EXPLOITING ITEM CO-UTILITY TO IMPROVE RECOMMENDATIONS

ALINE BESSA

EXPLOITING ITEM CO-UTILITY TO IMPROVE RECOMMENDATIONS

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Nivio Ziviani
Co-Advisor: Adriano Veloso

Belo Horizonte
February 18, 2014

Bessa, Aline Duarte

B557e Exploiting item co-utility to improve recommendations / Aline Duarte Bessa.— Belo Horizonte, 2013.
xiii, 49 f.: il.; 29 cm.

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.

Orientador: Nívio Ziviani.
Coorientador: Adriano Alonso Veloso.

1. Computação - Teses. 2. Sistemas de recomendação
– Teses. 3. Recuperação da informação – Teses.
I..Orientador. II. Coorientador. III. Título.

CDU 519.6*73
(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Exploiting item co-utility to improve recommendations

ALINE DUARTE BESSA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

[illegible]

PROF. NIVIO ZIVIANI - Orientador
Departamento de Ciência da Computação - UFMG

[illegible]

PROF. ADRIANO ALONSO VELOSO - COORIENTADOR
Departamento de Ciência da Computação - UFMG

[illegible]

PROF. BERTHIER RIBEIRO DE ARAÚJO NETO
Departamento de Ciência da Computação - UFMG

[illegible]

PROF. RODRYGO LUIS TEODORO SANTOS
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 18 de fevereiro de 2014.

To Raja and Lico.

Acknowledgments

First to my advisor, Nivio Ziviani, who supported me throughout this process. I would also like to especially thank Adriano Veloso, for being a dedicated co-advisor. Looking back over the past two years I have learned undeniably much, thanks to these two professors in particular, but also to UFMG in general. I also thank professor Juliana Freire from NYU-Poly, who held an advisory role for me during my internship in 2013 and substantially changed my viewpoints on how to perform research.

Most of this thesis would not have been possible without the fruitful suggestions of my colleagues. First Filipe Arcanjo, who helped me on understanding the relevance of what I was studying, and on whom I bounced ideas since my first months in Belo Horizonte. Special thanks to Fernando Mourão, who have given me substantial ideas for experiments and validation. I have learned much from my peers at LATIN: Aécio Santos, Adolfo Guimarães, Wladmir Brandão, Thales Costa, Cristiano Carvalho, Arthur Camara, Anisio Lacerda, Sabir Ribas, and Leonardo Resende; and at NYU-Poly: Fernando Chirigati, Tuan-Anh, and Kien Pham. I also would like to thank CNPQ and CAPES for funding my research.

If not for friends, I would not have had enough motivation to go through my master's. I would like to thank Denise Neri, Fernanda Wanderley, Roy Hopper, and Anthony Farnham for supporting me in so many ways during my time in Belo Horizonte and New York City. To my old yet always present friends, Clara Fernandes, Julianna Oliveira, Amine Portugal, Renata Amoedo, Rafael Saraiva, Alexandre Passos, thanks for hitting me up constantly through Skype and e-mails, you are all very important to me.

I also thank my family, Marcia Bessa, Andrea Bessa, and Adriano Bessa, who have inspired me and taught me how to stay strong even when I think I cannot.

And last, but not least, I want to thank Davi, my love and best friend, for reviewing this work and giving me invaluable advice and support all the time.

"Life is the sum of all your choices."
(Albert Camus)

Abstract

In this thesis, we consider that a recommendation was useful if the associated user's feedback was positive – e.g., the user purchased the recommendation, gave it a high rating, or clicked on it. We then formalize the concept of co-utility, stated as the property any two items have of being useful to a user, and exploit it to improve recommendations. We then present different ways of estimating co-utility probabilities, all of them independent of content information, and compare them with each other. We embed these probabilities, as well as normalized predicted ratings, in an instance of an $\mathcal{NP} - hard$ problem named Max-Sum Dispersion Problem. A solution to this problem corresponds to a set of items for recommendation. We study two heuristics and one exact solution to the Max-Sum Dispersion Problem and perform comparisons among them. According to our experiments, the three solutions have similar performance in practice. We also contrast our method to different baselines by comparing the ratings users give to different recommendations. We obtain expressive gains in the utility of recommendations, up to 106%, and our method also recommends higher rated items to the majority of users. Finally, we show that our method is scalable in practice and does not seem to affect recommendations' diversity.

Keywords: Recommender Systems, Co-Utility, Max-Sum Dispersion Problem.

Contents

Chapter 1

Introduction

People from widely varying backgrounds are inundated with options that lead to a situation known as "information overload", where the presence of too much information interferes with decision-making processes [Toffler, 1970]. To circumvent it, content providers and electronic retailers have to identify a small yet effective amount of information that matches users' expectations. In this scenario, Recommender Systems have become tools of paramount importance, providing a few personalized recommendations that intend to suit user needs in a satisfactory way.

One type of such systems, known as Collaborative Filtering, makes predictions about the interests of a user by gathering taste information from many other users. It generally works as follows: (i) prediction step - keeps track of users' known preferences and processes them to predict items that may be interesting to other users; (ii) recommendation step - selects predicted items, optionally ranks them, and recommends them to users [Adomavicius and Tuzhilin, 2005]. The scope of this work is limited to Collaborative Filtering. We also do not perform ranking, and therefore do not analyse the impact that the order of items may have on recommendations.

In the prediction step, scores are independently assigned to items by taking user's historical data into account [Ricci et al., 2011]. The higher the score the higher the estimated compatibility between the item and user's known preferences. It is therefore intuitive to think that the highest scored items should be the ones selected in the recommendation step. This is indeed what happens in the recommendation step of many systems, where the selection of items is based exclusively on how well they match users' known preferences. Nonetheless, by neglecting relations between predicted items, such systems may generate less useful recommendation lists. A large body of research, dating back from early studies in the 1960s, draws attention to the importance of exploiting dependencies between items [Bookstein, 1983; Carbonell and Goldstein, 1998].

In this work, we focus on improving the utility of recommendations by exploiting relations between items. In our case, the relation in question is co-utility, and other examples of item relations are diversity or competition [Zhang and Hurley, 2008; Xiong et al., 2012]. Throughout this work, we consider that a recommendation was useful if the associated user's feedback was positive – e.g., she purchased the recommendation, gave it a high rating, or clicked on it. Breese et al. [1998] and Passos et al. [2011] also relate utility to positive feedback – high ratings, in particular. With respect to relations between items, we focus on their co-utility, a concept that is defined in the following.

Definition 1. Two items are co-useful with respect to a user if she considered both of them useful. Co-utility is the property of being co-useful.

For each pair of items, we compute their probabilities of being co-useful, as detailed in Chapter 3, and embed this information into methods designed to generate recommendations. As we explain in Chapter 2, other works exploit relations between items for recommendations, especially when they account for diversity. Nonetheless, to the best of our knowledge, our work is the first to address co-utility relations in the context of items' recommendation.

This work is motivated by the Theory of Choice of Amos Tversky [Tversky, 1972], which indicates that preference among items depends not only on the items' specific features, but also on the presented alternatives. In our case, the selection of an item is based on its independently predicted rating and on how likely it is to be co-useful with other selected items. The simplicity of estimating co-utility probabilities and the fact that they are underexploited in the literature to date were also important motivations behind this work.

1.1 Work Contributions

Some of the specific contributions of this work include:

- A definition of co-utility and methods for estimating co-utility probabilities. Such methods are time efficient and do not depend on content information.
- An optimization objective function that combines prediction values and co-utility probabilities, its reduction to a popular Facility Location Analysis problem [Borodin et al., 2012], and different algorithms to tackle it. We show heuristics and an exact method to optimize this objective function.

- A thorough evaluation of our method. We compare different algorithms that adopt co-utility probabilities with methods that neglect relations between items and with methods that take them into consideration. Our comparisons are mainly focused on recommendations' utility, albeit we briefly address diversity as well.

- An analysis of the scalability of our method, which indicates that it is applicable for real-world recommender systems.

1.2 Research Development

This informal description aims to help you, the reader, understand the trajectory of this work. We portray not only the choices that led to the theme of this dissertation but also our main difficulties.

This research started as a study on diversity in the context of recommender systems. The abundance of work in this area, alongside the fact that it is hard to understand to what extent diversity should be regarded [Pu et al., 2012], has motivated us to focus on other types of relation between items. The idea of modelling co-utility probabilities came up naturally, and after a thorough investigation we concluded that it was rather original in our context.

Initially, we combined co-utility probabilities and predicted scores in a bayesian fashion. The selection of an item for recommendation depended on its *prior probability* of being useful, and its *posterior probability* of being useful given the items that were already selected for recommendation. For example, the selection of the first item, i_1, depended on its *prior probability*; the selection of the second item, i_2, depended on its *prior probability* and on its *posterior probability* given that i_1 was selected etc. These prior probabilities were straightforward normalizations of predicted scores, and the posterior probabilities were computed as co-utility probabilities.

We then examined a form of mapping these prior and posterior probabilities into a bayesian network [Koller and Friedman, 2009]. We built a directed acyclic graph with nodes corresponding to the candidate items, each associated to its prior probability of selection, and edges corresponding to their co-utility probabilities. To determine which items should be selected, we had to run a Maximum a Posteriori (MAP) inference algorithm with a restriction on the size of the selected set, as we wanted the recommendation lists to be comprised by a specific number of items [Koller and Friedman, 2009]. An adequate option was HOP-MAP, an efficient message passing algorithm proposed by Tarlow et al. [2010]. Nonetheless, this model was still not practical enough due to slow convergence, and we were not convinced as to whether it would work in real-time situations.

Some time later, we noticed that we could linearly combine predicted scores and co-utility probabilities, pose the combination as an optimization problem, and then tackle it under an Operations Research paradigm. Our combination trivially reduced to a Facility Location Analysis problem named Max-Sum Dispersion Problem [Borodin et al., 2012]. We detail the connection between our work and Max-Sum Dispersion Problem in Chapters 2, 3, and 4.

After the definition of our optimization problem, we performed experiments to analyse if the exploitation of co-utility probabilities could improve recommendations' utility. Although we were no longer focusing on diversity, we did not want to generate redundant recommendations, so our experiments also analysed whether our recommendation lists were redundant. Finally, we studied the scalability of our method.

The results of these experiments, and comparisons with different baselines, were published as a full-paper in SPIRE 2013, namely Bessa et al. [2013]. By that time, co-utility probabilities bore the name of *mutual influence*, but after SPIRE we concluded that this term was not very accurate. After SPIRE 2013, we studied more algorithms to the Max-Sum Dispersion Problem, a different way of estimating co-utility probabilities, and ran more experiments. Throughout this entire research, we had a difficulty with the choice of baselines, as we explain in Chapter 2. Albeit some of them were recommended by SPIRE reviewers and other researchers, we still think it would be better if at least one of them addressed co-utility in a way that is similar to ours.

1.3 Work Outline

The remainder of this work is organized as follows.

Chapter 2 [Related Work] Related work is discussed and connected to our study.

Chapter 3 [Basic Concepts] Basic definitions, notations, and functions concerning our combination of predictions and co-utility probabilities are presented.

Chapter 4 [Algorithms and Validation] Algorithms to tackle the optimization problem detailed in Chapter 3, which are a key part of our method, are presented. We also detail baselines and our validation methodology.

Chapter 5 [Experimental Results] Experiments that demonstrate the efficiency and efficacy of the algorithms discussed in Chapter 4 are presented.

Chapter 6 [Conclusions and Future Work] Contributions are presented and conclusions, limitations, and future work are discussed.

Chapter 2

Related Work

In this chapter, we present works that are related to ours in different ways. Section 2.1 discusses rating predictors that are associated to state-of-the-art $Top - N$ recommendations. Section 2.2 summarizes works that exploit dependencies among items in the contexts of Information Retrieval and Recommender Systems. Section 2.3 addresses works that relate with our optimization task.

2.1 Predictors for $Top - N$ Recommendations

According to Cremonesi et al. [2010], $Top - N$ is a recommendation task where the "best bet" items are shown, but the predicted rating values are not. In this work, $Top - N$ stands for *the recommendation task where the items shown are the ones with the N highest predicted values*, as in Deshpande and Karypis [2004] and Cremonesi et al. [2010]. No relation among candidate items for recommendation is taken into account to select them. Consequently, their presences in a recommendation list are independent. Although $Top - N$ may refer to any method for recommending N items to users, we use this definition – which is rather popular – throughout this work.

Several predictors have been studied for collaborative filtering. They are grouped into two general classes: *memory-based* and *model-based* [Breese et al., 1998]. *Memory-based* predictors operate over the entire database to compute similarities between users or items, usually by applying distance metrics such as the cosine distance, and then come up with predictions. *Memory-based* predictors usually provide a more concise and intuitive justification for the computed predictions and are more stable, being little affected by the addition of users, items, or ratings [Ricci et al., 2011]. A rather popular *memory-based* predictor is Amazon's item-to-item collaborative filtering [Linden et al., 2003]. It scales independently of the numbers of customers and items in the product catalog. *Model-based* predictors use the database to learn models, usually by applying a Machine Learning or Data Mining technique, and then use the learned model for predictions. *Model-based* predictors have recently enjoyed much interest due to related outstanding results in the Netflix Prize competition, a popular event in the recommender systems field that took place between 2006 and 2009. [1] The SVD++ *model-based* predictor has received a lot of attention since 2008, when it came up as a key algorithm in the Netflix Prize competition [Koren, 2008]. SVD++ is a matrix factorization model that is optimized for minimizing error metrics such as RMSE (Root-Mean-Square Error).

When explicit feedback – such as ratings or likes – is available, the state-of-the-art prediction algorithms for $Top-N$ recommendations are NNCosNgbr (Non-normalized Cosine Neighborhood) and PureSVD (Pure Singular Value Decomposition) [Cremonesi et al., 2010]. NNCosNgbr is *memory-based* and works upon the concept of neighborhood, computing predictions according to the feedback given to similar users or items. PureSVD is *model-based* and works on latent factors, i.e., users and items are modeled as vectors in a same vector space and the score of user u for item i is predicted via the inner-product between their corresponding vectors. When only implicit feedback – such as browsing activity or purchase history – is available, *model-based* predictor WRMF is very effective [Pan et al., 2008].

In this work, competitive predictors for $Top-N$ recommendations are important for two reasons. First, the optimization problem we tackle uses individual item scores that are the ratings generated by such predictors. Second, our work extends $Top-N$ by addressing dependencies among items, and thus it is important to use $Top-N$ as a baseline. We are especially concerned with $Top-N$'s utility, and not with how accurate predictions are, so we decided to use NNCosNgbr and PureSVD as predictors in all our experiments.

2.2 Exploiting Relations among Items

Attempts to abandon the assumption that items are independent date back from Information Retrieval studies in the 1980s. By that time, researchers started questioning the Probability Ranking Principle (*PRP*), according to which documents should be retrieved in decreasing order of their predictive probabilities of relevance [Robertson, 1977]. Bookstein [1983], for instance, presented decision-theoretic models for Information Retrieval that take document interactions into account iteratively.

[1] http://en.wikipedia.org/wiki/Netflix_Prize

Later on, researchers started to focus on diversity-based re-ranking, and they also had to address relations among items to diminish inter-similarities. In particular, Carbonell and Goldstein [1998] have come up with the concept of Maximal Marginal Relevance (*MMR*) to strive redundancy while maintaining query relevance. At each iteration, *MMR* returns the highest-valued item with respect to a tradeoff between relevance and diversity.

In the context of Recommender Systems, several papers exploit relations among items to improve diversity. Zhang and Hurley [2008] model the competing goals of maximizing diversity while maintaining similarity as a binary optimization problem, relaxed to a trust-region problem. Wang [2009] presents a document ranking paradigm, inspired by the Modern Portfolio Theory in finance [Elton et al., 2009], where both the uncertainty of relevance predictions and correlations between retrieved documents are taken into account. Wang [2009] theoretically shows how to quantify the benefits of diversification and how to use diversity to reduce the risk of document ranking. Zuccon et al. [2012] show how Facility Location Analysis, taken from Operations Research, works as a generalization of state-of-the-art retrieval models for diversification in search. They treat the $Top - N$ search results as facilities that should be dispersed as far as possible from each other.

Relations among items other than diversity are also exploited to improve aspects of search results or recommendations. Tversky [1972] proposed a model according to which preference among items is influenced by the presented alternatives. The model, called Elimination By Aspects (*EBA*), states that a consumer chooses among options by sets of aspects, eliminating items that do not satisfy such aspects. Aspects in common among items can be exploited to change the selected results. A related work that presents a variation of *EBA* for commerce search is Ieong et al. [2012]. They propose a new model of ranking, the Random Shopper Model, where each item feature is a Markov Network over the items to be ranked, and the goal is to find a weighting of the features that best reflects their importance.

Another work that is somewhat close to ours is Xiong et al. [2012], where they observed that the Click-Through Rate (CTR) of an ad is often influenced by the other ads shown alongside. They designed a Continuous Conditional Random Field for click prediction focusing on how similarities influence items' CTRs. Weston and Blitzer [2012] also incorporated inter-item similarity during ranking to improve results' recall. They used a latent structured model to learn the structure of the ranked list while assigning scores to items, merging prediction and recommendation steps. Hansen and Golbeck [2009] address the task of recommending collections of items – music lists and mix tapes, for example. This task is different from the one we tackle, given that in their problem each recommended item is actually a collection of items (mix tapes, for instance). In spite of that, they also consider relations between items as an aspect that contributes to the overall value of a collection. In particular, they model the value of individual items, co-occurrence interaction effects, and order effects including placement and arrangement of items.

In this thesis, we adopt Wang [2009] and Zuccon et al. [2012] as baselines. The former is close to ours because it exploits correlations between documents in a collaborative filtering scenario, even though its focus is on ranking and diversity. The latter relates to our work because they also use Facility Location Analysis as a framework, though focused on diversity. Given that our method and theirs share the same theoretical framework, we think it is straightforward to compare both works. We do not compare our method with Weston and Blitzer [2012] because what they present is a specific improvement over latent factor models. As for Ieong et al. [2012], we discarded it because it requires information about item features, and therefore is not a pure collaborative filtering method.

2.3 Max-Sum Dispersion Problem (*MSDP*)

The idea of considering pairwise relations among items is becoming popular in the Recommender Systems literature. Some works, including Zuccon et al. [2012] and Vieira et al. [2011], address diversity by using a formulation that is popular in the Operations Research area. They consider the setting where they are given a set of candidate items I and a set valuation function f defined on every subset of I. For any subset $R \subseteq I$, the overall objective is a linear combination of $f(R)$ and the sum of dissimilarities induced by items in R. The goal is to find a subset R with a given cardinality constraint – e.g. $|R| = 5$ if 5 items must be selected out of I – that maximizes the overall objective [Borodin et al., 2012]. Our overall objective, as discussed in Chapter 3, is similar to this. Our valuation function is the sum of predicted ratings for items in R and we combine it with the sum of co-utility probabilities induced in R.

These overall objectives map into a well-known Facility Location Analysis problem: the weighted version of the Max-Sum Dispersion Problem (*MSDP*). As presented in Gollapudi and Sharma [2009], there are several papers in the Operations Research area that tackle *MSDP*. A common scenario is the placement of facilities in a given area in such way that the distances between them, as well as their individual relevances, are maximized. Analytical models for *MSDP* assume that an area is represented by a set $V = \{v_1, \ldots, v_K\}$ of K vertices with metric distance between every pair of vertices. The objective is to locate $N \leq K$ facilities such that some function of distances between facilities, combined with individual relevances, is maximized.

MSDP is known to be $\mathcal{NP}$-hard, but it admits approximation algorithms in some cases. As we show in Chapter 3, approximations are not admitted in our case. To the best of our knowledge, our work is the first application of *MSDP* for Recommender Systems that focuses on recommendations' utility.

Chapter 3

Basic Concepts

There are two fundamental sources of evidence that we use to select which items should be recommended to a certain user: (i) individual scores ϕ, that correspond to ratings predicted by either PureSVD or NNCosNgbr, and (ii) pairwise scores θ that quantify co-utility probabilities among items. Scores ϕ and θ are always real values in the interval $[0, 1]$, and they are combined in a bi-criteria optimization problem.

In the context of collaborative filtering, a component named *predictor* is used to estimate the feedback a user would give to an item. As an example, a predictor could estimate that a certain user *Sonia* would give 3 stars out of 5 to *Titanic* in a movies recommender system. Traditionally, generated predictions are considered independent from each other, i.e., it is not assumed that the value of a prediction may interfere with any other. Throughout this chapter, we assume that predictions are generated to K items, and then $N \leq K$ items must be selected to compose a recommendation list. Typical values for N are 5 and 10, and depending on the prediction algorithm K can be equivalent to the total number of items in the dataset [Ricci et al., 2011]. We also assume that users explicitly give feedback to items, and depending on the system it can be a rating, a "like" etc.

This chapter is divided into three sections. In Section 3.1, we describe predictors PureSVD and NNCosNgbr. In Section 3.2, we address different techniques for estimating pairwise scores θ. Finally, in Section 3.3, we present a formulation to *MSDP*.

3.1 Individual Scores

In this work, individual scores ϕ – i.e., predicted ratings – are generated by prediction algorithms NNCosNgbr and PureSVD. They are both state-of-the-art methods for $Top - N$ recommendations when explicit feedback is available [Cremonesi et al., 2010].

As already mentioned in Section 2.1, NNCosNgbr is *memory-based* – i.e., it uses rating data to compute the similarity between users or items. PureSVD, on the other hand, is *model-based* – i.e., it uses singular value decomposition to uncover latent factors that explain observed ratings.

3.1.1 NNCosNgbr

NNCosNgbr generates its predictions based on similarity relationships among either users or items. Working with item similarities usually leads to better accuracy rates and more scalability [Papagelis and Plexousakis, 2005]. In this case, predictions can be explained in terms of the items that users have already interacted with by rating them, liking them etc [Papagelis and Plexousakis, 2005]. Due to these reasons, we focus on item-based NNCosNgbr, i.e., our NNCosNgbr implementation exploits item similarities. The prediction of an individual score ϕ_i, given a user u and an item i, is computed as follows:

$$\phi_i = b_{ui} + \sum_{j \in D^l(u;i)} d_{ij}(r_{uj} - b_{uj}) \tag{3.1}$$

where b_{ui} is a combination of user and item biases, as in Koren [2008]; $D^l(u;i)$ is the set of $l \in \mathbb{N}$ items rated by u that are the most similar to i; d_{ij} is the similarity between items i and j, which is computed by taking users' feedback exclusively; r_{uj} is an actual feedback given by u to j; and b_{uj} is the bias related to u and j. Before being used in our optimization problem, ϕ_i is divided by the maximum value it can assume, so it always lies in the real interval $[0, 1]$.

Biases are taken into consideration as they mask fundamental relations between items. Item biases include the fact that certain items tend to receive better feedback than others. Similarly, user biases include the tendency of certain users to give better feedback than others. Finally, the similarity among items, used to compute both $D^l(u;i)$ and d_{ij}, is measured with the adjusted cosine similarity [Cremonesi et al., 2010].

3.1.2 PureSVD

The input for PureSVD is a User $\times$ Item matrix M filled up as follows:

$$M_{ui} = \begin{cases} \text{numerical feedback, if user } u \text{ gave feedback to item } i, \\ 0, \text{if not.} \end{cases} \tag{3.2}$$

PureSVD consists in factorizing M via SVD as $M = U \times E \times Q$, where U is an orthonormal matrix, E is a diagonal matrix with the first γ singular values of M, and Q is also an orthonormal matrix. The prediction of an individual score ϕ_i for a user u is thus given by:

$$\phi_i = M_u \times Q^T \times Q_i, \tag{3.3}$$

where M_u is the u-th row of M corresponding to user u latent factors; Q^T is the transpose of Q; and Q_i is the i-th row of Q corresponding to item i's latent factors [Cremonesi et al., 2010]. Again, ϕ_i is normalized to the real interval $[0, 1]$ before being used in our optimization problem.

3.2 Pairwise Scores

Pairwise scores θ_{ij} represent the probability of items i and j being co-useful to any user. If we consider E_{ij} as a random variable that represents the event *"Items i and j are co-useful to l_{ij} users"*, and assume that E_{ij} follows a Binomial distribution, then its probability mass function is given by:

$$f(l_{ij}; f_{ij}, \theta_{ij}) = \binom{l_{ij}}{f_{ij}} \theta_{ij}^{l_{ij}} (1 - \theta_{ij})^{f_{ij} - l_{ij}}, \tag{3.4}$$

where f_{ij} is the number of users that gave feedback to both i and j.

To estimate θ_{ij}, we analysed estimators Maximum Likelihood and Empirical Bayes [Bishop, 2006]. Maximum Likelihood gives the maximum of $f(l_{ij}; f_{ij}, \theta_{ij})$ by using the point where its derivative is zero and its second derivative is negative. It turns out that working with $\log(f(l_{ij}; f_{ij}, \theta_{ij}))$ is more practical, and the results hold for the original function trivially. Assuming that $f(l_{ij}; f_{ij}, \theta_{ij}) \neq 0$, then the derivation of Maximum Likelihood works as follows:

$$\log(f(l_{ij}; f_{ij}, \theta_{ij})) = \log \binom{l_{ij}}{f_{ij}} + l_{ij} \log(\theta_{ij}) + (f_{ij} - l_{ij}) \log(1 - \theta_{ij}),$$

$$\frac{\partial \log(f(l_{ij}; f_{ij}, \theta_{ij}))}{\partial \theta_{ij}} = \frac{l_{ij}}{\theta_{ij}} - \frac{f_{ij} - l_{ij}}{1 - \theta_{ij}}.$$

To find the maximum, we set the derivative to zero:

$$\frac{l_{ij}}{\theta_{ij}} - \frac{f_{ij} - l_{ij}}{1 - \theta_{ij}} = 0,$$

$$\frac{f_{ij} - l_{ij}}{1 - \theta_{ij}} = \frac{l_{ij}}{\theta_{ij}},$$

$$(f_{ij} - l_{ij})\theta_{ij} = (1 - \theta_{ij})l_{ij},$$

$$\theta_{ij} = \frac{l_{ij}}{f_{ij}} \in [0, 1]. \tag{3.5}$$

As $f(l_{ij}; f_{ij}, 0) = f(l_{ij}; f_{ij}, 1) = 0$, and $f(l_{ij}; f_{ij}, \frac{l_{ij}}{f_{ij}}) \geq 0$, then $\frac{l_{ij}}{f_{ij}}$ is a maximum. Also, when $f(l_{ij}; f_{ij}, \theta_{ij}) = 0$, then either $\theta_{ij} = 0$ or $\theta_{ij} = 1$. In the former case, l_{ij} must also be 0, and in the latter, l_{ij} must be equal to f_{ij}. Consequently, Equation (3.5) also holds in these cases.

Maximum Likelihood is simple and straightforward, but it is not always suitable for scenarios where pairs of items have poor support. This is very common in recommender systems, as users give feedback to a very small fraction of items. Empirical Bayes has the advantage of being more robust when not much data is available. To estimate scores with Empirical Bayes, we consider a simple prior distribution on θ_{ij}:

$$\pi(\theta_{ij}) = 6\theta_{ij}(1 - \theta_{ij}),$$

which is symmetric around $\frac{1}{2}$. This choice of prior is for convenience, as it simplifies the ensuing calculations. We calculate the posterior distribution of θ_{ij} given l_{ij} as:

$$\pi(\theta_{ij}|l_{ij}) = \frac{\Gamma(f_{ij} + 4)}{\Gamma(l_{ij} + 2)\Gamma(f_{ij} - l_{ij} + 2)} \times \theta_{ij}^{l_{ij}+1}(1 - \theta_{ij})^{f_{ij}-l_{ij}+1},$$

which is a form of the Beta distribution [Casella, 1985]. The estimate we effectively use is a point estimate given by the mean of $\pi(\theta_{ij}|l_{ij})$:

$$E(\theta_{ij}|l_{ij}) = \int_0^1 \theta_{ij}\pi(\theta_{ij}|l_{ij})d\theta_{ij} = \frac{f_{ij}}{f_{ij} + 4} \times \frac{l_{ij}}{f_{ij}} + \left(1 - \frac{f_{ij}}{f_{ij} + 4} \times \frac{1}{2}\right). \tag{3.6}$$

To compute θ_{ij} by using either Maximum Likelihood or Empirical Bayes, we assume implicitly that the random variable E_{ij} is the same for any user u and therefore θ_{ij} is independent of the user in question. Another important consideration is that, ideally, E_{ij} should only imply that i and j were co-useful if they were presented in a same recommendation list. Unfortunately, it is not possible to track at what times i and j were selected together in none of the studied datasets. As a consequence, it is not possible to know which items were presented to users in a same list. Hence we compute

θ_{ij} by considering f_{ij} and l_{ij} regardless of temporality. To give an example, if a user liked *Titanic* in November, 2012 and *Matrix* in June, 2011, we consider that they were co-useful to her even though she was not presented with them simultaneously.

It is crucial to point out that scores θ differ from collaborative filtering item-to-item similarities. In the first place, these similarities take all feedback into account. For instance, if a set of common users rated two items negatively, this contributes to their cosine similarity as much as positive ratings would. In the case of scores θ, what is measured is co-utility – not similarity –, and only feedback attesting that items were actually useful is taken into consideration. Following said example, only positive, useful ratings given by a set of common users would be considered.

Another critical distinction between scores θ and item-to-item similarities has to do with their scopes. Item-to-item similarities are computed between two sets of items in the prediction step: (i) items that are already part of the user's historical data and (ii) items to which the user has not given feedback yet. The idea is to retrieve candidates for recommendation that are likely to match the user's taste. In this step, no relation among the retrieved candidates is taken into account. Scores θ, on the other hand, capture the co-utility probabilities of pairs of retrieved candidates. Figure 3.1 illustrates this semantical distinction.

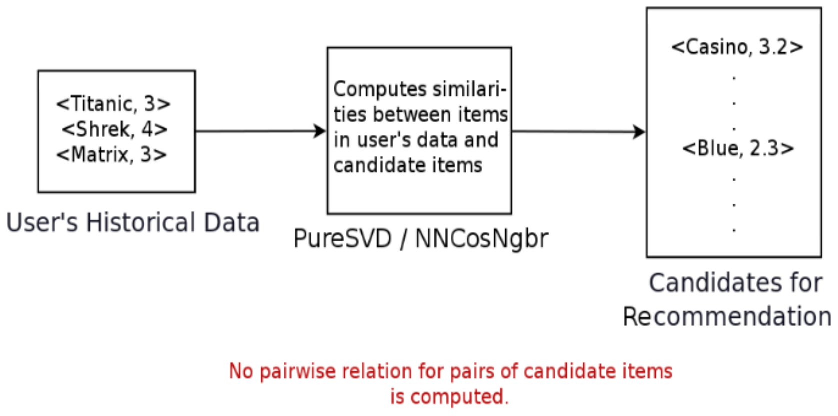
<Titanic, 3>
<Shrek, 4>
<Matrix, 3>
User's Historical Data
Computes similarities between items in user's data and candidate items
PureSVD / NNCosNgbr
<Casino, 3.2>
<Blue, 2.3>
Candidates for Recommendation
No pairwise relation for pairs of candidate items is computed.

Figure 3.1. While item-to-item similarities capture relations between user's historical data and candidates, scores θ are computed among pairs of candidates.

The first box in Figure 3.1 corresponds to movies the user in question has already rated. They are the inputs for box 2, which represents a generic predictor – in our case, it is either PureSVD or NNCosNgbr. The predictor then computes similarities between its inputs and candidate items for recommendation. A list comprised by these candidate items, alongside their predicted ratings, is the output of the predictor, and corresponds to box 3. The predictor does not compute similarities, or any relation, between pairs of candidate items.

3.3 Combining scores

In this work, we combine individual and pairwise scores to select N items out of K for recommendation. Our maximization problem is therefore posed as selecting a set of items $R = \{i_1, ..., i_N\}$ that maximizes the following function:

$$\frac{1}{|R|}\sum_{i_j \in R} \phi_{i_j} + \frac{1}{|R|^2}\sum_{(i_k,i_l)\in R^2} \theta_{i_k i_l}, \tag{3.7}$$

where the normalization in both summations is important to keep their contributions fair. Scores ϕ and θ are also normalized to the interval $[0, 1]$.

Structurally, this problem is an instance of *MSDP*. As previously mentioned, *MSDP* is a Facility Location Analysis, $\mathcal{NP}$-hard problem. When pairwise scores θ satisfy the triangle inequality, *MSDP* admits a 2-approximation algorithm. The proof for this can be seen in Borodin et al. [2012] and Hassin et al. [1997]. On the other hand, it was demonstrated that if the triangle inequality is not satisfied, there is no polynomial time approximation algorithm to *MSDP* unless $P = \mathcal{NP}$ [Ravi et al., 1994]. The co-utility probabilities that we exploit in this work, namely pairwise scores θ, do not satisfy the triangle inequality, as illustrated in Figure 3.2. Hence none of the algorithms we analyse in this thesis have bounds on solution quality.

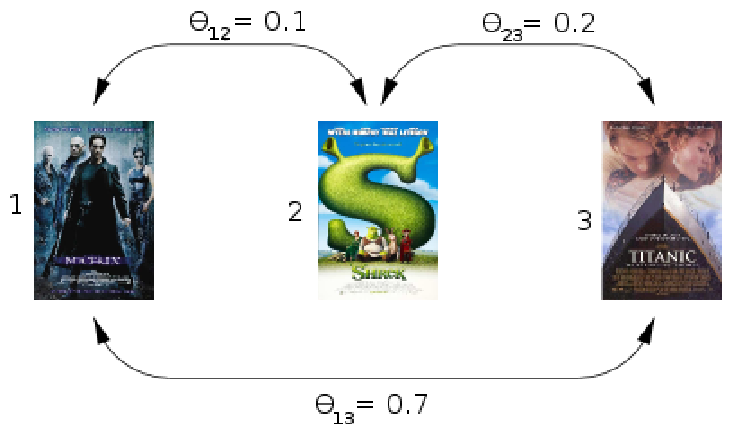

Figure 3.2. The triangle inequality is not satisfied, as $\theta_{13} \geq \theta_{12} + \theta_{23}$.

There are some different ways of obtaining an exact solution to this optimization problem. For instance, one can trivially enumerate all N-combinations of a set with K

items and choose the one that sums up to the highest value. It is also possible to use integer programming to solve it [Nemhauser and Wolsey, 1988]. We describe how we obtain exact solutions to *MSDP* in Chapter 4.

Chapter 4

Algorithms and Validation

To tackle *MSDP* under a practical viewpoint, we studied two suboptimal, polynomial algorithms that are widely related to this problem. We also studied an integer programming approach to *MSDP*. These algorithms are the focus of Section 4.1. This problem cannot be solved efficiently by exact algorithms, albeit it is important to understand how it can be optimally solved. Besides addressing *MSDP*, we detail our baselines in Section 4.2. Finally, in Section 4.3, we present our validation schema for assessing the quality of our experimental results.

4.1 Algorithms to *MSDP*

In this section, we address suboptimal and optimal algorithms to solve *MSDP*. Concerning the former category, we studied two popular heuristics with no bounds on solution quality. With respect to the latter, we describe how to model *MSDP* following the integer programming paradigm.

The algorithms addressed in this section are compatible with any recommender system where it is possible to estimate individual scores ϕ and pairwise scores θ to candidate items. Therefore, these algorithms are *a priori* compatible with systems that employ both matrix factorization techniques and sketching/fingerprinting methods.

4.1.1 Heuristic *Greedy1*

A widely used heuristic to *MSDP* is Greedy Best-First Search [Zuccon et al., 2012]. Henceforth, we will abbreviate it as *Greedy1*. In spite of not having solution quality bounds, it runs fast and yields acceptable solutions in practice. *Greedy1* is shown in Algorithm 1. I is a set of items, I_ϕ corresponds to their individual scores, and I_θ

corresponds to their pairwise scores. The output R is a set with N selected items, where $N \leq K$.

Algorithm 1 *Greedy1* Algorithm

```
Input: $I = \{i_1, \dots, i_k\}$, $I_\phi = \{\phi_{i_1}, \dots, \phi_{i_k}\}$, $I_\theta = \{\theta_{i_1 i_2}, \theta_{i_1 i_3}, \dots, \theta_{i_{k-1} i_k}\}$ , and $N \geq 1$, $|I| \geq N$
Output: Selected items $R$
1: $i \Leftarrow \underset{i \in I}{\operatorname{argmax}} \, \phi_i$
2: $R \Leftarrow \{i\}$
3: $I \Leftarrow I \setminus \{i\}$
4: while $|R| < N$ do
5:     $j \Leftarrow \underset{j \in I}{\operatorname{argmax}} \, \phi_j + \frac{1}{|R|} \sum_{k \in R} \theta_{jk}$
6:     $R \Leftarrow R \cup \{j\}$
7:     $I \Leftarrow I \setminus \{j\}$
8: end while
9: return $R$
```

Greedy1 starts by selecting the item that has the best individual score i. All other $N - 1$ selected items are chosen in a way that maximizes the equation in line 5, where the maximized set is comprised by all items R, that were already chosen, and the new item itself.

Greedy1 runs in polynomial time. The loop in line 4 will be executed exactly $N - 1$ times. In line 5, an item is chosen out of $K - 1$ in the worst case; in the best case, out of $K - N + 1$ ones. It means that $O(K)$ items need to be analysed at each time. In line 5, the first part of the summation is performed in $O(1)$ time; the second part, in $O(|R|)$ time. An upper bound for the time complexity of *Greedy1* is therefore $O(N \times (K \times |R|)) = O(KN^2)$, given that $|R| \leq N$.

4.1.2 Heuristic *Greedy2*

The second heuristic we studied, abbreviated as *Greedy2*, was proposed by Borodin et al. [2012]. It corresponds to Algorithm 2. *Greedy2* is popular because, when pairwise scores θ satisfy the triangle inequality, it is a 2-approximation algorithm to *MSDP*. It is almost the same as *Greedy1*, with a subtle yet relevant difference on line 5 – ϕ_j is multiplied by $\frac{1}{2}$. This difference makes *Greedy2* "non-oblivious", as it is not selecting the next element with respect to the original *MSDP* objective function (Equation (3.7) on page 18).

With respect to how *Greedy2* works, it starts by setting R, the output, as an empty set. From lines 2 to 6, it selects N items, one at a time, by picking the ones that

Algorithm 2 *Greedy2* Algorithm

Input: $I = \{i_1, \ldots, i_k\}$, $I_\phi = \{\phi_{i_1}, \ldots, \phi_{i_k}\}$, $I_\theta = \{\theta_{i_1 i_2}, \theta_{i_1 i_3}, \ldots, \theta_{i_{k-1} i_k}\}$, and $N \geq 1$, $|I| \geq N$

Output: Selected items R

```
1: $i \Leftarrow \operatorname*{argmax}_{i \in I} \phi_i$
2: $R \Leftarrow \{i\}$
3: $I \Leftarrow I \setminus \{i\}$
4: while $|R| < N$ do
5:     $j \Leftarrow \operatorname*{argmax}_{j \in I} \frac{1}{2}\phi_j + \frac{1}{|R|}\sum_{k \in R} \theta_{jk}$
6:     $R \Leftarrow R \cup \{j\}$
7:     $I \Leftarrow I \setminus \{j\}$
8: end while
9: return $R$
```

maximize the objective function in line 3. In this loop, *Greedy2* also updates sets I and R. An upper bound for the time complexity of *Greedy2* is $O(N \times (K \times |R|)) = O(KN^2)$, given that $|R| \leq N$.

4.1.3 Exact Solution

Since *MSDP* is $\mathcal{NP}$-hard, it can only be solved efficiently by suboptimal algorithms. Despite that, it is important to understand how to model an exact algorithm to *MSDP*, especially if comparisons between optimal and suboptimal solutions are of interest. We decided to model *MSDP* under the integer programming paradigm because of the rather fast exact solvers available. It was also quite simple to map our objective function (Equation (3.7)) into an equivalent integer programming problem.

The parameters to model our integer programming problem are a set of items $I = \{i_1, \ldots, i_k\}$, their corresponding individual scores $I_\phi = \{\phi_{i_1}, \ldots, \phi_{i_k}\}$, the pairwise scores for all combinations of items in I, $I_\theta = \{\theta_{i_1 i_2}, \ldots, \theta_{i_{k-1} i_k}\}$, and the number of items for selection N. We come up with binary variables $Y = \{y_1, \ldots, y_k\}$ to represent which items are selected ($y_j = 1$ if and only if i_j is selected), and rewrite *MSDP* as:

$$\begin{aligned} \text{maximize} \quad & \frac{1}{|I|}\sum_{j \in I} y_j \phi_j + \frac{1}{|I|^2}\sum_{j \in I}\sum_{k \in I | k \neq j} y_j y_k \theta_{jk}, \\ \text{subject to} \quad & y_i \in \{0, 1\} \quad \forall i, \\ & \sum_{y_i \in Y} y_i = N. \end{aligned}$$

To frame this program in the integer programming paradigm, we have to linearize

MSDP's products $y_j y_k$ as variables $x_{jk} = y_j y_k \; \forall j, \forall k$. Considering that y_j and y_k are binary variables, we have the following constraints for variables x_{jk}:

$$\begin{aligned} &x_{jk} \leq y_j \\ &x_{jk} \leq y_k \\ &x_{jk} \geq y_j + y_k - 1 \end{aligned}$$

That being stated, we rewrite our problem as:

$$\begin{aligned} \text{maximize} \quad & \frac{1}{|I|} \sum_{j \in I} y_j \phi_j + \frac{1}{|I|^2} \sum_{j \in I} \sum_{k \in I | k \neq j} x_{jk} \theta_{jk} \\ \text{subject to} \quad & y_i \in \{0, 1\} \quad \forall i \\ & x_{jk} \leq y_j \\ & x_{jk} \leq y_k \\ & x_{jk} \geq y_j + y_k - 1 \\ & \sum_{y_i \in Y} y_i = N \end{aligned}$$

4.2 Baselines

In this section, we describe the baselines with which we compare our method. *Top* $-$ *N* adopts the Probability Ranking Principle (*PRP*) by selecting items according to their individual scores ϕ, generated by predictors such as PureSVD, exclusively. *Mean-Variance Analysis*, proposed by Wang [2009], and *Maximal Marginal Relevance*, widely exploited by works in diversity, break with the *PRP* by considering relations among candidate items for recommendation. Our method exploits co-utility probabilities under an *MSDP* framework and semantically differs from these baselines. To the best of our knowledge, there is no general-purpose, collaborative filtering selection techniques that model co-utilities in a way that is similar to ours. Despite that, comparisons among our method and these baselines are worthwhile, as we discuss in Chapter 5.

4.2.1 *Top* $-$ *N*

Top $-$ *N* is described in Algorithm 3. Input I corresponds to the set of K candidate items, I_ϕ is the related set of predicted individual scores, and N is the number of items for selection. Output R is the set of selected items for recommendation.

Top $-$ *N* is rather simple, which is one of its competitive aspects. Its time complexity is $O(K \log N)$ if a heap is used to store the N items with highest individual

Algorithm 3 *Top − N* Algorithm

```
Input: $I = \{i_1, \ldots, i_k\}$, $I_\phi = \{\phi_{i_1}, \ldots, \phi_{i_k}\}$, and $N \geq 1$, $|I| \geq N$
Output: Selected items $R$
1: $R \Leftarrow \{i_j \in I \mid \phi_{i_j}\ is\ one\ of\ the\ N\ highest\ scores\ in\ I_\phi\}$
2: return $R$
```

scores ϕ [Baeza-Yates and Ribeiro-Neto, 2011]. Alternatively, it is $O(K \log K)$ if a complete ordering of items, according to their corresponding scores ϕ, is required.

4.2.2 Mean-Variance Analysis

Inspired by the Modern Portfolio Theory in finance, Wang [2009] proposes a method for ranking a list of items on the basis of its expected mean relevance and its variance. In that context, the variance works as a measure of risk. Based on this mean-variance principle, they devised a document ranking algorithm, abbreviated henceforth as *MVA*.

MVA is shown in Algorithm 4. The input I corresponds to the set of K candidate items; I_ϕ is the related set of individual scores, learned from a certain predictor; C is a covariance matrix estimated from users' historic data; α is a real-valued risk regulator; N is the number of items for selection [Wang, 2009]. The output R is the set of selected items for recommendation.

Algorithm 4 *MVA* Algorithm

```
Input: $I = \{i_1, \ldots, i_k\}$, $I_\phi = \{\phi_{i_1}, \ldots, \phi_{i_k}\}$, $C = \{c_{11}, c_{12}, \ldots, c_{k-1k}, c_{kk}\}$, $\alpha$, and $N \geq 1$, $|I| \geq N$
Output: Selected items $R$
1: $R \Leftarrow \emptyset$
2: while $|R| < N$ do
3:     $j \Leftarrow \underset{j \in I}{\operatorname{argmax}}\ \phi_j - \alpha \times b_j^2 - 2\alpha \times \sum_{k \in R} b_k b_j c_{kj}$
4:     $R \Leftarrow R \cup \{j\}$
5:     $I \Leftarrow I \setminus \{j\}$
6: end while
7: return $R$
```

The function in line 3 originally contains weights w for ranking regularization. We omitted these weights because we do not evaluate ranking aspects in this work – we focus on the selected set of items exclusively. Also in line 3, when $\alpha > 0$, the selection is risk-averse, while when $\alpha < 0$, it is risk-loving [Wang, 2009].

MVA is a polynomial time algorithm. As in *Greedy2*, the loop from lines 2 to 6 is executed N times and, in line 3, an item is chosen out of K in the worst case. It means that $O(K)$ items need to be tested for selection at each time. For each test in line 3, the objective function is computed in $O(|R|)$ time. An upper bound for the time complexity of *MVA* is thus $O(N \times (K \times |R|)) = O(KN^2)$, given that $|R| \leq N$.

4.2.3 Maximal Marginal Relevance

Maximal Marginal Relevance (*MMR*) is a criterion that has been widely adopted in search and recommendation contexts as a means of diminishing redundancy while maintaining relevance [Carbonell and Goldstein, 1998; Vargas and Castells, 2011; Vieira et al., 2011; Zuccon et al., 2012]. *MMR* consists in a ranking formula that, as well as our method, takes the individual relevance of items and relations among them both into account. Given the wide scope of applications for *MMR*, there are different ways of implementing it. The implementation we use in this work is described in Zuccon et al. [2012] and is shown in Algorithm 5. Input I corresponds to the set of K candidate items; I_ϕ is the related set of individual scores, learnt from a certain predictor; I_D is the set of pairwise dissimilarities computed for all items in I via users' historic data; λ is a real-valued term regulator; and N is the number of items for selection. Output R is the set of selected items for recommendation.

Algorithm 5 *MMR* Algorithm

```
Input: $I = \{i_1, \ldots, i_k\}$, $I_\phi = \{\phi_{i_1}, \ldots, \phi_{i_k}\}$, $I_D = \{D_{i_1 i_2}, \ldots, D_{i_{k-1} i_k}\}$, $\lambda$, and $N \geq 1$, $|I| \geq N$
Output: Selected items $R$
1: $R \Leftarrow \emptyset$
2: while $|R| < N$ do
3:     $j \Leftarrow \underset{j \in I}{\operatorname{argmax}}\ \lambda\phi_j + (1-\lambda)\min_{k \in R} D_{jk}$
4:     $R \Leftarrow R \cup \{j\}$
5:     $I \Leftarrow I \setminus \{j\}$
6: end while
7: return $R$
```

In this work, we follow the choice of Zuccon et al. [2012] and use Kullback-Leibler Distance as the dissimilarity metric for pairs of items. Basically, this metric outputs how divergent the feedback that two items have received is. *MMR* is a polynomial time algorithm with an $O(KN^2)$ upper bound for its time complexity. The derivation of this bound is analogous to *MVA*'s time complexity upper bound.

4.3 Validation Metrics

To validate our work, we use the explicit feedback users give over items as a utility metric: the better it is, the more useful the recommendations are [Breese et al., 1998]. For example, in movie ratings, where a 5-star movie is considered an excellent movie, we can assume that recommending a 5-star movie is more useful than recommending a 4-star one [Ricci et al., 2011]. Furthermore, we consider that the utility of a recommendation system can be quantified by the utility of the recommendations it actually makes – rather than how close predictions are from the actual feedback given by users [Passos et al., 2011].

For all studied datasets, as we discuss in Chapter 5, feedback consists of ratings. Considering that we are focused on recommendations' utility, and that we use ratings as a utility metric, we compare different algorithms by contrasting the ratings their selected items receive.

All the datasets we study consist of tuples $(user, item, rating)$. We applied cross-validation in all experiments, and randomly partitioned the datasets into training and test data. Consequently, we ignored rating timestamps, whenever they were present, while splitting the data. Cross-validation is interesting in our case because we only analyse three datasets, and by crossing training and data partitions we increase the number of different scenarios on which we run experiments. Considering that recommendation lists are generated over items in the test data, to which we know the actual ratings, our experiments simulate scenarios where users would rate all recommended items. Other works that opt for cross-validation are Vargas and Castells [2011] and Sarwar et al. [2001].

For each dataset, the training data is explored by predictors PureSVD and NNCosNgbr to generate individual scores ϕ. The training data is also used to estimate pairwise scores θ, as well as other pairwise information required by the baselines. For all tuples $(user, item, rating)$ in the test data, we hide the corresponding $rating$: a ground-truth information that will be used as a metric of utility. The union of all $(user, item)$ test pairs with a user in common corresponds to the set of candidate items for this user. This set, namely I, is one of the input parameters for all previously detailed algorithms. Individual scores are then predicted to all items in I, as well as pairwise scores for pairs of items in I.

We then run one of the studied algorithms and end up with a set of selected items R. At this point, we can retrieve the ground-truth rating for each selected item in R and analyse how useful the selection was in practice.

In this work we also make comparisons under a diversity perspective. In our scope, diversity is defined as the opposite of similarity, and hence as a synonym for dissimilarity. Although not the focus of our work, we briefly investigate whether our method hurts recommendations' diversity. The diversity metric we apply, intra-list distance (ILD), was proposed by Zhang and Hurley [2008] and works as follows:

$$\text{ILD} = \frac{2}{|R|(|R|-1)} \sum_{i_k, i_l \in R, l<k} 1 - \text{sim}(i_k, i_l), \tag{4.1}$$

where R is comprised by all selected items and $\text{sim}(i_k, i_l)$ is a generic similarity measurement for items i_k and i_l. Further discussions on how we computed items' similarities and performed experiments with ILD are presented in Chapter 5.

Chapter 5

Experimental Results

We examine different algorithms to *MSDP*, discussed in Chapter 4, by taking three different datasets into consideration: MovieLens 100K,[1] MovieLens 1M,[2] and Jester 1.[3] For the MovieLens datasets, we considered that movies were liked by users – i.e., they were useful – if their ratings were equal or higher than 4; [4] in the case of Jester 1, if they were equal or higher than 5.00. [5] The choice of these values is detailed in Section 5.1.

For all experiments, PureSVD was executed with 50 latent factors, and the number of neighbors in $D^k(u; i)$ in Equation 3.1 was fixed in 60. After grid searches, these two values yielded the best $Top - N$ results for both PureSVD and NNCosNgbr. Recommendation lists have sizes $N = 5, 10, 20$ because they are popular values in the related literature. Considering that we use cross-validation, reported results are means of values per fold. Additionaly, we also compute means of ratings in some experiments. According to the Central Limit Theory, sampling distributions of means always follow the Gaussian distribution, so the distributions that we analyse, comprised by means, can be compared to each other via paired Student's t-tests [Hastie et al., 2001].

This chapter is organized as follows. Section 5.1 brings a characterization of the datasets that we studied in this work. Section 5.2 compares different statistical estimators for pairwise scores θ. Section 5.3 contrasts different algorithms to *MSDP* and compares our method with different baselines in terms of utility. Section 5.4 examines how our method scales in time. Finally, Section 5.5 investigates the relation between our method and recommendations' diversity.

[1]`http://www.grouplens.org/system/files/ml-100k.zip`
[2]`http://www.grouplens.org/system/files/ml-1m.zip`
[3]`http://goldberg.berkeley.edu/jester-data/jester-data-1.zip`
[4]Values range from 1 to 5.
[5]Values range from -10.00 to 10.00.

5.1 Studied Datasets

We performed experiments to validate our method over three different datasets: MovieLens 100K, MovieLens 1M, and Jester 1. We chose to work with these datasets due to their popularity in the collaborative filtering literature. It is worth pointing out that the MovieLens datasets have some content and demographic data available. Nonetheless, we did not exploit these data when learning recommendation models because our scope is limited to collaborative filtering. In this section we present a characterization of them in order to facilitate posterior experiment analyses. To start off, Table 5.1 summarizes some of the datasets' main features.

Feature	MovieLens 100K	MovieLens 1M	Jester 1
Domain	Movies	Movies	Jokes
Feedback	Ratings (1 - 5)	Ratings (1 - 5)	Ratings (-10.00 - 10.00)
Number of users	943	6,040	24,983
Number of items	1,682	3,900	100
Number of given ratings	100,000	1,000,209	1,810,455
Minimum ratings/user	20	20	36
Sparsity rate	0.937	0.958	0.275
Mean rating value	3.588	3.703	1.877

Table 5.1. Characterization of the studied datasets.

As Table 5.1 shows, while ratings in the MovieLens dataset are discretized and vary from 1 to 5, users in Jester 1 can assign any real number from -10.00 to 10.00 to any joke. Another key difference is that the MovieLens datasets are significantly sparser than Jester 1. In the former, users rated at least 20 movies, whereas in the latter feedback was given to at least 36 jokes. Considering that there are only 100 jokes in Jester 1, this value corresponds to a minimum of 36%. Regarding MovieLens 1M, the sparsest dataset, it is comprised by many more users and items than MovieLens 100K, and its number of given ratings is smaller than Jester 1's. As for the mean rating value, Table 5.1 indicates that MovieLens users tend to give average-to-good ratings to movies. This reveals that, in this scenario, users prefer to manifest their tastes by rating movies they find enjoyable. As for Jester 1, the mean rating is more neutral. Figure 5.1 conveys information about the distribution of rating values in the datasets.

Figure 5.1 draws attention to the fact that MovieLens users tend to give higher ratings to items, when compared to Jester 1's. The distributions for both MovieLens datasets are rather similar, and some studies have shown that the assumption of Gaussian distributions for these ratings holds [Goyal and Lakshmanan, 2012; Lipcon, 2007]. As for Jester 1, the authors of its original paper indicate that its rating distribution

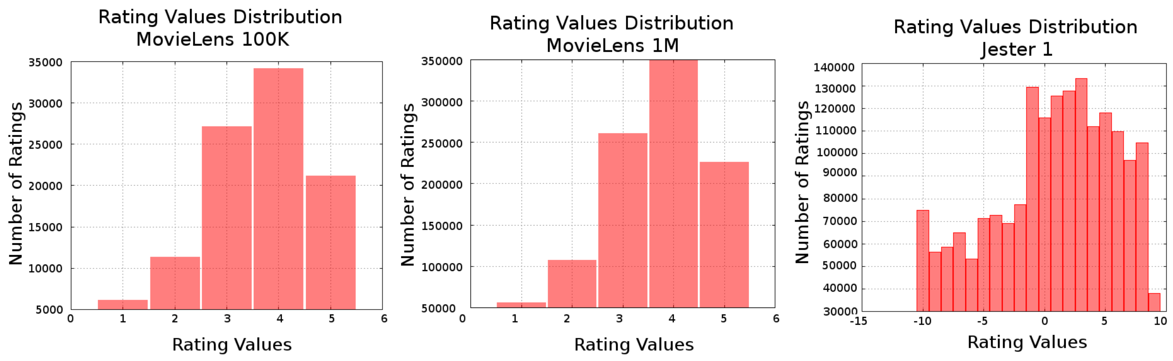

Figure 5.1. Distributions of ratings in the studied datasets. In the graph that corresponds to Jester 1, each bar consists of ratings in intervals $[-10.00; -9.00), [-9.00; -8.00), \ldots, [9.00; 10.00]$.

can be approximated by a Gaussian distribution [Goldberg et al., 2001]. Considering these results, we henceforth assume that the hypothesis of Gaussian distributions is applicable for these ratings. Figure 5.1 also indicates that Jester 1 presents a higher rating variance, implying that rating values distant from the mean are more common in it than in the MovieLens datasets.

We adopted 4 as a threshold for high ratings, with respect to the MovieLens datasets, because this value is considered high by relevant works [Ricci et al., 2011; Jannach et al., 2011]. As for Jester 1, we decided to adopt 5.00 as a threshold because this value is significantly higher than the interval $[2.00; 3.00)$, which is associated with most ratings in this dataset, and therefore seemed to be a safe choice. A more refined approach would consider that these thresholds vary from user to user. For example, users who tend to give lower ratings may find a 3-star movie useful. In spite of that, the generic thresholds that we adopted yielded useful recommendations to most users, as we discuss in this chapter. Consequently, we did not prioritize personalized thresholds.

5.2 Comparing Estimators for Pairwise Scores

In Chapter 3, we present two estimators for pairwise scores θ: Maximum Likelihood and Empirical Bayes.

We examined Maximum Likelihood and Empirical Bayes estimators considering all datasets. To assess the quality of these estimators, we contrasted mean ratings obtained with solutions that use either Maximum Likelihood or Empirical Bayes to estimate θ. In all cases, individual scores ϕ were generated by PureSVD exclusively, as analysing predictors is not in the scope of this experiment. Table 5.2 summarizes the obtained results.

List Size (N)	Method	MovieLens 100K	MovieLens 1M	Jester 1
$N = 5$	*ML – Greedy1*	3.974	4.175	1.518
	EB – Greedy1	3.976	4.175	1.518
	ML – Greedy2	3.977	4.184	1.689
	EB – Greedy2	3.987	4.187	1.688
$N = 10$	*ML – Greedy1*	3.874	4.057	1.188
	EB – Greedy1	3.876	4.057	1.188
	ML – Greedy2	$\underline{3.871}$	4.065	1.323
	EB – Greedy2	$\underline{3.881}$	4.065	1.323
$N = 20$	*ML – Greedy1*	3.765	3.939	0.911
	EB – Greedy1	3.765	3.939	0.911
	ML – Greedy2	3.766	3.942	0.923
	EB – Greedy2	3.768	3.942	0.923

Table 5.2. Mean ratings computed with different methods. *ML – Greedy1* and *ML – Greedy2* use Maximum Likelihood estimates for pairwise scores. *EB – Greedy1* and *EB – Greedy2* use Empirical Bayes estimates instead. Reported results are averages of results obtained with 5-fold cross-validation.

Regarding Table 5.2, for each ML/EB pair of results with algorithm ($Greedy1$ or $Greedy2$), dataset, and N in common, we performed a paired t-test with a 95% confidence interval. Aside from the underlined result, all outcomes were statistically equivalent. This provides strong evidence that both estimators lead to very similar recommendations in terms of utility – i.e., mean rating given by users.

5.3 Comparing Algorithms to *MSDP* and Baselines

In this section, we investigate the effectiveness of different non-exact algorithms to *MSDP*. We also present a statistical comparison between non-exact and exact algorithms to *MSDP*. Finally, we compare our method with three different baselines.

5.3.1 Comparing Non-Exact Algorithms to *MSDP*

To compare *Greedy1* and *Greedy2*, two non-exact algorithms to *MSDP*, we computed individual scores ϕ using PureSVD and NNCosNgbr as predictors, and pairwise scores θ were estimated with Empirical Bayes exclusively. Results are showed in Table 5.3, and statistical differences are underlined.

For each *Greedy1*/*Greedy2* pair of results with predictor, dataset and N in common, we performed a paired t-test with a 95% confidence interval. As shown in Table 5.3, *Greedy1* and *Greedy2* led to statistically equivalent solutions in most cases. It indicates that *Greedy1* may perform as well in practice as *Greedy2*. When *Greedy1* and

Predictor	Method	MovieLens 100K	MovieLens 1M	Jester 1
PureSVD	*Greedy1 − 5*	3.976	4.175	$\underline{1.518}$
	Greedy2 − 5	3.987	4.187	$\underline{1.688}$
	Greedy1 − 10	3.876	4.057	$\underline{1.188}$
	Greedy2 − 10	3.881	4.065	$\underline{1.323}$
	Greedy1 − 20	3.763	3.938	0.911
	Greedy2 − 20	3.768	3.941	0.923
NNCosNgbr	*Greedy1 − 5*	3.873	$\underline{4.065}$	2.362
	Greedy2 − 5	3.896	$\underline{4.082}$	2.356
	Greedy1 − 10	3.803	$\underline{3.973}$	1.579
	Greedy2 − 10	3.818	$\underline{3.988}$	1.578
	Greedy1 − 20	3.722	$\underline{3.890}$	0.940
	Greedy2 − 20	3.732	$\underline{3.902}$	0.939

Table 5.3. Mean ratings generated by solutions *Greedy1* and *Greedy2* for recommendation lists with sizes $N = 5, 10, 20$. $Greedy1 - 5$ corresponds to solutions with size $N = 5$ generated by $Greedy1$, for example. Reported results are averages of results obtained with 5-fold cross-validation.

Greedy2 generated statistically different results, *Greedy2* was responsible for the best values. For the Jester 1 dataset, whose ratings range from -10.00 to 10.00, the gains were up to 11% with *Greedy2 − 5* outperforming *Greedy1 − 5*. As for the MovieLens datasets, whose ratings range from 1 to 5, the gains were up to 0.4% for MovieLens 1M, with *Greedy2 − 5* winning over *Greedy1 − 5*.

It is important to notice that the best mean ratings for the MovieLens datasets were generated with predictor PureSVD. As for the Jester 1 dataset, NNCosNgbr yielded better results. In the latter case, it is likely that NNCosNgbr has benefited from Jester 1's low sparsity, as it is a memory-based predictor sensitive to high sparsity scenarios. Finally, it is clear in Table 5.3 that *Greedy1* / *Greedy2* differences tend to get smaller as N grows. This is a consequence of the fact that, with larger N values, the difference between N and the number of items available for selection becomes smaller, thus allowing higher overlaps between solution.

5.3.2 Comparing Non-Exact and Exact Algorithms to *MSDP*

Greedy2 is a polynomial time, non-exact algorithm to *MSDP*. It is thus useful to compare it with an exact algorithm, namely *Exact*, presented in Section 4.1.3, to understand how close their results are to each other in practice. We do not compare *Greedy1* with *Exact* because, in previous experiments, its results were either statistically equivalent or worse than *Greedy2*'s. In the following experiment, we used PureSVD for the MovieLens datasets, as it yielded the best results in previous experiments. We did not use both predictors because comparing their performances is not the goal of this experiment and generating exact solutions to *MSDP* is very time-consuming. For the same reason, we relied exclusively on predictor NNCosNgbr for the Jester 1 dataset. After the variables, parameters, and restrictions of *Exact* were defined, we passed them to IBM's CPLEX optimizer [ILOG, Inc, 2013].

To contrast *Greedy2* and *Exact*, we divided the dataset MovieLens 100K into 5 folds with approximately the same size randomly. As for MovieLens 1M and Jester 1, the number of folds was 10, as generating exact solutions to a test set is an exponential time task and takes many hours to be completed. In all cases, one of the folds was chosen for test and the others were used for training. We did not perform cross-validation either due to time constraints.

To make it possible to compute the exact solutions and perform comparisons that could make sense in practical scenarios, we adopted a timeout of 20 seconds. We discarded all exact solutions that would take more than that, and only compared optimal solutions obtained below this time threshold with their corresponding suboptimal ones. Solutions that take more than 20 seconds to compute may share specific characteristics that could change our analysis, but at least they corresponded to less than 15% of all cases. The mean ratings obtained by different solutions are listed in Table 5.4.

Method	MovieLens 100K	MovieLens 1M	Jester 1
Greedy2 − 5	3.971	4.166	2.136
Exact − 5	4.003	4.184	2.149
Greedy2 − 10	3.900	4.137	1.369
Exact − 10	3.907	4.148	1.370
Greedy2 − 20	3.770	3.947	1.075
Exact − 20	3.771	3.951	1.078

Table 5.4. Mean ratings computed with *Greedy2* and *Exact* for recommendation lists with sizes $N = 5, 10, 20$. $Greedy2{-}5$ corresponds to solutions with size $N = 5$ generated by $Greedy2$, for example.

We performed a paired t-test with a 95% confidence interval and all mean ratings obtained with *Greedy2* were statistically equivalent to the corresponding ones prompted by *Exact*. These results draw attention to the idea that, in practice, *Greedy2* is a good approach to *MSDP*.

A case where *Greedy2* and *Exact* lead to different solutions works as follows. Let us consider a set of candidate items $I = \{i_1, i_2, i_3\}$ with corresponding sets of scores $I_\phi = \{\phi_{i_1} = 0.9, \phi_{i_2} = 0.85, \phi_{i_3} = 0.85\}$ and $I_\theta = \{\theta_{i_1 i_2} = 0.7, \theta_{i_1 i_3} = 0.6, \theta_{i_2 i_3} = 0.9\}$. If we want to select $N = 2$ items out of I, *Exact* will select i_2 and i_3, whereas *Greedy2* will select i_1 and i_2. The greedy choice of starting the selection by choosing the item with the highest score ϕ not necessarily leads to the optimal solution, as the example illustrates. This situation happens in practice, but Table 5.4 indicates that, regardless of it, selections prompted by both *Exact* and *Greedy2* are likely to be similarly useful.

5.3.3 Comparing Our Method to Baselines

We decided to compare *Greedy2* with $Top - N$, MVA, and *MMR* in order to understand some of its different aspects. With respect to $Top - N$, we wanted to evaluate how the exploitation of co-utility alone can improve recommendations. As for MVA and *MMR*, we were interested in contrasting *Greedy2* with methods that also abandon the assumption that items are independent. *MMR*, in particular, also maps into $MSDP$, although the semantics of its pairwise scores is related to diversity – not to co-utility.

For all methods, individual scores were predicted by PureSVD and NNCosNgbr. *Greedy2*'s pairwise scores were calculated with the Empirical Bayes estimator. We set MVA's parameter $\alpha = 0.05$ after a grid search involving values that ranged from -5.0 to 5.0, i.e., MVA is slightly risk-lover in our experiments. The parameter $\alpha = 0.05$ prompted the best MVA's results. As to MVA's covariance matrix, we computed it by considering, for every pair of items, the ratings they received by a common set of users.

With respect to *MMR*, we followed Zuccon et al. [2012] and adopted Kullback-Leibler distance for computing the dissimilarity between items' rating distributions. There are several ways of measuring items' dissimilarity, and even though this implementation of *MMR* uses differences in rating distributions as a proxy for diversity, many methods for measuring diversity are based on item content [Ricci et al., 2011]. The use of rating distributions is particularly indispensable when a strict collaborative filtering schema has to be adopted, or when no information about the items is available. One of the reasons why we chose this implementation of *MMR* is because it does not use content information to devise pairwise scores, so the comparison becomes fairer.

Results that contrast *Greedy2* and $Top - N$ are listed in Table 5.5, Table 5.6 and Figure 5.2. As for MVA and *MMR*, they are compared with *Greedy2* in Table 5.7, Table 5.8, Figure 5.3, and Figure 5.4. For all experiments reported in tables, we performed a paired t-test with a 95% confidence interval. Underlined results are statistically equivalent.

Table 5.5 presents strong evidence that the exploitation of co-utility alone yields better recommendations than those obtained with competitive $Top - N$ baselines. In all cases, either *Greedy2* led to superior mean ratings or it was statistically equivalent to the corresponding $Top - N$ results. For the Jester 1 dataset, gains were up to 31%

Predictor	Method	MovieLens 100K	MovieLens 1M	Jester 1
PureSVD	*Top − 5*	3.924	4.127	1.292
	Greedy2 − 5	3.987	4.187	1.688
	Top − 10	3.837	4.004	1.031
	Greedy2 − 10	3.881	4.065	1.323
	Top − 20	3.738	3.908	0.892
	Greedy2 − 20	3.768	3.941	0.923
NNCosNgbr	*Top − 5*	3.821	4.027	2.312
	Greedy2 − 5	3.896	4.082	2.356
	Top − 10	3.775	3.928	1.529
	Greedy2 − 10	3.818	3.988	1.578
	Top − 20	3.691	3.836	$\underline{0.939}$
	Greedy2 − 20	3.732	3.902	$\underline{0.939}$

Table 5.5. Mean ratings computed with *Top − N* and *Greedy2* for recommendation lists with sizes $N = 5, 10, 20$. *Greedy*2 − 5 corresponds to solutions with size $N = 5$ generated by *Greedy*2, for example. Reported results are averages of results obtained with 5-fold cross-validation.

with *Greedy2 − 5* outperforming *Top − 5*. As for the MovieLens datasets, they were up to 2% with *Greedy2 − 5* outperforming *Top − 5* for MovieLens 100K. Although the difference between *Greedy2* and *Top − N* in Table 5.5 may seem small on average, it has an impact on recommender systems [Bell and Koren, 2007].

In absolute terms for Jester 1, *Greedy2* outperforms *Top − N* by 0.396 points in the best case (*Greedy2 − 5* and *Top − 5*) and by 0.000 in the worst case (*Greedy2 − 20* and *Top − 20*). With respect to the MovieLens datasets, *Greedy2* wins over *Top − N* by 0.228 points in the best case (*Greedy2 − 10* and *Top − 10* for MovieLens 1M) and by 0.030 in the worst case (*Greedy2 − 20* and *Top − 20* for MovieLens 100K).

Some works suggest that it is worse to recommend an item the user dislikes than to not recommend an item she likes [Hansen and Golbeck, 2009; Ricci et al., 2011]. In order to give continuity to our analysis, we exploit this idea by assuming that low ratings are given to disliked items and compare the lowest ratings obtained with *Top − N* and *Greedy2*. Instead of focusing on all recommended items, this experiment concerns only the worst rated item in each recommendation. Results are arranged in Table 5.6. Underlined values are statistically equivalent.

Results in Table 5.6 indicate that the worst item recommended by *Greedy2* tends to be better rated than the corresponding one for *Top − N*. In all cases, *Greedy2* led to superior or statistically equivalent lowest ratings. In terms of ratings for Jester 1, *Greedy2* exceeds *Top − N* by 0.496 points in the best case (*Greedy2 − 5* and *Top − 5*) and by 0.004 in the worst case (*Greedy2 − 20* and *Top − 20*). As for the MovieLens datasets, *Greedy2* exceeds *Top − N* by 0.142 points in the best case (*Greedy2 − 5*

Predictor	Method	MovieLens 100K	MovieLens 1M	Jester 1
	Top − 5	2.881	3.127	-3.766
	Greedy2 − 5	3.003	3.217	-3.270
PureSVD	*Top − 10*	2.404	2.624	-5.589
	Greedy2 − 10	2.482	2.683	-5.231
	Top − 20	2.100	2.223	-6.318
	Greedy2 − 20	2.123	2.263	-6.285
	Top − 5	2.670	2.963	-2.178
	Greedy2 − 5	2.812	3.057	-2.200
NNCosNgbr	*Top − 10*	2.303	2.472	-4.777
	Greedy2 − 10	2.360	2.557	-4.789
	Top − 20	2.035	2.117	-6.257
	Greedy2 − 20	2.066	2.201	-6.261

Table 5.6. Lowest ratings generated by *Top − N* and *Greedy2* for recommendation lists with sizes $N = 5, 10, 20$. Reported values are averages of results obtained with 5-fold cross-validation.

and *Top − 5* for MovieLens 100K) and by 0.023 in the worst case (*Greedy2 − 20* and *Top − 20* for MovieLens 100K). Finally, recommendations generated for Jester 1 with NNCosNgbr were particularly similar for both *Greedy2* and *Top − N*, with equivalent mean ratings for all N values.

Thus far, we have based our analysis on averages of ratings. Albeit useful, averages are not statistically robust measures [Zaki and Meira, 2014]. Hence, to have a better idea of the difference between *Greedy2* and *Top − N*, we extend our analysis to a percentual approach. Specifically, we computed how many times each method yielded the highest ratings and reported percentages in Figure 5.2.

Figure 5.2 leads to a succint Win/Loss analysis. *Greedy2* generates the highest mean ratings to approximately 65% of users. The percentages associated with *Greedy2* tend to increase as N grows, which suggests that our method brings gain to more users when more recommendations are generated. We also investigated whether our method works best to users with small historical data, or to users who tend to give particularly high/low ratings, but no patterns were noticed.

As to what concerns *MVA* and *MMR*, results in Table 5.7 indicate that *Greedy2* is likely to recommend items that receive better feedback from users. *MVA*'s mean ratings were particularly low with respect to Jester 1, and the results yielded by *MMR* were very close to those obtained with *Top − N*. The gains obtained with *Greedy2* for Jester 1, when compared to *MVA*, were up to 106% with *Greedy2 − 5* outperforming *MVA − 5*. With respect to the MovieLens datasets, they were up to 2% with *Greedy2 − 20* outperforming *MVA − 20* for MovieLens 1M. As for *MMR*, the gains were up to 31% for Jester 1, also with *Greedy2 − 5* outperforming *MMR − 5*. These

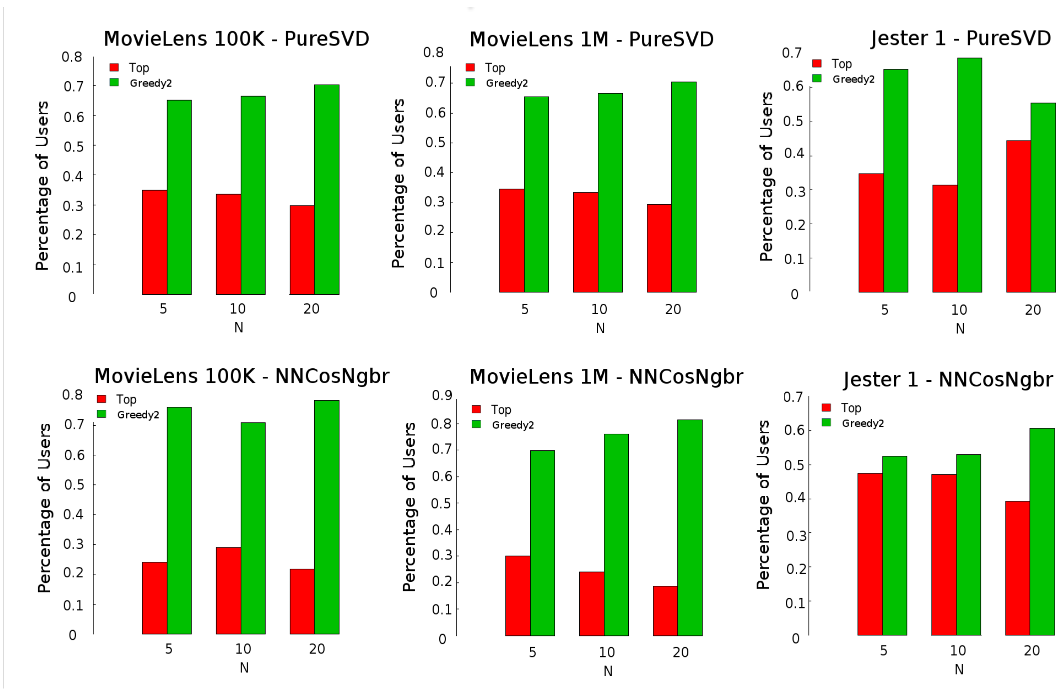

Figure 5.2. Percentages of users to which *Top* − *N* and *Greedy2* have won over each other, in terms of highest mean rating given to generated recommendations.

gains were up to 3% for the MovieLens datasets, with *Greedy2* − *5* outperforming *MMR* − *5* for MovieLens 100K.

In absolute terms for Jester 1, *Greedy2* surpasses *MVA* by 1.210 points in the best case (*Greedy2* − *5* versus *MVA* − *5*) and by 0.059 in the worst case (*Greedy2* − *20* versus *MVA* − *20*). As for the MovieLens datasets, *Greedy2* outperforms *MVA* by 0.235 points in the best case (*Greedy2* − *10* versus *MVA* − *10* for MovieLens 1M) and by 0.040 in the worst case (*Greedy2* − *20* versus *MVA* − *20* for MovieLens 1M). With respect to *MMR* for Jester 1, *Greedy2* wins over it by 0.396 points at most (*Greedy2* − *5* versus *MMR* − *5*) and by 0.000 at least (*Greedy2* − *20* versus *MMR* − *20*). *Greedy2* outperforms *MMR* for the MovieLens datasets by 0.103 points in the best case (*Greedy2* − *5* versus *MMR* − *5* for MovieLens 100K) and by 0.041 in the worst case (*Greedy2* − *20* versus *MMR* − *20* for MovieLens 1M).

We repeated the analysis of lowest scores and percentages, performed for *Top* − *N* and *Greedy2*, over *MVA*, *MMR*, and *Greedy2*. Results are summarized in Table 5.8, Figure 5.3, and Figure 5.4. Underlined values are statistically equivalent, according to a paired t-test with a 95% confidence interval.

Results in Table 5.8 lead to the conclusion that the worst item recommended by *Greedy2* tends to be better rated than the corresponding ones for *MVA* and *MMR*.

Predictor	Method	MovieLens 100K	MovieLens 1M	Jester 1
PureSVD	*MVA − 5*	3.923	4.128	1.092
	MMR − 5	3.884	4.120	1.292
	Greedy2 − 5	3.987	4.187	1.688
	MVA − 10	3.830	4.013	0.950
	MMR − 10	3.799	4.012	1.031
	Greedy2 − 10	3.881	4.065	1.323
	MVA − 20	3.720	3.901	0.864
	MMR − 20	3.698	3.900	0.892
	Greedy2 − 20	3.768	3.941	0.923
NNCosNgbr	*MVA − 5*	3.833	4.027	1.146
	MMR − 5	3.801	4.022	2.312
	Greedy2 − 5	3.896	4.082	2.356
	MVA − 10	3.768	3.927	1.338
	MMR − 10	3.760	3.926	1.559
	Greedy2 − 10	3.818	3.988	1.578
	MVA − 20	3.677	3.824	0.859
	MMR − 20	3.677	3.832	<u>0.939</u>
	Greedy2 − 20	3.732	3.902	<u>0.939</u>

Table 5.7. Mean ratings computed with *MVA*, *MMR*, and *Greedy2* for recommendation lists with sizes $N = 5, 10, 20$. $Greedy2 - 5$ corresponds to solutions with size $N = 5$ generated by $Greedy2$, for example. Reported results are averages of results obtained with 5-fold cross-validation.

Once again, values obtained with *MMR* were somewhat similar to those prompted by $Top - N$. *MVA* performed better than *MMR* with respect to the MovieLens datasets and the opposite was noticed with respect to Jester 1. In all cases, *Greedy2* led to superior or statistically equivalent lowest ratings, when compared with both baselines.

In terms of ratings for Jester 1, *Greedy2* exceeds *MVA* by 2.087 points in the best case (*Greedy2 − 5* and *MVA − 5*) and by 0.064 in the worst case (*Greedy2 − 20* and *MVA − 20*). As for the MovieLens datasets, *Greedy2* outperforms *MVA* by 0.133 points at most (*Greedy2 − 5* and *MVA − 5* for MovieLens 100K) and by 0.048 in the worst case (*Greedy2 − 20* and *MVA − 20* for MovieLens 100K). With respect to *MMR* for Jester 1, *Greedy2* surpassed it by 0.495 points in the best case (*Greedy2 − 5* and *MMR − 5*) and was surpassed by it by 0.012 points in the worst case (*Greedy2 − 10* and *MMR − 10*). Regarding the MovieLens datasets, *Greedy2* outperformed *MMR* by 0.205 points at most (*Greedy2 − 5* and *MMR − 5* for MovieLens 100K) and by 0.054 points at least (*Greedy2 − 20* and *MMR − 20* for MovieLens 100K).

As noticed in Table 5.6, recommendations generated for Jester 1 with NNCosNgbr were particularly similar for *Greedy2* and all baselines, with equivalent mean ratings for

Predictor	Method	MovieLens 100K	MovieLens 1M	Jester 1
PureSVD	*MVA − 5*	2.870	3.129	-4.295
	MMR − 5	2.798	3.114	-3.766
	Greedy2 − 5	3.003	3.217	-3.271
	MVA − 10	2.405	2.608	-5.918
	MMR − 10	2.339	2.612	-5.590
	Greedy2 − 10	2.481	2.683	-5.232
	MVA − 20	2.075	2.208	-6.349
	MMR − 20	2.037	2.209	-6.319
	Greedy2 − 20	2.123	2.263	-6.285
NNCosNgbr	*MVA − 5*	2.711	2.962	-4.287
	MMR − 5	2.663	2.953	-2.179
	Greedy2 − 5	2.812	3.057	-2.200
	MVA − 10	2.290	2.457	-5.937
	MMR − 10	2.272	2.466	-4.778
	Greedy2 − 10	2.359	2.556	-4.790
	MVA − 20	2.016	2.087	-6.350
	MMR − 20	2.011	2.111	-6.257
	Greedy2 − 20	2.065	2.201	-6.261

Table 5.8. Lowest ratings generated by *MVA*, *MMR*, and *Greedy2* for recommendation lists with sizes $N = 5, 10, 20$. Reported values are averages of results obtained with 5-fold cross-validation.

all N values. Figures 5.3 and 5.4 ratify the results illustrated by Figure 5.2. *Greedy2* wins over *MVA* for approximately 65% of users, and it is more effective when the adopted predictor is NNCosNgbr. With respect to *MMR*, *Greedy2* outperforms it for approximately 65% of users as well. Nonetheless, results varied more for *MMR*: in the graph associated with Jester 1 and NNCosNgbr, in particular, it won over *Greedy2* for approximately 48% of users. The percentages associated with *Greedy2* tended to increase as N growed as well, as noticed in Figure 5.2.

With respect to the predictors, NNCosNgbr is consistently associated with the best results for Jester 1. Regarding the MovieLens datasets, the reported absolute gains are similar for both predictors. In general, absolute and percentual gains were much higher for the Jester 1 dataset. Despite that, the gains obtained with *Greedy2* were consistent even when they were small.

Finally, the best baseline we studied in general, according to our experiments, was *MMR*, and the worst was *MVA*. It is important to bear in mind, nonetheless, that *MVA* is an application of Modern Portfolio Theory that was designed to improve ranking [Wang, 2009]. With respect to *MMR*, its main purpose is to increase recommendations' diversity without hurting their utility [Zuccon et al., 2012]. Ranking and diversity are aspects that are outside the scope of our work, and therefore these baselines were chosen exclusively because they also break with the *PRP* principle.

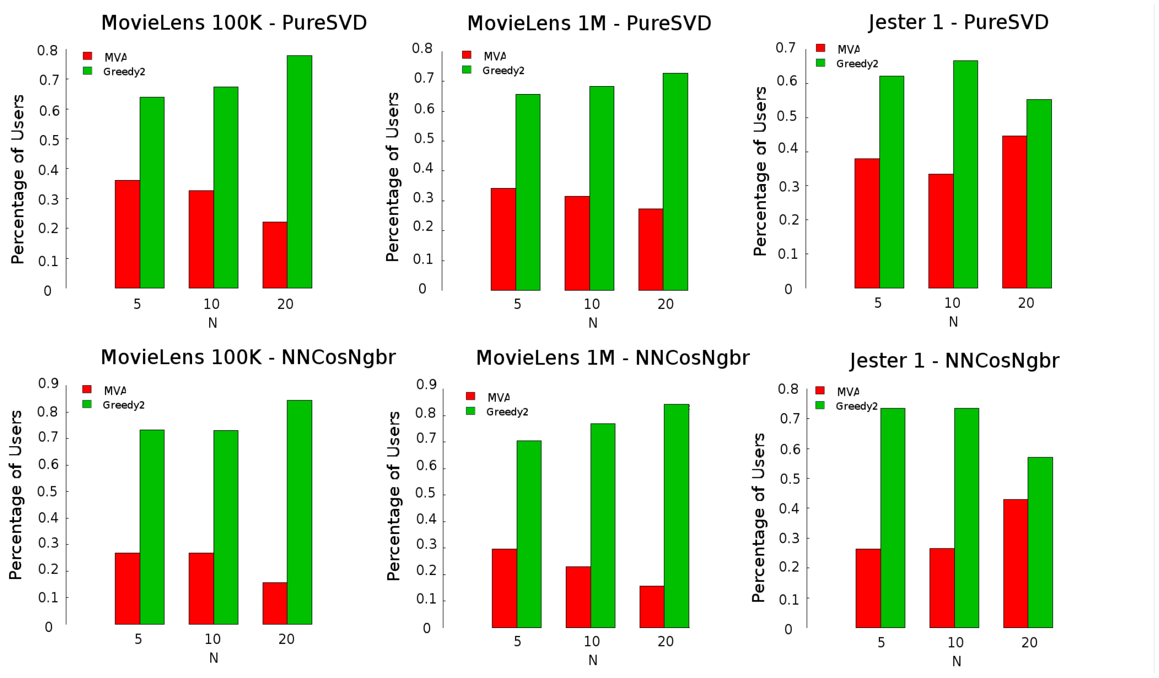

Figure 5.3. Percentages of users to which *MVA* and *Greedy2* have won over each other, in terms of highest mean rating given to generated recommendations.

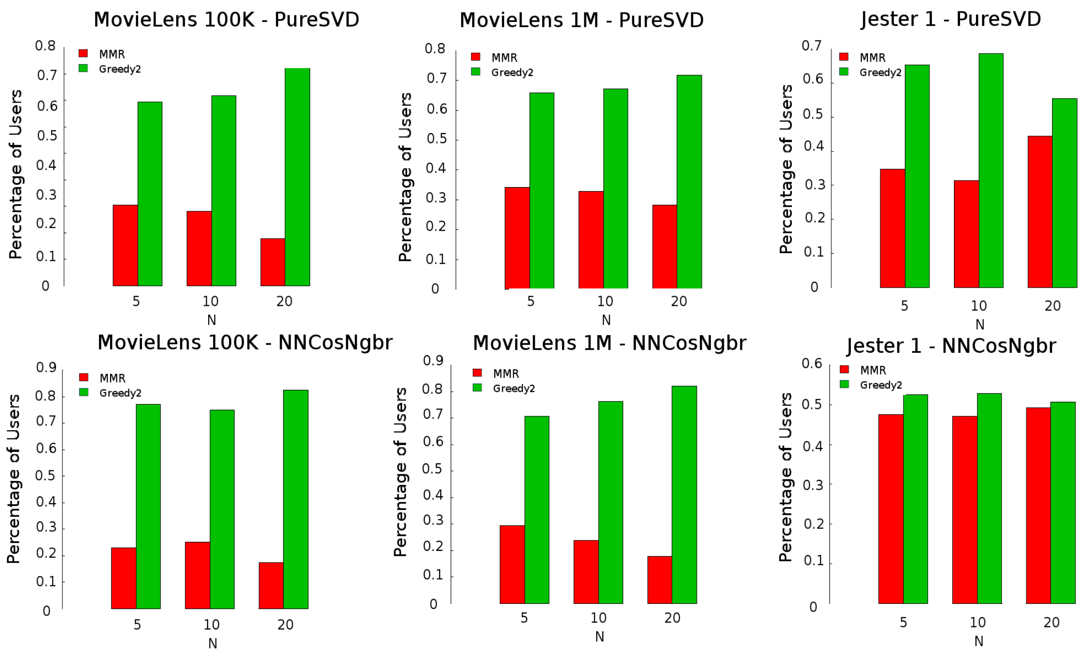

Figure 5.4. Percentages of users to which *MMR* and *Greedy2* have won over each other, in terms of highest mean rating given to generated recommendations.

5.4 Analysing the Scalability of Our Method

Results presented in Section 5.3 consistently indicate that *Greedy2*, by employing co-utility probabilities in its selection strategy, can improve recommendations. As a consequence, it is important to devise competitive implementations for *Greedy2* that scale in real-time situations.

Although *Greedy2* is polynomial and rather fast, there are some easy and important optimizations that make it scalable and competitive in practice. It is important, for example, to precompute and store all pairwise scores θ in a hash table as a pre-processing step. This offline computation speeds up the generation of solutions to *MSDP* by avoiding redundant computations of pairwise scores. Another improvement involves the use of memoization to reuse partial summations. Figure 5.5 illustrates the mean computation time per validation fold for each dataset, varying N and the predictor algorithm. All experiments were performed in a Pentium Dual-Core 2.0GHz with 2GB RAM. We decided to compare *Greedy2* with $Top - N$ because, from all studied methods, $Top - N$ is the fastest one in practice. [6]

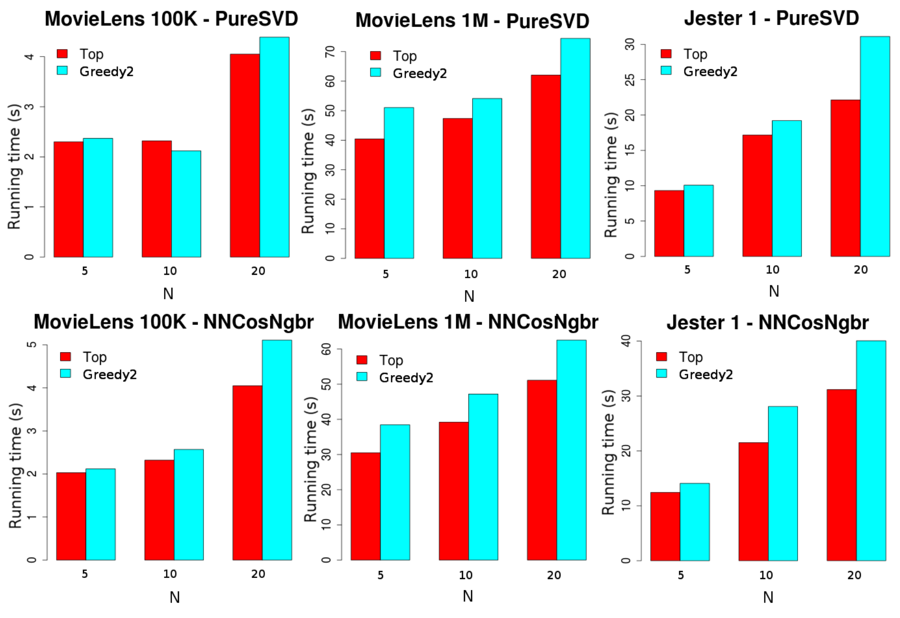

Figure 5.5. Mean running times per validation fold, in seconds, for different combinations of datasets and predictors, with $N = 5, 10, 20$.

Results in Figure 5.5 correspond to the mean aggregated running time for the generation of all recommendation lists concerning a validation fold. For higher values of N, the time difference between *Greedy2* and $Top - N$ could increase, but such analysis

[6]We used a $Top - N$ implementation with time complexity $O(K \log N)$, as explained in Section 4.2.1.

is not useful in real-world scenarios because N values are not big in practice [Ricci et al., 2011]. Therefore, for realistic values of N, *Greedy2* scales well and its mean running times per validation fold are only slightly worse than those obtained with $Top - N$. In spite of that, the time difference for generating a single recommendation list with all methods is irrelevant. Given that in real-world systems recommendation lists are generated once at a time via the interaction with users, *Greedy2* is a feasible alternative.

5.5 Relating Co-Utility and Diversity

In this section, we investigate whether items that are co-useful are necessarily similar. To perform this task, we rely on content similarity measures. We also compare the level of diversity in recommendations generated by *Greedy2*, $Top - N$, and *MMR*. We opted to contrast *Greedy2* with $Top - N$ to analyse whether the pairwise scores θ would hamper the diversity of recommendations generated by predictors PureSVD and NNCosNgbr. As for *MMR*, we wanted to understand how its results differ from those prompted by *Greedy2* and $Top - N$ – two methods that do not focus on diversity. Finally, we analyse if pairwise scores θ – i.e., co-utility probabilities – correlate with the cosine similarity.

There are several methods to measure recommendations' diversity [Vargas and Castells, 2011; Ricci et al., 2011]. In our scenario, it is important to choose a method that explores item content, as this information was not used by any of the studied algorithms. The use of content dissimilarities allows more impartial comparisons among these algorithms – especially with respect to *MMR*, as it already embeds rating distributions' dissimilarities in its optimization.

An advantage of exploring content instead of rating distributions is the interpretability of diversity results. For instance, stating that two books are different because their genres and authors are not the same is more interpretable than affirming it because their ratings are not alike. Most experiments in this section are exclusive to the MovieLens datasets because they have content information with which we can compute movie dissimilarities, in contrast to the Jester 1 dataset. To have an intuition about whether co-useful items share the same genres, we listed the pairs of movies with highest co-utility probabilities alongside their genres in common in Table 5.9.

Table 5.9 indicates that movies that are highly co-useful to users are not necessarily similar in terms of genres. 5 out of 10 movies have no genre in common. Although there may be other sources of similarities that we did not consider, such as actors in

MovieLens 100K		MovieLens 1M	
Pairs of Movies	Genres in Common	Pairs of Movies	Genres in Common
The Third Man and *Casablanca*	None	*Seven Samurai* and *Sanjuro*	*Action*
A Close Shave and *Wallace & Gromit*	*Animation*	*The Boat* and *Sanjuro*	*Action*
The Wrong Trousers and *Wallace & Gromit*	*Animation*	*GoodFellas* and *Sanjuro*	None
To Kill a Mockingbird and *Vertigo*	None	*Casablanca* and *Sanjuro*	None
Paths of Glory and *Dr. Strangelove*	*War*	*The Wrong Trousers* and *A Close Shave*	*Animation/Comedy*
My Life as a Dog and *His Girl Friday*	None	*The Great Escape* and *Sanjuro*	*Adventure*
12 Angry Men and *To Kill a Mockingbird*	*Drama*	*The Wrong Trousers* and *Wallace & Gromit*	*Animation*
Rear Window and *Vertigo*	*Thriller/Mistery*	*Yojimbo* and *The Bridge on the River Kwai*	*Drama*
When We Were Kings and *Star Wars Episode IV*	None	*Yojimbo* and *A Clockwork Orange*	None
Taxi Driver and *Wallace & Gromit*	None	*To Live* and *Boys Don't Cry*	None

Table 5.9. Top 10 pairs of movies with highest co-utility probabilities, computed with Empirical Bayes. Along with these pairs, the genres in common.

common, these results strengthen the hypothesis that co-utility does not imply similarity. For instance, *My Life as a Dog* is a drama released in the 1980s whereas *His Girl Friday* is reported as a comedy from 1940. It is important to note that most of these movies are very popular and received high ratings in websites such as IMDb and Rotten Tomatoes. [7] [8]

To further our understanding of how co-utility may relate to diversity, we aggre-

[7] `http://www.imdb.com/`

[8] `http://www.rottentomatoes.com/`

gated the diversity levels of recommendations generated with *Greedy2*, *Top* $-$ N, and *MMR* in Table 5.10. To compute movie dissimilarities, we calculated Jaccard's coefficient over movies' corresponding genres. To compare the diversity levels of *Greedy2*, *Top* $-$ N, and *MMR*, we used the ILD metric described in Section 4.3.

Predictor	Method	MovieLens 100K	MovieLens 1M
PureSVD	*Top* − *5*	0.8558	0.8305
	MMR − *5*	0.8572	0.8309
	Greedy2 − *5*	0.8554	0.8302
	Top − *10*	0.8619	0.8392
	MMR − *10*	0.8621	0.8395
	Greedy2 − *10*	0.8615	0.8392
	Top − *20*	0.8645	0.8444
	MMR − *20*	0.8648	0.8445
	Greedy2 − *20*	0.8643	0.8444
NNCosNgbr	*Top* − *5*	0.8571	0.8360
	MMR − *5*	0.8585	0.8365
	Greedy2 − *5*	0.8565	0.8358
	Top − *10*	0.8628	0.8429
	MMR − *10*	0.8621	0.8428
	Greedy2 − *10*	0.8628	0.8429
	Top − *20*	0.8649	0.8458
	MMR − *20*	0.8646	0.8458
	Greedy2 − *20*	0.8650	0.8458

Table 5.10. Mean ILD values generated by for recommendation lists with sizes $N = 5, 10, 20$. *Greedy2* $-$ *5* corresponds to solutions with size $N = 5$ generated by *Greedy2*, for example. Reported values are averages of results obtained with 5-fold cross-validation.

We performed paired t-tests for each *Top* $-$ N/*Greedy2* and *MMR*/*Greedy2* pair in Table 5.10 with a 95% confidence interval and none of the results were statistically different. This is an evidence that *Greedy2* is not likely to hurt recommendations' diversity when compared to *Top* $-$ N and may generate as diversified recommendations as the *MMR* algorithm implemented with the Kullback-Leibler distance. Nonetheless, it is important to highlight that different implementations of *MMR* may yield mean ILD values that are statistically better than those related to *Greedy2*. As the focus of this work is not on diversity, though, we only implemented *MMR* with the Kullback-Leibler distance. Another important consideration is that, if *Greedy2* starts hampering diversity in any specific scenario, it is always possible to penalize pairwise scores that correlate with very redundant pairs of items.

To finish our analysis, we investigated whether co-utility probabilities correlate positively with the cosine similarity, a very popular metric for assessing similarities in recommender systems [Ricci et al., 2011; Jannach et al., 2011]. For each pair of items i and j, we computed its co-utility probability via Empirical Bayes and its cosine similarity. We then correlated the resulting distributions using the Pearson correlation coefficient. For datasets MovieLens 100k, MovieLens 1M, and Jester 1, correlations were -0.472, -0.497, and 0.968 respectively. These values show that the correlation between co-utility probabilities and the cosine similarity can be negative – moderately negative, in particular, which may indicate a mild discorrelation or no correlation at all [Zaki and Meira, 2014]. This implies that co-utility probabilities cannot necessarily be substituted by the cosine similarity. Furthermore, if one assumes that a high cosine similarity between two items is a proxy for low diversity, then co-utility probabilities do not necessarily worsen recommendations' diversity, as they are not always positively correlated with the cosine similarity.

The high positive correlation for Jester 1 means that co-useful jokes also share a similar rating pattern. Then again, if we assume that a high cosine similarity implies low diversity, then Jester 1 users have a redundant taste for jokes, as recommendations get better rated when co-utility is taken into consideration. The higher the co-utility, the more similar jokes would potentially be. Moreover, in the case of Jester 1, this high positive correlation indicates that it is possible to adopt the cosine similarity in lieu of co-utility probabilities.

Chapter 6

Conclusions and Future Work

In this thesis, we investigated how co-utility probabilities can be estimated and exploited in order to improve recommendations' utility. The main intuition behind this project is that not only individual predicted scores should be taken into account by recommender systems. Relations between all candidate items also play an important role on the utility of recommendations.

We proposed two ways to estimate co-utility probabilities: Maximum Likelihood and Empirical Bayes [Bishop, 2006]. Empirical Bayes is theoretically more robust in scenarios where data is sparse. Despite that, it prompted results that were statistically equivalent to those generated by Maximum Likelihood, as detailed in Chapter 5.

We modelled the combination of individual predicted scores and co-utility probabilities as a linear combination. Afterwards, we posed the task of finding the best subset of candidate items for recommendation as an optimization problem. The problem mapped trivially into the Max-Sum Dispersion Problem, abbreviated as *MSDP* [Borodin et al., 2012]. This problem has been thoroughly studied by the Operations Research community and was previously used to capture the semantics of diversity in Information Retrieval and Recommender Systems contexts.

We implemented two heuristics and one exact formulation to *MSDP*: *Greedy1*, *Greedy2*, and *Exact*. We also implemented three baselines: $Top - N$, *MVA*, and *MMR*. We describe our method and the baselines in Chapter 4. Comparisons between *Greedy1* and *Greedy2* were performed and the mean ratings obtained from recommendation lists yielded by both heuristics were mostly statistically equivalent. We then contrasted *Greedy2* to *Exact* and results were also statistically equivalent, which indicates that *Greedy2* is likely to be a good heuristics in practice.

With respect to the baselines, we compared *Greedy2* with them by contrasting mean ratings, lowest ratings, and percentages of users to which each method has won over each other. *Greedy2* has performed consistently better than all baselines. By comparing the running times of *Greedy2* and $Top - N$, we also present evidence that *Greedy2* is scalable and therefore useful for real-world scenarios. Finally, by contrasting *Greedy2*, $Top - N$, and *MMR* in terms of diversity, we show that the exploitation of co-utility probabilities does not necessarily hurt recommendations' diversity.

Throughout this thesis, we showed that co-utility probabilities are an important evidence for recommender systems. Hence we intend to develop Learning to Rank algorithms that embed them in a near future. We also want to extend our method to hybrid recommenders, by using content information to compute co-utility probabilities.

Bibliography

Adomavicius, G. and Tuzhilin, A. (2005). Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734--749.

Baeza-Yates, R. and Ribeiro-Neto, B. (2011). *Modern Information Retrieval: The Concepts and Technology behind Search (2nd Edition)*. ACM Press Books.

Bell, R. and Koren, Y. (2007). Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2).

Bessa, A., Veloso, A., and Ziviani, N. (2013). Using mutual influence to improve recommendations. In *Proceedings of the 20th String Processing and Information Retrieval Symposium*, pages 17--28.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bookstein, A. (1983). Information retrieval: A sequential learning process. *Journal of the American Society for Information Science*, 34(5):331--342.

Borodin, A., Lee, H. C., and Ye, Y. (2012). Max-sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st Symposium on Principles of Database Systems*, pages 155--166.

Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithm for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43--52.

Carbonell, J. and Goldstein, J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335--336.

Casella, G. (1985). An introduction to empirical bayes data analysis. *The American Statistician*, 39(2):83--87.

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM conference on Recommender Systems*, pages 39--46.

Deshpande, M. and Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143--177.

Elton, E. J., Gruber, M. J., Brown, S. J., and Goetzmann, W. N. (2009). *Modern Portfolio Theory and Investment Analysis*. Wiley.

Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133--151.

Gollapudi, S. and Sharma, A. (2009). An axiomatic approach for result diversification. In *Proceedings of the 18th international conference on World Wide Web*, pages 381--390.

Goyal, A. and Lakshmanan, L. V. S. (2012). Recmax: Exploiting recommender systems for fun and profit. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1294--1302.

Hansen, D. L. and Golbeck, J. (2009). Mixing it up: Recommending collections of items. In *Proceedings of the 27th ACM Conference on Human Factors in Computing Systems*, pages 1217--122.

Hassin, R., Rubinstein, S., and Tamir, A. (1997). Approximation algorithms for maximum dispersion. *Operations Research Letters*, 21:133--137.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer.

Ieong, S., Mishra, N., and Sheffet, O. (2012). The random shopper model: Predicting preference flips in commerce search. In *Proceedings of the 29th International Conference on Machine Learning*.

ILOG, Inc (2013). ILOG CPLEX: High-performance software for mathematical programming and optimization. See `http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r5/index.jsp`.

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2011). *Recommender Systems: An Introduction*. Cambridge University Press.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426--434.

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

Lipcon, T. (2007). *Algorithms for Collaborative Prediction*. PhD thesis, Brown University.

Nemhauser, G. and Wolsey, L. (1988). *Integer and combinatorial optimization*. Wiley.

Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q. (2008). One-class collaborative filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 502--511.

Papagelis, M. and Plexousakis, D. (2005). Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. *Engineering Applications of Artificial Intelligence*, 18(7):781--789.

Passos, A., Gael, J. V., Herbrich, R., and Paquet, U. (2011). A penny for your thoughts? the value of information in recommendation systems. In *Proceedings of the 1st Neural Information Processing Systems Workshop on Bayesian Optimization, Experimental Design, and Bandits*, pages 9--14.

Pu, P., Chen, L., and Hu, R. (2012). Evaluating recommender systems from the user's perspective: survey of the state of the art. *User Modeling and User-Adapted Interaction*, Volume 22(4–5):317--355.

Ravi, S., Rosenkrantz, D., and Tayi, G. (1994). Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299--310.

Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors (2011). *Recommender Systems Handbook*. Springer.

Robertson, S. E. (1977). The probability ranking principle in information retrieval. *Journal of Documentation*, 33(4):294--304.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285--295.

Tarlow, D., Givoni, I. E., and Zemel, R. S. (2010). Hop-map: Efficient message passing with high order potentials. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*.

Toffler, A. (1970). *Future Shock*. Random House.

Tversky, A. (1972). Elimination by aspects: A theory of choice. *Psychological Review*, 79(4):281--299.

Vargas, S. and Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the 5th ACM conference on Recommender Systems*, pages 109--116.

Vieira, M. R., Razente, H. L., Barioni, M. C. N., Hadjieleftheriou, M., Srivastava, D., Jr., C. T., and Tsotras, V. J. (2011). On query result diversification. In *Proceedings of the 27th IEEE International Conference on Data Engineering*, pages 1163--1174.

Wang, J. (2009). Mean-variance analysis: A new document ranking theory in information retrieval. In *Proceedings of the 31st European Conference on Information Retrieval*, pages 4--16.

Weston, J. and Blitzer, J. (2012). Latent structured ranking. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, pages 903--913.

Xiong, C., Taifeng Wang, Wenkui Ding, Y. S., and Liu, T.-Y. (2012). Relational click prediction for sponsored search. In *Proceedings of the 5th International Conference on Web Search and Web Data Mining*, pages 493--502.

Zaki, M. and Meira, W. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press.

Zhang, M. and Hurley, N. (2008). Avoiding monotony: Improving the diversity of recommendation lists. In *Proceedings of the 2nd ACM conference on Recommender Systems*, pages 123--130.

Zuccon, G., Azzopardi, L., Zhang, D., and Wang, J. (2012). Top-k retrieval using facility location analysis. In *Proceedings of the 34th European Conference on Information Retrieval*, pages 305--316.