Università degli Studi di Cagliari

Dipartimento di Matematica e Informatica

Corso di Dottorato in

Informatica

Group Recommendation With Automatic Detection and Classification of Groups

Tesi di Dottorato di

Ludovico Boratto

Supervisor:

Salvatore M. Carta

Anno Accademico 2010/2011

Contents

Introduction

A group of people decides to dine together. In order to choose the restaurant in which the group should go, each participant expresses his/her preferences about different types of food. Such preferences have to be combined, in order to choose a restaurant that maximizes the group satisfaction (i.e., a restaurant that has a type of food that satisfies most of the group).

Group recommendation is a type of recommendation designed for contexts in which more than a person is involved in the recommendation process [Jameson and Smyth, 2007a]. While the objective of a classic recommender system is to produce personalized content for users, in the form of suggestion of items that users might like [Ricci et al., 2011], group recommender systems suggest items that a group might like, by combining individual models that contain a user's preferences [Masthoff, 2011]. At the time when the work for this PhD thesis started, Group Recommendation was highlighted as a challenge in the recommendation research [Jameson and Smyth, 2007a].

> *A company decides to print recommendation flyers that present suggested products. Even if the data to produce a flyer that contains individual recommendations for each customer is available, the process of printing a different flyer for everyone would be technically too hard to accomplish and costs would be too high. A possible solution would be to set a number of different flyers to print, such that the printing process can be affordable in terms of costs and the recipients of the same flyer are interested by its content.*

With respect to classic group recommendation, the first step that such systems have to compute is the detection of groups of people with similar preferences, in order to respect the constraint on the number of recommendations that can be produced and maximize users' satisfaction.

This PhD thesis presents *ART (Automatic Recommendation Technologies)*, a set of group recommendation algorithms that detect groups of users with similar preferences.

Formally, the problem of group recommendation with automatically detected groups can be stated as follows: let $U = \{u_1, u_2, ..., u_n\}$ be a set of users, $I = \{u_1, u_2, ..., u_m\}$ be a set of items and R be a totally ordered set that expresses the possible values for a *rating* (e.g., $R = [1, 5]$ or $R = \{like, dislike\}$). A rating indicates how a particular user liked a specific item. Given a value k, that denotes the maximum number of group recommendations that can be generated, for each $q \in \{1, ..., k\}$, set the group G_q to be the set of users in U, such that each user in the group is more similar to the users in its group than to any user in another group[1]. The objective is to find a function $f : G_q \times I \rightarrow R$ that measures the usefulness of an item i for the users in G_q.

According to [Jameson and Smyth, 2007a], a system can generate group recommendations using three different approaches to aggregate individual preferences:

- prediction of the ratings for the items not rated by each user and merging of the individual recommendations made for the members of a group;
- aggregation of individual preferences into group preferences;
- construction of group preference models and prediction of the missing ratings for each group using the model.

This PhD thesis will analyze these approaches in the previously described domain.

Masthoff presented several studies [Masthoff, 2002, Masthoff, 2004, Masthoff, 2005, Masthoff, 2011] related to *group modeling*, i.e., the process used to combine multiple user models into a group model. Group modeling allows to merge

[1] The metrics that define how similar two users are and how users should be grouped are defined by the *clustering* algorithm chosen to create the partition. Clustering algorithms will be presented in Chapter 3.

the preferences of the individual members of a group and derive a group preference for each item, by using different strategies. As highlighted in [Pizzutilo et al., 2005a], “there is no strategy useful in every context independently from the environment” and the choice of the strategy that best models the group should be made after a deep analysis of the context in which the group is modeled. In every algorithm proposed, different strategies to model the groups will be analyzed.

Classic group recommender systems work with different types of groups, that are either *established* (i.e., people explicitly choose to be a part of a group, because of shared, long-term interests), *occasional* (i.e., a number of persons who do something occasionally together, like visiting a museum, and have a common aim in a particular moment), or *random* (i.e., a number of persons who share an environment in a particular moment, without explicit interests that link them). The properties of such groups, like for example the heterogeneity of an occasional group (in terms of age and interests) are managed by the systems, in order to produce the group recommendations. The question that arises with automatically detected groups is: which properties of this type of groups affect the quality of the system? A study is conducted, in order to study the per-group effectiveness of a group recommender system.

When a group recommender system deals with the suggestion of items like movies, a peculiar issue arises, i.e., *novelty* of the recommended items. In fact, if an item was already evaluated by a great part of the group, the system should limit its recommendation, since users who already considered the item would be bored to watch/read/listen to it often and it wouldn't be a real recommendation for them. A study that shows how novelty of the recommended items affect the performances of a system is presented in this thesis.

Developing the work on group recommendation, two studies which aimed at finding additional information about the detected groups were conducted. All the studies focus on finding the features that characterize a group.

The first study is a related to *market segmentation*, i.e., the process that leads to an identification of groups of people with similar interests in terms of products or services. Such groups are usually called *market segments*. This thesis presents a technique to automatically identify market segments and classify users using query logs.

The second approach is a tag clustering technique that groups related tags in a tagging system, by monitoring the activity of the users in its search engine. This allows to define sets of related tags that help the identification of a context that would make resources retrieval easier.

Specifically, the contribution of this PhD thesis are summarized as follows.

- Study of the approaches to aggregate individual preferences and generate group recommendations and identification of the approach the works best in a scenario in which groups are automatically detected.

- Study of the strategies to model groups, in order to find the strategy that works best with automatically detected groups.
- Study of the per-group effectiveness of an algorithm, in order to understand which properties of a group affect the quality of a group recommendation algorithm.
- Study of how the novelty of the recommended content affects the quality of a group recommendation algorithm.
- A technique to automatically segment markets based on query logs.
- A tag clustering technique to simplify the exploration of a tagging system.

The thesis is organized as follows: Chapter 1 presents recommender systems; Chapter 2 presents the state-of-the-art on group recommender systems; Chapter 3 presents the different algorithms for group recommendation with detection of groups proposed in the thesis; Chapter 4 illustrates the experiments conducted on the algorithms and shows the obtained results; Chapter 5 presents the study conducted on the novelty of the recommended content; Chapter 6 presents a technique to automatically identify market segments and classify users using query logs; Chapter 7 presents a technique to group tags in a tagging systems; Chapter 8 contains comments, conclusions and future work.

Chapter 1

Recommender Systems

The development of the World Wide Web and its explosive diffusion in the 1990s, caused a sudden growth in the amount of data available, overreaching the capacity of users to handle it. A new problem, usually known as *information overload*, cropped up. To overcome this limit, techniques to process data and transform it into knowledge were developed.

Recommender systems are a type of technology designed to deal with information overload and, since their appearance in the mid-1990s [Hill et al., 1995, Resnick et al., 1994, Shardanand and Maes, 1995], the interest in this field has constantly increased [Ricci et al., 2011].

This chapter introduces recommender systems, presents a survey of the state-of-the-art and the current challenges in this research area.

1.1 What is a recommender system?

Recommender systems are "a personalized information filtering technology used to either predict whether a particular user will like a particular item (prediction problem) or to identify a set of N items that will be of interest to a certain user (top-N recommendation problem)" [Bigdeli and Bahmani, 2008]". So, recommender systems aim at finding items that are likely of interest to a user, by exploiting different types of information sources related to both the users and the items.

Formally, the recommendation problem can be stated as follows: let $U = \{u_1, u_2, ..., u_n\}$ be a set of users, $I = \{u_1, u_2, ..., u_m\}$ be a set of items and R be a totally ordered set that expresses the possible values for a *rating* (e.g., $R = [1, 5]$ or $R = \{like, dislike\}$).

The recommendation problem is a classification problem, whose objective is the learning of a function $f : U \times I \to R$, that predicts the rating $p_{ui} = f(u, i)$ of a user u for an item i that the user has not rated yet[1].

So, recommender systems are directed towards users who lack in experience or cannot evaluate the huge number of alternative items that a Web site contains [Resnick and Varian, 1997]. Therefore, recommender systems are largely diffused on popular Web sites, in order to help users find what

[1] Note that a predicted rating for a user u and an item i is indicated by p_{ui}, to have a different annotation with respect to the preferences expressed by users in the form of ratings, indicated by r_{ui}.

they might be interested on. The most famous example is the website amazon.com, that employs a recommender system [Linden et al., 2003] that analyze the activity of users (item purchased, rated or liked) to personalize the store for each user.

1.2 Types of Recommender Systems

Recently [Burke, 2007, Ricci et al., 2011], a partitioning of the different types of recommender systems into six classes has been made. The recommendation techniques considered in this classification are the following.

Collaborative filtering. It is the most widely used type of recommendation and also the first developed. It works assuming that two people who had similar preferences in the past will also have similar preferences in the future. In its most simple formulation, collaborative filtering algorithms consider the preferences expressed by users in order to derive similarities between users (i.e., users are considered similar if they have similar ratings for a set of items).

Content based. Content-based recommender systems predict ratings considering how similar two items are, based on the features associated to each item. In order to predict a rating for an item not yet considered by the active user, a content-based algorithm looks for items similar to those the active user likes.

Demographic. Demographic recommendation predicts ratings considering the demographic attributes associated to a user. Like Collaborative filtering, these algorithms calculate "user-to-user correlations" but in this case correlations are not built considering the preferences expressed by users, but their personal attributes (i.e., the recommendation process is based on demographic classes).

Knowledge-based. Knowledge-based systems, similarly to how Utility-based recommendation works, try to suggest items based on a user's needs and preferences. The input of these algorithms is a description of what the active user is interested in (e.g., a query typed in a search engine). The user's need is then compared with the features of the items, in order to find a relationship between the user's need and an item to recommend.

Community-based. This type of systems base their recommendation considering the preferences of the friend of the users. Community-based algorithm consider the fact that a user tends to trust more in the recommendations made by friends than on recommendations built considering similar but anonymous users. Such systems consider both user profiles and relationships between users.

Hybrid recommender systems. These systems combine the techniques previously mentioned. This is done in order to avoid the limitations that each type of recommendation has and produce a recommendation using an hybrid system, that involves different recommendation approaches.

Chapter 2

State of the Art on Group Recommendation

This chapter describes the first challenge presented in the previous chapter: group recommendation.

The different domains of application in which group recommender systems are used are described in detail and the state-of-the-art for this class of algorithms is described. This survey was also presented in [Boratto and Carta, 2010a].

2.1 Introduction

Recommender systems aim to provide information items (web pages, books, movies, music, etc.) that are of potential interest to a user. To predict the items to suggest, the systems use different sources of data, like preferences or characteristics of users.

However, there are contexts and domains where classic recommender systems cannot be used, because people operate in *groups*. Here are some examples of such contexts:

- a system has to provide recommendations to an established group of people who share the same interests and do something together;
- recommendations are provided to an heterogeneous group of people who has a common, specific aim and shares the system on a particular occasion;
- a system tries to recommend items in an environment shared by people who do not have anything in common (e.g., background music in a room);
- when a limitation in the number of available recommendations to be provided is given, individuals with similar preferences have to be grouped.

To manage such cases, group recommendation was introduced. These systems aim to provide recommendations to groups, considering the preferences and the characteristics of more than a user. But what is a *group*? As we can see from the list above, there are at least four different notions of group:

1. **Established group**: a number of persons who explicitly choose to be a part of a group, because of shared, long-term *interests*;

2. **Occasional group**: a number of persons who do something occasionally together, like visiting a museum. Its members have a common *aim* in a particular moment;

3. **Random group**: a number of persons who share an environment in a particular moment, without explicit interests that link them;

4. **Automatically detected group**: groups that are automatically detected considering the preferences of the users and/or the resources available, in order to face the limitations imposed by a system for the recommendation process.

Of course the way a group is formed affects the way it is modeled and how recommendations are predicted. The next sections introduce related work on group recommendation and the different types of group recommendation.

2.2 State-of-the-art in group recommendation

This section will present a survey of the state-of-the-art in group recommendation. A few years ago [Jameson and Smyth, 2007b] presented a state-of-the-art survey too, dividing the group recommendation process into four subtasks and describing how each system handles each subtask. Here we will try to describe the existing approaches, focusing on the different notions of group and how the type of group affects the way the system works.

The rest of the section is organized as follows: section 2.2.1 describes approaches that consider groups with an a priori known structure; section 2.2.4 considers systems that automatically detect groups.

2.2.1 Group recommendation for groups with an a priori known structure

Systems that consider established groups

An *established group* is formed by people who share common interests for a long period of time. According to [O'Connor et al., 2001] established groups have the property to be *persistent* and users actively *join* the group.

As Table **??** shows, group recommender systems that aim to established groups are designed for domains of recommendation like:

- entertainment/cultural items (books, music and movies);
- documents (web pages and conferences documents).

Group recommender systems for entertainment/cultural items *GRec_OC (Group Recommender for Online Communities)* [Kim et al., 2009] is a book recommender system for online communities (i.e., people with similar interests that share information). The system aims to improve satisfaction of individual users.

The approach works in two phases. Since the system aims to established groups, the first phase uses a classic Collaborative Filtering (CF) method to build a group profile, by merging the profiles of its members. Each group's nearest neighbors are found and a "candidate recommendation set" is formed by selecting the top-*n* items. To achieve satisfaction of each member, the second phase evaluates the relevance of the books in the candidate recommendation set for each member. Items not preferred by any member are eliminated and a list of books is recommended to the group.

Jukola [O'Hara et al., 2004] and *PartyVote* [Sprague et al., 2008] are two systems able to provide music to an established social group of people attending a party/social event.

The type of group and the context in which the systems are used, make these systems work without any user profiles. In fact, in order to select the music to play, each user is allowed to express preferences (like the selection of a song, album, artist or genre) in a digital musical collection. The rest of the group votes for the available selections and a weight/percentage is associated to each song (i.e., the probability for the song to be played). The song with the highest vote is selected to be played.

The system proposed in [Recio-García et al., 2009] aims to produce personality aware group recommendations, i.e., recommendations that consider the personality of its members ("group personality composition") and how conflicts affect the recommendation process.

To measure the behaviors of people in conflicts, each user completes a test and a profile is built computing a measure called *Conflict Mode Weight (CMW)*. Recommendations are calculated using three classic recommendation algorithms, integrated with the CMWs of the group members.

Group recommender systems for documents *I-SPY* [Smyth et al., 2005, Smyth and Balfe, 2006, Smyth et al., 2003a, Smyth et al., 2003c, Briggs and Smyth, 2005, Freyne and Smyth, 2006, Coyle and Smyth, 2005] is a search engine that personalizes the results of a web search, using the preferences of a community of like-minded users.

When a user expresses interest in a search result by clicking on it, I-SPY populates a *hit matrix* that contains relations between the query and the results pages (each community populates its own matrix). Relations in the hit matrix are used to re-rank the search results to improve search accuracy.

Glue [Carta et al., 2008] is a collaborative retrieval algorithm that monitors the activity of a community of users in a search engine, in order to exploit implicit feedbacks.

A feedback is collected each time a user finds a relevant resource during a search in the system. The algorithm uses the feedback to dynamically strengthen associations between the resource indicated by the user and the keywords used in the search string. Retrieval is based on feedbacks, so it is not just dependent on the resource's content, making it possible for the system to retrieve even non-textual resources and update its performances dynamically (i.e., the community of users decides the keywords that describe a resource).

CAPS (Context Aware Proxy based System) [Sharon et al., 2003] is an agent that recommends pages and annotates links, based on their popularity among a user's colleagues and the user's profile. The system focuses on two aspects: page enhancement, with symbols that indicate its popularity, and search queries augmentation, with the addition of relevant links for a query. Since the system was designed to enhance the search activity of a user considering the experience of a user's colleagues, a CF approach and a zero-input interface (able to gather implicit information) were used.

The approach proposed in [Baskin and Krishnamurthi, 2009] was developed to help a group of conference committees selecting the most suitable items in a large set of candidates.

The approach is based on the *relative preference* of each reviewer, i.e., a rank of the preferred items, with no numeric score given to express the preferences. All the preferences ordering of the reviewers are aggregated through a variable neighborhood search algorithm improved by the authors for the recommendation purpose.

2.2.2 Systems that consider occasional groups with a particular aim

There are lots of contexts in which a group of people is not established but might be interested in getting together for a common aim. This is for example the case of people traveling together: they might not know each other, but they share interest for a common place. In such cases, a group recommender system could be useful, since it would be able to put together the preferences of an heterogeneous group, in order to achieve the common aim. As mentioned in Table **??**, group recommender systems that work for occasional groups were developed for the following domains:

- movies;
- tourist destinations;
- TV programs;

Group recommender systems for TV programs consider occasional groups that get together for a specific aim (watch TV together) and randomly share an environment (approaches for random groups are described next). Since the approaches focus on the group's aim, this category of systems was placed in this subsection.

Group recommendation for movies *PolyLens* [O'Connor et al., 2001] is a system built to produce recommendations for groups of users who want to see a movie.

To produce recommendations for each user of the group a CF algorithm is used. In order to model the group, a "least misery" strategy is used: the rating used to recommended a movie to a group is the lowest predicted rating for that movie, to ensure that every member is satisfied.

The system proposed in [Chen et al., 2008] considers interactions among group members, assuming that in a group recommender system ratings are not given just by individuals, but also by subgroups. If a group G is composed of members u_1, u_2 and u_3, ratings might be given by both individuals and subgroups (e.g., $\{u_1, u_2\}$ and $\{u_1, u_3\}$).

The system learns the ratings of a group using a Genetic Algorithm (GA), that uses the ratings of both individuals and subgroups to learn how users interact. For example, if an item is rated by users u_1 and u_2 as 1 and 5 but as a whole they rate the item as 4, it is possible to derive that u_2 plays a more influential role in the group.

The group recommendation methodology used by the system combines an item-based CF algorithm and the GA, to improve the quality of the system.

In [Amer-Yahia et al., 2009] an approach to compute group recommendation that introduces *disagreement* between group members as an important aspect to efficiently compute group recommendations is presented. The authors introduce a *consensus function*, which combines *relevance* of the items for a user and *disagreement* between members. After the *consensus function* is built, an algorithm to compute group recommendation (based on the class of Threshold algorithms) is proposed.

The system proposed in [de Campos et al., 2007, de Campos et al., 2009] presents a group recommendation approach based on Bayesian Networks (BN). The system was developed to help a group of people making decisions that involve the whole group (like seeing a movie) or in situations where individuals must make decisions for the group (like buying a company gift). The system was empirically tested in the movie recommendation domain.

To represent users and their preferences a BN is built. The authors assume that the composition of the groups is a priori known and model the group as a new node in the network that has the group members as parents. A collaborative recommender system is used to predict the votes of the group members. A posteriori probabilities are calculated to combine the predicted votes and build the group recommendation.

Group recommendation for tourist destinations In [McCarthy et al., 2007, McCarthy et al., 2006a, McCarthy et al., 2006b, Mccarthy et al., 2006, McCarthy et al., 2006] a group recommender system called *CATS (Collaborative Advisory Travel System)* is presented. The aim of the system is to help a group of friends plan and arrange ski holidays.

To achieve the objective, users are positioned around a device called "DiamondTouch table-top" [Dietz and Leigh, 2001] and the interactions between them (since they physically share the device) help the development of the recommendations.

To produce the recommendations, the system collects *critiques*, which are feedbacks left by users while browsing the recommended destinations (e.g., a user might specify that he/she is looking for a cheaper hotel, by *critiquing* the price feature).

Interactions with the DiamondTouch device are used to build an individual personal model (IM) and a group user model (GUM). Individual recommendations are built using both the IM and the GUM to maximize satisfaction of the group, whereas group recommendations are based on the critiques contained in the GUM.

INTRIGUE (INteractive TouRist Information GUidE) [Ardissono et al., 2003, Ardissono et al., 2005] is a system that recommends sightseeing destinations using the preferences of the group members.

Heterogeneity of a group is considered in several ways. Each group is subdivided into homogeneous subgroups of similar members that fit a stereotype (e.g., children). Recommendations are predicted for each subgroup and an overall preference is built considering some subgroups more influential (e.g., disabled people).

Travel Decision Forum [Jameson et al., 2003, Jameson, 2004, Jameson et al., 2004] is a system that helps groups of people plan a vacation. Since the system aims to find an agreement between the members of a group, asynchronous communication is possible and, through a web interface, a member can view (and also copy) other members' preferences. Recommendations are made using a simple aggregation (the *median*) of the individual preferences.

In [Lorenzi et al., 2008] a multiagent system in which agents work on behalf of a group of customers, in order to produce group recommendations, is presented. A formalism, named DCOP (Distributed Constraint Optimization Problem), is proposed to find the best recommendation considering the preferences of the users.

The system works with two types of agents: a user agent (UA), who works on behalf of a user and knows his preferences, and a recommender agent (RA), who works on behalf of suppliers of travel services. An optimization function is proposed to handle the agents' interactions and find the best recommendation.

e-Tourism [Garcia et al., 2009] is a system that plans tourist tours for groups of people. The system considers different aspects, like a group tastes, its demographic classification and places previously visited. A taxonomy-driven recommendation tool called GRSK (*Generalist Recommender System Kernel*), provides individual recommendations using three techniques: demographic, content-based and preference-based filtering. For each technique group preferences are computed using aggregation, intersection and incremental intersection methods and a list of recommended items is filtered.

Pocket RestaurantFinder [McCarthy, 2002] is a system that suggests restaurants to groups of people who want to dine together. The system was designed for contexts like conferences, where an occasional group of attendees decides upon a restaurant to visit.

Each user fills a profile with preferences about restaurants, like the price range or the type of cuisine they like (or don't like). Once the group composition is known, the system estimates a user's individual preference for each restaurant and averages those values to build a group preference and produce a list of recommendations.

Group recommendation for TV programs *FIT (Family Interactive TV System)* [Goren-Bar and Glinansky, 2004] is a recommender system that aims to filter TV programs considering the preferences of the viewers.

The only input required by the system is a stereotype user representation (i.e., a class of viewers that would suit the user, like *women*, *businessmen*, *students*, etc.), along with the user preferred watching time. The system automatically updates a profile, by collecting implicit feedbacks from the watching habits of the user.

When someone starts watching TV, the system looks at the probability of each family member to watch TV in that time slot and predicts who there might be watching TV. Programs are recommended through an algorithm that combines such probabilities and users' preferences.

The system proposed in [Vildjiounaite et al., 2009] recommends TV programs to a family.

To protect the privacy of each user and avoid the sharing of information, the system observes the habits of a user and adds contextual information about what is monitored. By observing indicators like the amount of time a TV program has been watched, a user's preferences are exploited and a profile is built.

To estimate the interests of the users in different aspects, the system trains on each family history three Support Vector Machine (SVM) models for program name, genre and viewing history. After the models are trained, recommendation is performed with a Case-Based Reasoning (CBR) technique.

TV4M [Yu et al., 2006] is a TV programs recommender system for multiple viewers.

To identify who is watching TV, the system provides a login feature. To build a group profile that satisfies most of its members, all the current viewers' profiles are merged, by doing a total distance minimization of the features available (e.g., genre, actor, etc.). According to the built profile, programs are recommended to the group.

2.2.3 Systems that consider random groups who share an environment

A random group is formed by people who share an environment without a specific purpose. Its nature is *heterogeneous* and its members might not share interests.

Group recommender systems that work with random groups calculate the list of predicted items frequently, as people might join or leave the environment at any moment. This section will describe group recommender systems that work with random groups. Two main recommendation domains are related to this type of systems:

- multimedia items (e.g., music) broadcast in a shared environment;
- information items (e.g., news or web pages).

Group recommendation for broadcast multimedia items *Adaptive Radio* [Chao et al., 2005] is a system that broadcasts songs to a group of people who share an environment. The approach tries to improve satisfaction of the users by focusing on *negative preferences*, i.e., it keeps track of which songs a user does not like and avoids playing them. Moreover, the songs similar to the ones rejected by a user are reject too (the system considers two songs similar if they belong to the same album). The highest rated between the remaining songs is automatically played.

In-Vehicle Multimedia Recommender [Zhiwen et al., 2005] is a system that aims to select multimedia items for a group of people traveling together.

The system aggregates the profiles of the passengers and merges them using a notion of *distance* between the profiles. Once the profiles are merged, a content-based recommender system is used to compare multimedia items and group preferences.

Flytrap [Crossen et al., 2002] is a group recommender system that selects music to be played in a public room. Since people in a room (i.e., the group members) change frequently, the system was designed to predict the song to play considering the preferences of the users present in the room at the moment of the song selection.

A 'virtual DJ' agent is used to automatically decide the song to play. To build a model of the preferences of each user the agent analyzes the MP3 files played by a user in his/her computer and considers the information available about the music (like similar genres, artists, etc.). The song is selected through a voting system in which an agent represents each user in the room and rates the candidate tracks.

MusicFX [McCarthy and Anagnost, 2000] is a system that recommends music to members of a fitness center, etting them influence (but not control) the music selected.

Since the group structure (i.e., the people in the room) varies continuously, the system gives the users working out in the fitness center the possibility to login. To let users express their preferences about a particular genre, the system has a database of music genres. The music to play is selected considering the preferences of each user in a summation formula.

Group recommendation for information items *Let's Browse* [Lieberman et al., 1999] is a system that recommends pages to people browsing the web together. Since the group is random (a user might join or leave the group at any time), the system uses an electronic badge to detect the presence of a user.

The system builds a user profile analyzing the words present in his/her homepage. The group is modeled by a linear combination of the individual profiles and the system analyzes the words that occur in the pages browsed by the group.

The system recommends pages that contain keywords present in the user profile. Such keywords are listed in the recommended page.

GAIN (Group Adapted Interaction for News) [Pizzutilo et al., 2005b, Carolis and Pizzutilo, 2009] is a system that selects background information to display in a public shared environment.

The authors assumed that the group of users may be totally unknown, partially or completely known. The group is modeled by splitting it in two subgroups: the *known subgroup* (i.e., people that are certainly near the display for a period of time) and the *unknown subgroup* (i.e., people not recognized by the system). Recommendations are predicted using a statistical dataset built from the group modeling.

2.2.4 Group recommendation with automatic group detection

As shown in Table **??**, aside from the contribution developed for this thesis, there is just a group recommender system that automatically detects groups of users. Such an approach is interesting for various reasons: (I) people change their mind frequently, so a user membership in a group might not be long-term, or (II) technological constraints might allow the system to handle only a certain number of groups (or a maximum number of members per group).

Group recommendation with Communities of Interest detection

The approach proposed in [Cantador et al., 2008] aims to automatically discover Communities of Interest (CoI) (i.e., a group of individuals who share and exchange ideas about a given interest) and produce recommendations for them.

CoI are identified exploiting the preferences expressed by users in personal ontology-based profiles. Each profile measures the interest of a user in concepts of the ontology. The interest expressed by users is used to cluster the concepts.

User profiles are then split into subsets of interests, to link the preferences of each user with a specific cluster of concepts. Hence it is possible to define relations among users at different levels, obtaining a multilayered interest network that allows to find multiple CoI. Recommendations are built using a content-based CF approach.

Chapter 3

Algorithms

3.1 Overview

This chapter presents *ART (Automatic Recommendation Technologies)*, i.e., a set of group recommendation algorithms, able to produce suggestions respecting a constraint on the number of recommendations that can be generated. In order to consider the preferences of each user and respect the constraint, groups of users with similar preferences have to be detected. Individual preferences should then be combined, in order to derive a group model that allows to predict group preferences.

ART identifies five algorithms, that implement in different ways all the different approaches to generate recommendations for a group.

The first part of the chapter describes the methods used by each task performed by a group recommendation algorithm developed for this thesis. In particular, Section 3.2 presents an overview of clustering algorithms, Section 3.8 presents the group modeling strategies implemented for this work and Section 3.4 presents an overview of the families of approaches to generate group predictions.

Then, the algorithms that compose *ART* are described in detail. Section 3.5 describes the algorithms based on the construction of group models, Section 3.6 describes the algorithm that merges individual recommendations and Section 3.7 describes the algorithms based on the aggregation of individual preferences.

3.2 Clustering

Clustering, also known as data classification, unsupervised learning or unsupervised classification [Kleinberg and Tardos, 2002], is the partitioning of unlabeled data into groups (named *clusters*), such that objects in a cluster are very similar and objects that belong to different clusters are highly dissimilar. That means that data is classified into two or more classes, without a priori knowledge of its structure and based on a distance function that allows to capture how similar objects are.

Note that in this section only the aspects of clustering related to the studies conducted for this PhD thesis will be considered.

In particular two important classes of clustering algorithms are considered.

Clustering in a metric space. A set of data and a distance function that satisfies the property of a metric are determined, in order to measure the similarity between objects and divide data into homogeneous groups.

Graph clustering. The objective is to divide the points of a graph into clusters, considering the edges between the points. The partitioning is such that there are many edges within each cluster and a few between the clusters.

Reader should refer to [Xu and II, 2005] for a survey on algorithms that cluster on a metric space and to [Schaeffer, 2007] for a deep analysis of graph clustering algorithms.

3.2.1 Clustering in a metric space

A metric space (X, ρ) consists of a set of data X and a distance function $\rho : X \times X \to \mathbb{R}$ that satisfies the three properties of a metric:

1. Reflexivity: $\rho(x, y) \geq 0$, with $\rho(x, y) = 0$ iff $x = y$

2. Symmetry: $\rho(x, y) = \rho(y, x)$

3. Triangle inequality: $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$

One of the most used and popular algorithms that cluster in a metric space is *k-means* [MacQueen, 1967]. A brief description of the algorithm is now presented.

The k-means clustering algorithm

The k-means clustering algorithm partitions a set of data points into cluster. Let $X = \{x_1, x_2, ..., x_n\}$ be a set of points in $\mathbb{R}^d$. Using a set of k centers $c_1, c_2, ...c_k$ in $\mathbb{R}^d$, the algorith works as follows.

1. For each $i \in 1, ..., k$, let C_i be the set of points in X closer to c_i than they are to any other center.

2. For each $i \in 1, ..., k$, set c_i as the new center for C_i, calculated as follows: $c_i = \frac{1}{|c_i|} \sum_{x_j \in C_i} x_j$.

3. Repeat steps 1 and 2 until neither c_i nor C_i change. If that happens, return the clusters C_i.

3.2.2 Clustering a graph

Graph clustering algorithms inspect the structure of a graph, in order to find a group of points in which the number of links inside the group is much higher than the links between the groups. A cluster in a graph is called *community* and graph clustering is also known as *Community Detection*

[Fortunato, 2009]. Next, two widely used graph clustering algorithms are presented.

Louvain method

Louvain algorithm [Blondel et al., 2008] was developed to be very fast and be capable to cluster very large networks in linear time (the analysis of a network of 2 million nodes takes two minutes on a PC). The method generates a hierarchical structure of communities. It is one of the most widely used method for detecting communities in large networks.

The algorithm is based on the optimization of a function called *modularity* [Newman and Girvan, 2004], defined as the number of edges that are within groups minus the expected number in an equivalent random network.

The algorithm is divided into two steps, iteratively repeated. At first each node of the graph is assigned to a different community (i.e., there's a community for each node). For each node i, each neighbor j is considered and i is moved to the community j that allows to have a positive and maximum gain of modularity (if that is not possible, i remains in its community). This step is repeat until an improvement in modularity is possible.

The next step considers each detected community as a new node in the network. Nodes are linked with a weight equal to the sum of the edges of the nodes of the two communities.

MCL

Markov Clustering (MCL) [van Dongen, 2000] is an algorithm based on a bootstrapping procedure applied to a stochastic matrix (also known as Markov matrix), derived from the adjacency matrix of the graph. The approach is based on the intuition that if nodes belong to the same cluster, the longest path between them is relatively short. On the contrary, for nodes that belong to different clusters, its value is relatively high. That means that it should be difficult to move from one cluster to another with a random walk.

To explain how a random walk on a weighted graph works, suppose that a random walker is, at a certain instant, in a node i. Node j, where he/she will be at the following instant, is chosen among the first neighbors of i, with a probability proportional to the weight of the edge between i and j. In such a way it is possible to create a transition matrix M of size $N \times N$, in which each element is as follows:

$$M_{ij} = \frac{a_{ij}}{\sum_m a_{im}} \tag{3.1}$$

It can be trivially verified that M is a stochastic matrix. A matrix is *stocastic* if the following requirements are satisfied:

- it is squared;
- all its elements belong to interval [0, 1];
- the sum of all the elements of each column is equal to 1.

As previously mentioned, the algorithm is based on a bootstrapping procedure, i.e., the probability of random walks in the graph is calculated iteratively using two sets of operators, named operators of *expansion* and *inflation* and applied to the stochastic matrices.

The expansion operator computes the square of the matrix (the product of that matrix with itself). Inflation operator is the entry-wise Hadamard-Schur product of the matrix combined with a diagonal scaling, to allow the resulting matrix to be a stochastic matrix.

Formally, let $M \in R^{k \times k}$ be a stochastic matrix and $r > 0$ a real number. Inflation operator $\Gamma_r : R^{k \times k} \rightarrow R^{k \times k}$ acts on M as follows:

$$(\Gamma_r M)_{pq} = \frac{(M_{pq})^r}{\sum_{i=1}^{k} M_{iq}^r} \tag{3.2}$$

The algorithm consists on the subsequent application of the inflation operator and the expansion operator and converges quadratically in the neighborhood of doubly idempotent stochastic matrices, i.e., matrices that do not change under the action of the two operators. The obtained matrix returns a disconnected graph, in which each component contains nodes that belong to the same cluster. The inflation operator **??** depends from a parameter r, known as *granularity*. By incrementing this operator, the strength of the inflation operator and causes a higher number of clusters.

3.3 Group modeling

In order to manage the information related to a group and provide recommendations, it is necessary to first *model* the group. A group is composed of individuals that get together for a particular aim. *Group modeling* is the process used to combine multiple user models into a group model. So, the first aspect to consider when modeling a group is an individual user model, made of the user's interest for a set of items. A group model can be considered as a "synthesis" of the user models, built by combining the preferences of the group members.

In group recommendation, building a group model is strongly related to the idea of collective choice, i.e., making a choice for a group taking into account the opinions of the users that belong to it. The aggregation of individual preferences is made using a particular strategy and, as stated in the Introduction and highlighted in [Pizzutilo et al., 2005a], the usefulness of a strategy has be evaluated in the environment in which the modeling is done and no strategy can be used in every context. In fact there are different aspects that have to be evaluated when modeling a group, like its size and the members that belong to it, in order to combine the individual ratings properly. For example, there are strategies that work best with small groups or strategies that consider some categories of users as more important than others (like children or disabled people).

In the domain of application presented in this thesis, the choice of the right group modeling strategy is particularly important, since a group modeling strategy should be able to level the preferences of groups of possibly very different sizes (given a set of users, the size of each group is related to both the number of groups and their preferences). Moreover, the users in a group do not interact with each other.

This section presents the group modeling strategies evaluated in the algorithms developed for this PhD thesis. We considered all the strategies presented in [Masthoff, 2004] and implemented all the ones that could be applied to our domain. In fact there are some voting strategies that do not produce an explicit rating, but just a ranked list of the items evaluated by the group (i.e., the *Plurality Voting*, *Copeland Rule* and *Fairness* strategies), and there is a strategy (*Most Respected Person*) where just the ratings of the most respected person are considered (the idea of a "most respected person" is not meaningful in a context where a group of people is automatically detected).

Even though the *Multiplicative Utilitarian* strategy, that produces a group preference by multiplying all the ratings for an item, was implemented, it could not be tested. This is because of the limit on the maximum number that can be calculated by a computer[1]. Therefore, the strategy is not even presented in this section.

After the description of each strategy, an example of how individual ratings are combined is presented. In the examples, three users (u_1, u_2 and u_3) rate ten items with a rating from 1 to 10.

3.3.1 Additive Utilitarian Strategy

Individual ratings for each item are summed and a list of the group ratings is produced. The ranked group list of items is exactly the same that would be produced when averaging the individual ratings, so this strategy is also called 'Average strategy'.

	A	B	C	D	E	F	G	H	I	J
u_1	8	10	7	10	9	8	10	6	3	6
u_2	7	10	6	9	8	10	9	4	4	7
u_3	5	1	8	6	9	10	3	5	7	10
Group	20	31	21	25	26	28	22	15	14	23

3.3.2 Borda Count

Each item gets a number of points, according to the position in the list of each user. The least favorite item gets 0 points and a point is added

[1] A 64 bits machine cannot calculate numbers higher than 2^{52}. That would mean that even if the strategy was tested with a very small dataset of 55 users and they all gave a very small rating for an item, like 2, an overflow would occur.

each time the next item in the list is considered. If a user gave the same rating to more items, points are distributed. So, for example, items H and I were rated by user u_2 with the lowest rating and should "share" the lowest positions with 0 and 1 points, so both the items get (0+1)/2=0.5 points. A group preference is obtained by adding the individual points of an item.

	A	B	C	D	E	F	G	H	I	J
u_1	4.5	8	3	8	6	4.5	8	1.5	0	1.5
u_2	3.5	7.5	2	6.5	5	7.5	6.5	0.5	0.5	3.5
u_3	2.5	0	5	3	6	7.5	1	2.5	4	7.5
Group	10.5	15.5	10	17	17	19.5	15.5	4.5	4.5	12.5

3.3.3 Approval Voting

Each user can vote for as many items as the want. To show how the strategy works, we are going to suppose that each user votes for all the items with a rating above a certain treshold (let's say 5). A group preference is obtained by adding the individual points of an item.

	A	B	C	D	E	F	G	H	I	J
u_1	1	1	1	1	1	1	1	1		1
u_2	1	1	1	1	1	1	1			1
u_3			1	1	1	1			1	1
Group	2	2	3	3	3	3	2	1	1	3

3.3.4 Least Misery Strategy

The rating assigned to an item for a group is the lowest rating expressed for that item by a member of the group. This strategy is usually used to model small groups, to make sure that every member is satisfied. A drawback of this strategy is that if the majority of the group really likes something, but one person doesn't, the item will not to be recommended to the group. This is what happens in the example for items B and G.

	A	B	C	D	E	F	G	H	I	J
u_1	8	10	7	10	9	8	10	6	3	6
u_2	7	10	6	9	8	10	9	4	4	7
u_3	5	1	8	6	9	10	3	5	7	10
Group	5	1	6	6	8	8	3	4	3	6

3.3.5 Most Pleasure Strategy

The rating assigned to an item for a group is the highest rating expressed for that item by a member of the group.

	A	B	C	D	E	F	G	H	I	J
u_1	8	10	7	10	9	8	10	6	3	6
u_2	7	10	6	9	8	10	9	4	4	7
u_3	5	1	8	6	9	10	3	5	7	10
Group	8	10	8	10	9	10	10	6	7	10

3.3.6 Average Without Misery Strategy

The rating assigned to an item for a group is the average of the ratings assigned by each user for that item. All the items that were evaluated with a rating under a certain threshold are not considered (in the example the threshold rating is 4).

	A	B	C	D	E	F	G	H	I	J
u_1	8	10	7	10	9	8	10	6	3	6
u_2	7	10	6	9	8	10	9	4	4	7
u_3	5	1	8	6	9	10	3	5	7	10
Group	20	-	21	25	26	28	-	15	-	23

3.4 Families of approaches for the group recommendation process

Given a set of individual preferences, group preferences can be generated using one of three families of approaches [Jameson and Smyth, 2007a]: (a) generation of a group model that combines individual preferences, (b) merging of recommendations built for individual users, or (c) aggregation of individual preferences.

3.4.1 Construction of Group Preference Models

This approach builds a group model using the preferences expressed by each user, then predicts a rating for the items not rated by the group using that model.

Description of the approach

1. Construct a model M_g for a group g, that represents the preferences of the whole group.
2. For each item i not rated by the group, use M_g to predict a rating p_{gi}.

3.4.2 Merging of Recommendations Made for Individuals

The approach presents to a group a set of items, that is the merging of the items preferred by each member of the group.

Description of the approach

1. For each member of the group u:
    - For each item i not rated by the user, predict a rating p_{ui}.
    - Select the set C_i of items with the highest predicted ratings p_{ui} for u_i
2. For each group produce $\bigcup_i C_i$, the union of the sets of items with the highest predicted rating of each member.

3.4.3 Aggregation of Individual Preferences

The approach first predicts individual preferences for all the items not rated by each user, then aggregates individual preferences for an item to derive a group preference.

Description of the approach

1. For each item i:

    - For each member u of the group g that did not rate i, predict a rating p_{ui}.
    - Calculate an aggregate rating r_{gi} from the set $\{r_{ui}\}$.

3.5 Algorithms Based on Group Models Construction

3.5.1 MART (Model-based Automatic Recommendation Technology)

MART (Model-based Automatic Recommendation Technology) is an algorithm that detects groups of similar users, models each group using the preferences of its members and predicts group preferences, according to the approach presented in 3.4.1.

The algorithm works in four steps:

1. In order to create groups of users, the algorithm takes as input the ratings expressed by each user and evaluates through a standard metric (i.e., cosine similarity) how similar the preferences of two users are. The result is a weighted graph where nodes represent users and each weighted edge represents the similarity value of the users it connects. A post-processing technique is then introduced to remove noise from the network and reduce its complexity.

2. To identify intrinsic communities of users, a Community Detection algorithm proposed by [Blondel et al., 2008] is applied to the graph that contains the similarities between users and partitions of different granularities are generated.

3. Once groups have been detected, a group model is built for each group g, using one of the modeling strategies presented in 3.8.

4. A rating is predicted for each item not rated by a group, using the model that contains its preferences.

Each step will now be described in detail.

Users' Similarities Graph

A graph that describes the connections between users in terms of similarity can be built considering the individual preferences expressed by each user for the items.

Similarity between two users can be measured by calculating the *cosine similarity* between them. The metric compares the ratings of all the items rated by both the two considered users (*corated* items). Cosine similarity between a user u and a user v is given in Equation 3.3. $CR_{u,v}$ is the set of corated items between u and v.

$$userSim(u,v) = \frac{\sum_{i \subset CR_{u,v}} r_{ui} \times r_{vi}}{\sqrt{\sum_{i \subset CR_{u,v}} (r_{ui})^2} \times \sqrt{\sum_{i \subset CR_{u,v}} (r_{vi})^2}} \tag{3.3}$$

The resulting graph (*users' similarities graph*) links each couple of associated users with a weighted edge.

As highlighted by [Gfeller et al., 2005], in graph like this, edges have intrinsic weights and no information is given about the real associations between the nodes. Edges are usually affected by noise, which leads to ambiguities in the detection of the groups. Moreover, the weights of the edges in the graph are calculated considering the ratings and it is well known that people have different rating tendencies, i.e., some users tend to express their opinion using just the end of the scales, expressing if they loved or hated an item. In order to eliminate noise from the graph and reduce its complexity by removing weak edges, a parameter called *noise* was set in the algorithm. The parameter indicates the weight that is subtracted by every edge of the graph.

Groups Detection

This step of the algorithm has the goal of finding groups of users with similar preferences, accepting as input the weighted users' similarities graph that was built in the previous step. In 2004 a new optimization function has been introduced, the modularity, that measures for a generic partition of the set of nodes in the network, the number of internal (in each partition) edges respect to the random case. The optimization of this function gives, without a previous assessment of the number and size of the partitions [Fortunato and Castellano, 2007], the natural community structure of the network. Moreover it is not necessary to embed the network in a metric space like in the k-means algorithm. A notion of distance or link weight can be introduced but in a pure topological fashion [Newman, 2004].

Recently a very efficient algorithm has been proposed, based on the optimization of the weighted modularity, that is able to easily handle networks with millions of nodes, generating also a dendrogram; a community structure at various network resolutions [Blondel et al., 2008]. Since the algorithm had all the characteristics we were looking for, it was chosen to create the groups of users used by our group recommendation algorithm.

Group Modeling

To create a model that represents the preferences of a group g, the strategies previously described are taken into account. As can be noticed in the examples, the group ratings produced by each strategy are in completely different scales of representation (in the examples individual preferences are expressed with a rating between 1 and 10, while individual ratings can be much higher than 10). In order to evaluate how each group rating reflects the individual preferences, it is necessary that both individual and group ratings are in the same domain of ratings. This can be obtained with a simple reduction:

$$group\_rating : max\_group\_rating = new\_group\_rating : max\_rating \tag{3.4}$$

where:

group_rating is the rating produced by a modeling strategy;

max_group_rating is the maximum rating that a user can express for an item;

max_rating is the maximum value of *group_rating* that can be obtained for an item.

So a *new_group_rating* can be obtained calculating:

$$new\_group\_rating = \frac{group\_rating \cdot max\_rating}{max\_group\_rating} \tag{3.5}$$

The formula is not necessary for all the considered strategies or has to be adapted for some of them. Below there is a description of the particular cases.

- *Least Misery* and *Most Pleasure* strategies already produce a rating that belongs to the same domain of the original ratings, so the reduction is not necessar.

- *Additive Utilitarian* and *Average without Misery* strategies sum the individual ratings. The reduction previously presented can be also rewritten as:

$$new\_group\_rating = \frac{group\_rating \cdot max\_rating}{num\_ratings \cdot max\_rating} = \frac{group\_rating}{num\_ratings} \tag{3.6}$$

  which is the arithmetic mean of the ratings.

- Considering the ratings produced by the *Borda Count* and *Approval Voting* strategies it is clear that the value of *max_group_rating* has to be evaluated each time a rating is produced.

After a group has been modeled, in order to calculate meaningful ratings for a group g, an aggregate rating r_{gi} is a part of the model only if a consistent part of the group has rated item i. This is done by setting a parameter, named *coratings*, which expresses the minimum percentage of group members who have to rate an item, in order to include the rating in the model.

Prediction of the Missing Ratings Using a Group Model

In the models created by the previous step, for a subset of items there is no preference, because the item is rated by a small part of the group and cannot be considered representative of the preferences of the group as a whole. In order to estimate such preferences, a rating p_{gi} for an item i not rated by a group g is predicted through the model that contains a group's preferences. This is done by using an Item-Based Nearest Neighbor Collaborative Filtering Algorithm. The algorithm predicts a rating p_{gi} for each item i that was not evaluated by a group g, considering the rating r_{gj} of the most similar items rated by the group. Equation 3.7 gives the formula used to predict the ratings:

$$p_{gi} = \frac{\sum_{j \in topItems(g)} itemSim(i, j) \cdot r_{gj}}{\sum_{j \in topItems(g)} itemSim(i, j)} \tag{3.7}$$

Similarity *itemSim*() between two items is calculated using the cosine similarity. The metric is computed considering all users who rated both item i and item j. Equation 3.8 gives the formula for the similarity (note that $RB_{i,j}$ is the set of users that rated both item i and j).

$$itemSim(i,j) = \frac{\sum_{u \subset RB_{i,j}} r_{ui} \times r_{uj}}{\sqrt{\sum_{u \subset RB_{i,j}} (r_{ui})^2} \times \sqrt{\sum_{u \subset RB_{i,j}} (r_{uj})^2}} \tag{3.8}$$

In order to compute the similarity between items, the original ratings given by the individual users are considered (i.e., the metric is not computed considering the aggregate group preferences).

The *topItems* list is a selection of the most similar items to the one for which the algorithm predicts the rating. A parameter, called *top*, indicates how many similarities the algorithm considers to predict the ratings.

An example of how the *top* similar items are selected is shown in Figure 3.1. The algorithm needs to predict a rating for Item 1. The most similar items are shown in the list. For each similar item j, the table indicates the similarity with Item 1 (column t_{1j}) and the rating expressed by the group (column r_j). In the example, the *top* parameter is set to 3 and items with similarity 0.95, 0.88 and 0.71 are selected.

The choice of using an Item-based Collaborative Filtering approach is because the algorithm deals with group models. Since groups might be very large, a group model might put together a lot preferences and it would not be significant to make a prediction with a User-based approach, that would look for "similar groups" [2].

[2] Think of an example with 6000 users and 10 groups. If groups were homogeneous,

Item j	t_{1j}	r_j
Item 2	0.95	3.5
Item 3	0.95	4.2
Item 4	0.88	2.8
Item 5	0.71	2.6
Item 6	0.71	3.9
Item 7	0.71	4.3
Item 8	0.63	1.2
Item 9	0.55	3.2

Figure 3.1: Top similar items of an unrated item

To make meaningful predictions, it would be useful to evaluate how ”reliable” the calculated predictions are. This is done by calculating the mean of the *top* similarities and by setting a *trust* parameter. The parameter indicates the minimum value the mean of the similarities has to get, in order to be considered reliable and consider the predicted rating. The mean of the similarities in the previous example is 0.85 so, to consider the predicted rating, the *trust* parameter has to be lower than 0.85.

there would be around 600 users per group. If a User-based approach was used, when looking for neighbors, the algorithm would look for a two similar models, that represent 600 users each.

3.5.2 SMART (State-of-the-art Model-based Automatic Recommendation Technology)

The previously proposed algorithm, *MART* presents several aspects that can be improved, listed below.

- It was recently highlighted [Amatriain et al., 2011] that in literature the k-means clustering algorithm [MacQueen, 1967] is by far the most used clustering algorithm in recommender systems, producing improvements on various aspects. Moreover, according to the authors, alternative to the k-means algorithm are rarely used in the recommendation research.

- If a graph clustering algorithm does not allow to set a fixed number of groups and generates a hierarchical structure that contains the natural partitioning of the users, like the one previously used, it might be impossible to repeat the experiments multiple times under the same conditions, in order to do a k-fold cross validation and conduct statistical tests to validate the results.

- When calculating similarities between items (like the *MART* algorithm does when predicting group ratings), Adjusted-cosine similarity is the most popular measure and believed to be the most accurate [Schafer et al., 2007].

- In literature [Schafer et al., 2007], the items used for the prediction in an Item-based Nearest Neighbors algorithm are all the items rated by a user. Therefor, there is no need to use a parameter to select the most similar items.

In order to overcome this limitation, an updated version of *MART*, called *SMART (State-of-the-art Model-based Automatic Recommendation Technology)*, was developed. As the name says, the algorithm uses the state-of-the-art approaches previously mentioned to predict preferences using a group model.

The algorithm works in three steps:

1. Using a set of individual preferences, groups of users with similar preferences are detected through the k-means clustering algorithm [MacQueen, 1967].

2. Once groups have been detected, a group model is built for each group g, using one of the modeling strategies presented in 3.8.

3. A rating is predicted for each item not rated by a group, using the model that contains its preferences.

Ratings expressed by the users for the evaluated items will be used by the k-means clustering algorithm [MacQueen, 1967] to detect groups of users with similar preferences. Groups will be modeled in the same way *MART* does, therefore this step will not be described. The step that predicts group ratings using the model is described next.

Prediction of the Missing Ratings Using a Group Model

A rating p_{gi} is predicted with an Item-Based Nearest Neighbor Collaborative Filtering Algorithm presented in [Schafer et al., 2007]. The algorithm predicts a rating p_{gi} for each item i that was not evaluated by a group g, considering the rating r_{gj} of each similar item j rated by the group. Equation 3.9 gives the formula used to predict the ratings:

$$p_{gi} = \frac{\sum_{j \in ratedItems(g)} itemSim(i, j) \cdot r_{gj}}{\sum_{j \in ratedItems(g)} itemSim(i, j)} \tag{3.9}$$

According to [Schafer et al., 2007], some authors do not consider all the items rated by a group in the model, but just the top n, correlations. This is the approach used also for this algorithm.

As previously mentioned, in order to compute similarity between items, adjusted-cosine similarity will be used. The metric is computed considering all users who rated both item i and item j. Equation 3.10 gives the formula for the similarity (note that $RB_{i,j}$ is the set of users that rated both item i and j).

$$itemSim(i, j) = \frac{\sum_{u \subset RB_{i,j}} (r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_{u \subset RB_{i,j}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \subset RB_{ui,j}} (r_{uj} - \bar{r}_u)^2}} \tag{3.10}$$

3.6 Algorithm Based on Merging Individual Recommendations

3.6.1 APART (Aggregated Preferences-based Adaptive Recommendation Technology)

The algorithm named *APART (Aggregated Preferences-based Adaptive Recommendation Technology)*, detects groups of similar users, predicts individual preferences and selects the items with the highest predicted ratings for each user, using the approach presented in 3.4.2.

The algorithm works in three steps:

1. Using individual preferences, groups of similar users are detected through the k-means clustering algorithm.

2. Individual predictions are calculated for each user with a User-Based Collaborative Filtering Approach.

3. A list of items that contains the set of items with the highest predicted rating for each user is produced.

Detection of the Groups

The first step uses the same approach previously presented for the *SMART* algorithm, i.e. the k-means clustering algorithm.

Prediction of the Missing Ratings

Ratings for a group's members can be predicted using a classic User-Based Nearest Neighbor Collaborative Filtering Algorithm, presented in [Schafer et al., 2007]. The algorithm predicts a rating p_{ui} for each item i that was not evaluated by a user u, considering the rating r_{ni} of each similar user n for the item i. A user n similar to u is called a *neighbor* of u. Equation 3.11 gives the formula used to predict the ratings:

$$p_{ui} = \bar{r}_u + \frac{\sum_{n \subset neighbors(u)} userSim(u,n) \cdot (r_{ni} - \bar{r}_n)}{\sum_{n \subset neighbors(u)} userSim(u,n)} \tag{3.11}$$

Values $\bar{r}_u$ and $\bar{r}_n$ represent, respectively, the mean of the ratings expressed by user u and user n. Similarity *userSim*() between two users is calculated using the Pearson' correlation, a coefficient that compares the ratings of all the items rated by both the target user and the neighbor (*corated* items). Pearson' correlation between a user u and a neighbor n is given in Equation 3.12. $CR_{u,n}$ is the set of corated items between u and n.

$$userSim(u,n) = \frac{\sum_{i \subset CR_{u,n}} (r_{ui} - \bar{r}_u)(r_{ni} - \bar{r}_n)}{\sqrt{\sum_{i \subset CR_{u,n}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \subset CR_{u,n}} (r_{ni} - \bar{r}_n)^2}} \tag{3.12}$$

The values of the metric range from 1.0 (that indicates complete similarity) and -1.0 (that indicates complete dissimilarity). As highlighted in [Herlocker et al., 1999], negative correlations do not help increasing the prediction accuracy and can be discarded.

Generation of the Group Predictions

For each user, the items for which a rating is predicted are ranked in descending order based on the ratings, then the top-n items are selected. The union of the individual lists that contain the items preferred by each user is then produced. Note that if an item appears in the list of more members of the same group, the average of the of the predicted ratings for that item is calculated, in order to derive the preference of that group for the item.

3.7 Algorithms Based on Individual Preferences' Aggregation

3.7.1 BART (Baseline Adaptive Recommendation Technology)

BART (Baseline Adaptive Recommendation Technology), is an algorithm built to detect groups of similar users, predict individual preferences and aggregate the preferences expressed for each item into a group preference, according to the approach presented in 3.4.3.

The algorithm works in three steps:

1. Using a set of individual preferences, groups of users with similar preferences are detected through the k-means clustering algorithm.
2. Individual predictions are calculated for each user with a User-Based Collaborative Filtering Approach.
3. A group preference is computed by aggregating the preferences of the individual users.

The first two steps use the same algorithms previously presented, i.e., the k-means clustering algorithm and the User-Based Collaborative Filtering algorithm. Therefore only the details of the approach used to aggregate individual preferences will now be described.

Aggregation of the Individual Preferences

This step combines the preferences of each user that belongs to a group for an item.

The same modeling strategies used for *MART* and *SMART* can be exploited by the algorithm. The only difference is that this algorithm models individual predictions. Since a prediction is calculated for all the items not rated by a user, there is a rating for every item and every user of the group and there is no need to remove any group prediction from the model.

3.7.2 HEART (Highly Enhanced Adaptive Recommendation Technology)

BART detects groups of similar users using the preferences expressed by users for the evaluated items.

However, the number of items rated by users is much lower than the number of available items. This leads to the sparsity problem that is common in clustering.

HEART (Highly Enhanced Adaptive Recommendation Technology) was conceived to improve the quality of the clustering step of *BART*[3]. *HEART* detects groups giving as input to the k-means algorithm not the original ratings explicitly expressed by users, but also the predicted values of the unrated items for each user.

In order to do so, the individual predictions are predicted by *HEART* at the beginning of the computation. Using more values as input for the clustering, the algorithm should be able to identify better groups, i.e., groups composed by users having more correlated preferences. This should lead to a higher overall quality of the group recommendations.

In conclusion, *HEART* performs the same steps performed by *BART* but computes individual recommendations before clustering the users. This al-

[3] The adverb "highly" should be intended both as an intention to make great improvements with respect to the previous algorithm and as a synonym of "favourably", to intend that the enhancement suggests a good outcome.

lows to cluster the users using more preferences and identify better groups. The preferences expressed by users and the individual recommendations are also used to model the group.

3.8 Contribution

All the algorithms previously presented present several scientific contributions, listed below.

- None of the existing approaches in the group recommendation literature works with automatically detected groups and is able to adapt to constraints imposed by the systems.
- The algorithms explore the different ways to produce group recommendations in such a context, by using the different families of approaches to produce the recommendation presented in 3.4. This allows to discover the best way to build a recommendation for a group for automatically detected groups.
- Different classes of clustering algorithms and dinstance metrics are considered (*BART* and *SMART* build groups in completely different ways), in order to find the best way to group similar users.
- All the existing strategies to model a group and presented in will be deeply studied, in order to find the best way to model automatically detected groups and the properties that characterize them.

3.9 Conclusions

Chapter 4

Experimental Evaluation

This chapter presents the experiments conducted to evaluate the algorithms previously proposed. The first objective of this study is to find the best configuration for each algorithm, by properly setting its parameters. Then all the proposed algorithms will be compared, in order to find the best way to produce group recommendations for automatically detected groups. The last part of the chapter presents a study on the property of the groups that characterize the quality of the obtained results.

4.1 Experimental framework

This section presents the framework used to conduct the experiments. The dataset used and the preprocessing made on the data are first described.

Then the metrics used to make the evaluations is presented. After, the strategy and aims that drove our experiments are described.

4.1.1 Dataset

The dataset used to conduct the experiments is MovieLens-1M[1], which is composed of 1 million ratings, expressed by 6040 users for 3900 movies.

For this framework, only the file `ratings.dat` that contains the actual ratings given by users is considered (the other files available in the dataset contain features that describe the users and the movies). The file contains four features: *UserID*, that contains user IDs in a range between 1 and 6040, *MovieID*, that contains IDs of the movies in a range between 0 and 3592, *Rating*, that contains values in a scale between 1 and 5 and *Timestamp*, that contains a timestamp of the moment in which a user rated an item. Each user rated at least 20 movies.

4.1.2 Preprocessing

The file `ratings.dat` was preprocessed for the experimentation. Out of all the features available, just the first three features were selected (i.e., *UserID*, *MovieID* and *Rating*), since none of the presented algorithms uses a timestamp. The feature *UserID* was mapped in a new set of IDs between 0 and 6039, to facilitate the computation using data structures.

[1] http://www.grouplens.org/

In order to conduct statistical tests to validate the obtained results, experiments were repeated five times with a 5-fold cross-validation. In this approach, each rating available in the dataset is used four times for training and once for the testing. In order to do so, the dataset is split into five subsets with a random sampling technique (each subset contains 20% of the ratings). During each run of experiments, one of the subsets becomes test set and the rest is used for training.

4.1.3 Metrics

The quality of the predicted ratings was measured through the Root Mean Squared Error (RMSE). The metric compares the test set with the predicted ratings, by comparing each rating r_{ui} expressed by a user u for an item i with the rating p_{gi} predicted for the item i for the group in which user u is. The formula is shown below:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n}(r_{ui} - p_{gi})^2}{n}}$$

where n is the number of ratings available in the test set. The metric was chosen because, as the organizers of the Netflix prize highlight[2], it is well-known and widely used, it allows to evaluate a system through a single number and it emphasizes the presence of large errors (both false

[2] http://www.netflixprize.com/faq

positive and false negatives). These are important properties, useful in the evaluation of a recommender system.

In order to evaluate the properties of a clustering that affect the quality of group recommendations, four properties of each cluster were analyzed. These properties are described in the list below:

- **sparsity**, i.e., the average of the number of ratings per user of a group;
- **size**, i.e., the number of users per group;
- **diameter**, i.e., the maximum distance between two users in a group, calculated with the Euclidian distance between the ratings;
- **average distortion**, i.e., mean of the distances of each user from the centroid of the group. Each distance is calculated with the Euclidian distance.

4.2 Overall effectiveness

This section describes the evaluation of each group recommendation algorithm proposed. For each algorithm, experiments to set the parameters and find the best configuration are conducted.

In order to evaluate the quality of the predicted ratings for different numbers of groups, in each experiment four different clusterings of the users in 20, 50, 200 and 500 groups were created, apart for *MART*, that uses a clustering algorithm that does not allow to set the number of clusters produced.

Moreover, we compared the results obtained with the previously mentioned four clusterings, with the results obtained considering a single group with all the users (predictions are calculated considering all the preferences collected for an item), and the results obtained by the system that calculates individual predictions for each user.

To generate the clusterings with k-means, a testbed program called KMlocal [Kanungo et al., 2002] was considered. The program contained a variant of the k-means algorithm, called *EZ Hybrid*, developed by the authors. The k-means algorithm minimizes the *average distortion*, i.e., the mean squared distance from each data point to its nearest center. With the data used for the experiments *EZ Hybrid* is the algorithm that returned the lowest distortion and therefor the one used to cluster the users.

4.2.1 MART (Model-based Automatic Recommendation Technology)

As previously mentioned, *MART* is an algorithm designed to produce group recommendations using a model that contains the preferences of each group.

In order to properly evaluate the algorithm, the following aspects have to be set:

- parameter *noise*, that allows to reduce the complexity of the users' similarity graph;
- the strategy used to model a group's preferences;
- parameter *coratings*, used to decide which ratings have to be included in a group model;
- parameter *top* used to select the items considered in the prediction of group ratings;
- parameter *trust*, that allows to evaluate how reliable a group prediction is.

In order do so, five experiments have been conducted. Each experiment evaluates the performances of the algorithm for different values of the parameter or strategy considered.

As previously mentioned, the step of the algorithm that detects the groups does not allow to set a fixed number of groups, so it was impossible to repeat the experiments multiple times and conduct statistical tests to validate the results. That means that for each experiment, the value of a parameter that allows to obtain the lowest RMSE is considered.

MART: setting the *noise* parameter and detecting the groups

The *noise* parameter is used to subtract weight from the edges of the users' similarities graph and remove weak links between users.

Experiments are not presented, since with even with small values like 0.1, the graph would become disconnected and a subset of users could not be clustered. So, groups were detected with a value 0.0 for the *noise* parameter and the step that detects the groups returned returned three partitions in 4, 13 and 40 groups.

MART: modeling the groups

In order to build a model that contains a group's preferences, individual preferences have to be combined.

In this set of experiments, groups are modeled according to each strategy previously presented, in order to evaluate the one that best models the groups. If a strategy presents a *threshold* value (i.e., Approval Voting and Average Without Misery), all the possible values are tested.

The other parameters that have yet to be tested are set with a fixed value, i.e., parameter *coratings* is set to 10% (if *coratings* was set to 0, it wouldn't be possible to predict ratings, since all the items would be evaluated in the model), parameter *top* is set to 2 (in order to avoid the possibility of predicting a rating considering just one neighbor) and parameter *trust* is set to 0.0.

Figure 4.1 shows the trend of the RMSE values for each modeling strategy and each partition of the users in groups.

An important aspect to consider when analyzing these results is that the groups formed are very large (in fact, 6040 users partitioned into 40 homogeneous groups would generate groups with 151 users each).

Note that results obtained for the Average Without Misery Strategy have been omitted, since a very small portion of the test set could be considered (less than 10%) and results were not reliable. In fact, the strategy discards a group prediction if at least a person has evaluated an item with a rating lower than the threshold value. It is clear that even with a small threshold value, like 2, the vast majority of the items is not modeled by the strategy in such a context.

For the rest of the strategies instead, it can be noticed that as the number of groups is higher, the quality of the recommendations improves. This phenomenon will be extensively studied later in the chapter and the explanation of this result will be omitted in this and next experiments conducted to set the parameters.

The attention can be focused on the different modeling strategies. As it can be noticed, in every partition Additive Utilitarian is the strategy that best models the groups. As previously mentioned, this corresponds to a modeling with an average of the ratings collected for an item.

Since the system deals with large groups, this is why an average, that is a single value that is meant to typify a set of different values, is best way to put together the ratings in this context.

This explanation is also strengthened by the fact that Least Misery (i.e., the strategy that assigns to the group the lowest value assigned by a user to an item) is the strategy the performs worse. In fact, in such a context, a group would be modeled just with very low ratings.

So, the strategy chosen to model the groups generated by *MART* is Additive Utilitarian.

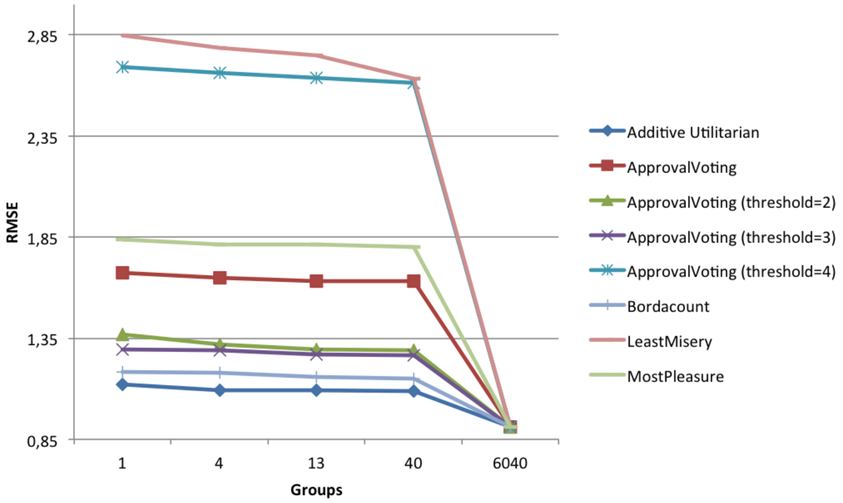


Figure 4.1: RMSE values for the different modeling strategies

MART: setting the *coratings* parameter

Since a model should reflect the preferences of the group, an aggregate rating produced by the modeling is considered only if a percentage of the group has rated the item. In order to do so, a *coratings* parameter should be set to a suitable value. The next experiment presents the values of

RMSE obtained by the algorithm for different values of the parameter. The modeling strategy used is Additive Utilitarian and the other parameters not yet tested keep the same values (*top*=2, *trust* = 0.0).

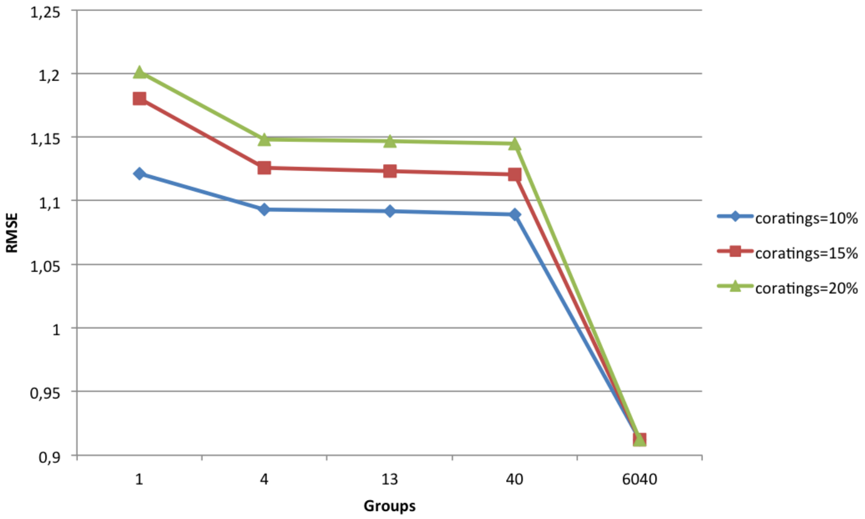


Figure 4.2: RMSE values for the different values of the *coratings* parameter

Figure 4.2 clearly shows that for every partition the initial value of *coratings*, i.e., 10%, is the one that allows to achieve better results. That means that as the higher is the value of *coratings*, the more ratings are eliminated for the model and the harder it is for the system to predict ratings for a group. Therefor, a 10% value will be used also for the next experiments.

The next experiment conducted is to evaluate the quality of recommendations for different values of the *top* parameter, i.e., the number of similarities considered to select the nearest neighbors of an item. The results won't be presented, since the results show that the quality of recommendations does not depend from this parameter and RMSE does not change. The initial value of 2 was kept to conduct the next experiment.

MART: setting the *trust* parameter

Parameter *trust* is used at the end of the computation, to evaluate if a group prediction can be considered. If the mean of the similarities of the items used to compute the prediction is higher than the value of *trust*, the prediction is considered.

An experiment is conducted to evaluate the performances of the system for different values of *trust*. Figure 4.3 illustrates the performances of the system for increasing values of the parameter. Performances improve for higher values of *trust*, i.e. when the ratings predicted can be considered more "reliable". However, the higher is the value of *trust*, the more predicted values are discarded. That means that for values of the parameter higher than 0.2, less than 50% of the ratings is considered in the test set (i.e., each RMSE value is calculated not considering more than half of the preferences). This is why experiments stopped and 0.2 is the chosen value for the parameter.

MART: conclusions

All the parameters used by *MART* have been tested and the algorithm will be next compared with the other presented, with the following configuration.

- parameter *noise* has been set to 0.0 (no edges were removed from the graph)
- *Additive Utilitarian* is the strategy selected to model a group's preferences;
- parameter *coratings* is set to 10%;
- parameter *top* is set to 2;
- parameter *trust*, is set to 0.2.

4.2.2 SMART (State-of-the-art Model-based Automatic Recommendation Technology)

SMART is an improvement of the model-based algorithm previously tested, that uses state-of-the-art algorithms and approaches. A few aspects, listed below, have to be set in order to run the algorithm.

- the strategy used to model a group's preferences;

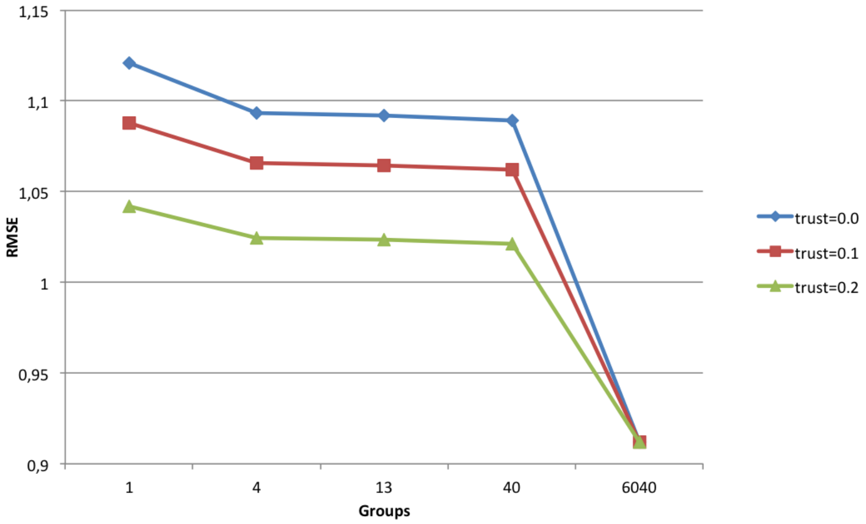


Figure 4.3: RMSE values for the different values of the *trust* parameter

- parameter *coratings*, used to decide which ratings have to be included in a group model;
- parameter *n* used to select the most similar items considered in the prediction of group ratings;

Three experiments allow to evaluate all the parameters and strategies that have to be set in the algorithm.

SMART: modeling the groups

In this experiment *SMART* models groups using different strategies, to evaluate the strategy that best models the preferences explicitly expressed by members of the groups created with the k-means algorithm. Again, if a strategy presents a *threshold* value (i.e., Approval Voting and Average Without Misery), all the possible values are tested and the other parameters re set with a fixed value, i.e., parameter *coratings* is set to 10% and parameter *n* is set to 10.

Figure 4.4 shows the trend of the RMSE values for each modeling strategy and each partition of the users in groups.

Results for the Average Without Misery Strategy are not presented, for the reason previously given (less than 10% of the test set is considered).

Again, Average Utilitarian is the modeling strategy that allows to achieve the best results. Once again, the modeling groups with Least Misery takes to the worst results.

Independent t-tests were conducted to compare the results obtained by *SMART* with each couple of modeling strategy. All the tests returned that there is a significant difference in the values obtained with each modeling strategy. For readability reasons, just the comparison between Additive Utilitarian will be reported.

For one group, there is a significant difference in the RMSE values for Additive Utilitarian (M = 1.059626, SD = 0.00) and Borda Count (M = 1.070556, SD = 0.00); t(7.92) = 5.72, p = 0.0.

The same happens for 20 groups, comparing the RMSE values for Additive Utilitarian (M = 1.040172, SD = 0.00) and Borda Count (M = 1.053482, SD = 0.00); t(6.92) = 6.28, p = 0.0.

For 50 groups, the test returned a complete statistical difference between the values obtained for Additive Utilitarian (M = 1.033472, SD = 0.00) and Borda Count (M = 1.053122, SD = 0.00); t(7.44) = 15.22, p = 0.0.

For 200 groups, the same happens when comparing the RMSE obtained for Additive Utilitarian (M = 1.026542, SD = 0.00) and Borda Count (M = 1.050044, SD = 0.00); t(4.28) = 24.61, p = 0.0.

Finally, even with 500 groups there is a significant difference in the RMSE values for Additive Utilitarian (M = 1.026246, SD = 0.00) and Borda Count (M = 1.054678, SD = 0.00); t(7.68) = 0.24, p = 0.0.

These results suggest modeling ratings averaging the preferences of the users in a group does lead to improvement with respect to the other strategies.

So, the strategy chosen to model the groups generated by *SMART* is Additive Utilitarian.

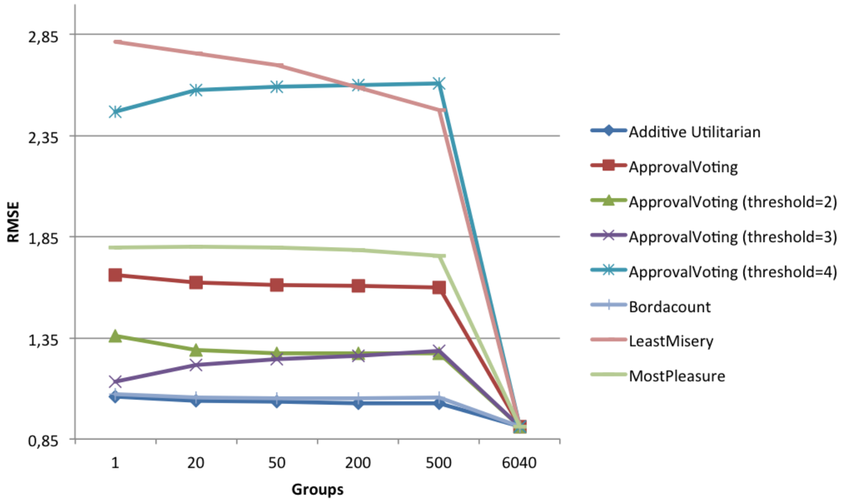


Figure 4.4: RMSE values for the different modeling strategies

SMART: setting the *coratings* parameter

The *coratings* parameter, which allows to consider in the model only the ratings rated by a certain part of the group has to be set. An experiment to evaluate a suitable value for the parameter is conducted. The modeling strategy used is Additive Utilitarian and parameter n is set to 10.

Figure 4.5 shows the same behaviour of the *coratings* parameter obtained with *MART*, i.e., for increasing values of *coratings* RMSE worsens, because an increasing number of aggregate ratings are eliminated from the model.

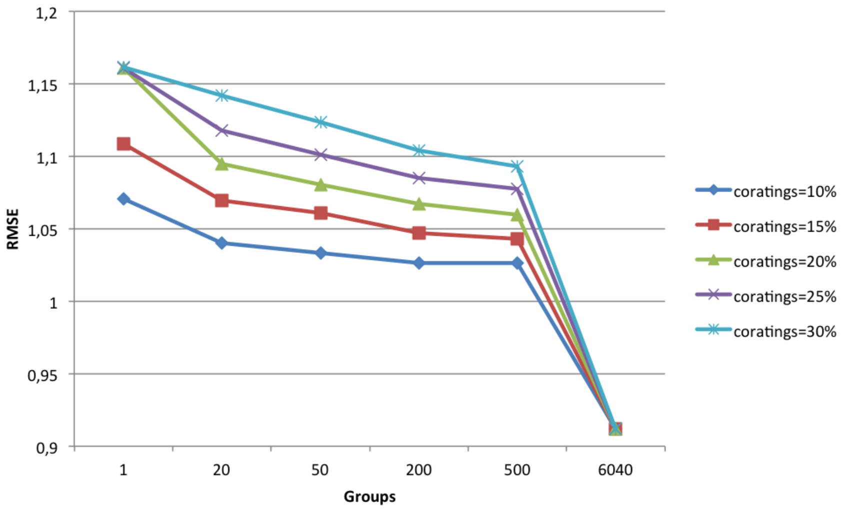


Figure 4.5: RMSE values for the different values of the *coratings* parameter

Again, independent-samples t-tests have been conducted to compare the results for different values of *coratings* in each clustering. All the tests returned that there is a significant difference in the values obtained with different values of the *coratings* parameter. The results obtained to compare the results obtained considering 10% and 15% of the group are presented next.

Considering 1 group, there is a significant difference in the RMSE values for $coratings$ = 10% (M = 1.070556, SD = 0.00) and $coratings$ = 15% (M =

1.108634, SD = 0.00); t(7.85) = 20.26, p = 0.0.

For 20 groups, the difference is also significant when comparing the RMSE values for *coratings* = 10% (M = 1.04019, SD = 0.00) and *coratings* = 15% (M = 1.069618, SD = 0.00); t(9.96) = 9.24, p = 0.0.

The test conducted for 50 groups returned a significant difference between *coratings* = 10% (M = 1.033476, SD = 0.00) and *coratings* = 15% (M = 1.06113, SD = 0.00); t(7.11) = 15.24, p = 0.0.

With 200 groups, the obtained results are *coratings* = 10% (M = 1.026542, SD = 0.00) and *coratings* = 15% (M = 1.047102, SD = 0.00); t(7.88) = 14.60, p = 0.0.

For 500 groups, there is a significant difference in the RMSE values for *coratings* = 10% (M = 1.026246, SD = 0.00) and *coratings* = 15% (M = 1.042848, SD = 0.00); t(7.68) = 13.80, p = 0.0.

The results suggest that lowering the *coratings* value allows to substantially improve the results. Specifically, these results suggest that the less ratings are removed from the model, the better the algorithm predicts the ratings for a group.

SMART: setting parameter n

In order to predict a rating for the group, the items most similar to the one currently predicted have to be selected. In order to do so, the right number of neighbors has to be selected when computing a prediction.

This is done with a parameter called n, tested in this set of experiments. The aspects previously tested are set as previously mentioned, i.e., the modeling strategy is Additive Utilitarian and $coratings = 10\%$.

Figure 4.6 shows the performances of the algorithm for different values of n, i.e., considering the selection of a different number of similar items.

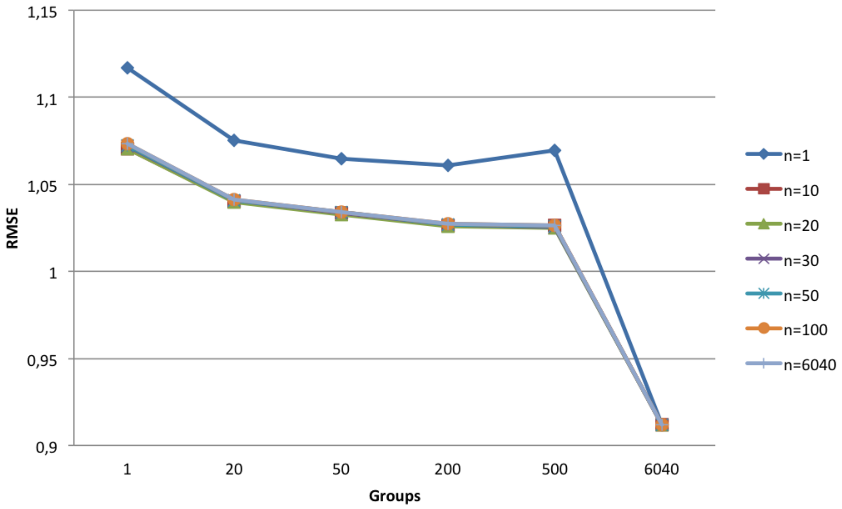


Figure 4.6: RMSE values for the different values of the n parameter

Unfortunately, results are not clear and it is impossible to see the value that allowed to obtain the best results. Therefor the figure has been zoomed, considering the part between 20 and 500 groups (Figure 4.7).

As the results show, there is an improvement up to $n = 20$, then results start worsening again (Figure 4.7 shows very similar results for $n = 10$, rep-

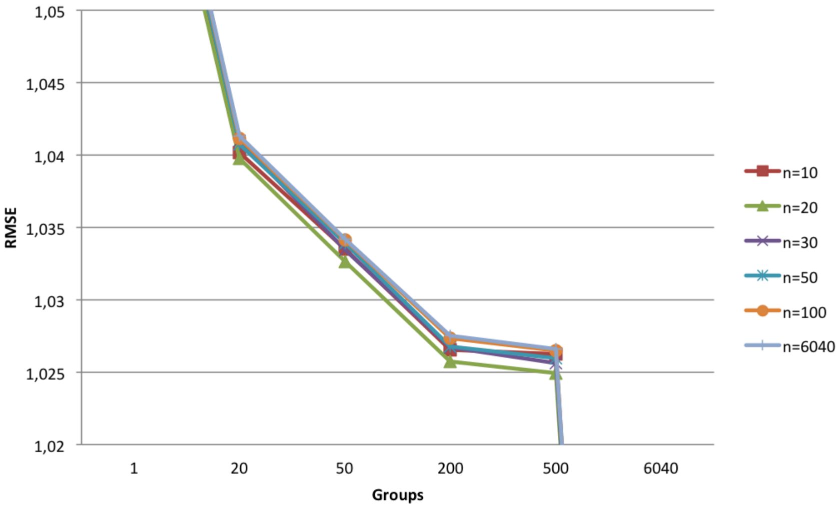


Figure 4.7: Detail of the experiment to study parameter *n*

resented by the red line, and n = 30, represented by the violet line). However RMSE value are very close, so it is important to conduct independent-samples t-tests to evaluate the difference between the results. In particular, the tests conducted to compare 20 and 30 groups are now presented.

Considering 1 group, there is a difference in the RMSE values for n = 20 (M = 1.070534, SD = 0.00) and n = 30 (M = 1.07217, SD = 0.00); t(9.92) = 0.87, p = 0.41.

For 20 groups, there is difference for the results obtained with n = 20 (M = 1.039798, SD = 0.00) and n = 30 (M = 1.040968, SD = 0.00); t(7.17) = 0.40, p = 0.70.

The test conducted for 50 groups returned a difference between n = 20 (M = 1.03344, SD = 0.00) and n = 30 (M = 1.033898, SD = 0.00); t(7.36) = 0.49, p = 0.63.

With 200 groups, there is a difference between the RMSE values obtained with n = 20 (M = 1.026698, SD = 0.00) and n = 30 (M = 1.026764, SD = 0.00); t(7.31) = 0.74, p = 0.48.

For 500 groups, the test returned a difference between n = 20 (M = 1.02493, SD = 0.00) and n = 30 (M = 1.025626, SD = 0.00); t(7.94) = 0.67, p = 0.52.

As it can be noticed, the results of the t-tests show that there is not enough confidence to reject the null hipotesys that the values obtained for n = 20 and n = 30 are different. However, the results obtained with n = 20 are always better in terms of RMSE and the t-tests returned that the probability that there is a difference for $n = 20$ ranges between 30% and 59%. Therefor, the value of n used to select the items similar to the one considered is 20.

SMART: conclusions

All the parameters used by *SMART* have been tested and the algorithm will be next compared with the other presented algorithm, with the following configuration.

- *Additive Utilitarian* is the strategy selected to model a group's preferences;
- parameter *coratings* is set to 10%;
- parameter n is set to 20.

4.2.3 APART (Aggregated Preferences-based Automatic Recommendation Technology)

As previously described, *APART* recommends to a group the top-n items that were predicted for each user of the group. In order to evaluate the algorithm, two parameters have to be set:

- parameter *neighbors*, used by the algorithm that calculates individual predictions;

- parameter *n*, that selects the number of items recommended to each user (top-*n* items).

APART: selecting the number of *neighbors*

In order to predict a rating for a user, the users most similar to the one currently considered have to be selected. In order to do so, the right number of neighbors has to be selected when computing a prediction. This is done with a parameter called *neighbors*, tested in this set of experiments.

Since we have to evaluate the number of neighbors for an algorithm that predicts individual ratings, this evaluation is done out of the group recommendation context. In other words, the RMSE values of the individual predictions for different values of *neighbors* are presented.

Figure 4.8 shows the RMSE values for increasing values of *neighbors*. As highlighted in [Desrosiers and Karypis, 2011] this is the common way to choose the value. Moreover, our results reflect the trend described by the authors, i.e., for low values of the parameter, great improvement can be noticed. As expected, RMSE takes the form of a convex function (Figure 4.9 shows a particular of Figure 4.8), that indicates that after a certain value improvement stops. In these experiment that value is 100.

Independent-samples t-tests can be conducted evaluate the difference between the results obtained between 100 and the other numbers of neighbors are now presented.

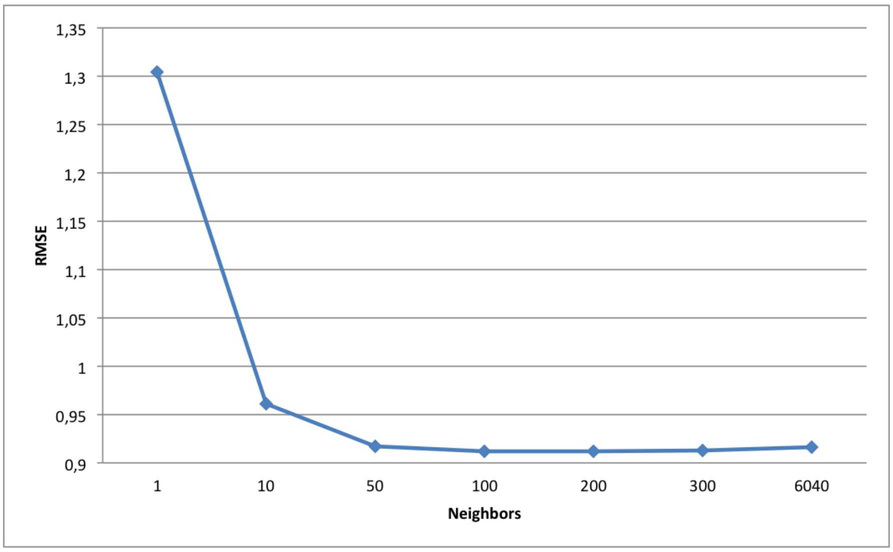


Figure 4.8: RMSE values for increasing number of *neighbors*

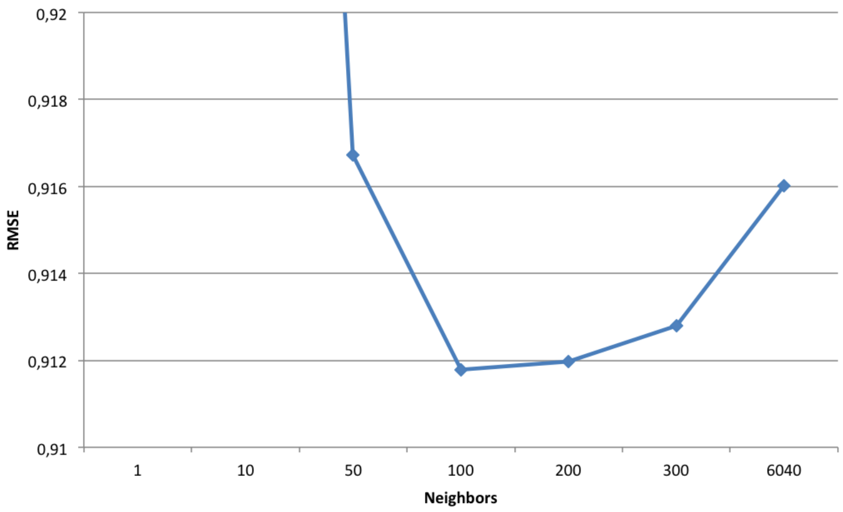


Figure 4.9: RMSE takes the form of a convex function.

There is a significant difference in the RMSE values for 1 neighbor (M = 1.304622, SD = 0.00) and 100 neighbors (M = 0.911785, SD = 0.00); $t(7.59) = 450.02$, $p = 0.00$.

There is a also a significant difference in the RMSE values for 10 neighbors (M = 0.961122, SD = 0.00) and 100 neighbors (M = 0.911785, SD = 0.00); $t(7.41) = 54.44$, $p = 0.00$.

A significant difference is also present in the RMSE values for 50 neighbors (M = 0.916725, SD = 0.00) and 100 neighbors (M = 0.911785, SD = 0.00); $t(7.97) = 6.02$, $p = 0.00$.

The RMSE values present a difference for 100 neighbors (M = 0.911785, SD = 0.00) and 200 neighbors (M = 0.911968, SD = 0.00); $t(7.99) = 0.24$, $p = 0.82$.

There is a also a difference in the RMSE values for 100 neighbors (M = 0.911785, SD = 0.00) and 300 neighbors (M = 0.912803, SD = 0.00); $t(7.97) = 1.06$, $p = 0.33$.

There is a significant difference in the RMSE values for 100 neighbors (M = 0.911785, SD = 0.00) and 6040 neighbors (M = 0.916022, SD = 0.03); $t(7.99) = 1.27$, $p = 0.24$.

As it can be noticed, for values of *neighbors* higher than 100, the probability that there is a difference between the values obtained for 100, 200 and 300 neighbors is between 18% and 67%. In particular, there seems to be no difference between choosing 100 and 200 neighbors. Since *neighbors* = 100 returned the best results and it is fast to compute predictions considering

100 neighbors instead of 200, this is the value chosen for the algorithm.

APART: choosing the top-n items

APART works combining recommendations made for individual users, in the form of the items that received the higher predicted rating (top-n items). This set of experiments allows to evaluate how big n should be, i.e., how many items should be selected from the predictions.

Figure 4.10 shows that selecting the top-5 out of all the predicted ratings allows to achieve the best results.

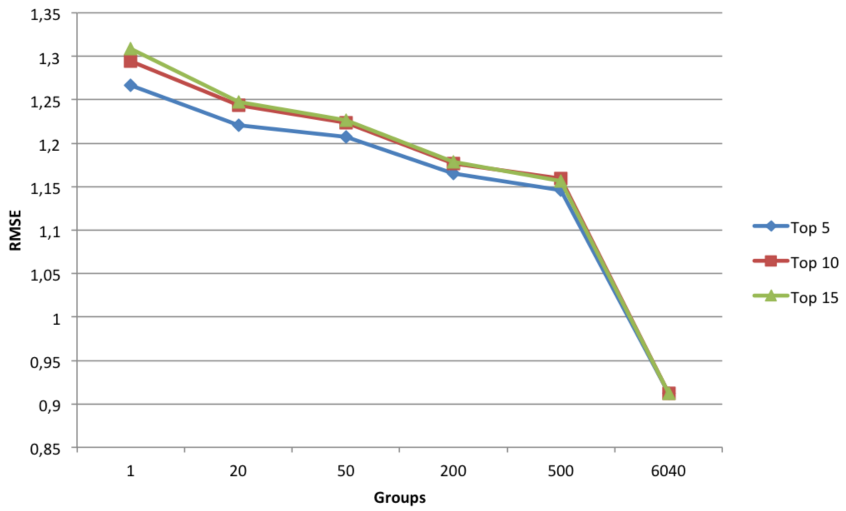


Figure 4.10: RMSE values for increasing values of n

Independent-samples t-tests have been conducted, in order to evaluate if there is a significant difference between the values obtained for the different values of n and different numbers of groups. Such a difference exists and the results of the tests that compare $n = 5$ and $n = 10$ are now presented.

Considering 1 group, there is a significant difference between $n = 5$ (M=1.266718, SD=0.00) and $n = 10$ (M=1.294696, SD=0.00); $t(7.74) = 3.39$, $p = 0.01$.

For 20 groups, there is a difference between $n = 5$ (M=1.220748, SD=0.00) and $n = 10$ (M=1.243682, SD=0.00); $t(7.99) = 1.46$, $p = 0.18$.

When 50 groups are considered, there is also a difference between $n = 5$ (M=1.207548, SD=0.00) and $n = 10$ (M=1.223508, SD=0.00); $t(7.98) = 0.65$, $p = 0.53$.

For 200 groups, there is also a difference between $n = 5$ (M=1.16532, SD=0.00) and $n = 10$ (M=1.176224, SD=0.00); $t(7.99) = 0.54$, $p = 0.60$.

With 500 groups, there is a difference between $n = 5$ (M=1.146128, SD=0.00) and $n = 10$ (M=1.159176, SD=0.00); $t(7.45) = 0.65$, $p = 0.54$.

Results show that when the number of groups increases, the significance of the difference between the value decreases. However, since for $n = 5$ the results are always lower and the lowest probability that the values are not different is 40%, the value was chosen for the algorithm.

APART: conclusions

All the parameters used by *APART* have been tested and the algorithm will be next compared with the other algorithms presented, with the following configuration.

- *Additive Utilitarian* is the strategy selected to model a group's preferences;
- parameter *neighbors* is set to 100;
- parameter *n* is set to 5.

4.2.4 BART (Baseline Automatic Recommendation Technology)

BART is a simple group recommendation algorithm that automatically detects groups according to the constrains imposed by the system, predicts individual ratings and models the groups.

Since the clustering algorithm uses the preferences explicitly expressed by users and the algorithm that predicts individual ratings is the same used by *APART* and was previously test, the only set of experiments that have to be conducted is the one that allows to decide with which strategy the groups should be modeled.

Figure 4.11 shows the obtained results with each modeling strategy. As it can be noticed, unfortunately a few modeling strategy could not be evaluated. This is the case of Average Without Misery (that could also not be tested previously for reasons already explained) and Approval Voting with threshold values 3 and 4, because considering only items with a high rating (i.e., with a rating above 3), too many ratings were discarded from the model and a low percentage of comparisons with the test set was done.

Once again, Additive Utilitarian is the strategy that best models the groups.

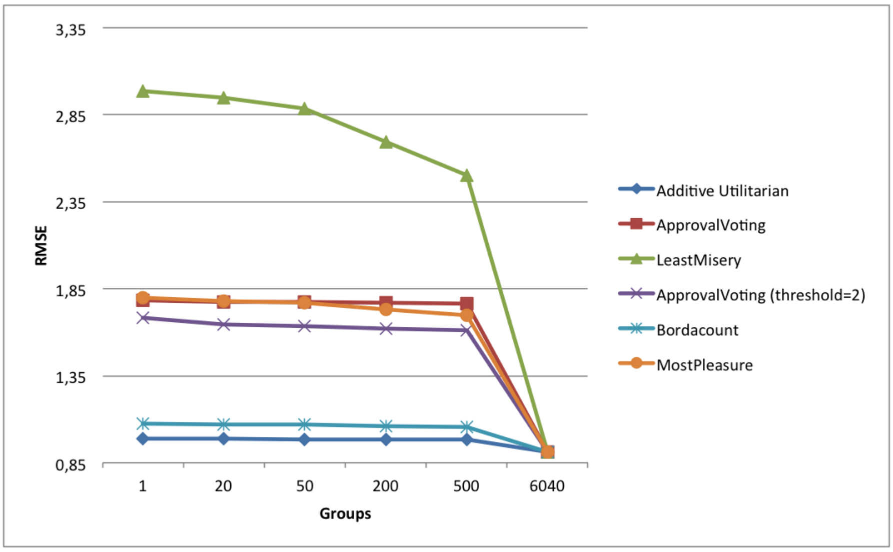


Figure 4.11: RMSE values for different group modeling strategies

Independent-samples t-test have been conducted, in order to compare the results obtained with each couple of modeling strategies. For readability reasons, just the tests that involve the strategy closer to Additive Utilitarian, i.e., Borda Count, are presented.

Considering 1 group, there is a significant difference between Additive Utilitarian (M=0.9894706, SD=0.00) and Borda Count (M=1.272588, SD=0.00); $t(5.56) = 559.80, p = 0.00$.

For 20 groups, there is a significant difference between Additive Utilitarian (M=0.9872284, SD=0.00) and Borda Count (M=1.263502, SD=0.00); $t(5.46)165.03 =, p = 0.00$.

When 50 groups are considered, there is a significant difference between Additive Utilitarian (M=0.9857, SD=0.00) and Borda Count (M=1.261276, SD=0.00); $t(5.55) = 320.60, p = 0.00$.

For 200 groups, there is a significant difference between Additive Utilitarian (M=0.9836828, SD=0.00) and Borda Count (M=1.250402, SD=0.00); $t(6.59) = 274.73, p = 0.00$.

With 500 groups, there is a significant difference between Additive Utilitarian (M=0.983241, SD=0.00) and Borda Count (M=1.24804, SD=0.00); $t(5.62) = 237.40, p = 0.00$.

Results clearly suggest that Additive Utilitarian is the best strategy to model groups, i.e., averaging individual ratings leads to produce the best group predictions with this algorithm.

BART: conclusions

Only one additional set of experiments was required to configure *BART*, which uses the same algorithm used by *APART* to produce individual predictions. All the group modeling strategies were tested and Additive Utilitarian was the one selected. The algorithm will be later compared with the other proposed algorithms.

4.2.5 HEART (Highly Enhanced Automatic recommendation Technology

HEART is an variant of the algorithm previously proposed, that clusters users considering individual predictions, in addition to the ratings explicitly expressed. So the clustering was repeated considering also the individual predictions calculated with *neighbors* = 100 and the modeling strategies were tested in a set of experiments.

Figure 4.12 shows the RMSE values obtained by each modeling strategy for different numbers of groups. The same modeling strategies that could not be evaluated with *BART*, i.e., Average Without Misery and Approval Voting with threshold values 3 and 4, could not be evaluated with this algorithm for the same reasons.

Results show that Additive Utilitarian is the strategy that allows to achieve the best results.

As always, independent-samples t-test have been conducted, in order to compare the results obtained with each couple of modeling strategies. For readability reasons, just the tests that involve the strategy closer to Additive Utilitarian, i.e., Borda Count, are presented.

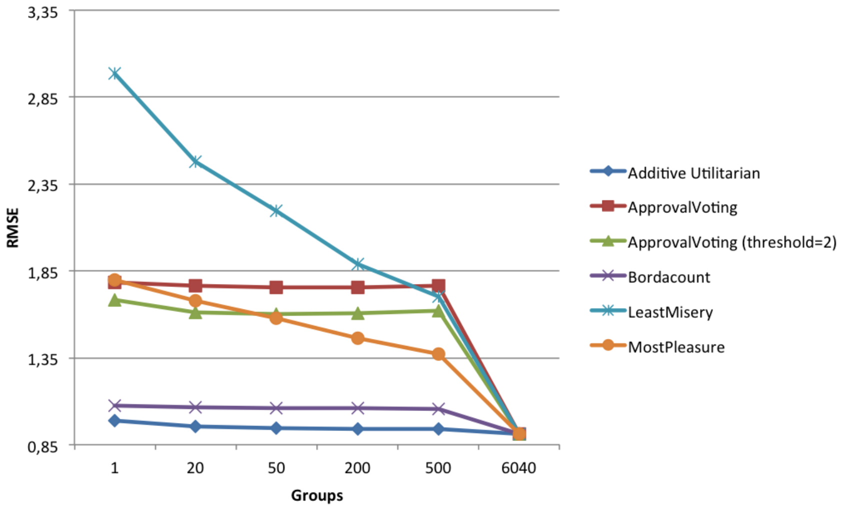


Figure 4.12: RMSE values for different group modeling strategies

Considering 1 group, there is a significant difference between Additive Utilitarian (M=0.989471, SD=0.00) and Borda Count (M=1.076738, SD=0.00); $t(6.91) = 230.75$, $p = 0.00$.

For 20 groups, there is a significant difference between Additive Utilitarian (M=0.955438, SD=0.00) and Borda Count (M=1.066654, SD=0.00); $t(7.03) = 55.53$, $p = 0.00$.

When 50 groups are considered, there is a significant difference between Additive Utilitarian (M=0.943494, SD=0.00) and Borda Count (M=1.062368, SD=0.00); $t(5.04) = 139.27$, $p = 0.00$.

For 200 groups, there is a significant difference between Additive Utilitarian (M=0.939531, SD=0.00) and Borda Count (M=1.059622, SD=0.00); $t(7.54) = 192.73$, $p = 0.00$.

With 500 groups, there is a significant difference between Additive Utilitarian (M=0.938527, SD=0.00) and Borda Count (M=1.056988, SD=0.00); $t(7.98) = 225.65$, $p = 0.00$.

Results clearly suggest that even for *HEART* Additive Utilitarian is the best strategy to model groups, i.e., averaging individual ratings leads to produce the best group predictions with the algorithm.

4.2.6 Comparing the algorithms

Once that all the parameters for each algorithm have been tested and the strategy that best models the groups has been selected, the performances of each proposed algorithm can be compared.

Figure 4.13 reports the results obtained by each algorithm with its best configuration.

An important aspect, not previously considered in the previous experiments, is that for all the algorithms, as the number of groups grows, the quality of the results improves (RMSE values get lower). This result will be deeply analyzed next, in order to understand which properties of a group cause this improvement.

As it can be noticed, the three families of approaches to produces group recommendation are clearly separated in the results.

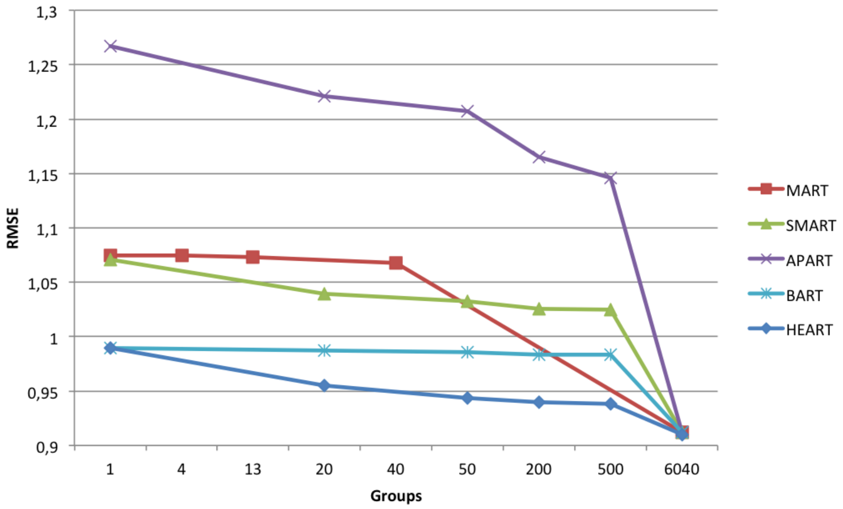


Figure 4.13: RMSE values obtained by each algorithm

In fact the approach that merges individual recommendations (*APART*) achieves the worst results. This is the sign that with large and automatically detected groups, if a user preferences are expressed just with a small subset of items (in this case five), a group recommendation algorithm is not able to properly satisfy users.

The approaches based on a group model (*MART* and *SMART*) lay in the middle. As depicted, the refinements introduced in *SMART* lead to great improvements in the quality of a model-based algorithm.

At the bottom of the figure, achieving the best results, are the algorithms that merge individual preferences (*BART* and *HEART*). Moreover, the performances of *HEART* are much better than the performances of *BART* and this proves that enhancing the clustering with individual predictions leads to great improvements in the quality of the predicted results.

Independent-samples t-tests, conducted to compare the results, confirm that there is a significant difference between the RMSE values obtained by *HEART* and the ones obtained.

4.3 Per-group effectiveness

The comparison of the algorithms previously presented illustrated an important results, i.e., that the more groups the system can handle, the more accurate predictions are. A question that arises when considering this aspect is: why does that happen? Is it because groups get smaller and the system has to put together the preferences of a small amount of users, making individual satisfaction easier to achieve? Is it because groups are cohesive? Is it because there are no users that do no fit with the group?

This aspects will be analyzed in a set of experiments that evaluates the different properties of a group, in order to analyze the factors that affect the quality of a group recommendation algorithm that automatically detects groups.

The algorithm used for this analysis is *HEART*, because it is the one that achieved the best results.

For each group, its RMSE was measured and compared with one or more of the factors of quality previously mentioned, i.e., *sparsity*, *size*, *diameter*, *average distortion*.

Results will be reported in a figure that reports all the groups of all the clusterings considered, in order to analyze if there is a clear trend that characterizes the considered aspect.

4.3.1 Sparsity

Sparsity is considered to be the average of the number of ratings per user of a group.

Figure 4.14 compares the RMSE values obtained for different values of sparsity. Moreover, a Linear Regression line of RMSE was added, in order to capture the trend of data.

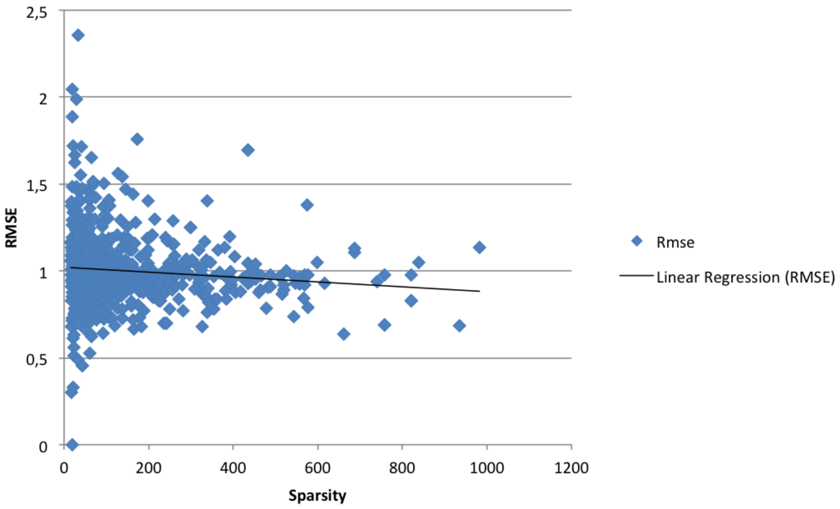


Figure 4.14: Effect of sparsity on the quality of the results

Results show that as sparsity grows (i.e., as the number of ratings per group grows), the quality of the predictions for that group improves. This results is pretty straightforward and indicates that if the system has more data to produce the recommendations, its performances improve.

4.3.2 Group size distribution

Size, i.e., the number of users per group, is the second aspect considered.

Figure 4.15 compares the RMSE values obtained for different values of size of the groups. Again, a Linear Regression line of RMSE was added, in order to capture the trend of data.

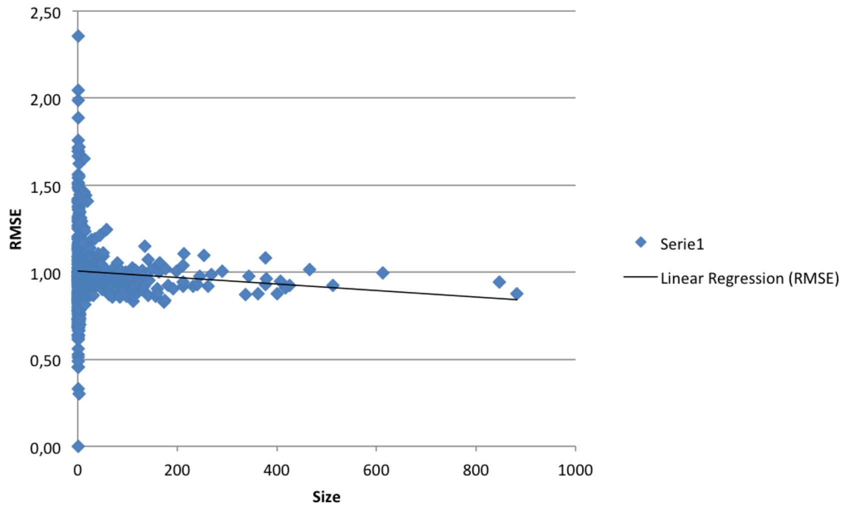


Figure 4.15: Effect of size on the quality of the results

Results indicate an unexpected trend. In fact one would expect that the overall quality of the results improve when the number of groups handled by the system is higher, because groups get smaller and it is easier for the system to put together a small amount of preferences.

However, results show that this is not what happens in this context and further experiments will be presented to understand the role of size.

4.3.3 Group diameter distribution

Diameter is the maximum distance between two users in a group.

Figure 4.16 compares the RMSE values obtained for different values of size of the groups and a Linear Regression line of RMSE is added, in order to capture the trend of data.

Results show that the higher is the diameter, the higher is RMSE. In other words, the higher is the distance between the two furthest users, the worse are the performances of the algorithm.

Distance between users has been investigated even more specifically. The next experiment will measure how distant users are from the center of the group and how that property affects the quality of the algorithm.

4.3.4 Average distortion distribution

Average distortion, i.e., the distance of each user from the centroid of the group, is the next aspect considered. Before analyzing the results of the

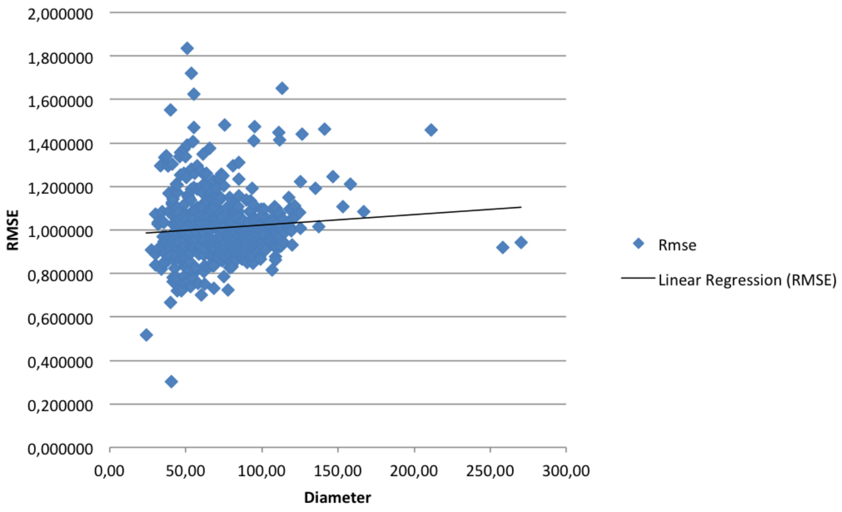


Figure 4.16: Effect of diameter on the quality of the results

experiment, reader should note that the higher is distortion, the less "cohesive" is the group.

Figure 4.17 compares the RMSE values obtained for different values of distortion and a Linear Regression line of RMSE is added, in order to capture the trend of data.

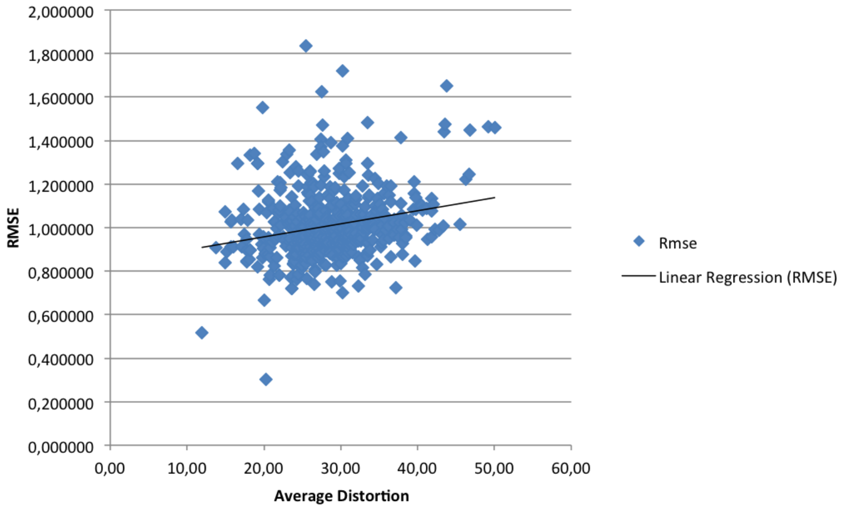


Figure 4.17: Effect of distortion on the quality of the results

Distortion is clearly the property that affects the most the performances of the algorithm. In fact, for higher values of distortion, RMSE worsens. So, the less cohesive is the group, the harder it is for the algorithm to predict for that group.

This result allows to understand why Additive Utilitarian is the strategy that works best to model the groups. In fact if a group has a low average distortion, its users are closer to the centroid. But the centroid is represented by the average of the ratings of the users. So, if users are closer to the centroids, their ratings are closer to the average and their RMSE is lower.

Now that the property the influences the most the quality of a group recommender system that automatically detects groups has emerged, it is important to understand what role does size play. In fact, it seems strange that for larger groups the system can actually improve its performances. This is why average distortion has been combined with size, in order to analyze how size affects the quality of a system.

4.3.5 Combining distortion and size

In order to understand how size affects the quality of a group recommendation algorithm, this experiment combines average distortion and size through their product and measures the resulting RMSE, in order to evaluate the distribution of the points.

Figure 4.18 shows that size does actually affect the performances of the system.

In order to inspect these results and understand how distortion and size affect the performances, all the cases in which more than a group had the same average distortion and all the cases in which more than a group

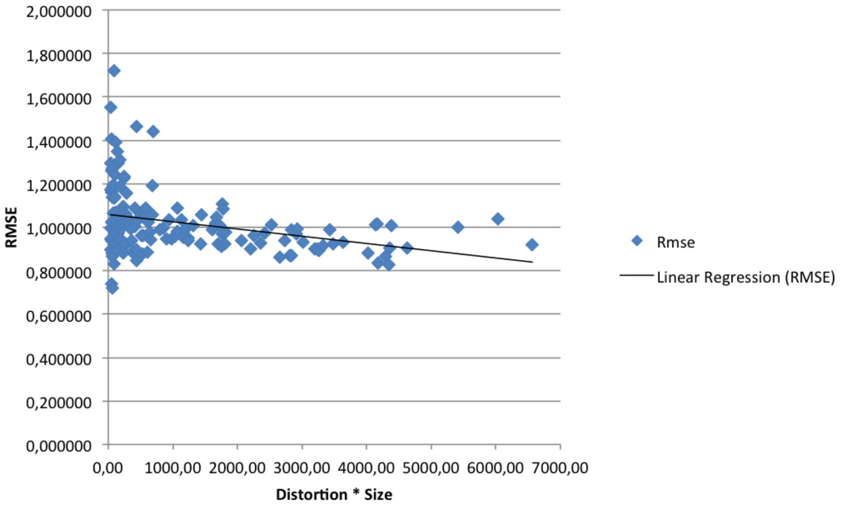


Figure 4.18: Combining distortion and size

had the same size were compared and the resulting RMSE values were analyzed. An example of each case is reported in Tables 4.1 and 4.2.

Average distortion	**Size**	**RMSE**
25.36	10	0.92725
34.37	10	1.013870

Table 4.1: Two groups with same size and different average distortion

Average distortion	**Size**	**RMSE**
30.85	3	1.00193
30.85	108	0.913884

Table 4.2: Two groups with same average distortion and different sizes

Table 4.1 shows that if two groups have the same size, the one with lower average distortion is the one for which better results can be achieved.

Table 4.2 shows a very interesting result, i.e., that if two groups have the same distortion, the one with larger size is the one that allows to achieve the best results. That means that if two groups have the same level of cohesion, it is easier for the system to build predictions for the larger one, since it can exploit more preferences.

4.4 Conclusions

This chapter presented a deep evaluation of all the proposed algorithms. All the algorithms have been test in order to find the configuration that allows that allows to achieve better performances.

All the algorithms have then been compared and experimental results showed that the family of algorithms that combines individual preferences is the one that works best with automatically detected. In particular *HEART* is the algorithm that performs best, indicating that the integration of the clustering input with individual predictions leads to improvements in the quality of the clustering and of the group recommendations.

Moreover, a set of factors of quality was inspected, in order to understand which properties affect a group recommendation algorithm that works with automatically detected groups. Per-group effectiveness was compared with these factors and average distortion of a group is the property that affects the quality of an algorithm. Moreover, the role of the size of a group was studied, in order to understand how it affects the performances.

Next chapter will present a study related to the *novelty* of the recommended content in a group recommender system.

Chapter 5

Effect on Novelty

This chapter presents a study conducted to evaluate how novelty of the recommended content affects the quality of group recommendations.

5.1 Introduction

A group recommendation approach that recommends the same content previously evaluated by users would be useful for content that is always renewed and ever-changing, like news items or TV series episodes. Users preferences for such types of content can be used to recommend items of the same type (e.g., news about the same topic or new episodes of the same series).

On the contrary, when a system produces group recommendations for types of content like movies, a new issue arises: *novelty* of the recommended items. In fact, if an item was already evaluated by a great part of the group, the system should limit its recommendation: users who already considered the item would be bored to watch/read/listen to it often and it wouldn't be a real recommendation for them.

This chapter presents a study that shows how novelty of the recommended content affects the quality of group recommendations. Recommending novel content creates a trade-off that involves an improvement in satisfaction of the users and a loss in the quality of the predicted ratings. Since groups of different sizes are automatically detected by the system we used, this study would allow a content provider to explore such a trade-off, considering also the level of personalization of the recommended content.

5.2 Experimental framework

This study will be conducted on the algorithm *HEART*, which as illustrated in the previous chapter, is the one that works best with automatically detected groups.

The main objective of the experiments is to measure how much novelty of the recommended content affects the quality of group recommendations, considering different partitions of the users in groups. To make the study, the MovieLens[1]-1M dataset was used. The dataset was pre-processed in

[1] http://www.grouplens.org/

the same way as previously explained and experiments were repeated multiple times, in order to conduct a k-fold cross-validation and independent samples t-tests.

Experimental methodology and setup

For each partition of the users in groups, ratings were predicted and the quality of the predictions was evaluated through the Root Mean Squared Error (RMSE). As previously described, the metric compares the test set with the predicted ratings: each rating r_{ui} expressed by a user u for an item i is compared with the rating p_{gi} predicted for the item i for the group in which user u is.

To evaluate the obtained RMSE values, the quality of the system that produces individual predictions for each user and the quality of the predictions made for a single group that contains all the users were considered. Inside that range it is reasonable to compare the different partitions, considering that recommendations predicted for a single user are the best result that can be obtained (predictions are tailored to a user's preferences) and a broadcast recommendation for a single group with no novelty of the content is still acceptable.

In each experiment the system performances were evaluated considering different values of a *novelty* parameter, i.e., the minimum percentage of users in a group that didn't previously rate an item, in order for it to be recommended. For example, if *novelty* was set to 50% and an item was rated by 60% of the group, the predicted rating for that item would be discarded, since the recommended item would not be novel just for 40% of the group.

Experimental results

Figure 5.1 shows RMSE for different values of content novelty for a set of groups, considering the same partitions of the users in 20, 50, 200 and 500 groups.

Result show that up to 30% it is hard to notice a worsening of the performances. This is because the groups handled by the algorithm are very large. In fact, it can be noticed that performances of 1 group do not vary until *novelty* > 60%.

In general performances start worsening when *novelty* > 30%. It can be noticed that, for higher numbers of groups (e.g., 200 and 500 groups), worsening of the results is faster, i.e., small groups are more affected by novelty and it is harder to recommend new items to a small groups.

5.3 Conclusions

This chapter presented a study on the novelty of the recommended content in group recommendation.

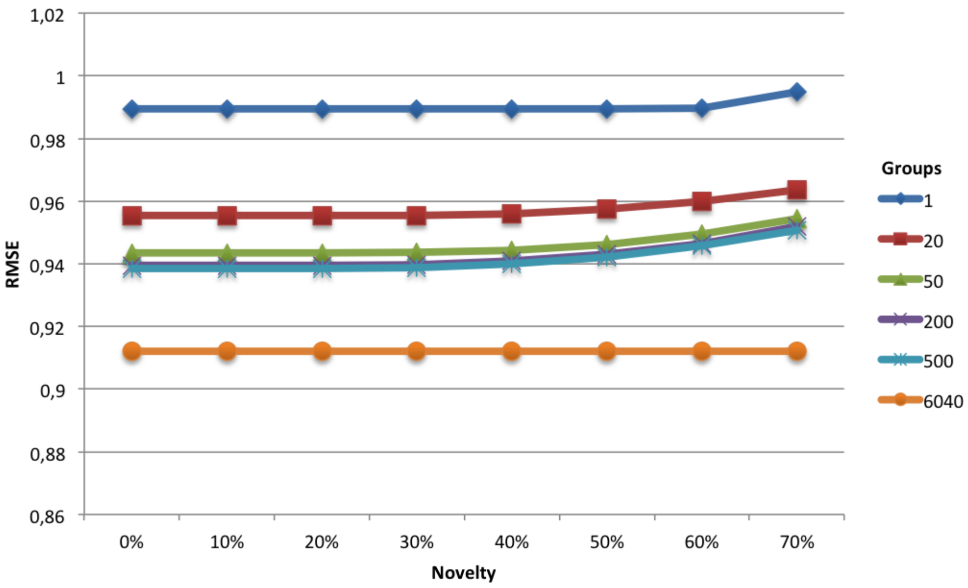


Figure 5.1: Performances for different values of novelty

Experimental results show that if the system tries to recommend novel content for more than 30% of the group, the performances start decreasing. Moreover, small groups are more affected by novelty.

Chapter 6

Market Segmentation

Market segmentation is the process that leads to an identification of groups of people with similar interests in terms of products or services. Such groups are usually called *market segments*. This chapter presents a technique to automatically identify market segments and classify users using query logs. Experimental results show that even observing just a few queries in a user's search history it is possible to classify users, detecting the market segments to which a user belongs.

6.1 Introduction

Web is in constant evolution and new content is reached by millions of users in a very little time.

With such a wide target of users it is hard to make effective advertising by targeting users that might be interested in a product. Market segmentation is the process that divides a large market into subsets, usually called *market segments*, that have common needs or have similar preferences. Market segments generally receive the same marketing campaigns, targeted on the preferences/needs of the segment.

Market segmentation techniques rely on several sources of data and require work of domain experts to detect the market segments. The result is usually a very complex classification of the users, that can be difficult to use for an advertiser.

Moreover, an effective classification of such a type way would be very hard for the web, because the amount of users is both big and ever-changing. No team of domain experts could actually analyze the preferences of web users and follow the trends in order to build an effective market segmentation.

In such a complex scenario, it would be useful to have a market segmentation that is automatically built and simple to analyze for advertisers.

In this section a technique to segment markets based on query logs is proposed. The search history of a user is analyzed and product queries are detected, in order to exploit the preferences of a user for certain product categories. A subset of this preferences is used to cluster users and identify market segments. Demographic and behavioral features are then used to classify users.

Experimental results show that even monitoring just a few product queries of the users can lead to the detection of the correct segment to which the user belongs.

The scientific contribution of the proposed technique is the capability to automatically detect market segments using implicit data that comes from query logs. Moreover, the approach is able to find a small number of segments, allowing advertisers to easily analyze how the market is composed and find the segments to reach.

6.2 Experimental framework

In order to build an automatic market segmentation, information about the interests of users have to be acquired. To do so, the queries typed on the US Yahoo! search engine from June 5th 2009 to June 7th 2010 were collected.

The subset of queries typed by registered users from the US was selected. Moreover, queries were ordered by decreasing popularity and only the top million queries were selected.

Out of all the data available for each query, only some information was selected. In particular, the query and the demographic information about the user who typed that query were kept.

For each query typed by a user, this was the considered data:

user id *year of birth* *ZIP* *gender* *query* where *user id* is a completely anonymous string of characters.

Since the approach aims to build an automatic market segmentation, we are interested in queries related to products. From now on such queries will be called *product queries*. From the market segmentation point of view, product queries are the only ones interesting, so they were the only ones considered. Product queries can be detected using Yahoo! categories, that allow also to derive the category to which each product belongs.

In order to classify users properly, enough information about a user's interests was needed. So we considered only users who typed at least 20 product queries. At this stage, the number of considered users was 588916 and the number of product categories was 100.

Knowing the number of product queries a user typed and the category of products associated to each product query, it is possible to derive the level of *interest* of a user u for a product category c:

$$i_{uc} = \frac{pq_{uc}}{pq_u} \tag{6.1}$$

where pq_{uc} is the number of product queries typed by a user u for a product category c and pq_u is the number of product queries typed by user u.

In the data considered for each query, the ZIP code of the user who typed the query was available. From the ZIP code it is possible to have some more demographic information about a user. In particular, the information available in the US Census 2000[1] was used. So the information available in a query about a user was augmented with the US Census 2000 data.

Each user can now be represented by a vast set of features, that also describe the characteristics of the area in which a user lives. The list of features is listed below:

user id *year of birth* *ZIP* *gender* *population*
average household size *mean travel time*
% *non − english language people* % *people bachelors degree*
% *people below poverty* *per − capita income* % *white people*
% *black people* % *asian people* % *hispanic people*

Merging demographic data with the US Census allowed us to process the data once more. In fact we were able to check the ZIP entered by each user and select only users who typed a real ZIP code. The number of considered users was reduced to 557766.

6.2.1 Behavioral features

Considering the *interest* of each user for a product category, it is possible to make a representation of the user considering behavioral features. Given n product categories, a user can be represented with behavioral features by:

[1] http://factfinder.census.gov/

user id i_{u1} i_{u2} ... i_{un}

where i_{u1} ... i_{un} represents the interest of a user u for each product category.

6.3 Clustering based on behavioral features

The clustering part of the approach is based on behavioral features. In fact the interest of each user for product categories was used as input for the clustering. Since behavioral features are interesting to consider for both the clustering and the classification, the input for the clustering algorithm was the set of behavioral features, built considering the queries a user typed after the tenth query.

The algorithm chosen to cluster users is k-means. In particular, we used a variant of the algorithm, called *EZ Hybrid*, developed inside a testbed program called KMlocal [Kanungo et al., 2002].

As mentioned in the Introduction, one of the objectives of this approach is to have a segmentation that can be easily inspected by humans. So the number of clusters cannot be too high. Considering a number of clusters between 2 and 20, an important aspect to consider is the choice of the right number of clusters. In fact the partition of users in clusters has to reflect the actual market segmentation.

To decide the number of clusters, different evaluations were done.

Size of each cluster. Having clusters of too different sizes would not be helpful, since too large clusters would be hard to inspect (i.e., the preferences of the users in it might not be clear). It would optimal to have a clustering where the size of each cluster is similar.

Features that characterize a cluster. For each cluster the number of discriminative features (i.e., the features that characterize a cluster) is considered. If a cluster has a high number of elevated features, the preferences of the users in it are not well-defined. If that happens in a lot of clusters, we can assume that the considered clustering is not a good partitioning of the users.

Elbow method. For each clustering in k clusters, its distortion is measured (i.e., the sum of squared distances for the centroids). If there is a drop of distortion for a value of k, that means that the true partition of the users is in k clusters.

After considering the aspects previously mentioned, we chose a clustering in 13 clusters.

6.4 Users classification

After users are clustered, they can be classified into segments. To classify users using the available data, three different choices could be made:

- Classification of the users considering only demographic data

- Classification of the users considering only behavioral data
- Classification of the users considering both demographic and behavioral data

Since the clustering step considered all the queries a user typed after the tenth, the classifications that consider behavioral data can be done considering the first ten typed queries. Moreover, the classification can be repeated ten times, starting from the first typed query and considering one more query at a time. Seeing how accuracy of the classification changes considering different subsets of queries can be useful to inspect how having more information about a user's interests affects the results.

The third classification can be also useful to understand how demographic and behavioral data affect the classification accuracy.

The classification was done using Weka with the Naive Bayes algorithm and a 10-fold cross-validation.

6.4.1 Classification based on demographic features

To classify users considering demographic data, the users representation with the demographic features was considered and the cluster to which a user belongs was added as the true class.

Accuracy of the classifier, knowing just the demographic data, was 12.37%.

6.4.2 Classification based on behavioral features

To classify users considering behavioral data, the users representation with the behavioral features was considered and the cluster to which a user belongs was added as the true class.

Since the first ten queries typed by each user were not considered for the clustering, it is now possible to use them to classify users. Moreover, it is possible to evaluate how accuracy changes classifying users considering always one more query, in respect to the previous classification. In the initial evaluation we calculate accuracy of the classifier considering just the first query, then the first two queries, and so on. These are the preliminary results of the classifier:

Behavioral (First query)	18.27%
Behavioral (First 5 queries)	30.77%
Behavioral (First 10 queries)	45.03%

As predictable, having more information about a user leads to great improvements in the classification accuracy.

6.4.3 Classification based on demographic and behavioral features

To classify users considering both demographic and behavioral data, the users representations with the demographic and behavioral features were merged and the cluster to which a user belongs was added as the true class.

One again users were classified adding one query at a time, to monitor how accuracy improves.

Demographic + Behavioral (First query)	19.34%
Demographic + Behavioral (First 5 queries)	30.62%
Demographic + Behavioral (First 10 queries)	42.45%

Once again, having more information about a user's interest leads to great improvements in the accuracy of the classifier.

Comparing the results

Here the obtained results will be put together, in order to evaluate how demographic and behavioral data affect the accuracy of the classifier.

Demographic	12.37%
Behavioral (First query)	18.27%
Demographic + Behavioral (First query)	19.34%
Behavioral (First 5 queries)	30.77%
Demographic + Behavioral (First 5 queries)	30.62%
Behavioral (First 10 queries)	45.03%
Demographic + Behavioral (First 10 queries)	42.45%

As it can be noticed, in order to improve accuracy of the classification, demographic information is useful just when little information about a user's behavior is known. When a user typed even just 5 queries, demographic information not only becomes useless but it also is misleading and accuracy of the classifier worsens.

6.5 Conclusions

Chapter 7

Tag Clustering

This chapter presents an approach to cluster tags of a tagging system, in order to facilitate the exploration of a tagging systems. The system outperforms the existing state-of-the-art techniques.

7.1 Introduction

The development of Web 2.0 applications, like blogs and wikis, led to a continuous growth of information sources, with daily uploaded resources shared by many users. Besides traditional techniques to categorize and index data, new approaches based on collaborative tagging have been effectively proposed and adopted. The success of those approaches is due to the fact that tagging does not require specific skills and seems a natural way for people to classify any kind of resource.

A set of tags (tagspace) can be explored in several ways and many tagging systems usually define sets of related tags, called *tag clouds*, that help the tagspace visualization. However, as highlighted in [Golder and Huberman, 2006], there are some well-known linguistic limitations that can inhibit information retrieval in those systems. In particular, the meaning or semantics of a tag is usually unknown. For instance, tag "orange" might refer either to a fruit or a color. Moreover, people use several tags to select the same resources. For example, a resource related to a pasta dish could be tagged as "Italian food", "spaghetti", "first course", etc. On the one hand, user can freely choose which tags classify resources in a useful way; on the other hand, the searching activity of other users within the tagspace could be limited. In fact, to find a resource it might be necessary to search several times using different keywords, and people should evaluate the relevance of the retrieved documents.

Grouping related tags together would avoid such limitations and simplify the exploration of a tagging system [Bielenberg and Zacher, 2005]. In fact, the definition of sets of related tags would help the identification of a context that would make resources retrieval easier.

In this section, RATC (Robust Automated Tag Clustering), a technique that monitors users activity in the search engine of a tagging system in order to exploit implicit feedbacks provided by users, is presented. A feedback is collected each time a user finds a relevant resource during a search in a tagging system. The algorithm uses the feedback to dynamically strengthen associations between the resource indicated by the user and the tags used in the search string. Tag-resource associations are then used to infer tag-tag associations by adopting a standard correlation measure. Tag-tag associations allow to cluster tags in order to find strongly related tag sets. Results have been compared with the ones obtained by adopting the state-of-the-art approach proposed in [Begelman et al., 2006] showing an improvement in the presence of strongly related tags in a cluster.

The main contribution of the proposed approach is that, by supervising users activity in a tagging system and monitoring their searches, we can progressively create and update tag-resource associations and tag-tag associations, rewarding the real semantic relations among tags and penalizing the misleading ones.

The rest of the section is organized as follows: in 7.2 the state-of-the-art in tag clustering is presented; 7.3 describes in detail the steps we followed to build the technique; in 7.4 the performed experiments are described and main results are outlined.

7.2 Related Work

In the literature, several techniques, aimed at grouping tags by adopting different clustering algorithms and heuristics, have been presented.

In [Specia and Motta, 2007], an approach that tries to infer the semantics behind a tag space is proposed. The corresponding collaborative tagging can help in finding groups of concepts and partial ontologies. This is achieved by using a combination of shallow pre-processing strategies and statistical techniques together with knowledge provided by ontologies available on the semantic web. This technique starts pre-processing the data and cleaning up the tag space, then, evaluating co-occurrences, it finds tag-tag associations and clusters them. Semantic relations are extracted from the clusters and the results consist of groups of highly related tags that conceptualize specific facets of knowledge and correspond to elements in ontologies. This approach differs from the one proposed in this chapter since the proposed technique does not pre-process the tagspace. The approach, in fact, is able to adaptively remove noisy tags by monitoring user interactions.

In [Hamasaki et al., 2008], being aimed at extracting ontologies, authors proposed a way to integrate a social network with collaborative tagging. The usual tripartite models of ontologies based on users, tags and instances, are integrated with user-user relations. Concepts in each community (called *p-concepts*) are considered different and this model was used to resolve the polysemy/homonymy problem. This technique aims to group p-concepts and find keywords associations by using an algorithm that considers the interactions among users and p-concepts. This approach differs in the sense that RATC considers users interaction just to link resources to tags, without creating explicit associations among users and resources.

In [Wu et al., 2006], an approach aimed at generating groups of semantically related tags through a probabilistic model is presented. The technique is based on evaluating co-occurrence of tags, resources, and users. The approach proposed here differs because it does not rely on a probabilistic model and it does not consider users.

In [Giannakidou et al., 2008], a co-clustering approach, based on the one proposed in [Dhillon, 2001], is employed. In this approach, tags and resources belonging to different datasets are clustered together. The clustering activity is based on a similarity metric that uses tag co-occurrences and semantic knowledge about the tags. The relations among the elements are used to enrich ontologies and to train multimedia processing algorithms. On the contrary, in this approach, the clustering activity is based just on tags and new knowledge is inferred by clustering elements of the same dataset.

In [Smyth et al., 2003b], a technique to exploit information from queries is presented. Associations between the keywords used in a query and the relevant resources retrieved by a search engine are exploited in order to rank search results based on the past users activity. The technique proposed creates associations also between a resource and the tags used to classify it when uploaded.

In [Baeza-Yates, 2005], an approach to create clusters of queries is presented. Related queries are clustered together, in order to recommend a better query to users. This is achieved by finding the most descriptive words in a cluster and recommending better queries to users. In this approach, queries are used in a different way. Associations among tags clustering queries are not inferred, but tags associations are derived considering the resources that they classify.

In [Begelman et al., 2006], a technique to cluster strongly related tags is presented. The algorithm is based on counting the number of co-occurrences (tags that are used for the same resource) of any pair of tags and a cut-off point is determined to decide if the co-occurrence count is significant enough to be used. Tags are clustered with an algorithm that is based on the spectral bisection and uses the modularity function to measure the quality of a partitioning. Related tags are then automatically discovered by incrementing a counter for each pair of tags that belong to the same cluster. Although the approach presented is quite similar, the main difference is that tag-resource associations are continuously updated during the use of the system.

7.3 RATC: Robust Automated Tag Clustering

RATC, which stands for Robust Automated Tag Clustering, monitors users activity in the search engine of a tagging system. The technique has been defined “robust” to put into evidence its ability to overwhelm the misleading resource classification problem. Robustness is the capability of an algorithm to remain stable in presence of fake information, usually added on purpose to influence its quality [**?**].

7.3.1 Top Level View of the Approach

RATC encompasses four main steps:

***Tag-resource* associations creation.** As in any tagging system, each time a new resource is put into the system, a *tag-resource* association is created among that resource and the tags used to describe it.

Dynamic *tag-resource* associations evaluation. Users activity in the tagging system search engine is monitored and exploited in order to update existing *tag-resource* associations and to create new ones.

***Tag-tag* associations creation and quantification** Dynamic *tag-resource* associations are exploited to create associations among tags (*tag-tag* associations). A standard cosine similarity measure [Baeza-Yates and Ribeiro-Neto, 1999] is used to evaluate the similarity among tags. The result of this process is a weighted graph (*tag similarity graph*) in which each node represents a tag and each weighted arc represents the similarity value of the tags it connects.

Clustering. The community detection algorithm proposed by [van Dongen, 2000] is applied to the tag similarity graph in order to detect the intrinsic communities of tags.

7.3.2 Representation of a Tagging System

A tagging system is a community driven tool that allows users to classify resources by using tags. It can be represented as a bipartite graph that contains:

- a set T of tags t;
- a set R of resources r;
- a set $A : (T \times R)$ of weighted arcs $t - r$, representing tag-resource associations. The weight of the tag-resource associations represents the number of times that a tag has been associated to a resource by users.

As depicted in Figure7.1, a tagging system is composed by a set of tags (rectangular nodes) linked by a weighted arc to a subset of resources (round nodes). In the example in figure there are three resources concerning with "goal actions" in a soccer game [1]. All of those resources has been classified with the tags *soccer* and *goal* and the weight of each arc represents the strength of the association between a tag and a resource. Each tag has some outgoing dotted arcs, which indicate that there are other resources linked to those tags, not depicted in the example.

As a final remark, this approach does not take into account different meaning associated to a same word (i.e. polysemy). For instance, in the example in Figure7.1, tag *goal* is used either as a successful attempt at

[1] Resources represent multimedia documents, like videos or pictures.

scoring in a soccer game or as the place designed at the end of a race.

7.3.3 Tag-Resource Associations Quantification

The standard search paradigm provided by tagging services is based on query strings containing one (or more) tag. The search returns a list of resources associated to these tags. To provide such list, a ranking of the results is derived according to the tag-resource associations available in the tagging system that can be considered as the strength of the association between a resource and each tag used to describe it. While tagging systems usually associate tags and resources at upload time, implicit user feedback, coming from its search activity, can be exploited to improve *tag-resource* associations.

To represent the strength of *tag-resource* associations we adopted an algorithm based on counters. The algorithm exploits users feedback to discover and emphasize correct associations strength, while making negligible the contribution of "noisy" associations. The strength of each association evolves according to an extremely simple and effective mechanism. A *tag-resource* association is created each time a resource is added to the system by a user. After a search operation based on a tag, each time the user selects a resource, the counter of the tag-resource association is increased (an example of tag-resource associations is shown in Figure7.1). Although a huge number of resources may be related to a single tag, their relevance will depend on the feedbacks provided by the community of users. In such a way the association of a misleading tag to a resource will give a negligible contribution.

In order to contain the counters relative to tag-resource associations, a matrix $W = \{w_{rt}\}$ is defined, where w_{rt} is the association between resource r and tag t (an example is depicted in Figure7.2).

Initial values are assigned when a new resource is uploaded and values are updated either when a user adds a tag already present in the database or when a feedback is given [2]. When a new resource is uploaded to the tagging system together with some tags, the corresponding *tag-resource* counter is set to 1. If such association is already present in the system, the corresponding w_{ij} is incremented. The matrix is also updated when a user performs a search in the tagging system and selects one of the results as relevant. At this stage, after the user selection took place, the counters between the selected resource and all the tags in the query list are incremented, namely $w_{rt} = w_{rt} + 1$.

The tagging system shown in Figure7.1 has been built using the tag-resource counters described above. Let us stress the fact that the strength of the relation between a tag and a resource in our tagging system is based on the feedbacks left by the users during the use of the system. For example, tag *soccer* has been used three times to classify and search the

[2] The initial values are just a starting point that evolve as the feedbacks are collected.

second resource concerning with the goal action of a player.

7.3.4 Tag-Tag Associations Quantification

Let v_i be the vector of associations among a tag i and its related resources and v_j be the vector of associations among a tag j and its related resources. The association a_{ij} between tag i and tag j can be measured by the cosine similarity between the vectors as follows:

$$a_{ij} = cos(v_i, v_j) = \frac{v_i \cdot v_j}{||v_i||_2 \times ||v_j||_2} = \frac{v_{i1} \cdot v_{j1} + ... + v_{ik} \cdot v_{jk}}{||v_i||_2 \times ||v_j||_2}$$

These associations can be represented in a graph, called *tag similarity graph*, which links each couple of associated tags with a weighted arc. An example, built using the associations among tags and the resources shown in Figure7.1, is represented in Figure7.2 [3].

7.3.5 Clustering

To perform clustering MCL (Markov Clustering Algorithm) [van Dongen, 2000] was adopted. MCL is a Community Detection [Porter et al., 2009] algorithm, built to find cluster structure in simple graphs, considering the similarity between vertexes. The MCL algorithm tries to simulate the flow within a graph, considering just the part where the flow is strong and

[3] The values of the associations in the figure have been calculated considering the whole tagging system.

removing weak connections. If natural groups are in the graph, the links between the groups disappear, leaving the cluster structure.

The flow is simulated by a transformation of the graph into a Markov graph (a graph where for all nodes the sum of the weights of outgoing arcs is 1) and is expanded by computing powers of the associated stochastic (Markov) matrix.

Since these operations are not enough and do not reveal the clusters in the graph, a new operator (inflation) is inserted. Flow inflation [van Dongen, 2000] is the entry-wise Hadamard-Schur product of the matrix combined with a diagonal scaling, while flow expansion is represented by the usual matrix product. The inflation operator has been introduced to strengthen and weaken the flow, and the expansion operator is responsible for allowing flow to connect different regions of the graph.

As the MCL algorithm basically consists of alternation of two different operators on matrices, followed by interpretation of the resulting limit, its formulation is quite simple. It is also possible to find clusters of different granularities, by varying its parameters.

A more detailed description of this algorithm is beyond the scope of this thesis. The interested reader could refer to [van Dongen, 2000] for further details.

7.4 Experiments and Results

To evaluate the proposed approach, first, a tagging system with an internal experimental search engine [Carta et al., 2008] was adopted, and then, the performances were compared with a state-of-the-art approach [Begelman et al., 2006].

Several aspects have been taken into account while performing comparisons regarding the robustness of the two approaches. In particular, to analyze the impact of noise in the performances, noisy tags were suitably added to the tagging system. The quality of the obtained clusters have been evaluated comparing results with the ones provided by a domain engineer in terms of precision and recall. The adaptive capability of our approach (i.e., the users activity monitoring and their feedback) has been evaluated measuring the temporal evolution of clusters quality.

7.4.1 Setting Up the Experiments

To conduct the experiments 10 volunteers populated a tagging system [Carta et al., 2008] (*resource acquisition* step). They were asked to select as many videos as they wanted from YouTube [4] and add them to the tagging system. The application domain was limited to "sport" as specific topic, which can be considered a concept domain. Each video was classified with four tags related to the resource and two tags (the noisy tags) not related

[4] http://www.youtube.com

to the resource. Noisy tags were added to simulate the noise that typically occurs in practice.

Once the tagging systems was populated, volunteers were asked to perform normal searches in the tagging system (*feedback collection* step). During this step, RATC improves its performances monitoring users search activity. Videos are chosen based on a preview shown to the user and their original description. This step started as soon as the resource acquisition step was completed. The reason was to neatly separate the initial values of the correlations from their evolutions caused by the feedbacks of the users.

Resources Acquisition

Each time a volunteer added a new video to the application, she/he had to create two sets of tags. The former is devoted to contain (at least) four characteristic tags, strongly related to the video; the latter is devoted to contain two tags not related to the video but in the same domain (in this experiments, “sport”). This tag set is required to create some verifiable noise and it has been used to monitor the progressive decreasing of their correlation with the video they had been initially introduced with. Such noise is useful to evaluate the clustering algorithm. In particular, in this way we are able to monitor how the clusters structure changes and to evaluate the quality of the clusters.

The tagging system was populated with a total of 406 videos, 1021 tags, 2597 video-tag correlations. Although in [Golder and Huberman, 2006] authors show that tags that identify qualities or characteristics can be effective in recommendation systems, being interested in clustering tags according to their meaning, we disregarded such kind of tags (i.e., those that express emotions or feelings). At the end of this step, the system involves 964 tags.

Feedback Collection

During this step, each volunteer performed 300 searches in the tagging systems. For each search, each volunteer: (i) entered a list of tags as query for the search; and (ii) selected, from the videos in the results list, the video most related with the query.

A feedback is then collected each time a user performed a search and consequently tag-resource counters are incremented. After entering a list of tags, she/he was free to analyze the videos resulting from the search (during this phase the user could also play all the videos to help her/his choice). At the end of this activity, the user had to pick a video from the output list providing a feedback. This emulates a real world scenario in which a user, after the result of a search is displayed, selects the resources she/he is interested in.

7.4.2 Benchmark algorithm description

The technique selected for comparison with RATC, i.e., the one proposed by [Begelman et al., 2006], hereinafter ATC.

ATC is aimed at clustering tags to improve user experience in the use of a tagging system and minimize the classical linguistic limitations. The approach defines an algorithm to find strongly related tags counting the number of tag co-occurrences used for a page. A cut-off point is determined to evaluate when a counter is useful. A spars matrix is produced and its elements are the similarities among tags.

A graph representation of the similarities is defined and the tags are grouped with a graph clustering algorithm based on the spectral bisection. The quality of the partitioning is measured with the "modularity function" Q [Newman and Girvan, 2004]. ATC performs the following steps: (i) it uses spectral bisection to split the graph into two clusters; (ii) it compares the value of the modularity function Q_0 of the original unpartitioned graph to the value of the modularity function Q_1 of the partitioned graph, if $Q_1 > Q_0$ accepts the partitioning, otherwise rejects the partitioning; and (iii) it proceeds recursively on each accepted partition.

A similarity counter is increased for each pair of tags that belong to the same cluster and the top similar pairs of tags are extracted.

The choice to compare RATC with this approach is motivated by the fact that both approaches use tag-resource counters to define associations among tags. Let us also note that the main difference is on the way the counter is incremented. In fact, as previously explained, RATC counter is incremented also during the search activity.

7.4.3 Evaluation Measures

To assess the ability of RATC to learn from users activity monitoring, the state of the tagging system (i.e. the current values of each tag-resource association) has been saved and used to evaluate clusters quality each 50 feedbacks. In this way, 6 tagging system sessions, which can be used to compare the two tag clustering approaches, are available. As already pointed out, a subset of known tags was added to the tagging system to create some verifiable noise. To evaluate the quality of the clusters created by each algorithm in presence of noise we conducted experiments considering both the original dataset and a dataset in which we removed the noisy tags. The only parameter that had to be set is the inflation value in the clustering step (set to 3.0).

To make fair comparisons, first, a domain engineer clustered the involved tags. Each cluster was created considering tags that refer to the same *concept*, i.e. a particular event or a clear 'topic' that groups tags. Subsequently, the tag clustering obtained by the domain engineer is compared with the clusters automatically generated by using RATC and the ones obtained by applying ATC. Each cluster produced by both RATC and

ATC was evaluated considering the most related cluster generated by the domain engineer and producing the following sets:

- *true positive tags* (TP): tags that appear both in a cluster generated by RATC (ATC) and in the cluster of the domain engineer partition.
- *true negative tags* (TN): tags that do not appear both in a cluster generated by RATC (ATC) and in the cluster of the domain engineer partition.
- *false positive tags* (FP): tags that appear in a cluster generated by RATC (ATC) and do not appear in the cluster of the domain engineer partition.
- *false negative tags* (FN): tags that do not appear in a cluster generated by RATC (ATC), but appear in the cluster of the domain engineer partition.

To validate the approach, we resort to classical information retrieval measures, such as micro- and macro-averaging of precision and recall [Sebastiani, 2002]. Let us recall here that micro- and macro-averaging are aimed at obtaining estimates of π and ρ relative to the whole category set. In particular, micro-averaging evaluates the overall π and ρ by globally summing over all individual decisions. In symbols:

$$\pi^{\mu} = \frac{TP}{TP + FP}; \quad \rho^{\mu} = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m}(TP_i + FN_i)} \tag{7.1}$$

where the "μ" superscript stands for microaveraging. On the other hand, macro-averaging first evaluates π and ρ "locally" for each category, and then "globally" by averaging over the results of the different categories. In symbols:

$$\pi^M = \frac{\sum_{i=1} mP_i}{m}; \qquad rho^M = \frac{\sum_{i=1} mP_i}{m} \tag{7.2}$$

where the "M" superscript stands for macroaveraging.

7.4.4 Results

Fig. 7.3 compares the results in terms of macro-averaging precision (Fig. 7.3-a) and recall (Fig. 7.3-b) obtained by adopting RATC and ATC with and without noisy tags. Fig. 7.4 compares the results in terms of micro-averaging precision (Fig. 7.4-a) and recall (Fig. 7.4-b) obtained by adopting RATC and ATC with and without noisy tags. Results show that RATC performs always better than ATC, and that such performances improve session by session, due to the fact that tag-resource associations and tag-tag associations get better with the use of the system (i.e., by applying the feedback mechanism).

It is worth to put into evidence that, in the first session, the tag-resource associations have the same values for both algorithms, as no search activity was done in the system. Considering that RATC achieves better results even in this session, it can be stated that cosine similarity represents a better way to measure associations among tags.

In order to measure the robustness of each system, i.e., the degree to which it can function correctly in presence of noise, the worsening of micro- and macro-averaging of precision and recall are measured. Micro-averaging worsening for RATC is between 7% and 9%, while worsening for ATC is 29.19%. Macro-averaging worsening for RATC is always around 8%, while worsening for RATC is 22.57%. That means that RATC is more robust thsan ATC, since it is less affected by the presence of noise.

7.5 Conclusions

This chapter presented a technique able to cluster tags in a tagging system, with the ability to dynamically improve its performances while the tagging system is being used. The algorithm monitors users activity and exploits implicit feedbacks left by users. Experimental results highlight the effectiveness of the approach in the presence of strongly related tags in a cluster.

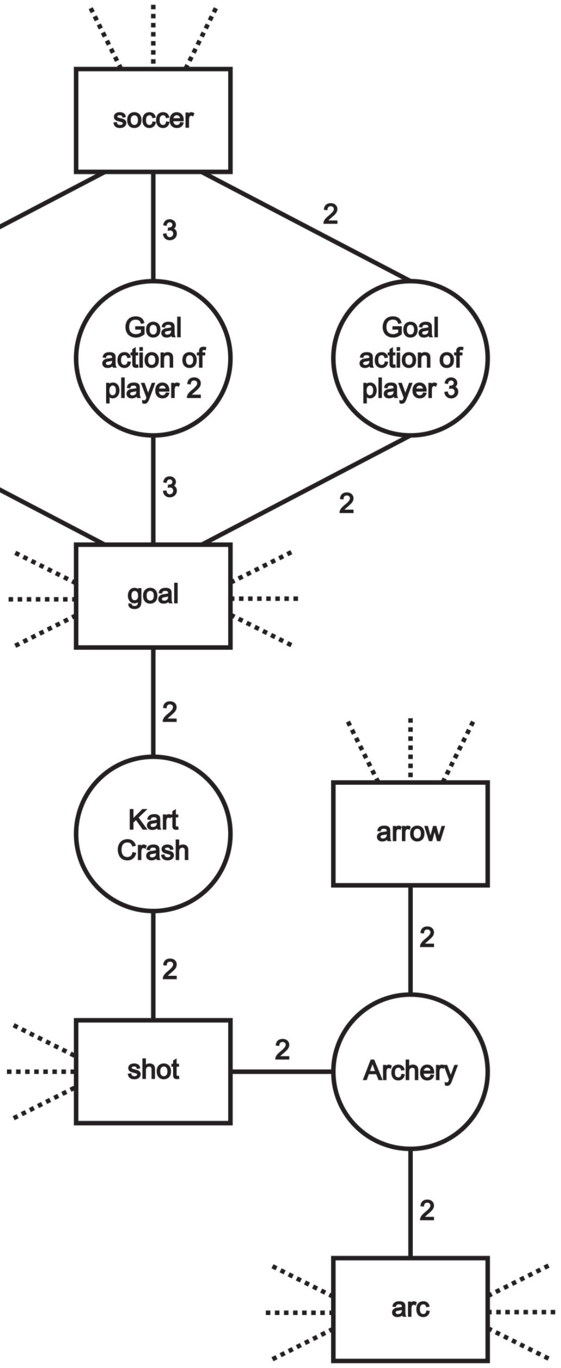


Figure 7.1: A tagging system example

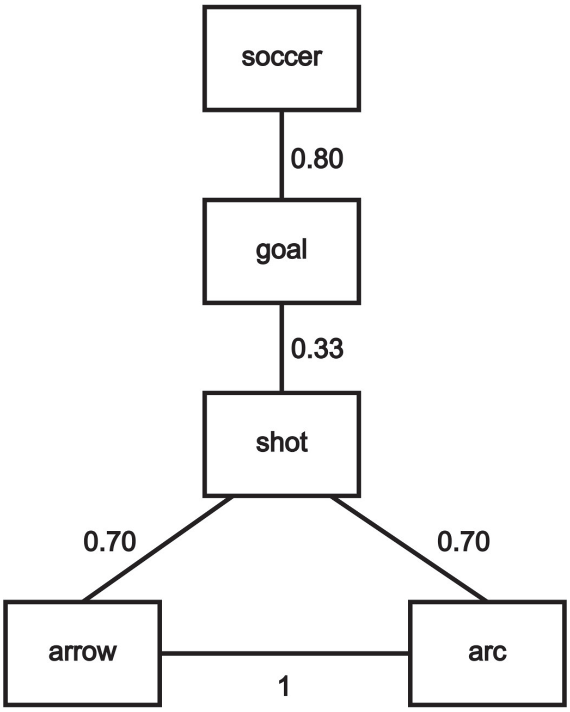
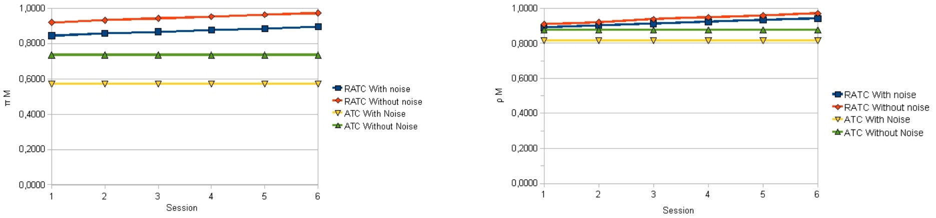


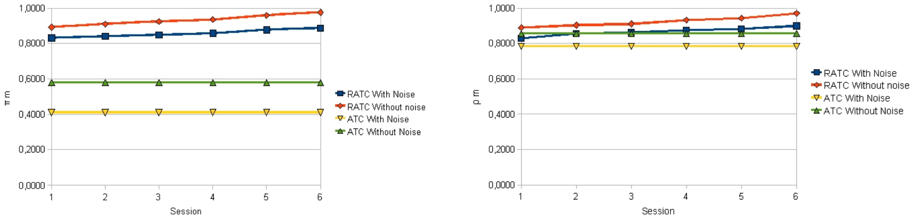
Figure 7.2: Similarities graph



(a)

(b)

Figure 7.3: Macro-averaging precision (a) and recall (b)



(a)

(b)

Figure 7.4: Micro-averaging precision (a) and recall (b)

Chapter 8

Conclusions

This PhD thesis focused on group recommendation algorithm that automatically detect groups of users with similar preferences, in contexts in which the number of recommendations that can be provided is limited.

8.1 Contributions

Specifically, the contribution of this PhD thesis are summarized as follows.

- Study of the approaches to aggregate individual preferences and generate group recommendations and identification of the approach the works best in a scenario in which groups are automatically detected.
- Study of the strategies to model groups, in order to find the strategy that works best with automatically detected groups.

- Study of the per-group effectiveness of an algorithm, in order to understand which properties of a group affect the quality of a group recommendation algorithm.
- Study of how the novelty of the recommended content affects the quality of a group recommendation algorithm.
- A technique to automatically segment markets based on query logs.
- A tag clustering technique to simplify the exploration of a tagging system.

Chapter 3 appeared in [Boratto and Carta, 2010b]. The algorithm presented in Chapter 3 as *MART* was presented in [Boratto et al., 2009a]. The algorithms presented as *BART* and *HEART* in Chapter 3 were proposed in [Boratto et al., 2010b].

Chapter 5 appeared in [Boratto et al., 2010a] and Chapter 7 in [Boratto et al., 2009b]. Algorithms *SMART* and *APART*, the vast majority of the experiments in Chapter 4 and the technique proposed in Chapter 6 are all novel contributions developed for this thesis.

Bibliography

[Amatriain et al., 2011] Amatriain, X., Jaimes, A., Oliver, N., and Pujol, J. M. (2011). Data mining methods for recommender systems. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 39–71. Springer.

[Amer-Yahia et al., 2009] Amer-Yahia, S., Roy, S. B., Chawla, A., Das, G., and Yu, C. (2009). Group recommendation: Semantics and efficiency. *PVLDB*, 2(1):754–765.

[Ardissono et al., 2005] Ardissono, L., Goy, A., Petrone, G., and Segnan, M. (2005). A multi-agent infrastructure for developing personalized web-based systems. *ACM Trans. Internet Technol.*, 5(1):47–69.

[Ardissono et al., 2003] Ardissono, L., Goy, A., Petrone, G., Segnan, M., and Torasso, P. (2003). Intrigue: Personalized recommendation of tourist attractions for desktop and handset devices. *Applied Artificial Intelligence*, 17(8):687–714.

[Baeza-Yates, 2005] Baeza-Yates, R. (2005). *Advances in Information Retrieval*, chapter Applications of Web Query Mining, pages 7–22. Springer Verlag.

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley.

[Baskin and Krishnamurthi, 2009] Baskin, J. P. and Krishnamurthi, S. (2009). Preference aggregation in group recommender systems for committee decision-making. In [Bergman et al., 2009], pages 337–340.

[Begelman et al., 2006] Begelman, G., Keller, P., and Smadja, F. (2006). Automated tag clustering: Improving search and exploration in the tag space. In *Proceedings of the Collaborative Web Tagging Workshop at the WWW 2006*.

[Bergman et al., 2009] Bergman, L. D., Tuzhilin, A., Burke, R. D., Felfernig, A., and Schmidt-Thieme, L., editors (2009). *Proceedings of the 2009 ACM Conference on Recommender Systems, RecSys 2009, New York, NY, USA, October 23-25, 2009*. ACM.

[Bielenberg and Zacher, 2005] Bielenberg, K. and Zacher, M. (2005). Groups in social software: Utilizing tagging to integrate individual contexts for social navigation.

[Bigdeli and Bahmani, 2008] Bigdeli, E. and Bahmani, Z. (2008). Comparing accuracy of cosine-based similarity and correlation-based similarity algorithms in tourism recommender systems. In *Management of Innovation and Technology, 2008. ICMIT 2008. 4th IEEE International Conference on*, pages 469–474. IEEE.

[Blondel et al., 2008] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of community hierarchies in large networks. *CoRR*, abs/0803.0476.

[Boratto and Carta, 2010a] Boratto, L. and Carta, S. (2010a). State-of-the-art in group recommendation and new approaches for automatic identification of groups. In Alessandro Soro, Eloisa Vargiu, G. A. and Paddeu, G., editors, *Information Retrieval and Mining in Distributed Environments*. Springer Verlag. In press.

[Boratto and Carta, 2010b] Boratto, L. and Carta, S. (2010b). State-of-the-art in group recommendation and new approaches for automatic identification of groups. In Alessandro Soro, Eloisa Vargiu, G. A. and Paddeu, G., editors, *Information Retrieval and Mining in Distributed Environments*. Springer Verlag. In press.

[Boratto et al., 2009a] Boratto, L., Carta, S., Chessa, A., Agelli, M., and Clemente, M. L. (2009a). Group recommendation with automatic identification of users communities. In *Proceedings of the 2009 IEEE/WIC/ACM*

International Conference on Web Intelligence and International Conference on Intelligent Agent Technology - Workshops, Milan, Italy, 15-18 September 2009, pages 547–550. IEEE.

[Boratto et al., 2010a] Boratto, L., Carta, S., and Satta, M. (2010a). Content novelty and group recommendation. In *Proceedings of the Fourth International Workshop on Distributed Agent-based Retrieval Tools*, pages 367–380, Webster University - Geneve, Switzerland.

[Boratto et al., 2010b] Boratto, L., Carta, S., and Satta, M. (2010b). Groups identification and individual recommendations in group recommendation algorithms. In Picault, J., Kostadinov, D., Castells, P., and Jaimes, A., editors, *Practical Use of Recommender Systems, Algorithms and Technologies 2010*, volume 676 of *CEUR Workshop Proceedings*.

[Boratto et al., 2009b] Boratto, L., Carta, S., and Vargiu, E. (2009b). Ratc: A robust automated tag clustering technique. In Noia, T. D. and Buccafurri, F., editors, *E-Commerce and Web Technologies, 10th International Conference, EC-Web 2009, Linz, Austria, September 1-4, 2009. Proceedings*, volume 5692 of *Lecture Notes in Computer Science*, pages 324–335. Springer.

[Briggs and Smyth, 2005] Briggs, P. and Smyth, B. (2005). Modeling trust in collaborative web search. In *AICS*, Coleraine, NI.

[Burke, 2007] Burke, R. D. (2007). Hybrid web recommender systems. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web,* *Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 377–408. Springer.

[Cantador et al., 2008] Cantador, I., Castells, P., and Superior, E. P. (2008). Extracting multilayered semantic communities of interest from ontology-based user profiles: Application to group modelling and hybrid recommendations. In *In: Computers in Human Behavior, special issue on Advances of Knowledge Management and the Semantic*. Elsevier. In press.

[Carolis and Pizzutilo, 2009] Carolis, B. D. and Pizzutilo, S. (2009). Providing relevant background information in smart environments. In [Noia and Buccafurri, 2009], pages 360–371.

[Carta et al., 2008] Carta, S., Alimonda, A., Clemente, M., and Agelli, M. (2008). Glue: Improving tag-based contents retrieval exploiting implicit user feedback. In Hoenkamp, E., de Cock, M., and Hoste, V., editors, *Proceedings Of The 8th Dutch-Belgian Information Retrieval Workshop (DIR 2008)*, pages 29–35.

[Chao et al., 2005] Chao, D. L., Balthrop, J., and Forrest, S. (2005). Adaptive radio: achieving consensus using negative preferences. In Pendergast, M., Schmidt, K., Mark, G., and Ackerman, M., editors, *GROUP*, pages 120–123. ACM.

[Chen et al., 2008] Chen, Y.-L., Cheng, L.-C., and Chuang, C.-N. (2008). A group recommendation system with consideration of interactions among group members. *Expert Syst. Appl.*, 34(3):2082–2090.

[Coyle and Smyth, 2005] Coyle, M. and Smyth, B. (2005). Explaining search results. In [Kaelbling and Saffiotti, 2005], pages 1553–1555.

[Crossen et al., 2002] Crossen, A., Budzik, J., and Hammond, K. J. (2002). Flytrap: intelligent group music recommendation. In *IUI*, pages 184–185.

[de Campos et al., 2007] de Campos, L. M., Fernández-Luna, J. M., Huete, J. F., and Rueda-Morales, M. A. (2007). Group recommending: A methodological approach based on bayesian networks. In *ICDE Workshops*, pages 835–844. IEEE Computer Society.

[de Campos et al., 2009] de Campos, L. M., Fernández-Luna, J. M., Huete, J. F., and Rueda-Morales, M. A. (2009). Managing uncertainty in group recommending processes. *User Model. User-Adapt. Interact.*, 19(3):207–242.

[Desrosiers and Karypis, 2011] Desrosiers, C. and Karypis, G. (2011). A comprehensive survey of neighborhood-based recommendation methods. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 107–144. Springer.

[Dhillon, 2001] Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, New York, NY, USA. ACM.

[Dietz and Leigh, 2001] Dietz, P. H. and Leigh, D. (2001). Diamondtouch: a multi-user touch technology. In *UIST*, pages 219–226.

[Fortunato, 2009] Fortunato, S. (2009). Community detection in graphs. *CoRR*, abs/0906.0612.

[Fortunato and Castellano, 2007] Fortunato, S. and Castellano, C. (2007). Community structure in graphs. *Springer's Encyclopedia of Complexity and System Science*.

[Freyne and Smyth, 2006] Freyne, J. and Smyth, B. (2006). B.: Cooperating search communities. In *Proc. of 4th International Conference on Adaptive Hypermedia and Adaptive WebBased Systems (AH'2006). Lecture Notes in Computer Science*, pages 101–111. Springer Verlag.

[Garcia et al., 2009] Garcia, I., Sebastia, L., Onaindia, E., and Guzman, C. (2009). A group recommender system for tourist activities. In [Noia and Buccafurri, 2009], pages 26–37.

[Gfeller et al., 2005] Gfeller, D., Chappelier, J. C., and Los, D. (2005). Finding instabilities in the community structure of complex networks. *Physical Review E*, 72(5 Pt 2):056135+.

[Giannakidou et al., 2008] Giannakidou, E., Koutsonikola, V., Vakali, A., and Kompatsiaris, Y. (2008). Co-clustering tags and social data sources. *Web-Age Information Management, 2008. WAIM '08. The Ninth International Conference on*, pages 317–324.

[Golder and Huberman, 2006] Golder, S. A. and Huberman, B. A. (2006). Usage patterns of collaborative tagging systems. *J. Information Science*, 32(2):198–208.

[Goren-Bar and Glinansky, 2004] Goren-Bar, D. and Glinansky, O. (2004). Fit-recommend ing tv programs to family members. *Computers & Graphics*, 28(2):149–156.

[Hamasaki et al., 2008] Hamasaki, M., Matsuo, Y., Nishimura, T., and Takeda, H. (2008). Ontology extraction by collaborative tagging with social networking. In *WWW2008, April, Beijing, China.*

[Herlocker et al., 1999] Herlocker, J., Konstan, J., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Research and Development in Information Retrieval*. American Association of Computing Machinery, American Association of Computing Machinery.

[Hill et al., 1995] Hill, W. C., Stead, L., Rosenstein, M., and Furnas, G. W. (1995). Recommending and evaluating choices in a virtual community of use. In Katz, I. R., Mack, R. L., Marks, L., Rosson, M. B., and Nielsen, J., editors, *Human Factors in Computing Systems, CHI '95 Conference Proceedings, Denver, Colorado, USA, May 7-11, 1995*, pages 194–201. ACM/Addison-Wesley.

[Jameson, 2004] Jameson, A. (2004). More than the sum of its members: Challenges for group recommender systems. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 48–54, Gallipoli, Italy. Available from http://dfki.de/~jameson/abs/Jameson04AVI.html.

[Jameson et al., 2003] Jameson, A., Baldes, S., and Kleinbauer, T. (2003). Enhancing mutual awareness in group recommender systems. In Mobasher, B. and Anand, S. S., editors, *Proceedings of the IJCAI 2003 Workshop on Intelligent Techniques for Web Personalization*. AAAI, Menlo Park, CA. Available from http://dfki.de/~jameson/abs/JamesonBK03ITWP.html.

[Jameson et al., 2004] Jameson, A., Baldes, S., and Kleinbauer, T. (2004). Two methods for enhancing mutual awareness in a group recommender system. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, Gallipoli, Italy. In press.

[Jameson and Smyth, 2007a] Jameson, A. and Smyth, B. (2007a). Recommendation to groups. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 596–627. Springer.

[Jameson and Smyth, 2007b] Jameson, A. and Smyth, B. (2007b). Recommendation to groups. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 596–627. Springer.

[Kaelbling and Saffiotti, 2005] Kaelbling, L. P. and Saffiotti, A., editors (2005). *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*. Professional Book Center.

[Kanungo et al., 2002] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:881–892.

[Kim et al., 2009] Kim, J. K., Kim, H. K., Oh, H. Y., and Ryu, Y. U. (2009). A group recommendation system for online communities. *International Journal of Information Management*, In Press, Corrected Proof:–.

[Kleinberg and Tardos, 2002] Kleinberg, J. M. and Tardos, É. (2002). Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. *J. ACM*, 49(5):616–639.

[Lieberman et al., 1999] Lieberman, H., Dyke, N. W. V., and Vivacqua, A. S. (1999). Let's browse: A collaborative web browsing agent. In *IUI*, pages 65–68.

[Linden et al., 2003] Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80.

[Lorenzi et al., 2008] Lorenzi, F., Santos, F., Ferreira, Jr., P. R., and Bazzan, A. L. (2008). Optimizing preferences within groups: A case study on travel recommendation. In *SBIA '08: Proceedings of the 19th Brazilian Symposium on Artificial Intelligence*, pages 103–112, Berlin, Heidelberg. Springer-Verlag.

[MacQueen, 1967] MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press.

[Masthoff, 2002] Masthoff, J. (2002). Modeling a group of television viewers. In *Proceedings of the Future tv: Adaptive instruction in your living room workshop*, pages 34–42.

[Masthoff, 2004] Masthoff, J. (2004). Group modeling: Selecting a sequence of television items to suit a group of viewers. *User Model. User-Adapt. Interact.*, 14(1):37–85.

[Masthoff, 2005] Masthoff, J. (2005). The pursuit of satisfaction: Affective state in group recommender systems. In Ardissono, L., Brna, P., and Mitrovic, A., editors, *User Modeling 2005, 10th International Conference, UM 2005, Edinburgh, Scotland, UK, July 24-29, 2005, Proceedings*, volume 3538 of *Lecture Notes in Computer Science*, pages 297–306. Springer.

[Masthoff, 2011] Masthoff, J. (2011). Group recommender systems: Combining individual models. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 677–702. Springer.

[McCarthy, 2002] McCarthy, J. F. (2002). Pocket restaurantfinder: A situated recommender system for groups. In *Workshop on Mobile Ad-Hoc Communication at the 2002 ACM Conference on Human Factors in Computer Systems*, Minneapolis.

[McCarthy and Anagnost, 2000] McCarthy, J. F. and Anagnost, T. D. (2000). Musicfx: an arbiter of group preferences for computer supported collaborative workouts. In *CSCW*, page 348.

[McCarthy et al., 2007] McCarthy, K., McGinty, L., and Smyth, B. (2007). Case-based group recommendation: Compromising for success. In Weber, R. and Richter, M. M., editors, *ICCBR*, volume 4626 of *Lecture Notes in Computer Science*, pages 299–313. Springer.

[McCarthy et al., 2006] McCarthy, K., McGinty, L., Smyth, B., and Salamó, M. (2006). The needs of the many: A case-based group recommender system. In Roth-Berghofer, T., Göker, M. H., and Güvenir, H. A., editors, *ECCBR*, volume 4106 of *Lecture Notes in Computer Science*, pages 196–210. Springer.

[Mccarthy et al., 2006] Mccarthy, K., Mcginty, L., Smyth, B., and Salamo, M. (2006). Social interaction in the cats group recommender. In Brusilovsky, P., Dron, J., and Kurhila, J., editors, *Workshop on the Social Navigation and Community-Based Adaptation Technologies at the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*.

[McCarthy et al., 2006a] McCarthy, K., Salamó, M., Coyle, L., McGinty, L., Smyth, B., and Nixon, P. (2006a). Cats: A synchronous approach to collaborative group recommendation. In Sutcliffe, G. and Goebel, R., editors, *FLAIRS Conference*, pages 86–91. AAAI Press.

[McCarthy et al., 2006b] McCarthy, K., Salamó, M., Coyle, L., McGinty, L., Smyth, B., and Nixon, P. (2006b). Group recommender systems: a critiquing based approach. In Paris, C. and Sidner, C. L., editors, *IUI*, pages 267–269. ACM.

[Newman and Girvan, 2004] Newman, M. E. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Phys Rev E Stat Nonlin Soft Matter Phys*, 69(2 Pt 2).

[Newman, 2004] Newman, M. E. J. (2004). Analysis of weighted networks. *Phys. Rev. E*, 70(5):056131.

[Noia and Buccafurri, 2009] Noia, T. D. and Buccafurri, F., editors (2009). *E-Commerce and Web Technologies, 10th International Conference, EC-Web 2009, Linz, Austria, September 1-4, 2009. Proceedings*, volume 5692 of *Lecture Notes in Computer Science*. Springer.

[O'Connor et al., 2001] O'Connor, M., Cosley, D., Konstan, J. A., and Riedl, J. (2001). Polylens: a recommender system for groups of users. In *ECSCW'01: Proceedings of the seventh conference on European Conference on Computer Supported Cooperative Work*, pages 199–218, Norwell, MA, USA. Kluwer Academic Publishers.

[O'Hara et al., 2004] O'Hara, K., Lipson, M., Jansen, M., Unger, A., Jeffries, H., and Macer, P. (2004). Jukola: democratic music choice in a public space. In *DIS '04: Proceedings of the 5th conference on Designing interactive systems*, pages 145–154, New York, NY, USA. ACM.

[Pizzutilo et al., 2005a] Pizzutilo, S., Carolis, B. D., Cozzolongo, G., and Ambruoso, F. (2005a). Group modeling in a public space: Methods, techniques and experiences. In *Proceedings of WSEAS AIC 05*, Malta. ACM.

[Pizzutilo et al., 2005b] Pizzutilo, S., De Carolis, B., Cozzolongo, G., and Ambruoso, F. (2005b). Group modeling in a public space: methods, techniques, experiences. In *AIC'05: Proceedings of the 5th WSEAS International Conference on Applied Informatics and Communications*, pages 175–180, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS).

[Porter et al., 2009] Porter, M. A., Onnela, J.-P., and Mucha, P. J. (2009). Communities in networks. *ArXiv*.

[Recio-García et al., 2009] Recio-García, J. A., Jiménez-Díaz, G., Sánchez-Ruiz-Granados, A. A., and Díaz-Agudo, B. (2009). Personality aware recommendations to groups. In [Bergman et al., 2009], pages 325–328.

[Resnick et al., 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *CSCW*, pages 175–186.

[Resnick and Varian, 1997] Resnick, P. and Varian, H. R. (1997). Recommender systems - introduction to the special section. *Commun. ACM*, 40(3):56–58.

[Ricci et al., 2011] Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to recommender systems handbook. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 1–35. Springer.

[Schaeffer, 2007] Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, 1(1):27–64.

[Schafer et al., 2007] Schafer, J. B., Frankowski, D., Herlocker, J. L., and Sen, S. (2007). Collaborative filtering recommender systems. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 291–324. Springer.

[Sebastiani, 2002] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–55.

[Shardanand and Maes, 1995] Shardanand, U. and Maes, P. (1995). Social information filtering: Algorithms for automating ″word of mouth″. In Katz, I. R., Mack, R. L., Marks, L., Rosson, M. B., and Nielsen, J., editors, *Human Factors in Computing Systems, CHI '95 Conference Proceedings, Denver, Colorado, USA, May 7-11, 1995*, pages 210–217. ACM/Addison-Wesley.

[Sharon et al., 2003] Sharon, T., Lieberman, H., and Selker, T. (2003). A zero-input interface for leveraging group experience in web browsing. In *IUI*, pages 290–292. ACM.

[Smyth and Balfe, 2006] Smyth, B. and Balfe, E. (2006). Anonymous personalization in collaborative web search. *Inf. Retr.*, 9(2):165–190.

[Smyth et al., 2005] Smyth, B., Balfe, E., Boydell, O., Bradley, K., Briggs, P., Coyle, M., and Freyne, J. (2005). A live-user evaluation of collaborative web search. In [Kaelbling and Saffiotti, 2005], pages 1419–1424.

[Smyth et al., 2003a] Smyth, B., Balfe, E., Briggs, P., Coyle, M., and Freyne, J. (2003a). Collaborative web search. In Gottlob, G. and Walsh, T., editors, *IJCAI*, pages 1417–1419. Morgan Kaufmann.

[Smyth et al., 2003b] Smyth, B., Freyne, J., Coyle, M., Briggs, P., and Balfe, E. (2003b). I-spy-anonymous, community-based personalization by collaborative meta-search. In *Proceedings of the 23rd SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 367–380. Citeseer.

[Smyth et al., 2003c] Smyth, B., Freyne, J., Coyle, M., Briggs, P., and Balfe, E. (2003c). I-SPY: Anonymous, Community-Based Personalization by Collaborative Web Search. In *Proceedings of the 23rd SGAI International Conference on Innovative Techniques*, pages 367–380. Springer. Cambridge, UK.

[Specia and Motta, 2007] Specia, L. and Motta, E. (2007). Integrating folksonomies with the semantic web. pages 624–639.

[Sprague et al., 2008] Sprague, D., Wu, F., and Tory, M. (2008). Music selection using the partyvote democratic jukebox. In *AVI '08: Proceedings of the working conference on Advanced visual interfaces*, pages 433–436, New York, NY, USA. ACM.

[van Dongen, 2000] van Dongen, S. (2000). *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht.

[Vildjiounaite et al., 2009] Vildjiounaite, E., Kyllönen, V., Hannula, T., and Alahuhta, P. (2009). Unobtrusive dynamic modelling of tv programme preferences in a finnish household. *Multimedia Syst.*, 15(3):143–157.

[Wu et al., 2006] Wu, X., Zhang, L., and Yu, Y. (2006). Exploring social annotations for the semantic web. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 417–426, New York, NY, USA. ACM.

[Xu and II, 2005] Xu, R. and II, D. C. W. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678.

[Yu et al., 2006] Yu, Z., Zhou, X., Hao, Y., and Gu, J. (2006). Tv program recommendation for multiple viewers based on user profile merging. *User Model. User-Adapt. Interact.*, 16(1):63–82.

[Zhiwen et al., 2005] Zhiwen, Y., Xingshe, Z., and Daqing, Z. (2005). An adaptive in-vehicle multimedia recommender for group users. In *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, volume 5, pages 2800–2804 Vol. 5.