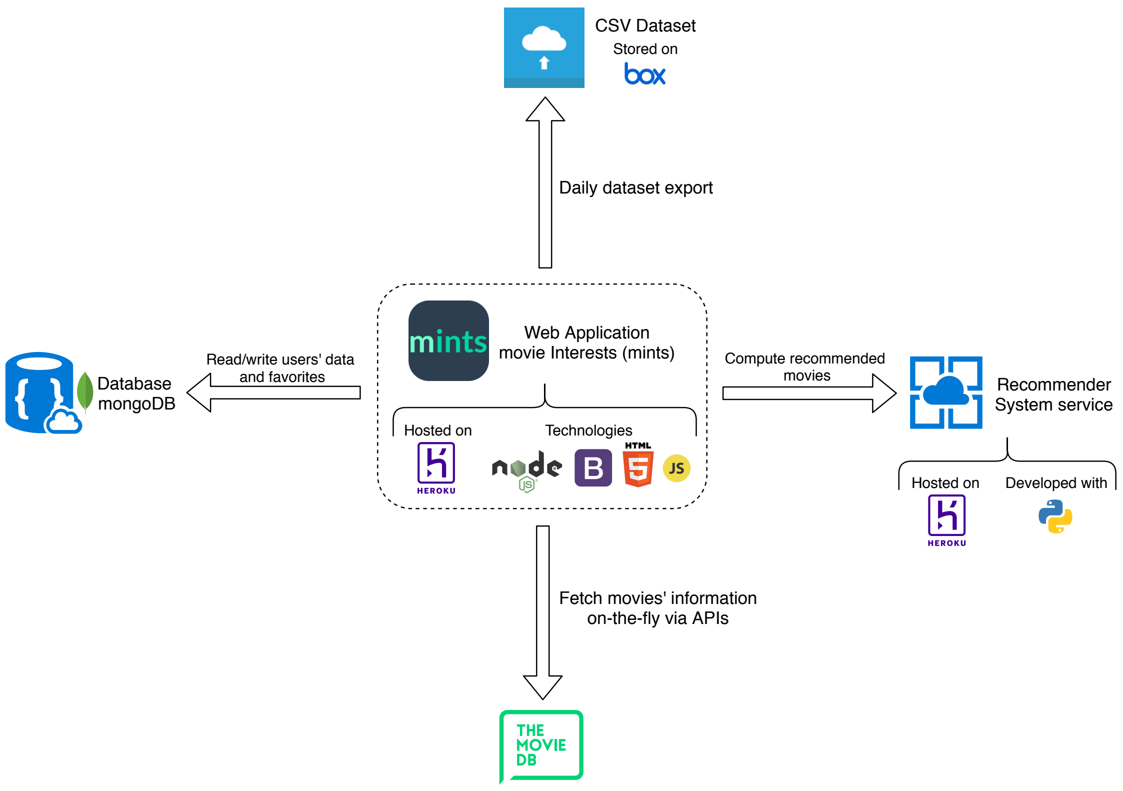


Figure 3.1: High-level architecture of our web application *mints*

Across the entire web application we use asynchronous JS requests for fetching or updating information, in order to make the actions and loadings faster. For instance, to read/write data related to users or favorites, asynchronous JS requests start from client's browser, directed to internal APIs in the back-end service of server, then the application communicates with `mongoDB`[4] database (left component of figure 3.1). Data on *mints* database are stored as json documents, for instance, a favorite actor is represented as:

Listing 3.1: Example of a json document representing a favorite actor

```
{
  "user"      : "00000111122222",
  "tmdb_id"   : "3223",
  "type"      : "actor",
  "name"      : "Robert Downey Jr."
  "createdAt" : "2019-04-06T18:22:56Z 20190406T182256Z",
  "updatedAt" : "2019-04-06T18:22:56Z 20190406T182256Z"
}
```

The information about movies and their features (plot, vote, genres, cast,

[4]mongoDB: `www.mongodb.com`

crew,...) are retrieved on-the-fly from TMDb via javascript requests to its APIs, starting directly from the client's browser. Then, these information are combined with the ones stored on internal database, in order to allow users to view and modify their favoured movies and attributes. An example of the sequence for the selection of a favorite actor, inside the web page of a specific movie, is illustrated with a sequence diagram in figure 3.2.

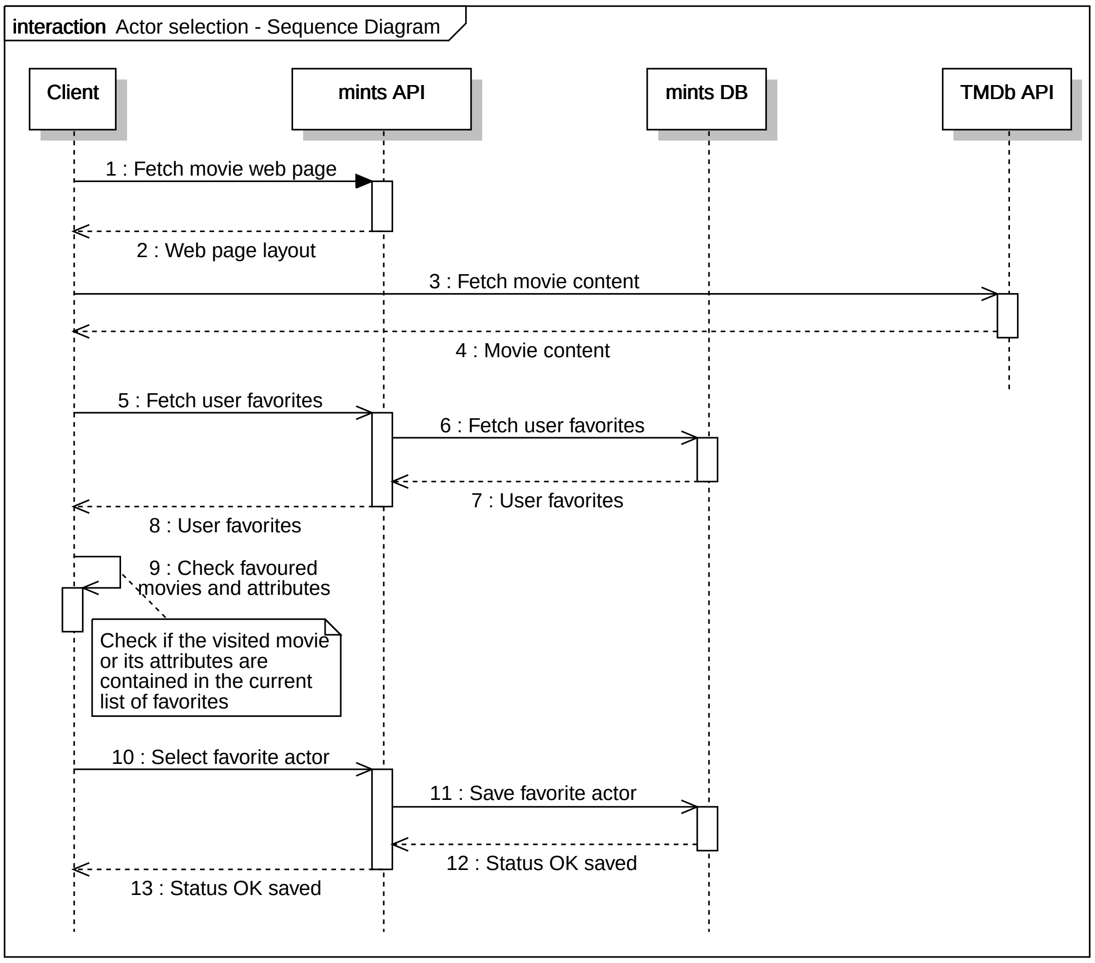


Figure 3.2: Sequence diagram for the selection of a favorite actor inside a movie details page

3.2 Data collection

As the system does not know about the preference of the target user, the goal is to invite him to choose a set of preferences which include movies and attributes, by repeatedly displaying the number of missing preferences to reach the minimum required. These required favorites include the movies, which can be found by direct search (through a search bar on top) or via navigation between popular movies (or

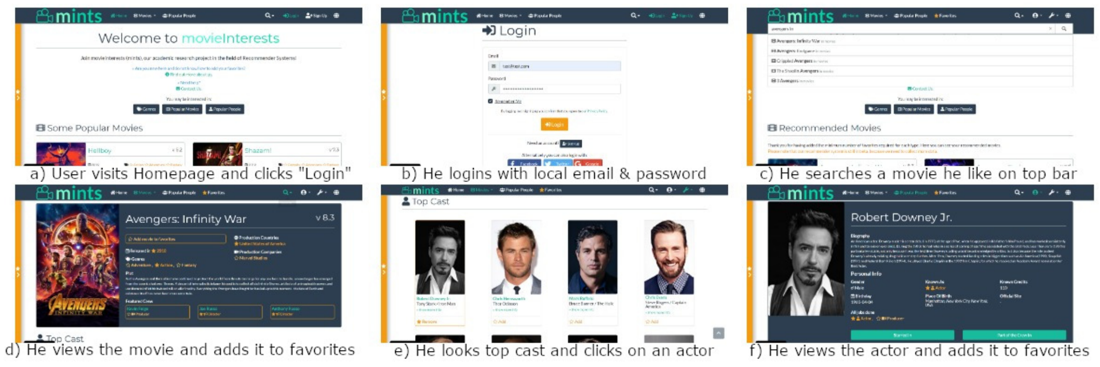

a) User visits Homepage and clicks "Login"
b) He logins with local email & password
c) He searches a movie he like on top bar
d) He views the movie and adds it to favorites
e) He looks top cast and clicks on an actor
f) He views the actor and adds it to favorites

Figure 3.3: Typical user scenario on *mints*

search with filters), and attributes, which can be found (and selected) independently from movies (e.g., by direct search an actor) or inside movies content (e.g., inside the page of movie details).

The collection of the data is divided into two phases: the first involved volunteers, the second, instead, involved users coming from the crowdsourcing platform Amazon Mechanical Turk (MTurk[5]). First, we started the data collection of volunteers, and then, after addressing the feedbacks received, we started the crowdsourcing phase. The user experience was different for the two groups, as we will show in the next sections.

3.2.1 Volunteer users

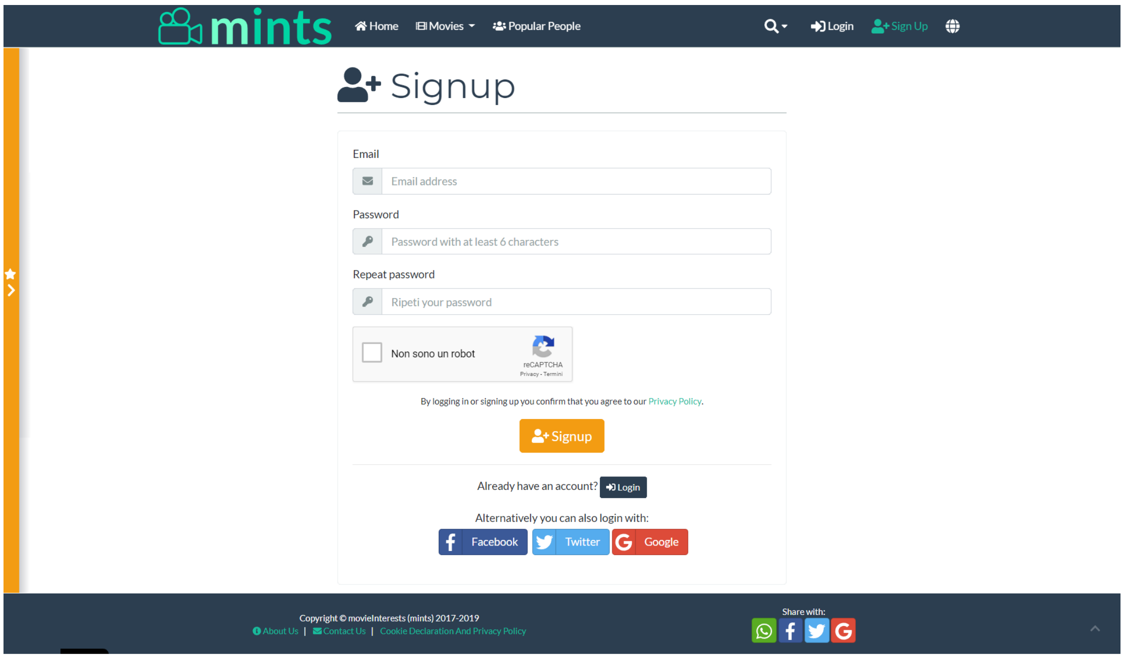
Collection

The group of volunteers is composed of family, friends and colleagues that wanted to freely contribute to the dataset. A typical user scenario on *mints* is shown in figure 3.3.

Volunteers have to register by creating a local account with email and password or by signing in with social accounts (Google +, Facebook or Twitter) (figure 3.4); then, in case of a local account (i.e., email and password), they had to confirm the activation with a link sent via mail. After that users could add all the favorites required during the navigation through the web appplication.

Initially, the preferences required were: 4 movies, 2 actors, 2 genres, 1 director, 1 production company, 2 release years. After collecting the first feedbacks from some registered users, which selected all the favoured elements required, we figured out that requiring some specific information as release years was too much difficult. Hence, we changed the required favorites to:

[5]Amazon Mechanical Turk (MTurk): www.mturk.com



By logging in or signing up you confirm that you agree to our Privacy Policy.

Already have an account? Login

Alternatively you can also login with:

Copyright © movieInterests (mints) 2017-2019

About Us | Contact Us | Cookie Declaration And Privacy Policy

Figure 3.4: Sign-up web page

- 5 movies,
- 3 actors,
- 1 director,
- 2 genres.

and they remained the same till the entire data collection phase.

Reward

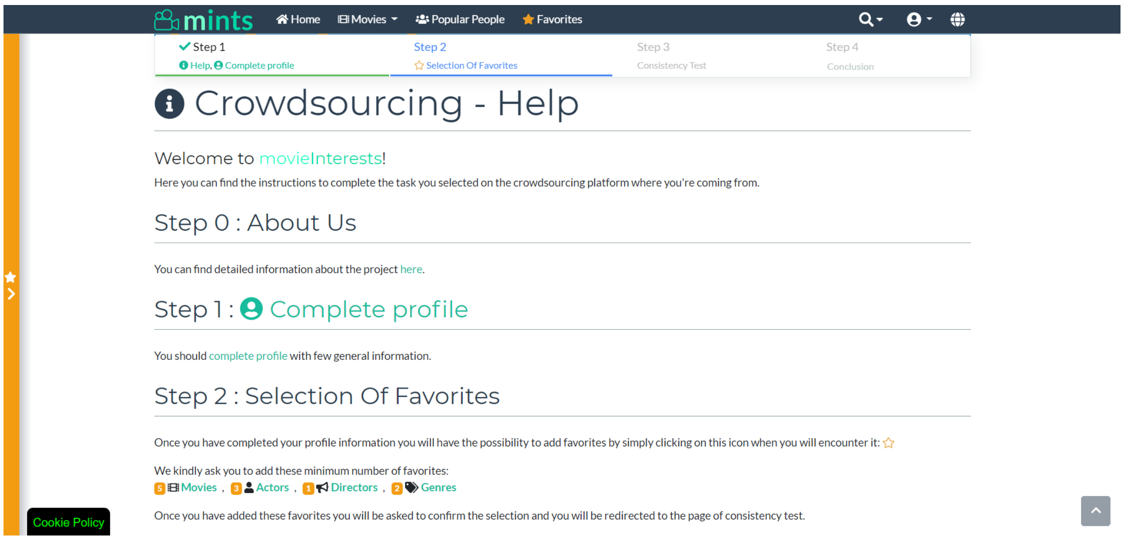
When all the preferences required were selected, users could see in homepage and in a dedicated page 20 recommended movies as a reward, computed with CBF algorithm based on favorite features selected during the data collection.

3.2.2 Crowdsourcing users

Collection

During the crowdsourcing phase, the users were found using the platform MTurk. We published on MTurk a task with brief instructions and a specific link to our web application [6]. By clicking on this link the users were redirected to a web page containing all the instructions to complete the crowdsourcing task on mints and

[6]https://movieinterests.herokuapp.com/crowdsourcing/mturk



Crowdsourcing - Help

Welcome to movieInterests!

Here you can find the instructions to complete the task you selected on the crowdsourcing platform where you're coming from.

Step 0 : About Us

You can find detailed information about the project here.

Step 1 : Complete profile

You should complete profile with few general information.

Step 2 : Selection Of Favorites

Once you have completed your profile information you will have the possibility to add favorites by simply clicking on this icon when you will encounter it: ☆

We kindly ask you to add these minimum number of favorites:
5 Movies , 3 Actors , 1 Directors , 2 Genres

Once you have added these favorites you will be asked to confirm the selection and you will be redirected to the page of consistency test.

Figure 3.5: First web page shown to crowdsourcing users with instructions

to be paid (figure 3.5). The difference between usage experience of crowdsourcing users and volunteer ones (depicted in figure 3.3) mainly concern the phases of registration, instruction and consistency test. In fact, crowdsourcing users did not have to complete the registration on mints[7], because they were already logged through an account with timed login, automatically created in background. Using timed login means that users couldn't log out or re-login at the end of the fixed time of 1 hour. This trick was used to guarantee that users completed the task only once and couldn't change their preferences a posteriori.

Following the initial instructions, the users added all the preferences and, when completed the task, they had to click on the submit button, to definitively confirm the favorites. After that, users couldn't change their preferences anymore.

Consistency test

In order to verify the reliability of the information collected, crowdsourcing users had to complete a final form called *"consistency test"*, to demonstrate that they had not randomly completed the task, but that they were conscious of their preferences. During the consistency test some of the preferences previously selected were proposed (only with name or title), mixed with other popular elements not chosen by users. They had to select only the features they remembered to have added before. Obviously, the more was the similarity between the preferences added before and the elements selected in the test, the more reliable was the user. In fact, the score was calculated as the *"precision"* of true favorites between the selected elements.

[7]movie Interests (mints): `movieinterests.herokuapp.com`

The consistency test was useful to understand which user deserved to be paid on Amazon MTurk platform and to filter collected information by reliable users, as depicted in chapter 5.

Chapter 4

Methodology

In the following chapter, we will show the methodology that we applied to study the user profiling stage. (i) First, in section 4.1, we will describe our adaptation of user profiling methods (introduced in section 2.4), to apply them in case of binary ratings (like in our dataset), giving particular attention to the assumption on threshold value (in methods 2, 3 and 4). We will explain the assumptions done for the two similarity indices, *cosine similarity* and *Jaccard similarity*, used to compute the overlap between the implicit preferences of user and their explicit selections. (ii) Finally, in section 4.2, we will present our proposed methodology to integrate feature preferences, explicit or implicitly modeled, into a recommender system.

4.1 User profiling with binary ratings

In chapter 2 we introduced 5 methods to build implicit user profiles. These methods are commonly related to datasets with scalar ratings belonging to a fixed range (e.g. from 1 to 5) or they are adjusted for specific datasets.

In our case, the dataset is composed by *binary ratings* with values 1 if an item (movie) or a feature is liked, 0 if the rating is missing, i.e. if a user didn't add an element to his favorites. We made some assumptions to create "implicit" user profiles on features from items.

4.1.1 Zhang Method

Zhang method presented in section 2.4.1 can be easily applied in our case of binary ratings. The relevance weight $h_{u,f}$ of each feature f for a user u is computed as:

$$h_{u,f} = \begin{cases} 1, & \text{if } r_{u,i} = 1 \text{ and } f \in i \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

where $r_{u,i}$ is the rating given by user u to item i.

4.1.2 Li Method

Li method presented in section 2.4.2 can be applied in case of binary ratings by setting the rating threshold r_{tau} to 0. Hence the relevance weight $h_{u,f}$ of each feature f for a user u is computed as:

$$h_{u,f} = \frac{N_{u,f}}{M_u} \tag{4.2}$$

where:

- $N_{u,f}$ is the number of items rated by user u containing feature f:

$$N_{u,f} = |\{\forall i \in \mathcal{I} : r_{u,i} > 0 \text{ and } f \in i\}| \tag{4.3}$$

- M_u is the number of items rated by user u:

$$M_u = |\{\forall i \in \mathcal{I} : r_{u,i} > 0\}| \tag{4.4}$$

4.1.3 Symeonidis Method

Symeonidis method presented in section 2.4.3 can be applied to binary ratings by setting the threshold rating r_τ to 0, like done for Li method. Hence the formula of feature frequency $FF(u,f)$ becomes the sum of items rated by user u containing feature f:

$$FF(u,f) = N_{u,f} = |\{\forall i \in \mathcal{I} : r_{u,i} > 0 \text{ and } f \in i\}| \tag{4.5}$$

The other steps are computed in the same way described in section 2.4.3:

$$IUF(f) = \log \frac{|\mathcal{U}|}{UF(f)} \tag{4.6}$$

$$h_{u,i} = FF(u,f) \cdot IUF(f) \tag{4.7}$$

4.1.4 Kim Method

Kim method presented in section 2.4.4 cannot be applied in our case of binary ratings, because if we set the rating threshold r_{tau} to 0, the two sets $\mathcal{I}_u^{pos}$ and $\mathcal{I}_u^{neg}$ would collapse into one unique set, obtaining the same results of Zhang method (except for a normalization factor). Therefore we will not apply this method to analyze the collected data.

4.1.5 TF-IDF Method

After having reviewed the state-of-art methods described above, we decided to investigate another variant of TF-IDF as a user profiling method. The Symeonidis method above is similar to TF-IDF, but it is user-centric because it considers the vector space of user profiles. Instead, our proposed TF-IDF method is item-centric as it considers the vector space of items (movies). In section 2.4.5 we described TF-IDF weighting in general, now we are going to present how we applied it as a user profiling method.

Let $\mathcal{U}$, $\mathcal{I}$ and $\mathcal{F}$ denote the set of users, items and all existing features of the items, respectively. First, we computed the IDF of each feature f as:

$$IDF(f) = \log \frac{|\mathcal{I}|}{n_f} \tag{4.8}$$

where n_f denotes the number of items in $\mathcal{I}$ in which the feature f occurs at least once. Then, for each user u we computed the relevance weight $h_{u,f}$ of a feature f as:

$$h_{u,f} = TF(f,u) \cdot IDF(f) \tag{4.9}$$

where $TF(u,f)$ is equivalent to $FF(u,f)$ of the Symeonidis method (i.e., number of times feature f occurs in items rated by user u).

4.1.6 Assumptions

Table 4.1: Common symbols for similarity indices applied to user profiles

Symbol	Description
$\mathcal{F}$	Set of all features
$\mathcal{T}_u$	Number of features explicitly selected as favorites by user u
$\mathbf{p}_u$	Implicit profile of user u
$\mathbf{p}'_u$	Explicit profile of user u

Since we asked users to add at least three types of features (genre, actor, director), we built the implicit profiles $\mathbf{p}_u$ with each method described above, for each of these three types of features. Then, in order to make a complete analysis

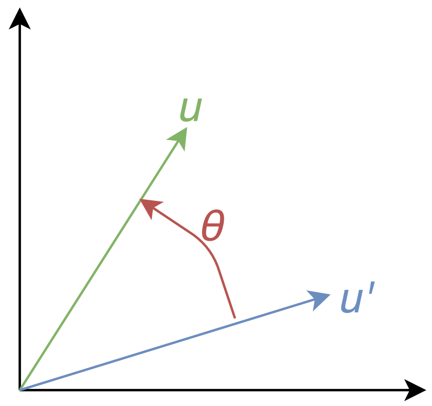


Figure 4.1: Cosine similarity of *explicit* and *implicit* user profiles

of each method in chapter 5, we compared the implicit profiles obtained, with the explicit profiles $\mathbf{p}'_u$, consisting of favorite features explicitly selected by the users on our web application. In this way we made statistical analysis in terms of pairwise similarity $sim(\mathbf{p}_u, \mathbf{p}'_u)$ between each couple of *user profiles*.

In order to build the *implicit profiles*, we needed an offline dataset of movies content so that we could extract their features. Therefore we started from a subset of $\tilde{4}5$,000 movies, belonging to "The Movies Dataset"[1] on Kaggle. Then we extended the dataset with the content of missing movies rated by users on our web application. We obtained a final dataset containing features of 45395 movies.

In order to use the *cosine similarity*, we considered user profiles as vectors. Therefore, given the *explicit profile* $\mathbf{p}'_u$ of the user u and the *implicit profile* $\mathbf{p}_u$ of the same user, built with one of the four methods, we computed the cosine similarity between these couple of profiles as:

$$sim(\mathbf{p}_u, \mathbf{p}'_u) = cos(\vartheta) = \frac{\mathbf{p}_u \cdot \mathbf{p}'_u}{\|\mathbf{p}_u\| \, \|\mathbf{p}'_u\|} \tag{4.10}$$

Regarding *Jaccard similarity*, since it can be applied only to binary attributes, but implicit profiles contain scalar weights for the relevance of features, we had to transform their resulting profiles to make them binary, through the following steps:

1. considering user u, we computed the weight $h_{u,f}$ of each feature f contained in the implicit profile u

2. given T_u, the number of features explicitly selected by user u, we assigned value 1 to the top T_u features with the highest weight, and 0 to the others.

[1]The Movies Dataset: https://www.kaggle.com/rounakbanik/the-movies-dataset/version/7

3. at the end we obtained $\mathbf{p}_u$ composed of T_u features with weight 1 and the others with 0.

Finally, the Jaccard similarity J between profiles $\mathbf{p}_u$ and $\mathbf{p}'_u$, transformed as described, was computed as:

$$J(\mathbf{p}_u, \mathbf{p}'_u) = \frac{|\mathbf{p}_u \cap \mathbf{p}'_u|}{|\mathbf{p}_u \cup \mathbf{p}'_u|} = \frac{|\mathbf{p}_u \cap \mathbf{p}'_u|}{|\mathbf{p}_u| + |\mathbf{p}'_u| - |\mathbf{p}_u \cap \mathbf{p}'_u|} = \frac{M_{11}}{M_{01} + M_{10} + M_{11}} \tag{4.11}$$

where:

- M_{11} is the total number of attributes whose value is equal to 1 both in implicit profile $\mathbf{p}_u$ and explicit profile $\mathbf{p}'_u$,
- M_{01} is the total number of attributes whose value is equal to 0 in $\mathbf{p}_u$ and 1 in $\mathbf{p}'_u$,
- M_{10} is the total number of attributes whose value is equal to 1 in $\mathbf{p}_u$ and 0 in $\mathbf{p}'_u$

For instance, if user u explicitly selected 5 favoured genres, we assigned value 1 to the 5 genres with the highest weight in the implicit profile $\mathbf{p}_u$, built with a considered method. Zhang method (section 4.1.1), instead, doesn't need any transformations, we didn't make assumptions on it because it already assign binary weights to features.

4.2 Recommender System

To further investigate user profiling methods in a concrete application, we evaluate the best performing user profiling method (among the ones described above) also via Recommender System, comparing accuracy of a RS which embeds implicitly modeled feature preferences, versus the same RS embedding true feature ratings. For the evaluation of the RSs, we divided the movie ratings randomly (with seed 2147483647) with a holdout strategy in two sets, 80% for the train set (1,389 movie ratings), 20% for the test set (347 movie ratings). We cut the output of recommendation list to 3 predicted items and we evaluated two scores for the accuracy of recommendations, AUC (Area Under the Curve) and MAP (Medium Average Precision).

4.2.1 Collaborative filtering

Before integrating feature ratings and user profiling methods, we decided to apply a typical item-based CF RS as a baseline. We considered in this case only movie ratings. The specific algorithm used is item-based CF KNN (k-Nearest Neighbors) with several values for k, where k is the number of neighbors considered.

4.2.2 Hybrid

After that, we experimented a hybrid approach with two models, one built from explicit feature ratings and the other one from implicit feature ratings modeled by the best-investigated user profiling method, i.e. Li method, as we report in section 5.3.1.

Regarding the first hybrid recommender, we extracted the favoured attributes of each user, then we built a user-user hybrid RS with item feature ratings treated like user attributes, composing in fact their explicit user profiles. The specific algorithm used is KNN with a cut-off size of predicted items equal to 3 and several values for k, where k is the number of neighbors considered. To find similar neighbors in the KNN algorithm the correlation was computed on user profiles.

The latter hybrid RS, that we call *implicit*, was built by applying the best-investigated user profiling method to extract the implicit user profiles from rated movies contained in the train set only. Then, we kept for each user and for each type of attribute (genres, actors, directors) only n implicit preferences with the highest relevance score (computed by profiling method), where n is the number of explicit preferences expressed for that attribute type. For instance, if a user explicitly selected 2 favoured genres on *mints*, then only the 2 implicit genres with the highest relevance score were considered.

Chapter 5

Experimental Results

In this chapter, we present the experimental results of the evaluation of user profiling methods, by using the dataset that we collected on *mints*, as described in section 3, and the methods discussed in section 4.1.

The structure of this chapter is structured as follows: (i) in section 5.1 we provide an overview about the characteristics of our collected dataset, including some simple statistics; (ii) in section 5.2 we analyze the global differences between the implicitly selected features (i.e. contained in their rated movies) and the real ones explicitly selected by users; (iii) in section 5.3 we study the user profiling step in-depth by investigating the 4 user profiling methods described in section 4.1, analyzing the similarity (i.e., the overlap) between the implicitly modelled user profiles and the real explicit tastes of users; (iiii) finally, in section 5.4 we show the comparison between the accuracy of some proposed recommender systems which integrate the preferences about movies and attributes, either implicit or explicit.

5.1 Data Characteristic

Users: After the data collection phase, the total number of participants registered on *mints* was 194 users. Before analyzing the 4,109 responses (preferences), we cleaned the data by taking a series of prepossessing steps:

- First, we removed the users who have not provided all the minimum number of required movies and attribute (genres, actors, directors) lists. *After this step, the remained number of users was equal to 180 (93%) users.*
- Afterwards, we removed the crowdsourcing users who scored less than 50% of precision during the consistency test (see section 3.2). *After this step, the remained number of users was equal to 155 users, we call them reliable users.*

The total number of their preferences is 3,341 (81%), including movies and attributes.

Among reliable users, 67 (43%) are *volunteers* and 88 (57%) are paid ones, referred to as *crowdsourcing* users, like shown in figure 5.1. The distribution of reliable users among volunteers and crowdsourcing ones is shown in table 5.1 and in figure 5.2. The reliable users are the ones kept after data cleaning step, while the unreliable are the ones discarded.

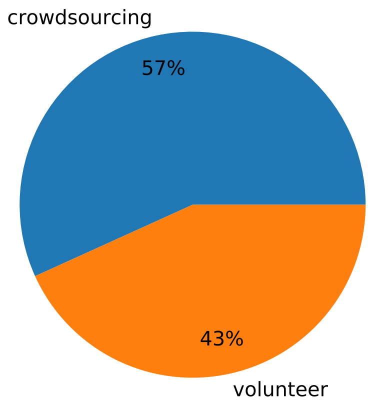


Figure 5.1: Distribution of users category

Table 5.1: Reliable and unreliable users of each category

User category	Reliable	Unreliable	Total
Crowdsourcing	88	25	113
Volunteers	67	14	81

Regarding users' gender, 92 users (59.4%) are male, 55 are female (35.5%) and 8 (5.2%) did not specify it, like shown in figure 5.3.

Most of the users (53%) are between 24 and 30 years old, as depicted in figure 5.4.

We received registrations from users coming from 10 different countries, mainly from Italy (43%), India (26%) and United States (21%), like shown in figure 5.5.

Preferred movies and attributes: We collected a total 4,109 preferred movies and attributes selected by participants, including 1,212 unique elements, i.e., elements selected by at least one user. In the following experiments, we include only the preferences of *reliable* users after the data cleaning step described above. This obtained number of preferences is 3,341 (81%), including 1,737 favoured movies, 461 genres, 698 actors, 198 directors, 74 production companies, 92 production countries, 39 producers, 17 screenwriters, 21 release years, and 4 sound crew members. The dataset is available at `http://bit.ly/2XzIYyL`.

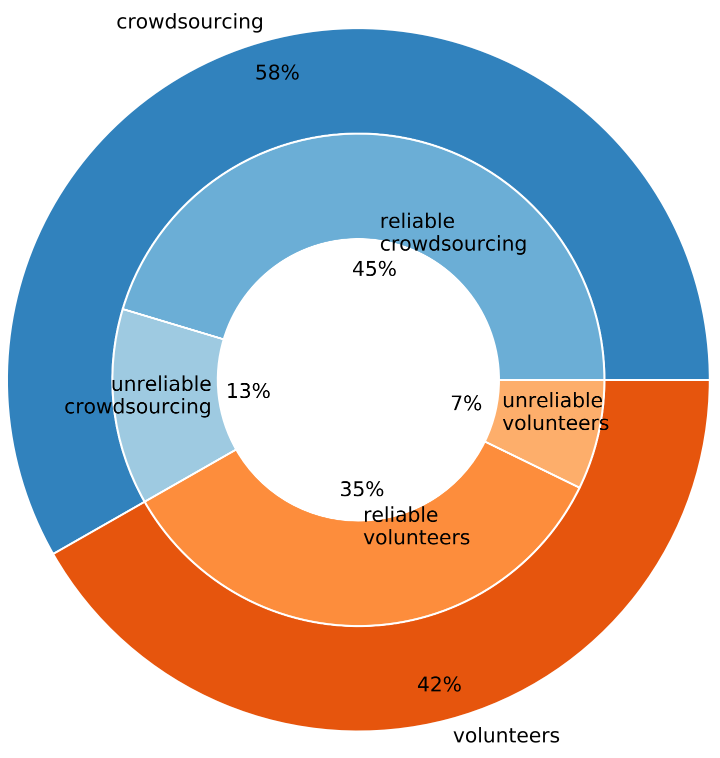


Figure 5.2: Distribution of reliable and unreliable users of each category

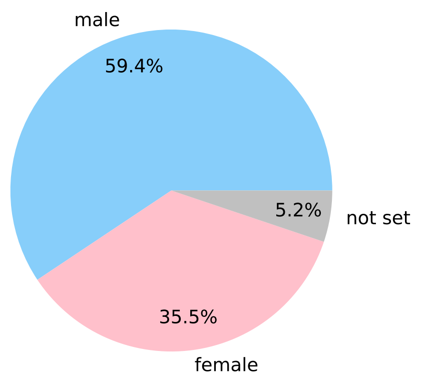


Figure 5.3: Distribution of users' gender

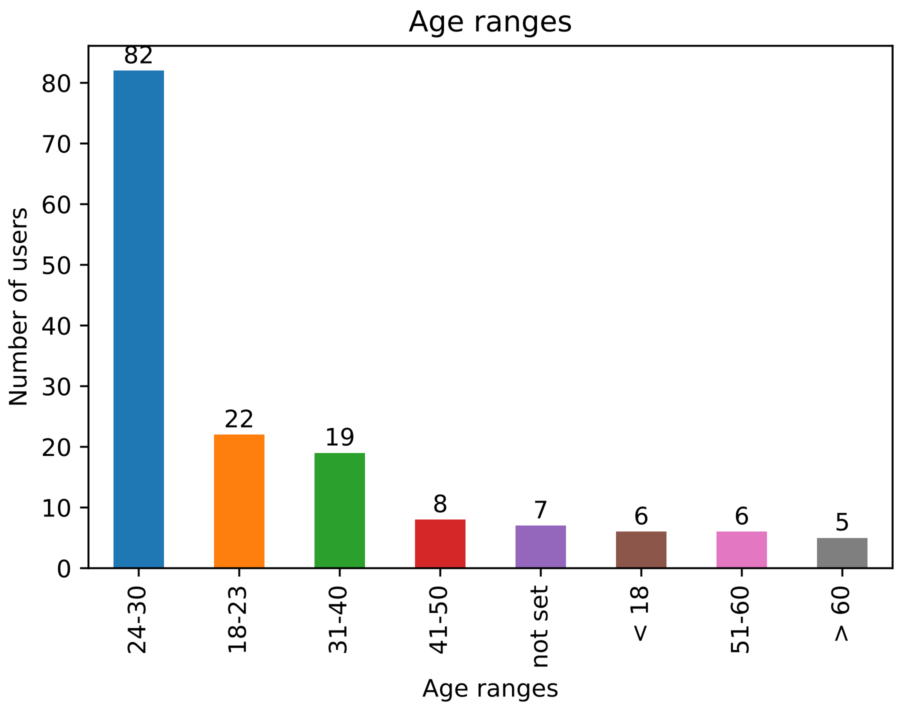


Figure 5.4: Distribution of users age ranges

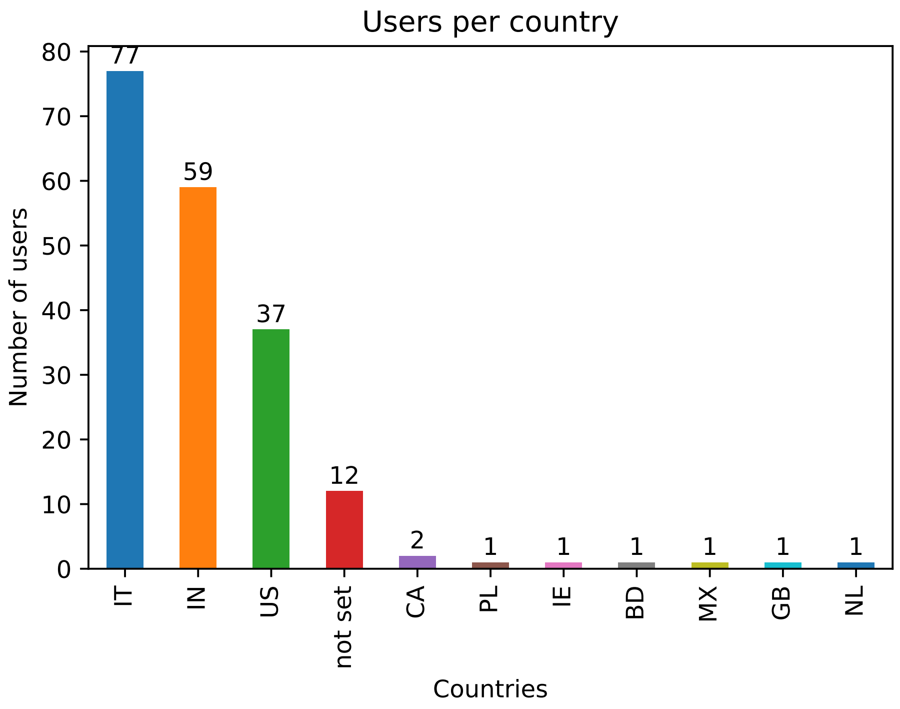


Figure 5.5: Distribution of users countries

We show the total selections and the unique selections of each favoured movie and attributes in figure 5.6.

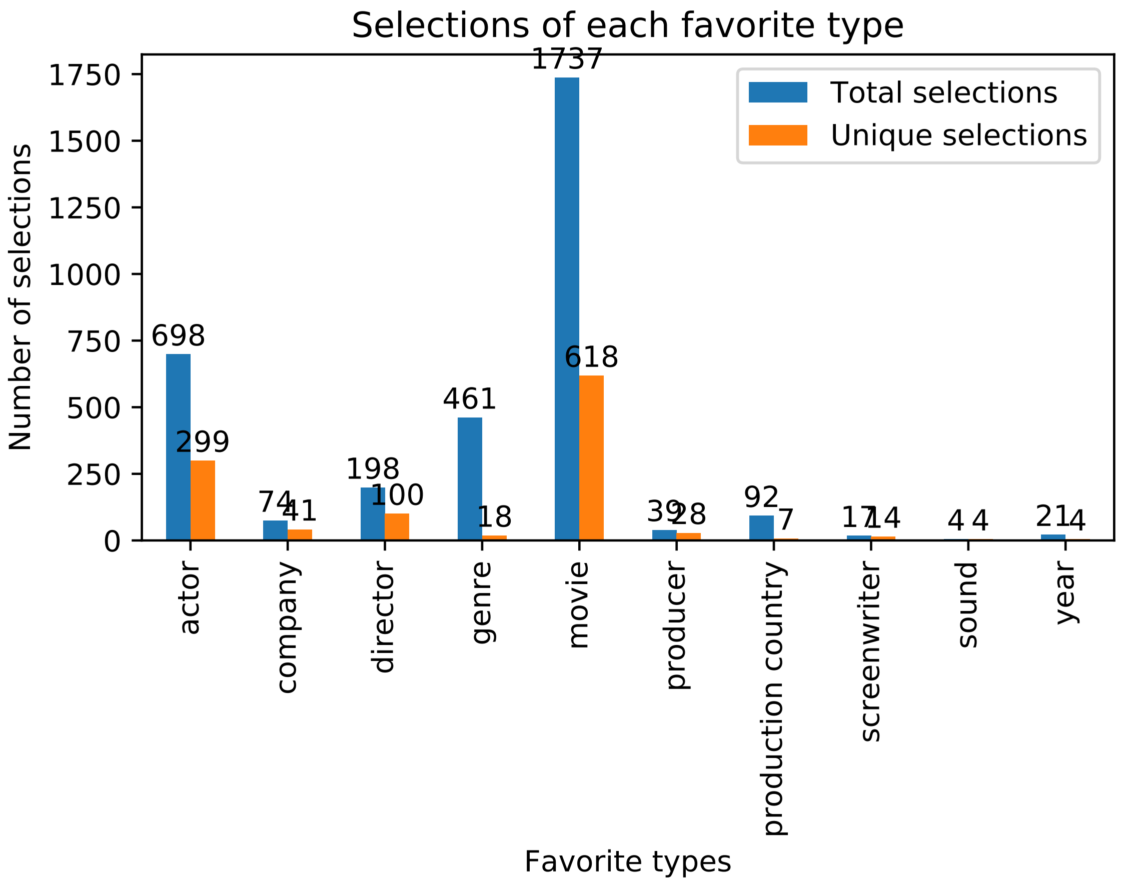


Figure 5.6: Total and unique selections of favoured elements

5.1.1 Summary statistics

Summarizing the statistics of the collected dataset, we have 155 reliable users, 45,392 movies considered as an offline dump for the analysis, 1,737 rated movies (618 unique), 1,604 rated features (515 unique), a mean number of movie ratings per user equal to 11.21 and a median equal to 5, a mean number of feature ratings per user equal to 10.35 and a median equal to 8. These statistics are represented also in table 5.2.

5.2 Global analysis of explicit/implicit preferences

In this section, we present an initial global comparison between all the features explicitly selected by users, which we call *explicit set*, and the features implicitly favored by extracting them from rated movies, which we call *implicit set*.

Table 5.2: Summary statistics of the collected dataset

Users	155
Movies	45,392
Movie ratings	1,737 (618 unq.)
Mean number of movie ratings per user	11.21
Median number of movie ratings per user	5
Feature ratings	1,604 (515 unq.)
Mean number of feature ratings per user	10.35
Median number of feature ratings per user	8

The *implicit set* is built by extracting for each user the set of unique features contained in his rated movies, that is its *implicit user profile* without any feature weighting algorithm. Then the total *implicit selections* that we show in the next subsections for each feature is the total number of its occurrences in the implicit user profiles.

We focus on the features that we consider most important and that we explicitly requested as minimum, that are *genres*, *actors* and *directors*, as explained in section 3.2. We want to show the difference between the implicit user profiles and explicit profiles. For this scope, we analyze separately for genres, actors and directors, the total number of features selected by the users and the implicit ones extracted by their favoured movies.

5.2.1 Movie

Users selected 1,737 favorite movies containing 618 unique movies. The top-20 of them are shown in table 5.3. All of these 618 unique movies are the ones which features are extracted from to build the *implicit set* and the *implicit user profiles* with the 4 methods in section 5.3.

As we can see in table 5.3, users' tastes are varied, even if the first movie is liked by a high number of users (38.7%) and the first 5-6 movies were really popular at the moment of acquiring data.

In the following sub-sections, we present separately for genres, actors and directors an initial statistical analysis, which highlights main differences between the set of all explicitly rated features and the set of all implicit features extracted from rated movies.

Table 5.3: Top-20 favoured movies

Pos.	Explicit movies	Explicit selections
1	Avengers: Infinity War	60
2	Aquaman	40
3	Glass	36
4	Bohemian Rhapsody	33
5	Venom	33
6	Mortal Engines	27
7	Creed II	27
8	How to Train Your Dragon: The Hidden World	26
9	A Star Is Born	24
10	Escape Room	24
11	Spider-Man: Into the Spider-Verse	22
12	Guardians of the Galaxy	22
13	Bumblebee	21
14	Robin Hood	21
15	Serenity	19
16	Widows	18
17	The Mule	18
18	The Avengers	18
19	Polar	18
20	T-34	17

5.2.2 Genre

In table 5.4 we report the total number of genres selected by users, hence that belong to the *explicit set*, and the total number of genres extracted from content of favorite movies, hence that belong to the *implicit set*.

Table 5.4: Genres extracted from *explicit set* and from *implicit set*

Explicit genres selected	Implicit genres extracted
461	1463

In Table 5.5 and in Figure 5.7, we present a comparison between the explicit and implicit sets of features, in percentage of common attributes (features), focusing on the k most frequently selected genres. We will repeat the same analysis for actors and directors in the next sections. These tables generally highlight a low overlap between the explicitly preferred features and the implicitly estimated ones (derived from favourite movies), in particular for actors and directors. The only exception is the genre attribute, which reveals a maximum overlap of 94.74% when considering all 19 genres.

Table 5.5: Common genres in the most selected k attributes, either explicitly or implicitly

k	No. common genres	% common genres
5	3	60.00%
10	8	80.00%
15	13	86.67%
All genres (19)	18	94.74%

We further provide a finer-grained analysis of the gap between explicit and implicit preferences of users according to their gender.

In Table 5.6, we compare the 10 most frequently selected genres, either *explicitly* or *implicitly*, by *male* and *female* users, respectively.

Regarding genre, the preferences of male and female users are similar. Furthermore, it is surprising that the genre “action” is highly ranked by female users. This could be due to the fact that the genre tastes of young women might be changing nowadays, especially because many popular action movies, like the Marvel ones, are liked by many people (especially under 30, i.e., the largest age group in our dataset), irrespective of gender.

We further provide an analysis of the convergence or divergence of users’ tastes, defined by explicitly rated genres or implicitly extracted. Let $\mathcal{F}$ denote the set of all possible genres contained in items and $\mathbf{d}_f$ the distribution of a genre f across

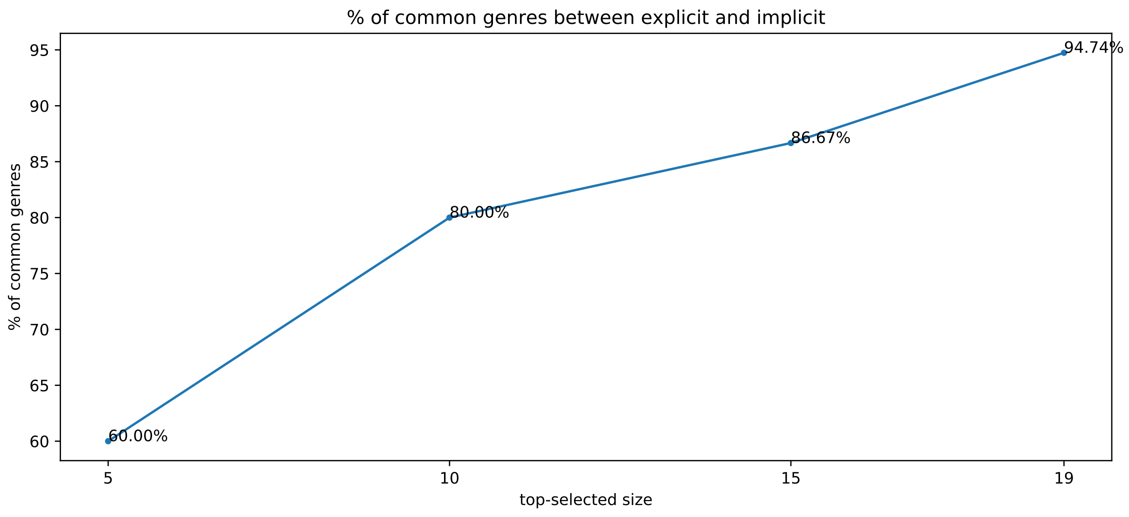


Figure 5.7: Common genres in the most selected k attributes, either explicitly or implicitly

Table 5.6: Most selected 10 genres, either explicitly or implicitly, by male and female users; $\left|R_f^{exp}\right|$ — number of explicit selections of genres, $\left|R_f^{imp}\right|$ — number of implicit selections of genres.

	Pos.	Explicitly selected	$\left\|R_f^{exp}\right\|$	Implicitly selected	$\left\|R_f^{imp}\right\|$
Male users	1	Action	51	Action	86
	2	Drama	31	Adventure	83
	3	Adventure	30	Drama	80
	4	Thriller	28	Science Fiction	76
	5	Science Fiction	28	Thriller	74
	6	Thriller	10	Thriller	25
	7	Fantasy	5	Fantasy	24
	8	Family	4	Crime	18
	9	Horror	4	Romance	17
	10	Western	3	Family	17
Female users	1	Drama	26	Drama	52
	2	Action	22	Adventure	48
	3	Adventure	14	Action	47
	4	Comedy	14	Fantasy	45
	5	Thriller	13	Science Fiction	43
	6	Thriller	10	Thriller	25
	7	Fantasy	5	Fantasy	24
	8	Family	4	Crime	18
	9	Horror	4	Romance	17
	10	Western	3	Family	17

all user profiles. $\mathbf{d}_f$ is a vector consisting of binary attributes $(f_1, f_2, ..., f_n)$ in which $f_k = 1$ if feature f is contained in the profile of user k. We computed the distribution of all possible genres across explicit and implicit user profiles and their corresponding standard deviation (std), denoted as $\sigma_{f,e}$ and $\sigma_{f,i}$, respectively.

In table 5.7 the genres with the highest std across explicit and implicit profiles are shown, i.e. the genres for which users' tastes diverge most. In table 5.8 instead we show the ones with the lowest std.

Table 5.7: Genres with the highest std across explicit and implicit user profiles

Genre with the highest std		$\sigma_{f,e}$	$\sigma_{f,i}$	$\lvert\sigma_{f,e} - \sigma_{f,i}\rvert$
Across explicit profiles	Action	0.501527	0.287573	0.213954
Across implicit profiles	Mystery	0.268122	0.501527	0.233405

Table 5.8: Genres with the lowest std across explicit and implicit user profiles

Genre with the lowest std		$\sigma_{f,e}$	$\sigma_{f,i}$	$\lvert\sigma_{f,e} - \sigma_{f,i}\rvert$
Across explicit profiles	Western	0.159071	0.329018	0.169947
Across implicit profiles	Action	0.501527	0.287573	0.213954

The most interesting result that can be noticed from tables 5.7 and 5.8 regards "Action", in fact, that is the genre for which users' tastes diverge most across explicit user profiles, but on the contrary it is the one for which they converge most across implicit profiles. The possible reason is the genre "action" is contained in many movies rated in our dataset, as already underlined in the previous analysis.

For the completeness of our analysis, in figure 5.8 we also show 4 charts of the top-5 genres with the highest std across explicit profiles and across implicit profiles (respectively, at top) and the top-5 with the lowest std (at bottom).

5.2.3 Actor

In table 5.9 we show the total number of actors selected by users, hence that belong to the *explicit set*, and the total number of actors extracted from content of favorite movies, hence that belong to the *implicit set*:

Table 5.9: Actors extracted from *explicit set* and from *implicit set*

Explicit actors selected	Implicit actors extracted
698	73558

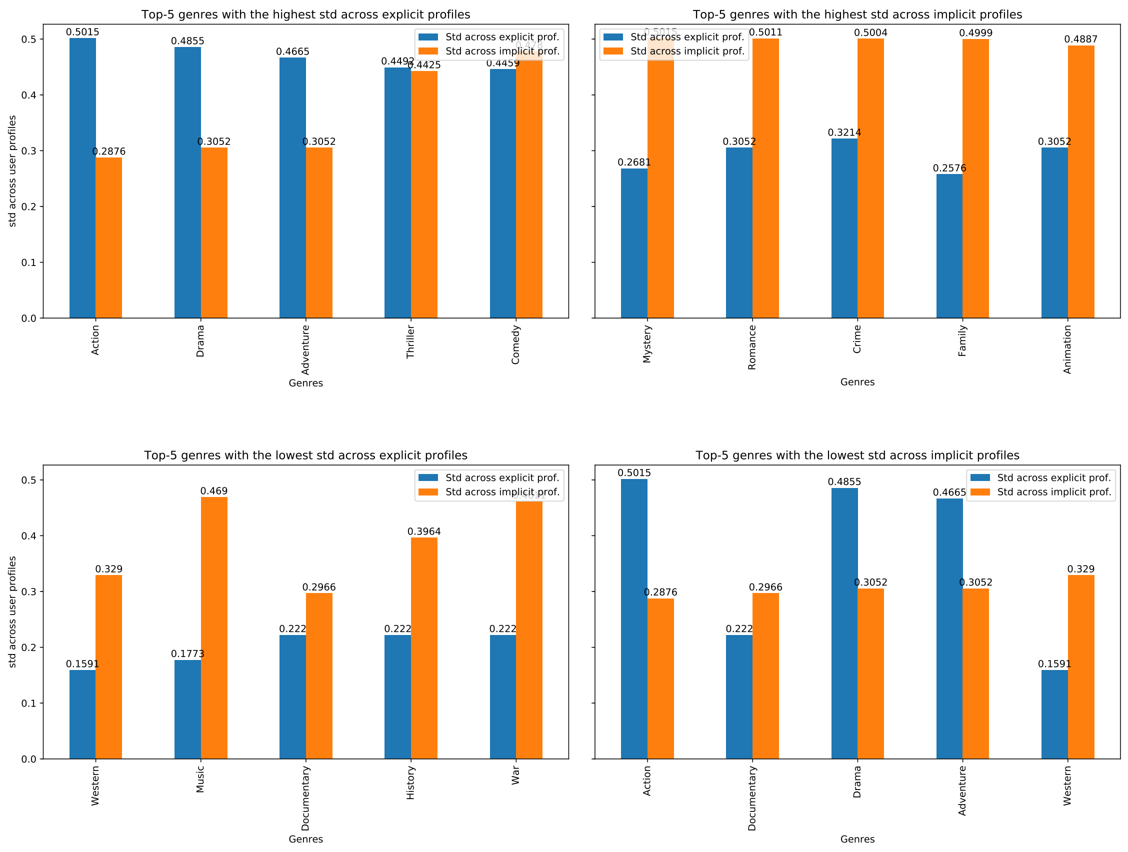


Figure 5.8: Top-5 genres with the highest std across explicit profiles and across implicit profiles (respectively, at top) and the top-5 with the lowest std (at bottom)

In Table 5.10 and in Figure 5.9, we present a comparison between the explicit and implicit sets of features, in percentage of common attributes (features), focusing on the k most frequently selected actors, like already analyzed for genres in previous section.

Table 5.10: Common quota of actors between the most selected k attributes, either explicitly or implicitly

k	No. of common actors	% of common actors
10	1	10.00%
20	4	20.00%
30	7	23.33%
40	9	22.50%
50	10	20.00%
60	10	16.67%
70	13	18.57%
80	16	20.00%
90	20	22.22%
100	20	20.00%

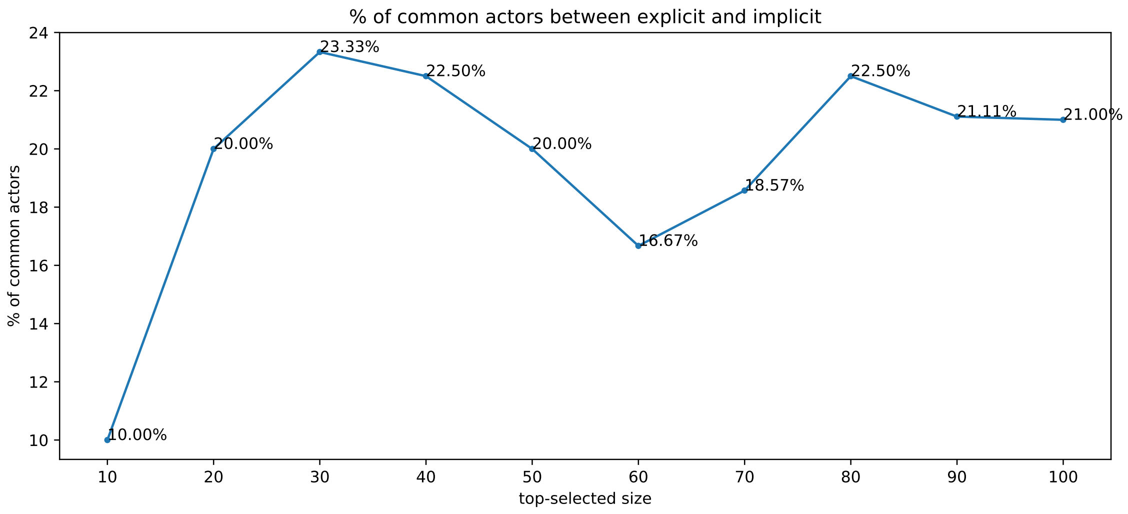


Figure 5.9: Common quota of actors between the most selected k attributes, either explicitly or implicitly

These results generally confirm the previous findings in [34] regarding existing gaps (i.e., low quota of common attributes) between explicitly selected features and implicitly estimated ones, with a different dataset containing more up-to-date movies and not limited to the most popular movies as used in [34].

In Table 5.11, we compare the 10 most frequently selected actors, either *explicitly* or *implicitly*, by *male* and *female* users, respectively.

Table 5.11: Most selected 10 actors, either explicitly or implicitly, by male and female users; $\left|R_f^{exp}\right|$ — number of explicit selections of considered feature, $\left|R_f^{imp}\right|$ — number of implicit selections of considered feature.

	Pos.	Explicitly selected	$\left\|R_f^{exp}\right\|$	Implicitly selected	$\left\|R_f^{imp}\right\|$
Male users	1	Robert Downey Jr.	16	Samuel L. Jackson	64
	2	Johnny Depp	15	Stan Lee	56
	3	Jason Statham	10	Bradley Cooper	51
	4	Leonardo DiCaprio	10	Paul Bettany	47
	5	Tom Hardy	8	Vin Diesel	47
	6	Harrison Ford	3	Sean Gunn	15
	7	Bruce Willis	3	Terry Notary	15
	8	Sean Bean	3	Bradley Cooper	15
	9	Vin Diesel	2	Zoe Saldana	15
	10	Leonardo DiCaprio	2	Scarlett Johansson	14
Female users	1	Robert Downey Jr.	12	Stan Lee	27
	2	Leonardo DiCaprio	7	Samuel L. Jackson	26
	3	Jennifer Lawrence	5	Bradley Cooper	23
	4	Chris Hemsworth	5	Djimon Hounsou	21
	5	Bruce Willis	4	James McAvoy	21
	6	Harrison Ford	3	Sean Gunn	15
	7	Bruce Willis	3	Terry Notary	15
	8	Sean Bean	3	Bradley Cooper	15
	9	Vin Diesel	2	Zoe Saldana	15
	10	Leonardo DiCaprio	2	Scarlett Johansson	14

Investigating the results for the preferences about actors, we notice a substantial difference between male and female users. it is surprising that in Table 5.11, for both male and female users' preferences, Stan Lee is among the top *implicitly* preferred actors even if he barely acted as a main character in any movie. The most probable reason is that even though he has not been selected explicitly as favourite actor by study participants, he appeared in all Marvel movies (in small "cameo roles"), so he is included in the implicit profiles.

Like previously done for genres (in section 5.2.2), we further provide an analysis of the convergence or divergence of users' tastes, defined by explicitly rated actors or implicitly extracted. Let $\mathcal{F}$ denote the set of all possible actors contained in items and d_f the distribution of an actor f across all user profiles. d_f is a vector consisting of binary attributes $(f_1, f_2, ..., f_n)$ in which $f_k = 1$ if feature f is contained in the profile of user k. We considered two subset of top-50 selected actors in explicit profiles and implicit profiles and we computed their corresponding distribution and standard deviation (std), denoted as $\sigma_{f,e}$ and $\sigma_{f,i}$, respectively.

In table 5.12 the actors with the highest std across explicit and implicit profiles are shown, i.e. the actors for which users' tastes diverge most. In table 5.13 instead we show the ones with the lowest std.

Table 5.12: Actors with the highest std across explicit and implicit user profiles

Actor with the highest std		$\sigma_{f,e}$	$\sigma_{f,i}$	$\lvert\sigma_{f,e} - \sigma_{f,i}\rvert$
Across explicit profiles	Robert Downey Jr.	0.401297	0.499853	0.098557
Across implicit profiles	Bradley Cooper	0.159071	0.501360	0.342288

Table 5.13: Actors with the lowest std across explicit and implicit user profiles

Actor with the lowest std		$\sigma_{f,e}$	$\sigma_{f,i}$	$\lvert\sigma_{f,e} - \sigma_{f,i}\rvert$
Across explicit profiles	Anthony Hopkins	0.113223	0.350115	0.236892
Across implicit profiles	Samuel l. Jackson	0.193527	0.487119	0.293592

Considering tables 5.11, 5.12 and 5.13, Robert Downey Jr is the most explicitly rated actor but it also the one for which explicit user tastes diverge most. Instead Bradley Cooper is the one for which implicit user tastes diverge most but it is not even contained in the top-10 list of explicitly rated actors.

For completeness of our analysis, in figure 5.10 we also show 4 charts of the top-5 actors with the highest std across explicit profiles and across implicit profiles (respectively, at top) and the top-5 with the lowest std (at bottom).

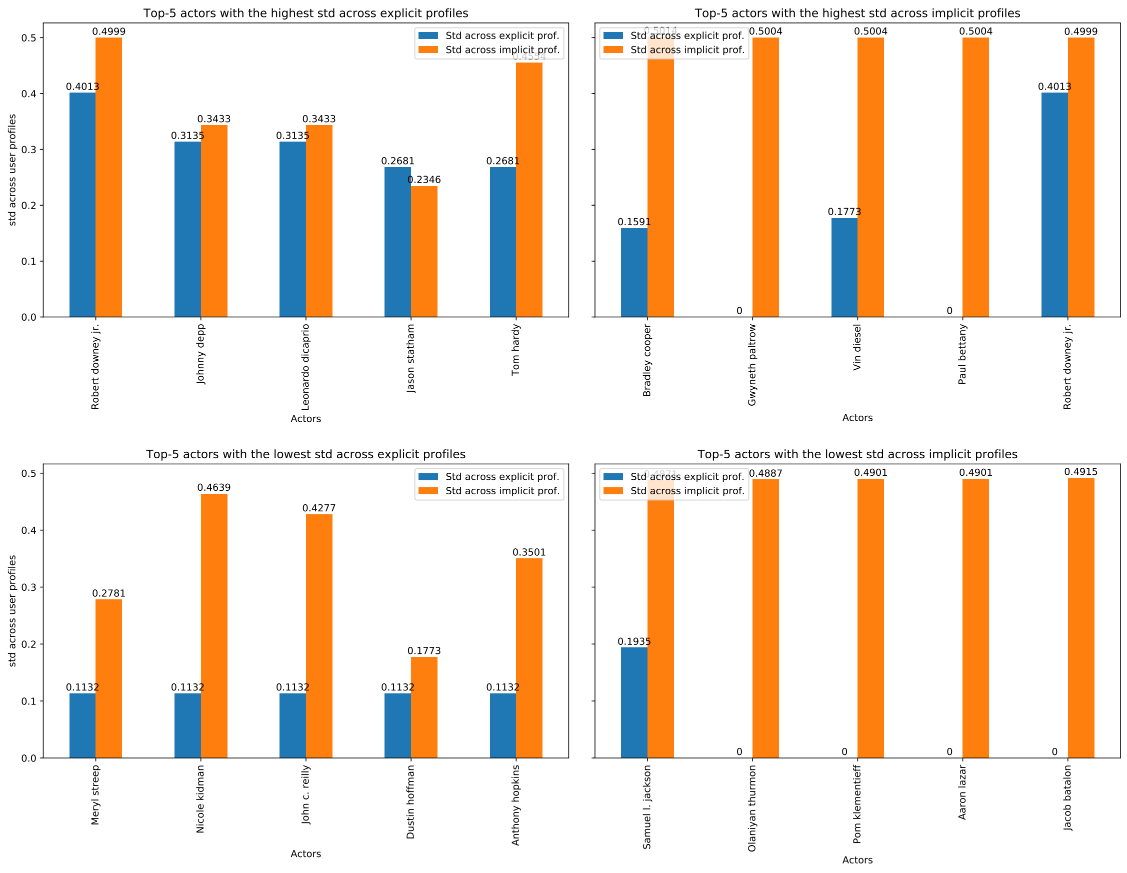


Figure 5.10: Top-5 actors with the highest std across explicit profiles and across implicit profiles (respectively, at top) and the top-5 with the lowest std (at bottom)

5.2.4 Director

In Table 5.14 we show the total number of directors selected by users, hence that belong to the *explicit set*, and the total number of directors extracted from content of favorite movies, hence that belong to the *implicit set*:

Table 5.14: Directors extracted from *explicit set* and from *implicit set*

Explicit directors selected	Implicit directors extracted
198	4349

In Table 5.15 and in Figure 5.11, we present a comparison between the explicit and implicit sets of features, in percentage of common attributes (features), focusing on the k most frequently selected directors, like already analyzed for genres and actors in previous sections.

Table 5.15: Common quota of directors between the most selected k attributes, either explicitly or implicitly

k	No. of common directors	% of common directors
10	3	30.00%
20	4	20.00%
30	7	23.33%
40	11	27.50%
50	16	32.00%
60	17	28.33%
70	20	28.57%
80	26	32.50%
90	30	33.33%
100	33	33.00%

These results show remarkable differences (i.e., low quota of common attributes) between explicitly selected directors and implicitly estimated ones, like the results reported for actors in section 5.2.3 and by Nasery et al. in [34].

In Table 5.16, we compare the 10 most frequently selected directors, either *explicitly* or *implicitly*, by *male* and *female* users, respectively.

The differences between tastes of male and female users, shown for preferences about directors in table 5.16 (but also for actors in previous section), suggest to embed gender information in a recommender system.

Like previously done for genres and actors (in sections 5.2.2 and 5.2.3), we further provide an analysis of the convergence or divergence of users' tastes, defined by explicitly rated directors or implicitly extracted. Now let $\mathcal{F}$ denote the set of

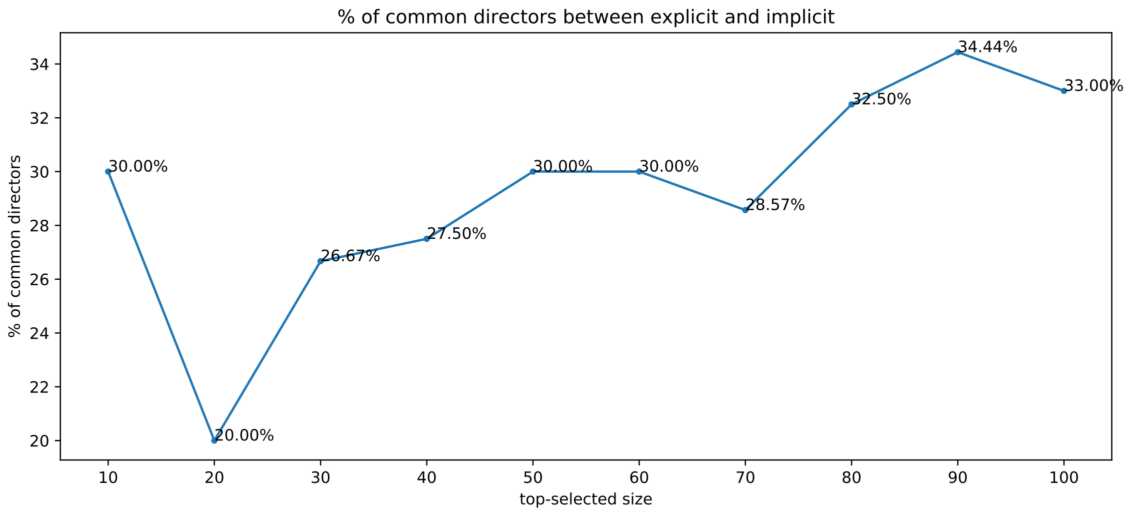


Figure 5.11: Common quota of directors between the most selected k attributes, either explicitly or implicitly

Table 5.16: Most selected 10 directors, either explicitly or implicitly, by male and female users; $\left|R_f^{exp}\right|$ — number of explicit selections of considered feature, $\left|R_f^{imp}\right|$ — number of implicit selections of considered feature.

	Pos.	Explicitly selected	$\left\|R_f^{exp}\right\|$	Implicitly selected	$\left\|R_f^{imp}\right\|$
Male users	1	Quentin Tarantino	11	Hajar Mainl	42
	2	Steven Spielberg	9	Chris Castaldi	41
	3	Joe Russo	7	Mark Rossini	41
	4	M. Night Shyamalan	6	Lori Grabowski	41
	5	Christopher Nolan	6	Eli Sasich	41
	6	Marcel Carné	1	Lori Grabowski	12
	7	Alfred Hitchcock	1	Hajar Mainl	12
	8	Ermanno Olmi	1	Chris Castaldi	12
	9	Elia Kazan	1	Steven Spielberg	10
	10	George Lucas	1	Bryan Singer	10
Female users	1	Joe Russo	4	Anthony Russo	16
	2	Christopher Nolan	4	Joe Russo	16
	3	Steven Spielberg	4	Bryan Singer	15
	4	Martin Scorsese	2	Hajar Mainl	14
	5	Ridley Scott	2	Chris Castaldi	14
	6	Marcel Carné	1	Lori Grabowski	12
	7	Alfred Hitchcock	1	Hajar Mainl	12
	8	Ermanno Olmi	1	Chris Castaldi	12
	9	Elia Kazan	1	Steven Spielberg	10
	10	George Lucas	1	Bryan Singer	10

all possible directors contained in items and d_f the distribution of a director f across all user profiles. d_f is a vector consisting of binary attributes $(f_1, f_2, ..., f_n)$ in which $f_k = 1$ if feature f is contained in the profile of user k. We considered two subset of top-50 selected directors in explicit profiles and implicit profiles and we computed their corresponding distribution and standard deviation (std), denoted as $\sigma_{f,e}$ and $\sigma_{f,i}$, respectively (like done for genres in section 5.2.2).

In table 5.17 the directors with the highest std across explicit and implicit profiles are shown, i.e. the directors for which users' tastes diverge most. In table 5.18 instead we show the ones with the lowest std.

Table 5.17: Directors with the highest std across explicit and implicit user profiles

Director with the highest std		$\sigma_{f,e}$	$\sigma_{f,i}$	$\|\sigma_{f,e} - \sigma_{f,i}\|$
Across explicit profiles	Quentin Tarantino	0.278093	0.296608	0.018516
Across implicit profiles	Anthony Russo	0.113223	0.491486	0.378263

Table 5.18: Directors with the lowest std across explicit and implicit user profiles

Director with the lowest std		$\sigma_{f,e}$	$\sigma_{f,i}$	$\|\sigma_{f,e} - \sigma_{f,i}\|$
Across explicit profiles	Bryan Singer	0.080322	0.435347	0.355025
Across implicit profiles	Aleksey Sidorov	0.000000	0.313500	0.313500

Investigating results of tables 5.16, 5.17 and 5.18, it can be noticed that "Quantin Tarantino" is the director for which explicit user tastes diverge most, but at the same time it is also the most explicitly rated director by male users, together with "Steven Spielberg". The same interesting result regards also "Anthony Russo" across implicit profiles, it is the most implicitly favoured by female users but the one with the highest implicit std (i.e., for which implicit users tastes diverge most).

For completeness of our analysis, in figure 5.12 we also show 4 charts of the top-5 directors with the highest std across explicit profiles and across implicit profiles (respectively, at top) and the top-5 with the lowest std (at bottom).

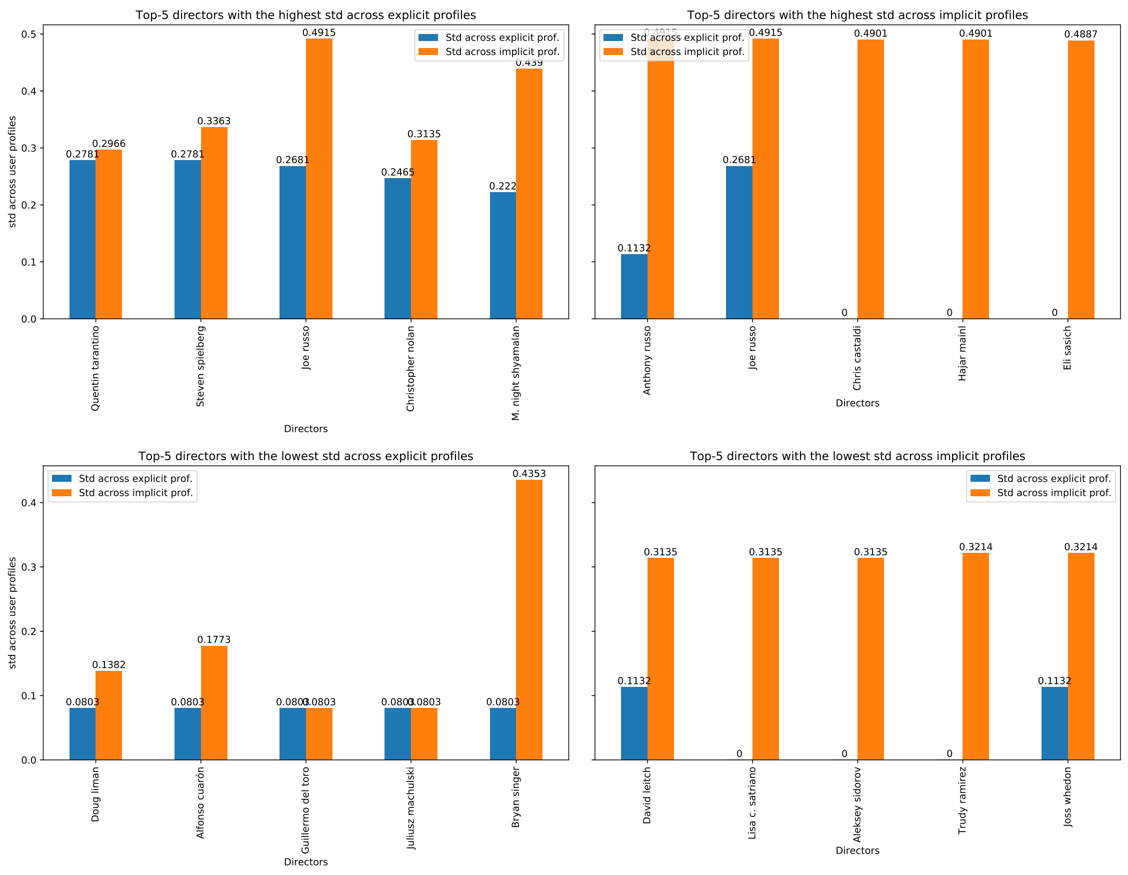


Figure 5.12: Top-5 directors with the highest std across explicit profiles and across implicit profiles (respectively, at top) and the top-5 with the lowest std (at bottom)

5.3 Evaluation of user profiling methods

In this section we study the user profiling step in-depth by investigating the 4 user profiling methods described in Section 4.1. Our aim is to analyze the similarity (i.e., the overlap) between the implicitly modelled user profiles and the real explicit tastes of users. For each target user u, we built his or her *explicit profile* $\mathbf{p}'_u$ as vector composed of relevance weights equal to 1, for all the features explicitly rated by u, and weight 0 for the ones not rated. Then we computed the pairwise similarity between the explicit user profiles and *implicit profiles* $\mathbf{p}_u$ produced by each method, using cosine similarity and Jaccard similarity, given the assumptions described in section 4.1.6. The highest is this similarity, the most accurate is the implicit user profile modelled.

In the first subsection we provide an overview of effectiveness of profiling methods, then in the next subsections we analyze in deep the methods considering only genres, actors and directors, separately.

5.3.1 Overview on pairwise similarity between user profiles

The average pairwise similarity $sim(\mathbf{p}_u, \mathbf{p}'_u)$ between implicit user profile $\mathbf{p}_u$ and explicit one $\mathbf{p}'_u$ is shown in Table 5.19 and in separated charts for each feature type, in figures 5.13, 5.14 and 5.15.

In table 5.20 we show the 95% confidence intervals for the pairwise similarity $sim(\mathbf{p}_u, \mathbf{p}'_u)$, for each user profiling method and each type of feature). To compute the confidence intervals for each method and feature type we used the t-distribution score, considering the average of pairwise similarity reported in Table 5.19 and the standard deviation $\sigma/\sqrt{|\mathcal{U}|}$, in which σ is the standard deviation of pairwise similarity of profiles and the denominator is the number of users (i.e., equal to the number of user profiles considered).

As revealed in the tables and already anticipated in Section 4.1, the TF-IDF method yields better results than Symeonidis even if they are intrinsically similar, hence the item-centric TF-IDF approach outperforms the user profle-based one. In general, the average pairwise similarities are remarkably low, even for the best investigated method, i.e., Li. The overlap between explicit and implicit profiles increases if we consider only genres; the reason is that the catalogue of all possible genres in the dataset is rather limited (19) compared to actors (567K) and directors (58K). The Jaccard measure yields lower similarities because it can be applied only to vectors composed of binary attributes while our tested profiling methods compute scalar weights (except for Zhang); hence we had to cut-off some feature weights by considering only the k most relevant features in the implicit profile of each user considered, in which k is the number of explicit features rated by that user.

The confidence intervals reported in table 5.20 show the intervals which the average pairwise similarities would fall into with a sample of user profiles different from the one analyzed (i.e., greater or less than our 155 users), with a confidence equal to 95%. For instance, the best performing method tested, i.e., Li, builds implicit user profiles whose average pairwise similarity with explicit profiles is between 54.72% and 61.42%, with confidence 95%.

Table 5.19: Average pairwise similarity between *explicit* and *implicit* user profiles, for all the methods.

Similarity	Feature	Zhang m.	Li m.	Symeonidis m.	tf-idf m.
Cosine	Genre	48.52%	58.07%	42.00%	53.08%
	Actor	7.03%	9.13%	6.50%	7.24%
	Director	15.17%	17.24%	15.32%	16.14%
Jaccard	Genre	27.49%	36.19%	18.54%	33.36%
	Actor	0.97%	5.73%	2.87%	4.64%
	Director	5.22%	10.24%	6.30%	8.17%

Table 5.20: 95% confidence intervals of pairwise similarity between *explicit* and *implicit* user profiles, for all the methods.

Cosine similarity				
Feature	Zhang m.	Li m.	Symeonidis m.	tf-idf m.
Genre	(45.99%, 51.04%)	(54.72%, 61.42%)	(38.59%, 45.42%)	(49.64%, 56.52%)
Actor	(6.10%, 7.96%)	(7.95%, 10.31%)	(5.49%, 7.51%)	(6.26%, 8.22%)
Director	(12.51%, 17.83%)	(14.20%, 20.27%)	(12.40%, 18.24%)	(13.25%, 19.02%)
Jaccard similarity				
Feature	Zhang m.	Li m.	Symeonidis m.	tf-idf m.
Genre	(25.19%, 29.78%)	(34.09%, 43.15%)	(15.31%, 21.77%)	(29.00%, 37.71%)
Actor	(0.74%, 1.19%)	(3.90%, 7.70%)	(1.65%, 4.09%)	(2.82%, 6.46%)
Director	(3.65%, 6.79%)	(5.45%, 14.00%)	(2.87%, 9.74%)	(4.25%, 12.08%)

In the next sub-sections we analyze in deep the number of user profiles within each range of pairwise similarity, considering only genres, actors and directors as features, separately.

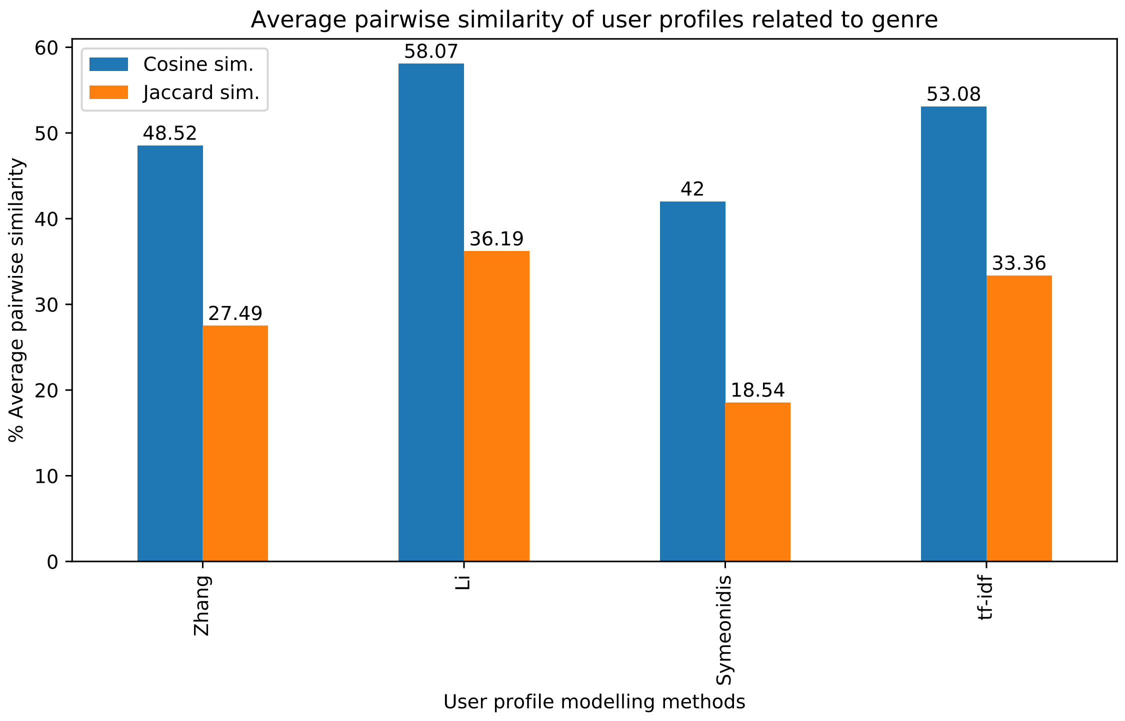


Figure 5.13: Average pairwise similarity between all *explicit* and *implicit* user profiles related to genres for all the methods

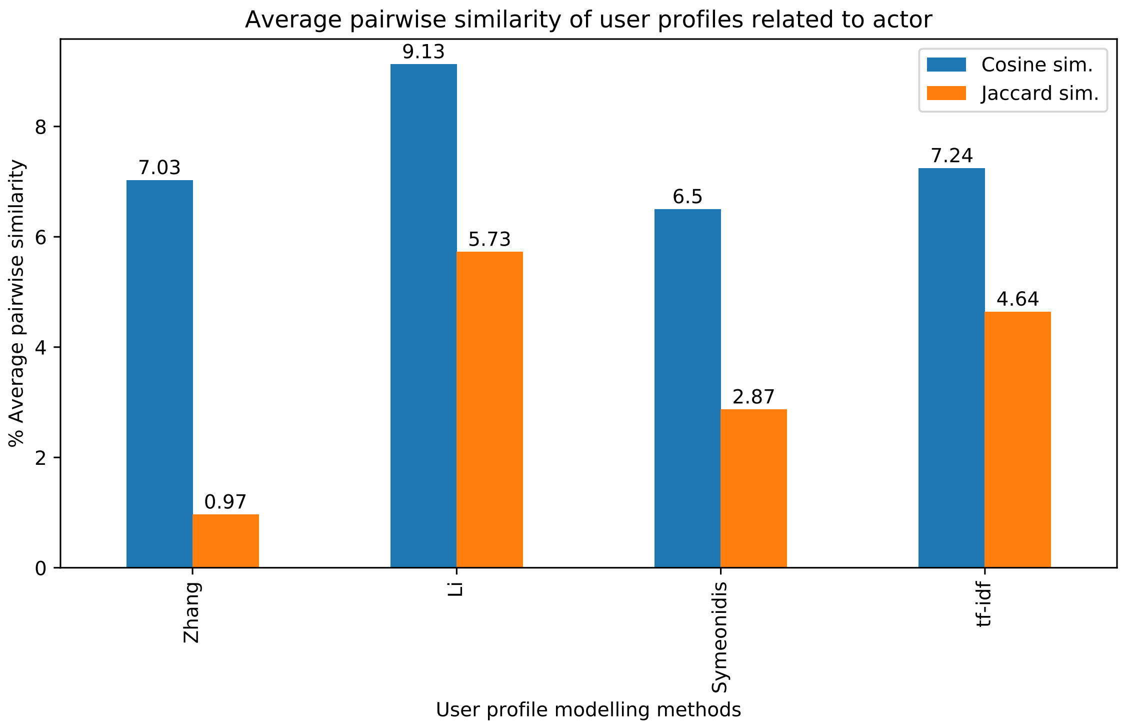


Figure 5.14: Average pairwise similarity between all *explicit* and *implicit* user profiles related to actors for all the methods

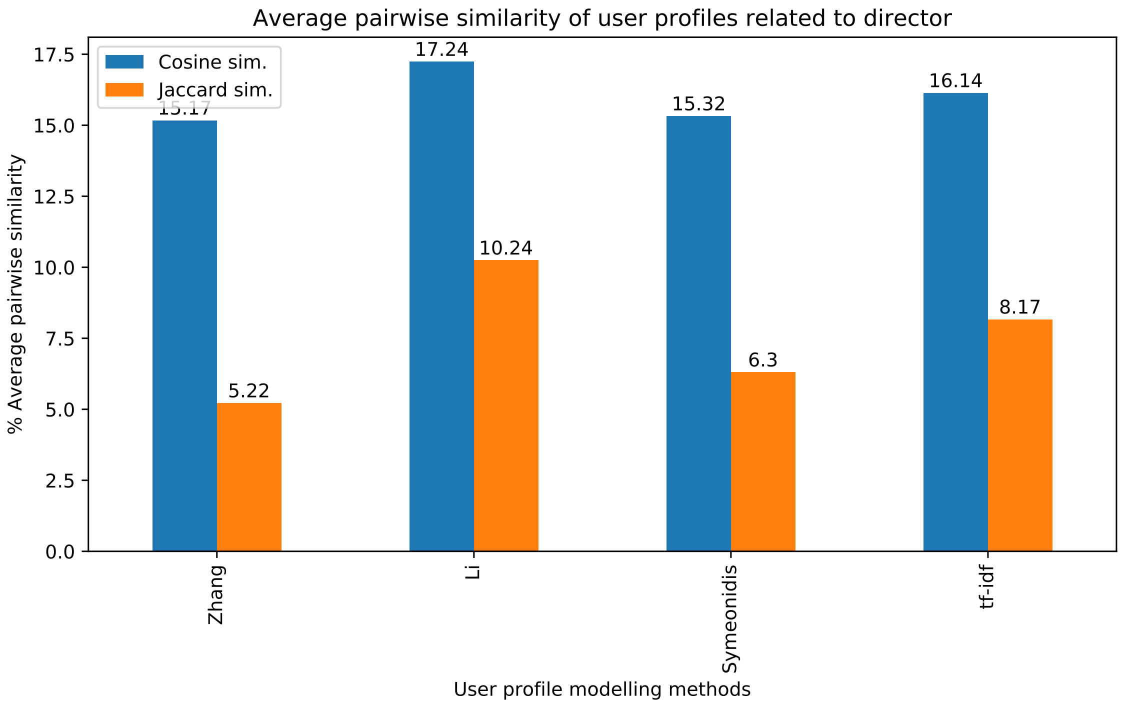


Figure 5.15: Average pairwise similarity between all *explicit* and *implicit* user profiles related to directors for all the methods

5.3.2 Genre

To compare the outcomes of the user profile modelling methods, considering only genres as features, we show the number of user profiles within each range of similarity between explicit and implicit.

The results for genres are shown in Table 5.21 and in figures 5.16 and 5.16.

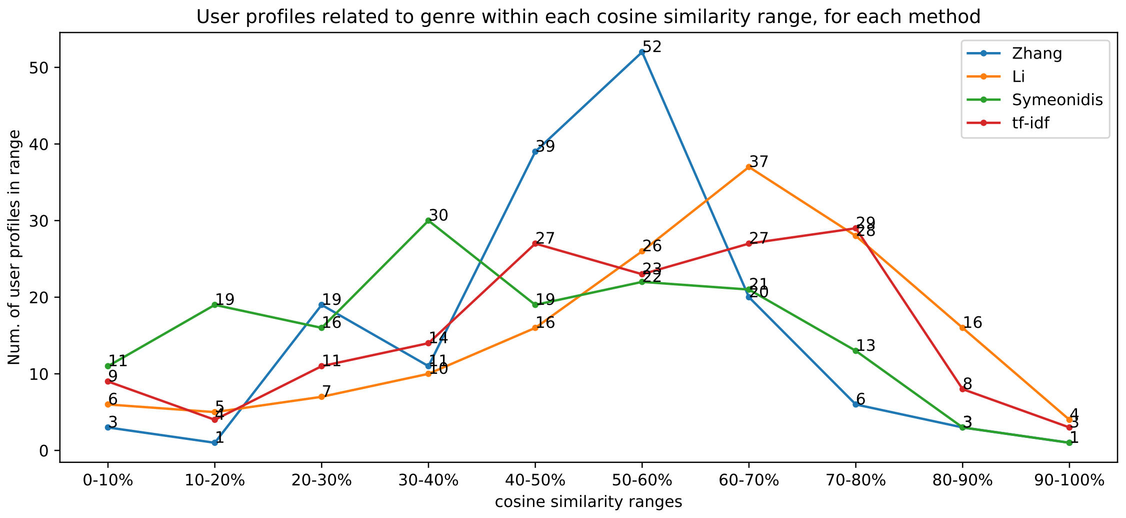


Figure 5.16: Number of user profiles within each range of pairwise cosine similarity between explicit and implicit, considering only genres as features

Table 5.21: Number of user profiles within each range of pairwise similarity between explicit and implicit, considering only genres as features

Similarity	Similarity ranges	Zhang m.	Li m.	Symeonidis m.	tf-idf m.
Cosine	0-10%	3	6	11	9
	10-20%	1	5	19	4
	20-30%	19	7	16	11
	30-40%	11	10	30	14
	40-50%	39	16	19	27
	50-60%	52	26	22	23
	60-70%	20	37	21	27
	70-80%	6	28	13	29
	80-90%	3	16	3	8
	90-100%	1	4	1	3
Jaccard	0-10%	9	33	71	36
	10-20%	30	2	8	9
	20-30%	72	11	17	9
	30-40%	22	57	39	53
	40-50%	15	9	6	10
	50-60%	4	19	7	16
	60-70%	3	7	5	8
	70-80%	3	1	1	1
	80-90%	1	0	0	0
	90-100%	0	16	1	13

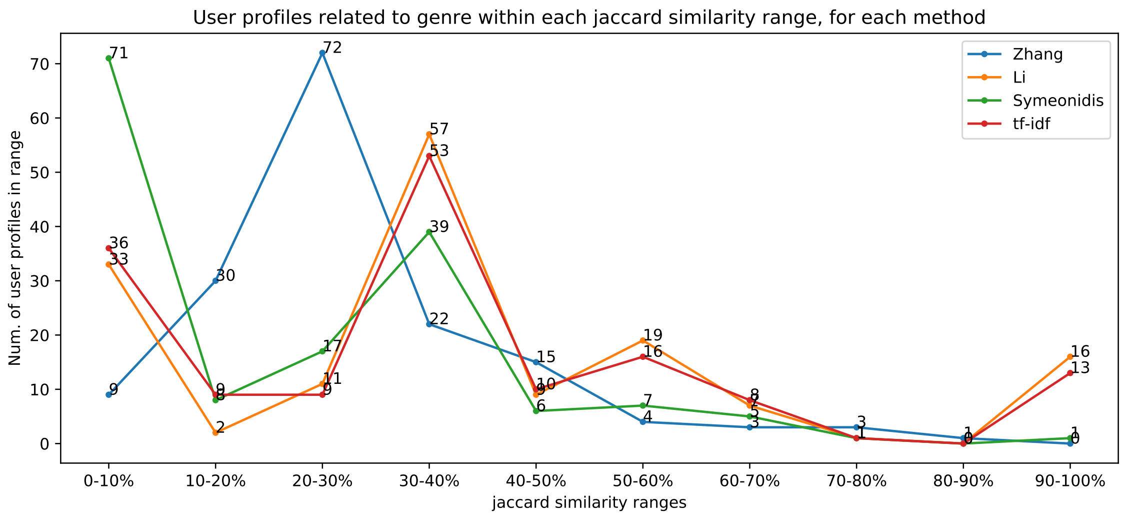


Figure 5.17: Number of user profiles within each range of pairwise Jaccard similarity between explicit and implicit, considering only genres as features

In particular, the number of user profiles with pairwise similarity exactly equal to 0% and 100% are shown in Table 5.22.

Table 5.22: Number of user profiles with pairwise similarity between explicit and implicit equal to 0% and 100%, considering only genres as features

Similarity	Similarity score	Zhang m.	Li m.	Symeonidis m.	tf-idf m.
Cosine	0%	3	3	3	3
	100%	0	0	0	0
Jaccard	0%	3	33	71	36
	100%	0	16	1	13

These results underline that Li and TF-IDF methods are the ones performing better, they produce the highest number of implicit user profiles with high range of similarity with explicit profiles, especially considering Jaccard similarity.

5.3.3 Actor

The results of pairwise similarity ranges of user profiles, considering only actors as features, are shown in Table 5.23 and in figures 5.18 and 5.18.

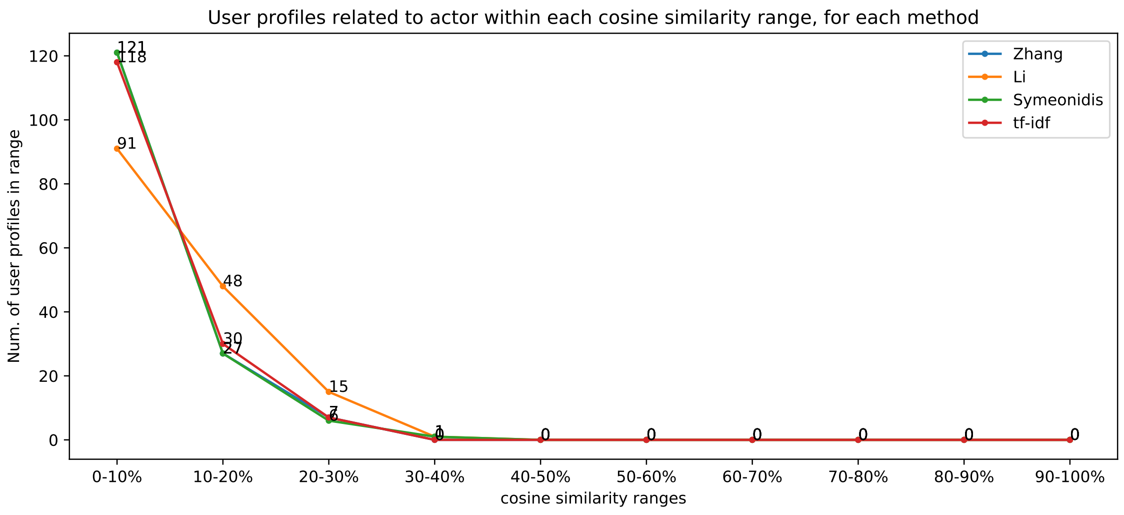


Figure 5.18: Number of user profiles within each range of pairwise cosine similarity between explicit and implicit, considering only actors as features

In particular, the number of user profiles with pairwise similarity exactly equal to 0% and 100% are shown in Table 5.24.

The pairwise similarities between user profiles are very low if we consider only actors, even with the best performing method (e.g., Li). Hence, these result underline the need of further research on user profiling methods, especially in case

Table 5.23: Number of user profiles within each range of pairwise similarity between explicit and implicit, considering only actors as features

Similarity	Similarity ranges	Zhang m.	Li m.	Symeonidis m.	tf-idf m.
Cosine	0-10%	121	91	121	118
	10-20%	27	48	27	30
	20-30%	7	15	6	7
	30-40%	0	1	1	0
	40-50%	0	0	0	0
	50-60%	0	0	0	0
	60-70%	0	0	0	0
	70-80%	0	0	0	0
	80-90%	0	0	0	0
	90-100%	0	0	0	0
Jaccard	0-10%	155	119	135	126
	10-20%	0	15	9	9
	20-30%	0	14	9	16
	30-40%	0	4	0	2
	40-50%	0	0	0	0
	50-60%	0	2	2	1
	60-70%	0	0	0	0
	70-80%	0	0	0	0
	80-90%	0	0	0	0
	90-100%	0	1	0	1

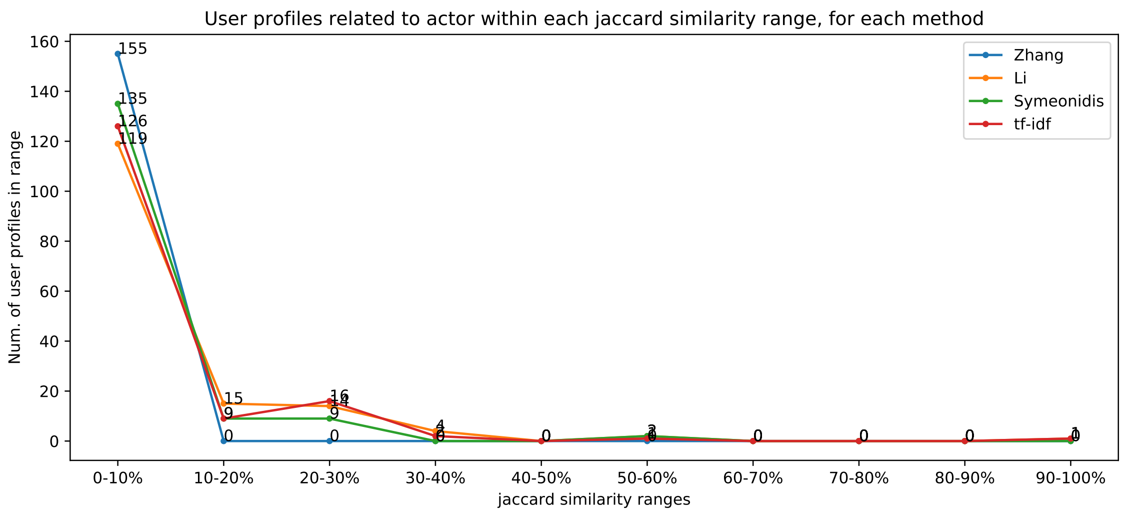


Figure 5.19: Number of user profiles within each range of pairwise Jaccard similarity between explicit and implicit, considering only actors as features

Table 5.24: Number of user profiles with pairwise similarity between explicit and implicit equal to 0% and 100%, considering only actors as features

Similarity	Similarity score	Zhang m.	Li m.	Symeonidis m.	tf-idf m.
Cosine	0%	30	30	32	32
	100%	0	0	0	0
Jaccard	0%	30	110	126	119
	100%	0	1	0	1

of features like actors because the cast of each movie contains a high number of actors, which are implicitly estimated (wrongly) as favorites.

5.3.4 Director

The results of pairwise similarity ranges of user profiles, considering only directors as features, are shown in Table 5.25 and in figures 5.20 and 5.20.

Table 5.25: Number of user profiles within each range of pairwise similarity between explicit and implicit, considering only directors as features

Similarity	Similarity ranges	Zhang m.	Li m.	Symeonidis m.	tf-idf m.
Cosine	0-10%	66	67	76	68
	10-20%	32	23	29	30
	20-30%	31	28	24	26
	30-40%	18	21	11	16
	40-50%	3	6	7	8
	50-60%	2	6	3	3
	60-70%	0	0	2	0
	70-80%	2	2	2	2
	80-90%	1	2	1	2
	90-100%	0	0	0	0
Jaccard	0-10%	129	131	137	135
	10-20%	19	2	4	1
	20-30%	3	1	1	2
	30-40%	1	6	5	6
	40-50%	0	2	1	1
	50-60%	2	0	0	0
	60-70%	0	1	0	1
	70-80%	1	0	0	0
	80-90%	0	0	0	0
	90-100%	0	12	7	9

In particular, the number of user profiles with pairwise similarity exactly equal to 0% and 100% are shown in Table 5.26.

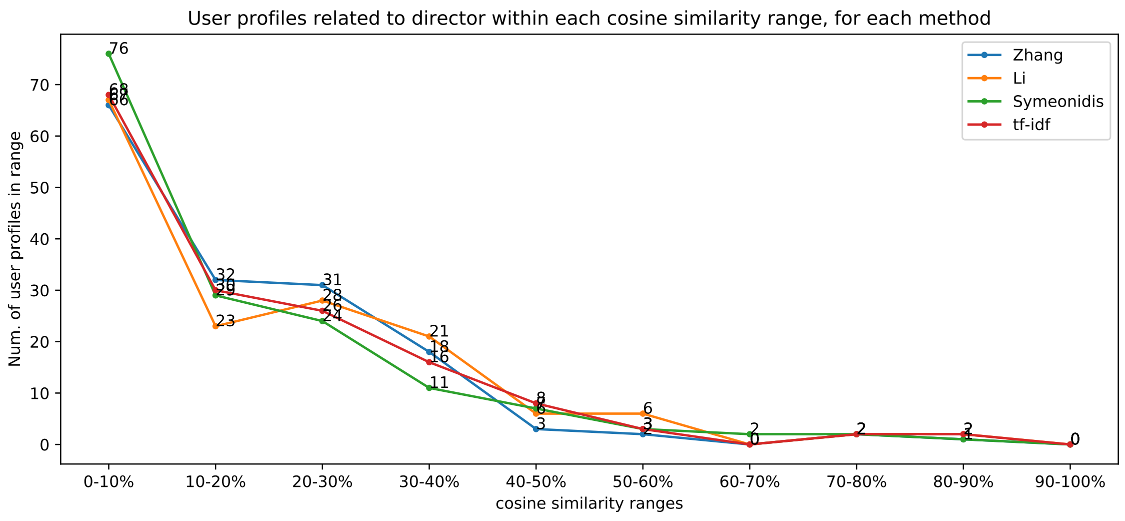


Figure 5.20: Number of user profiles within each range of pairwise cosine similarity between explicit and implicit, considering only directors as features

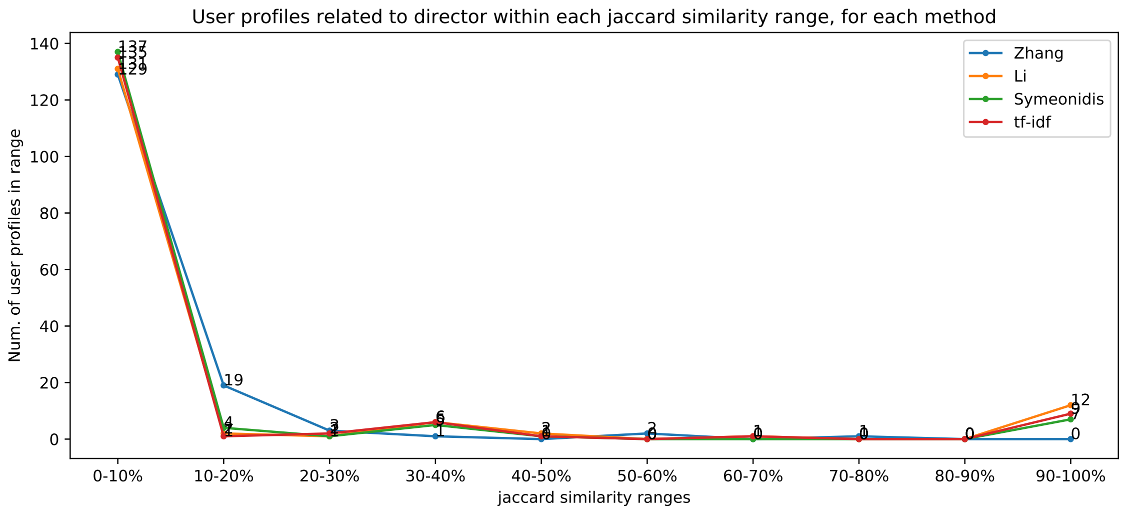


Figure 5.21: Number of user profiles within each range of pairwise Jaccard similarity between explicit and implicit, considering only directors as feature

Table 5.26: Number of user profiles with pairwise similarity between explicit and implicit equal to 0% and 100%, considering only directors as features

Similarity	Similarity score	Zhang m.	Li m.	Symeonidis m.	tf-idf m.
Cosine	0%	65	65	65	65
	100%	0	0	0	0
Jaccard	0%	65	131	137	135
	100%	0	12	7	9

Considering only directors as features, the overlap (i.e., the similarity) between implicit user profiles and explicit is slightly higher with respect to the results presented for actors (particularly with cosine similarity). The reason for this small improvement is that each movie contains only few directors, hence it's slightly more probable that these extracted attributes from rated movies are similar the ones explicitly rated by users. Anyway, all these 4 profiling methods have in general a low effectiveness, especially compared to the results reported for genres (in section 5.3.2).

5.4 Recommender System Evaluation

The results presented in previous sections for all the three types of features, underline the low effectiveness of the investigated user profiling methods to model real user tastes. This finding gives rise to the need of further research on this important user profiling step when devising recommender systems. If user profiles are not properly modelled before applying any RS technique, the accuracy of the final recommendations will likely be affected and lowered by an inaccurate representation of the user's tastes.

In this section we present the results obtained by experimenting 3 models of recommender systems: (i) collaborative filtering and (iii) 2 different hybrid RS. The settings for these RS models and for the evaluation are described in section 4.2. For all the RS models tested, we used the same train and test sets of movie ratings. We divided the movie ratings randomly (with seed 2147483647) with a holdout strategy, 80% for the train set (1,389 movie ratings), 20% for the test set (347 movie ratings). We cut the output of recommendation list to 3 predicted items.

Collaborative filtering (CF): We experimented an item-based CF recommender system, by considering only movies rated by the reliable users. The results shown in the first 2 columns of AUC and MAP in table 5.27 are obtained in specific with item-based CF KNN considering several values for k neighbors in the KNN.

As revealed in table 5.27, the best accuracy for item-based CF KNN with cosine similarity is obtained by setting $k = 2$ as number of neighbors considered, because it gives the highest values of AUC and MAP. Also with Jaccard similarity the best result is obtained with $k = 2$ and this is also the most accurate overall, among cosine and Jaccard results.

After that, we experimented an hybrid approach with two models, one built from explicit feature ratings and the other one from implicit feature ratings obtained by the best modelled user profiling method, i.e. Li method, as we revealed in section 5.3.1.

Explicit user-user hybrid: The first user-user hybrid RS, that we call *explicit*, is built with user-user hybrid KNN considering reliable users, their rated movies and their explicit profiles composed by their favoured attributes (genres, actors, directors) to compute the correlation. The results for the explicit hybrid RS are reported in the second pair of columns AUC and MAP in table 5.27.

Implicit user-user hybrid: The latter hybrid RS, that we call *implicit*, is built by applying the Li method to extract the implicit user profiles from rated movies contained in the train set only. Then, we kept for each user and for each type of attribute (genres, actors, directors) only n implicit preferences with the highest relevance score (computed by Li method), where n is the number of explicit preferences expressed for that attribute type. For instance, if a user explicitly selected 2 favoured genres on *mints*, then only the 2 implicit genres with the highest relevance score are considered. The results for the implicit hybrid RS are reported in the third (and last) pair of columns AUC and MAP in table 5.27.

The explicit hybrid RS provided the best results (i.e., the highest AUC and MAP values) by using Jaccard similarity and $k = 4$ neighbors in the KNN, while the *implicit* hybrid RS was more accurate with cosine similarity and $k = 4$. Overall, both the the two user-user hybrid RS performed much better than the CF KNN because their AUC and MAP values are much higher. In table 5.27 we highlighted in bold the best result for each RS, depending on k and similarity function.

In figures 5.22 and 5.23 we represent the comparison between AUC and MAP scores, respectively, for the explicit and implicit hybrid RS, by considering only the similarity function providing better results, i.e., cosine for the explicit RS and Jaccard for the implicit one.

5.5 Insights from user profiling and recommendation evaluation

In table 5.28 we summarize the best results of the evaluation of user profiling stage and of recommendations, extracted from what has been shown in the entire chapter 5.

The best investigated user profiling method, i.e., Li, provided, as average, poor

Table 5.27: Results of CF KNN, explicit user-user hybrid and implicit user-user hybrid, depending on k neighbors considered and similarity function

		Item-based CF		Explicit Hybrid		Implicit Hybrid	
Similarity	k	AUC	MAP	AUC	MAP	AUC	MAP
Cosine	2	0.60679	0.05799	0.61588	0.11263	0.62024	0.09106
	3	0.59391	0.04678	0.63739	0.11867	0.62477	0.11565
	4	0.59388	0.04760	0.64599	0.11867	**0.63340**	**0.11927**
Jaccard	2	**0.61119**	**0.06640**	0.61587	0.11215	0.60295	0.07512
	3	0.61111	0.06339	0.63312	0.12436	0.61179	0.09874
	4	0.61116	0.05872	**0.64174**	**0.12509**	0.61610	0.10236

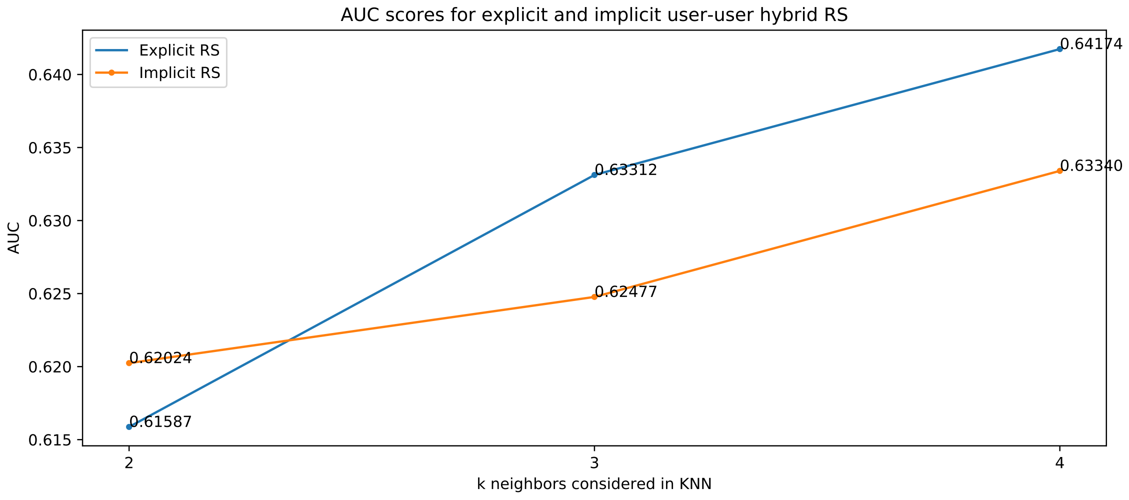


Figure 5.22: AUC scores for explicit and implicit user-user hybrid RS

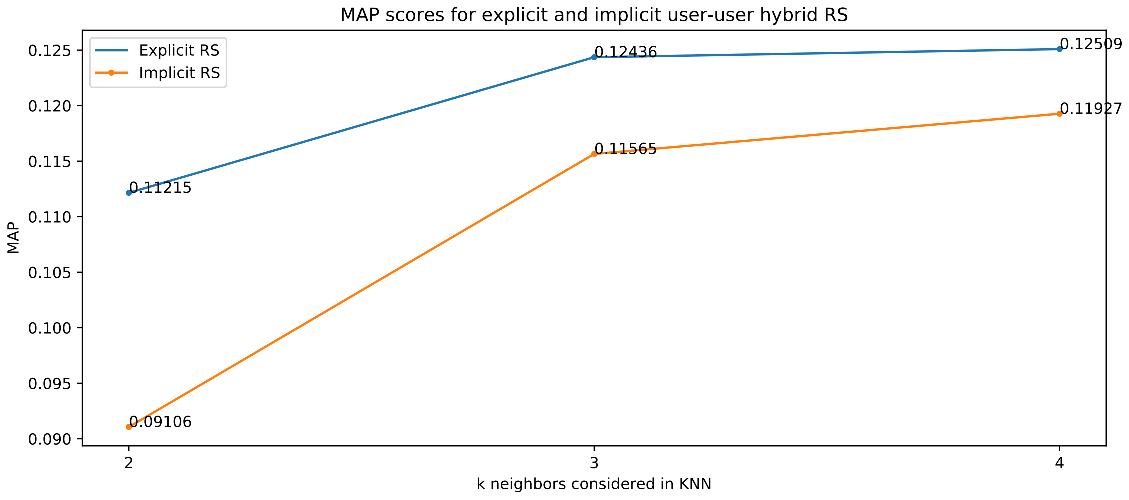


Figure 5.23: MAP scores for explicit and implicit user-user hybrid RS

results of similarity between the modeled implicit user profiles (by the method) and the explicit preferences selected by the users. The only exception is given by considering only genre attributes because the catalogue of existing genres is limited. Hence, our isolated evaluation of user profiling stage encourages to improve user profiling methods to better model the real users' tastes. The two hybrid RS models proposed, in order to integrate explicit feature ratings, in one case, and implicit preferences from modeled user profile, in the second case, confirmed the need and the usefulness of embedding the preferences about item attributes in a RS, as much accurate as possible (i.e., most similar to real user tastes). In fact, the two hybrid RS presented higher values of AUC and MAP than the CF algorithm, which did not integrate information on user profiles and item content. Furthermore, the explicit hybrid RS, embedding true explicit feature preferences, produced more accurate recommendations than the implicit one, embedding implicit user profile with Li method.

Overall, our evaluation results of user profiling step alone and of recommendations, highlight the importance and the need for more in-depth research on user profiling methods and on RS models to embed this information.

Table 5.28: Summary results of user profiling and recommendations evaluation

Best investigated profiling method	Li
Maximum avg. pairwise similarity between *explicit* and *implicit* user profiles, considering genres only	58.07%
Maximum avg. pairwise similarity between *explicit* and *implicit* user profiles, considering actors only	9.13%
Maximum avg. pairwise similarity between *explicit* and *implicit* user profiles, considering directors only	17.24%
Highest AUC and MAP of CF RS	0.61119 AUC, 0.06640 MAP
Highest AUC and MAP of *explicit* hybrid RS	0.64174 AUC, 0.12509 MAP
Highest AUC and MAP of *implicit* hybrid RS	0.63340 AUC, 0.11927 MAP

Conclusions

In this work, we analyzed in-depth the user profile modeling problem, which is a key part of personalized recommender systems based on content, by studying the differences between explicit user preferences and implicit user profiles. The explicit user profile is composed of feature-level preference scores provided by the user (e.g., through surveys or ratings), while the implicit profile is built by inferring feature-level preference scores starting from item-level ratings. Typically recommender systems use the implicit user profiles, mainly because they do not collect explicit feature-level preference scores from users. The goal of this work was, therefore, to investigate the importance and effectiveness of user profiling stage by studying the implicit profiles inferred via state-of-the-art user profiling techniques. To this end, we analyzed the gap between explicit and implicit profiles and we evaluated the final accuracy of recommendations based on these profiles.

We built a web application, called mints, in order to collect explicit feature-level ratings from users via a free navigation through any existing movie or related feature (genres, actors, directors,...). Then we used the collected ratings as a ground-truth to evaluate user profiling methods, by analyzing the pairwise similarity between explicit and implicit user profiles.

Our results showed a maximum average pairwise cosine similarity of 58.07% between the explicit feature preferences and the implicit user profiles modelled by the best investigated profiling method and considering movies' genres only. Considering actors and directors, this maximum similarity is only 9.13% and 17.24%, respectively. These results mean that there is a low overlap (i.e., a gap) between modelled implicit profiles and true explicit preferences, hence that the investigated user profiling methods do not model real users' tastes accurately.

To provide a finer-grained analysis on the quality/usefulness of user profiling stage we also experimented 3 RS models: (i) item-based collaborative filtering, considering only movie ratings, (ii) explicit user-user hybrid RS, embedding explicit feature ratings, and (iii) implicit user-user hybrid RS, embedding implicit user profiles built with the best-investigated method. We showed that both the hybrid RSs, which integrate feature preferences, resulted in better recommendation quality compared with a classical item-based CF, which considers only item ratings, with respect to AUC and MAP@3 metrics. Furthermore, our hybrid RS with explicit preferences performed better than the same model integrating the implicit user profiles, built with the best-investigated profiling method. Therefore, these results underline again that user profiling methods are extremely important in content-driven recommender systems and they should be studied more in-depth in order to reach the accuracy provided by true feature preferences, evaluated also on the final output of RS.

The imperial results of evaluation performed in this research showed that if user profile are not computed accurately, they can in turn affect the quality of recommendations. It is therefore our belief, that more efforts should be made on collecting novel preference elicitation methods that can collect reliable user opinion on content features and the core user modelling stage thereby.

As a contribution to research in this field, we publicly release the web application[1], which can be used in the future to acquire explicit preferences of more users in order to enrich the current dataset, which we provide at `http://bit.ly/2XzIYyL`. Other ways to acquire feature ratings, without asking them explicitly, could be, for instance, the collection of implicit feedbacks provided by users. For instance, if a user searches movies filtered by "action" genre, it could be estimated that the user implicitly likes this genre.

The limitations of our work are the following: (i) first, we evaluated only 4 user profiling methods proposed in the literature, (ii) second, we evaluated only two hybrid RS models. Surely future studies on more user profiling methods and hybrid RS models embedding user profiles could advance furthermore the research, also by using a larger dataset than ours.

[1] *mints*: movieinterests.herokuapp.com

Acronyms

RS	Recommender system
IR	Information retrieval
MRS	Movie recommender system
URM	User rating matrix The user rating matrix (URM) is matrix form used to represent the ratings of users to items (see section 2.1.1).
ICM	Item content matrix The item content matrix (ICM) is matrix form used to represent the profiles of the items (see section 2.1.1).
UCM	User content matrix
CS	Cold start
WS	Warm start
CBF	Content-based filtering
CF	Collaborative filtering
VSM	Vector space model
TF-IDF	Term Frequency-Inverse Document Frequency
IMDb	Internet Movie Database (IMDb): `www.imdb.com` IMDb (Internet Movie Database) is an online database of information related to films, television programs, home videos and video games, and internet streams, including cast, production crew and personnel biographies, plot summaries, trivia, and fan reviews and ratings. An additional fan feature, message boards, was abandoned in February 2017. Originally a fan-operated website, the database is owned and operated by IMDb.com, Inc., a subsidiary of Amazon. `www.imdb.com`

TMDb	The Movie Database (TMDb): www.themoviedb.org The Movie Database (TMDb) is a community built movie and TV database. Every piece of data has been added by our amazing community dating back to 2008. TMDb's strong international focus and breadth of data is largely unmatched and something we're incredibly proud of. Put simply, we live and breathe community and that's precisely what makes us different. www.themoviedb.org
MovieLens	MovieLens: movielens.org MovieLens is a research site run by GroupLens Research at the University of Minnesota. MovieLens uses "collaborative filtering" technology to make recommendations of movies that you might enjoy, and to help you avoid the ones that you won't. Based on your movie ratings, MovieLens generates personalized predictions for movies you haven't seen yet. MovieLens is a unique research vehicle for dozens of undergraduates and graduate students researching various aspects of personalization and filtering technologies. GroupLens Research has collected and made available rating data sets from the MovieLens web site. movielens.org
MTurk	Amazon Mechanical Turk (MTurk): www.mturk.com Amazon Mechanical Turk (MTurk) is a crowdsourcing marketplace that makes it easier for individuals and businesses to outsource their processes and jobs to a distributed workforce who can perform these tasks virtually. This could include anything from conducting simple data validation and research to more subjective tasks like survey participation, content moderation, and more. MTurk enables companies to harness the collective intelligence, skills, and insights from a global workforce to streamline business processes, augment data collection and analysis, and accelerate machine learning development. www.mturk.com
mints	movie Interests (mints): movieinterests.herokuapp.com movie Interests (mints) is the web application built by us for the project in order to collect the dataset of users and their explicit favorite movies and features. movieinterests.herokuapp.com
target user	target user: the user we are recommending items to

Bibliography

[1] Charu C Aggarwal. "An introduction to recommender systems". In: Recommender systems. Springer, 2016, pp. 1–28 (cit. on p. 1).

[2] Hyung Jun Ahn. "A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem". In: Information Sciences 178.1 (Jan. 2008), pp. 37–51. ISSN: 0020-0255. DOI: 10.1016/J.INS.2007.07.024. URL: https://www.sciencedirect.com/science/article/pii/S0020025507003751 (cit. on p. 10).

[3] James Allan et al. "Topic Detection and Tracking Pilot Study Final Report". In: Proceedings of the Darpa broadcast news transcription and understanding (1998), pp. 194–218. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.6373 (cit. on p. 20).

[4] Fahad Anwaar et al. "HRS-CE: A hybrid framework to integrate content embeddings in recommender systems for cold start items". In: J. Comput. Science 29 (2018), pp. 9–18. DOI: 10.1016/j.jocs.2018.09.008. URL: https://doi.org/10.1016/j.jocs.2018.09.008 (cit. on p. 2).

[5] Rounak Banik. The Movies Dataset. Dataset on Kaggle. Version 7. 2017. URL: https://www.kaggle.com/rounakbanik/the-movies-dataset (cit. on p. 3).

[6] Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, eds. The Adaptive Web, Methods and Strategies of Web Personalization. Vol. 4321. Lecture Notes in Computer Science. Springer, 2007. ISBN: 978-3-540-72078-2. DOI: 10.1007/978-3-540-72079-9. URL: https://doi.org/10.1007/978-3-540-72079-9.

[7] Francesca Carmagnola et al. “Tag-based user modeling for social multi-device adaptive guides”. In: User Modeling and User-Adapted Interaction 18.5 (Nov. 2008), pp. 497–538. ISSN: 0924-1868. DOI: 10.1007/s11257-008-9052-2. URL: http://link.springer.com/10.1007/s11257-008-9052-2 (cit. on p. 19).

[8] William Cohen and Haym Hirsh. “Joins that Generalize: Text Classification Using WHIRL”. In: Proc. of the 4th Int’l Conference on Knowledge Discovery and Data Mining. 1998, pp. 169–173 (cit. on p. 20).

[9] Paolo Cremonesi and Roberto Turrin. “Analysis of cold-start recommendations in IPTV systems”. In: Proceedings of the 3th ACM conference on Recommender systems. New York, New York, USA: ACM Press, 2009, p. 233. ISBN: 9781605584355. DOI: 10.1145/1639714.1639756. URL: http://portal.acm.org/citation.cfm?doid=1639714.1639756 (cit. on p. 10).

[10] Yashar Deldjoo and Markus Schedl. “Retrieving Relevant and Diverse Movie Clips Using the MFVCD-7K Multifaceted Video Clip Dataset”. In: Proceedings of the 17th Int. Workshop on Content-Based Multimedia Indexing. 2019 (cit. on p. 3).

[11] Yashar Deldjoo et al. “Audio-visual encoding of multimedia content for enhancing movie recommendations”. In: Proceedings of the 12th ACM Conference on Recommender Systems. Ed. by Sole Pera et al. ACM, 2018, pp. 455–459. ISBN: 978-1-4503-5901-6. DOI: 10.1145/3240323.3240407. URL: https://doi.org/10.1145/3240323.3240407 (cit. on p. 2).

[12] Yashar Deldjoo et al. “Content-Based Multimedia Recommendation Systems: Definition and Application Domains”. In: Proceedings of the 9th Italian Information Retrieval Workshop. Ed. by Nicola Tonellotto, Luca Becchetti, and Marko Tkalcic. Vol. 2140. CEUR Workshop Proceedings. CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2140/paper15.pdf (cit. on p. 2).

[13] Yashar Deldjoo et al. “MMTF-14K: a multifaceted movie trailer feature dataset for recommendation and retrieval”. In: Proceedings of the 9th ACM Multimedia Systems Conference. Ed. by Pablo César, Michael Zink, and Niall Murray. ACM, 2018, pp. 450–455. DOI:

10.1145/3204949.3208141. URL:
https://doi.org/10.1145/3204949.3208141 (cit. on p. 3).

[14] Yashar Deldjoo et al. "Movie genome: alleviating new item cold start in movie recommendation". In: User Model. User-Adapt. Interact. 29.2 (2019), pp. 291–343. DOI: 10.1007/s11257-019-09221-y. URL:
https://doi.org/10.1007/s11257-019-09221-y (cit. on p. 2).

[15] Fuhu Deng et al. "Leveraging Image Visual Features in Content-Based Recommender System". In: Scientific Programming 2018 (2018), 5497070:1–5497070:8. DOI: 10.1155/2018/5497070. URL:
https://doi.org/10.1155/2018/5497070 (cit. on p. 2).

[16] Mehdi Elahi et al. "Exploring the Semantic Gap for Movie Recommendations". In:
Proceedings of the 11th ACM Conference on Recommender Systems.
Ed. by Paolo Cremonesi et al. ACM, 2017, pp. 326–330. ISBN:
978-1-4503-4652-8. DOI: 10.1145/3109859.3109908. URL:
https://doi.org/10.1145/3109859.3109908 (cit. on p. 2).

[17] Ralph Jose Rassweiler Filho, Jonatas Wehrmann, and Rodrigo C. Barros. "Leveraging deep visual features for content-based movie recommender systems". In:
2017 International Joint Conference on Neural Networks, IJCNN. IEEE, 2017, pp. 604–611. ISBN: 978-1-5090-6182-2. DOI:
10.1109/IJCNN.2017.7965908. URL:
https://doi.org/10.1109/IJCNN.2017.7965908 (cit. on p. 2).

[18] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. "Beyond accuracy". In:
Proceedings of the fourth ACM conference on Recommender systems. New York, New York, USA: ACM Press, 2010, p. 257. ISBN: 9781605589060. DOI:
10.1145/1864708.1864761. URL:
http://portal.acm.org/citation.cfm?doid=1864708.1864761 (cit. on p. 12).

[19] DANIELA GODOY and ANALIA AMANDI. "User profiling in personal information agents: a survey". In: The Knowledge Engineering Review 20.04 (Dec. 2005), p. 329. ISSN: 0269-8889. DOI: 10.1017/S0269888906000397.
URL:
http://www.journals.cambridge.org/abstract_S0269888906000397
(cit. on p. 19).

[20] Asela Gunawardana and Guy Shani. “A Survey of Accuracy Evaluation Metrics of Recommendation Tasks”. In: Journal of Machine Learning Research 10.Dec (2009), pp. 2935–2962. ISSN: ISSN 1533-7928. URL: `http://www.jmlr.org/papers/v10/gunawardana09a.html` (cit. on p. 10).

[21] F. Maxwell Harper and Joseph A. Konstan. “The MovieLens Datasets: History and Context”. In: TiiS 5.4 (2016), 19:1–19:19. DOI: `10.1145/2827872`. URL: `https://doi.org/10.1145/2827872` (cit. on p. 3).

[22] Jonathan L. Herlocker et al. “Evaluating collaborative filtering recommender systems”. In: ACM Transactions on Information Systems 22.1 (Jan. 2004), pp. 5–53. ISSN: 10468188. DOI: `10.1145/963770.963772`. URL: `http://portal.acm.org/citation.cfm?doid=963770.963772` (cit. on pp. 10–12).

[23] Tae-Gyu Hwang et al. “An algorithm for movie classification and recommendation using genre correlation”. In: Multimedia Tools Appl. 75.20 (2016), pp. 12843–12858. DOI: `10.1007/s11042-016-3526-8`. URL: `https://doi.org/10.1007/s11042-016-3526-8` (cit. on p. 2).

[24] Xin Jin, Yanzan Zhou, and Bamshad Mobasher. “A maximum entropy web recommendation system”. In: Proc. of the eleventh ACM SIGKDD international conference... New York, New York, USA: ACM Press, 2005, p. 612. ISBN: 159593135X. DOI: `10.1145/1081870.1081945`. URL: `http://portal.acm.org/citation.cfm?doid=1081870.1081945` (cit. on p. 17).

[25] Byeong Man Kim, Qing Li, and Jong-Wan Kim. “Extraction of user preferences from a few positive documents”. In: Proc. of the 6th international workshop on Information retrieval with Asian... Vol. 11. Morristown, NJ, USA: Association for Computational Linguistics, 2003, pp. 124–131. DOI: `10.3115/1118935.1118951`. URL: `http://portal.acm.org/citation.cfm?doid=1118935.1118951` (cit. on p. 16).

[26] Heung-Nam Kim et al. “Collaborative user modeling with user-generated tags for social recommender systems”. In: Expert Syst. Appl. 38.7 (2011), pp. 8488–8496. DOI: `10.1016/j.eswa.2011.01.048`. URL: `https://doi.org/10.1016/j.eswa.2011.01.048` (cit. on pp. 14, 17, 19).

[27] Orges Leka. IMDB Movies Dataset. Dataset on Kaggle. 2016. URL: https://www.kaggle.com/orgesleka/imdbmovies (cit. on p. 3).

[28] Qing Li and Byeong Man Kim. “Constructing User Profiles for Collaborative Recommender System”. In: Advanced Web Technologies and Applications, 6th Asia-Pacific Web Conference. Ed. by Jeffrey Xu Yu et al. Vol. 3007. Lecture Notes in Computer Science. Springer, 2004, pp. 100–110. ISBN: 3-540-21371-6. DOI: 10.1007/978-3-540-24655-8_11. URL: https://doi.org/10.1007/978-3-540-24655-8_11 (cit. on pp. 14, 16).

[29] Xin Li and Hsinchun Chen. “Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach”. In: Decision Support Systems 54.2 (2013), pp. 880–890. DOI: 10.1016/j.dss.2012.09.019. URL: https://doi.org/10.1016/j.dss.2012.09.019 (cit. on pp. 6, 8).

[30] Haibo Liu, Shi Feng, and Ge Yu. “An interest propagation based movie recommendation method for social tagging system”. In: 2017 International Conference on Machine Learning and Cybernetics. IEEE, 2017, pp. 130–135. ISBN: 978-1-5386-0406-9. DOI: 10.1109/ICMLC.2017.8107754. URL: https://doi.org/10.1109/ICMLC.2017.8107754 (cit. on p. 2).

[31] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. “Content-based Recommender Systems: State of the Art and Trends”. In: Recommender Systems Handbook. Ed. by Francesco Ricci et al. Springer, 2011, pp. 73–105. ISBN: 978-0-387-85819-7. DOI: 10.1007/978-0-387-85820-3_3. URL: https://doi.org/10.1007/978-0-387-85820-3_3 (cit. on pp. 5, 13, 14, 19, 20).

[32] Juergen Mueller. “Combining aspects of genetic algorithms with weighted recommender hybridization”. In: Proc. of the 19th International Conference on Information Integration... Ed. by Maria Indrawan-Santiago et al. ACM, 2017, pp. 13–22. DOI: 10.1145/3151759.3151765. URL: https://doi.org/10.1145/3151759.3151765 (cit. on p. 2).

[33] Mona Nasery, Matthias Braunhofer, and Francesco Ricci. “Recommendations with Optimal Combination of Feature-Based and Item-Based Preferences”. In: Proceedings of the 2016 Conference on User Modeling Adaptation...

Ed. by Julita Vassileva et al. ACM, 2016, pp. 269–273. ISBN: 978-1-4503-4368-8. DOI: 10.1145/2930238.2930282. URL: https://doi.org/10.1145/2930238.2930282 (cit. on pp. ix, xi, 3).

[34] Mona Nasery, Mehdi Elahi, and Paolo Cremonesi. "PoliMovie: a feature-based dataset for recommender systems". In: Jan. 2015. DOI: 10.13140/RG.2.2.20636.49286 (cit. on pp. ix, xi, 3, 48, 52).

[35] Netflix. Netflix Prize Data. Dataset on Kaggle. Version 1. 2009. URL: https://www.kaggle.com/netflix-inc/netflix-prize-data (cit. on p. 3).

[36] Michael J. Pazzani and Daniel Billsus. "Content-Based Recommendation Systems". In: The Adaptive Web, Methods and Strategies of Web Personalization. Ed. by Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl. Vol. 4321. Lecture Notes in Computer Science. Springer, 2007, pp. 325–341. ISBN: 978-3-540-72078-2. DOI: 10.1007/978-3-540-72079-9_10. URL: https://doi.org/10.1007/978-3-540-72079-9_10 (cit. on pp. 1, 7).

[37] Alexandrin Popescul, David M Pennock, and Steve Lawrence. "Probabilistic Models for Unified Collaborative and Content-based Recommendation in Sparse-data Environments". In: Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence. UAI'01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 437–444. ISBN: 1-55860-800-1. URL: http://dl.acm.org/citation.cfm?id=2074022.2074076 (cit. on p. 16).

[38] Qi Qi et al. "Using inferred tag ratings to improve user-based collaborative filtering". In: Proceedings of the ACM Symposium on Applied Computing. Ed. by Sascha Ossowski and Paola Lecca. ACM, 2012, pp. 2008–2013. ISBN: 978-1-4503-0857-1. DOI: 10.1145/2245276.2232110. URL: https://doi.org/10.1145/2245276.2232110 (cit. on pp. 14, 16).

[39] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Recommender Systems: Introduction and Challenges". In: Recommender Systems Handbook. Ed. by Francesco Ricci, Lior Rokach, and Bracha Shapira. Springer, 2015, pp. 1–34. ISBN: 978-1-4899-7636-9. DOI: 10.1007/978-1-4899-7637-6_1. URL: https://doi.org/10.1007/978-1-4899-7637-6_1 (cit. on pp. 1, 5, 12).

[40] Francesco Ricci et al., eds. Recommender Systems Handbook. Springer, 2011. ISBN: 978-0-387-85819-7. URL: http://www.springerlink.com/content/978-0-387-85819-7.

[41] Diego Sánchez-Moreno et al. “Inferring User Expertise from Social Tagging in Music Recommender Systems for Streaming Services”. In: Hybrid Artificial Intelligent Systems - 13th International Conference, HAIS. Ed. by Francisco Javier de Cos Juez et al. Vol. 10870. Lecture Notes in Computer Science. Springer, 2018, pp. 39–49. ISBN: 978-3-319-92638-4. DOI: 10.1007/978-3-319-92639-1_4. URL: https://doi.org/10.1007/978-3-319-92639-1_4 (cit. on p. 14).

[42] Badrul Munir Sarwar et al. “Item-based collaborative filtering recommendation algorithms”. In: Proceedings of the 10th International World Wide Web Conference. Ed. by Vincent Y. Shen et al. ACM, 2001, pp. 285–295. ISBN: 1-58113-348-0. DOI: 10.1145/371920.372071. URL: https://doi.org/10.1145/371920.372071 (cit. on pp. 9, 10).

[43] J. Ben Schafer et al. “Collaborative Filtering Recommender Systems”. In: The Adaptive Web, Methods and Strategies of Web Personalization. Ed. by Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl. Vol. 4321. Lecture Notes in Computer Science. Springer, 2007, pp. 291–324. ISBN: 978-3-540-72078-2. DOI: 10.1007/978-3-540-72079-9_9. URL: https://doi.org/10.1007/978-3-540-72079-9_9 (cit. on pp. 1, 7–9).

[44] Markus Schedl et al. “Current challenges and visions in music recommender systems research”. In: International Journal of Multimedia Information Retrieval 7.2 (June 2018), pp. 95–116. ISSN: 2192-6611. DOI: 10.1007/s13735-018-0154-2. URL: http://link.springer.com/10.1007/s13735-018-0154-2 (cit. on pp. 10–13).

[45] Andrew I. Schein et al. “Methods and metrics for cold-start recommendations”. In: Proc. of the 25th annual international ACM SIGIR conference. New York, New York, USA: ACM Press, 2002, p. 253. ISBN: 1581135610. DOI: 10.1145/564376.564421. URL: http://portal.acm.org/citation.cfm?doid=564376.564421 (cit. on p. 10).

[46] Guy Shani and Asela Gunawardana. “Evaluating Recommendation Systems”. In: Recommender Systems Handbook. Ed. by Francesco Ricci et al. Springer, 2011, pp. 257–297. ISBN: 978-0-387-85819-7. DOI:

10.1007/978-0-387-85820-3_8. URL: https://doi.org/10.1007/978-0-387-85820-3_8 (cit. on p. 9).

[47] Yue Shi, Martha Larson, and Alan Hanjalic. "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges". In: ACM Computing Surveys (CSUR) 47.1 (2014), p. 3 (cit. on p. 1).

[48] Márcio Soares and Paula Viana. "The Semantics of Movie Metadata: Enhancing User Profiling for Hybrid Recommendation". In: Recent Advances in Information Systems and Technologies. Ed. by Álvaro Rocha et al. Vol. 569. Advances in Intelligent Systems and Computing. Springer, 2017, pp. 328–338. ISBN: 978-3-319-56534-7. DOI: 10.1007/978-3-319-56535-4_33. URL: https://doi.org/10.1007/978-3-319-56535-4_33 (cit. on p. 2).

[49] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. "Feature-Weighted User Model for Recommender Systems". In: User Modeling 2007, 11th International Conference, UM. Ed. by Cristina Conati, Kathleen F. McCoy, and Georgios Paliouras. Vol. 4511. Lecture Notes in Computer Science. Springer, 2007, pp. 97–106. ISBN: 978-3-540-73077-4. DOI: 10.1007/978-3-540-73078-1_13. URL: https://doi.org/10.1007/978-3-540-73078-1_13 (cit. on pp. 14, 17).

[50] Pooja Bhatt Vashisth, Purnima Khurana, and Punam Bedi. "A fuzzy hybrid recommender system". In: Journal of Intelligent and Fuzzy Systems 32.6 (2017), pp. 3945–3960. DOI: 10.3233/JIFS-14538. URL: https://doi.org/10.3233/JIFS-14538 (cit. on p. 2).

[51] Donghui Wang et al. "A content-based recommender system for computer science publications". In: Knowl.-Based Syst. 157 (2018), pp. 1–9. DOI: 10.1016/j.knosys.2018.05.001. URL: https://doi.org/10.1016/j.knosys.2018.05.001 (cit. on p. 14).

[52] Jian Wei et al. "Collaborative filtering and deep learning based recommendation system for cold start items". In: Expert Syst. Appl. 69 (2017), pp. 29–39. DOI: 10.1016/j.eswa.2016.09.040. URL: https://doi.org/10.1016/j.eswa.2016.09.040 (cit. on p. 2).

[53] Shouxian Wei et al. "A hybrid approach for movie recommendation via tags and ratings". In: Electronic Commerce Research and Applications 18 (2016), pp. 83–94 (cit. on p. 2).

[54] Yiming Yang. "An Evaluation of Statistical Approaches to Text Categorization". In: Information Retrieval 1.1/2 (1999), pp. 69–90. ISSN: 13864564. DOI: 10.1023/A:1009982220290. URL: http://link.springer.com/10.1023/A:1009982220290 (cit. on p. 20).

[55] Lisna Zahrotun. "Comparison Jaccard similarity, Cosine Similarity and Combined Both of the Data Clustering With Shared Nearest Neighbor Method". In: Computer Engineering and Applications Journal 5.1 (Jan. 2016), pp. 11–18. ISSN: 2252-5459. DOI: 10.18495/comengapp.v5i1.160. URL: http://www.comengapp.unsri.ac.id/index.php/comengapp/article/view/160 (cit. on p. 20).

[56] Chenyi Zhang et al. "Are Features Equally Representative? A Feature-Centric Recommendation". In: Proc. of the 29th AAAI Conference on Artificial Intelligence. Ed. by Blai Bonet and Sven Koenig. AAAI Press, 2015, pp. 389–395. ISBN: 978-1-57735-698-1. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9287 (cit. on pp. 14, 15).

[57] Shuai Zhang et al. "Deep Learning Based Recommender System: A Survey and New Perspectives". In: ACM Comput. Surv. 52.1 (2019), 5:1–5:38. URL: https://dl.acm.org/citation.cfm?id=3285029 (cit. on p. 2).

[58] Yuan Cao Zhang et al. "Auralist". In: Proc. of the 5th ACM international conference on Web search... New York, New York, USA: ACM Press, 2012, pp. 13–22. ISBN: 9781450307475. DOI: 10.1145/2124295.2124300. URL: http://dl.acm.org/citation.cfm?doid=2124295.2124300 (cit. on p. 13).

[59] Zi-Ke Zhang et al. "Solving the cold-start problem in recommender systems with social tags". In: EPL (Europhysics Letters) 92.2 (Oct. 2010), p. 28002. ISSN: 0295-5075. DOI: 10.1209/0295-5075/92/28002. URL: http://stacks.iop.org/0295-5075/92/i=2/a=28002?key=crossref.680c912446eac1481a3b9d5207671f1f (cit. on p. 10).

[60] Zhou Zhao et al. "Social-Aware Movie Recommendation via Multimodal Network Learning". In: IEEE Transactions on Multimedia (2017) (cit. on p. 2).