RECOMENDAÇÃO DE REVISÕES ORIENTADA A RANKING

LUCIANA BICALHO MAROUN

RECOMENDAÇÃO DE REVISÕES ORIENTADA A RANKING

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Mirella Moura Moro
Coorientadora: Jussara Marques de Almeida Gonçalves

Belo Horizonte
Março de 2016

LUCIANA BICALHO MAROUN

RANKING-ORIENTED REVIEW RECOMMENDATION

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

Advisor: Mirella Moura Moro
Co-Advisor: Jussara Marques de Almeida Gonçalves

Belo Horizonte
March 2016

Maroun, Luciana Bicalho

M356r Ranking-oriented review recommendation / Luciana Bicalho Maroun. — Belo Horizonte, 2016
xxvii, 124 f. : il. ; 29cm

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

Orientadora: Mirella Moura Moro
Coorientadora: Jussara Marques de Almeida Gonçalves

1. Computação – Teses. 2. Recuperação da informação. 3. Sistemas de recomendação. I. Orientadora. II. Coorientadora. II. 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

Ranking-oriented review recommendation

LUCIANA BICALHO MAROUN

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROFA. MIRELLA MOURA MORO - Orientadora
Departamento de Ciência da Computação - UFMG

PROFA. JUSSARA MARQUES DE ALMEIDA GONÇALVES - Coorientadora
Departamento de Ciência da Computação - UFMG

PROF. ANA PAULA COUTO DA SILVA
Departamento de Ciência da Computação - UFMG

PROF. MARCELO GARCIA MANZATO
Departamento de Ciências da Computação - USP

PROF. PEDRO OLMO STANCIOLI VAZ DE MELO
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 03 de março de 2016.

A Antônio Carlos, Valéria, Gabriel e Júlio, por tornarem tudo possível.

Agradecimentos

Agradeço aos meu pais, Antônio Carlos e Valéria, pelo amor, dedicação e suporte incondicionais, por sempre acreditarem em mim, e por serem os meus maiores exemplos. Nunca conseguirei retribuir tudo o que fizeram por mim, mas procurarei fazê-lo eternamente. Ao meu irmão, Gabriel, pela autenticidade e leveza, que me servem de inspiração, e por ser um presente em minha vida. A Júlio, pelo amor, companheirismo e apoio imensuráveis; por ser a minha completude. As palavras falham em descrever o quão abençoada sou por tê-lo ao meu lado. A Laís, pela compreensão, surpreendente afinidade e presença constante; pela amizade pura e sincera. Aos meus avós, Dodora, Rômulo, Regina e Michel, por serem os pilares essenciais. À minha família e à de Júlio, por torcerem por mim. A Deus, pelas oportunidades e pelas pessoas incríveis que colocou no meu caminho.

Não poderia deixar de agradecer à Mirella Moro e à Jussara Almeida, pela orientação, dedicação e ensinamentos que vão além da academia. Aos professores Pedro Olmo, Ana Paula Couto da Silva e Renato Assunção, pela colaboração e discussões. A Thanis, Izabela, Elverton e Osvaldo, pela partilha das dificuldades durante o trajeto que se estendeu da graduação até o mestrado. Aos colegas do LBD, pela solicitude em contribuir e tirar dúvidas; em especial, à Michele Brandão, ao Sérgio Canuto, ao Daniel Hasan e ao Daniel Xavier. À Marisa Vasconcelos, pela disponibilidade em ajudar. À UFMG, ao DCC e ao CNPq, pelo suporte. Por fim, agradeço a todos que contribuíram e acreditaram neste trabalho.

"A vida é assim: esquenta e esfria,
aperta e daí afrouxa,
sossega e depois desinquieta.
O que ela quer da gente é coragem. "
(Guimarães Rosa)

Resumo

A leitura de revisões *online* antes de efetuar uma compra se tornou habitual, uma vez que os usuários as utilizam para realizar escolhas mais bem fundamentadas. Entretanto, o crescente volume de revisões representa uma limitação para a sua eficácia de tal forma que muitas técnicas tentam predizer a qualidade global das mesmas. Porém, a utilidade pode não ser padronizada para todos os usuários devido a diferentes experiências e preferências. Dessa forma, a recomendação de revisões de modo personalizado é provavelmente mais precisa. Considerando essa hipótese, existem algumas estratégias personalizadas para recomendação de revisões que, no entanto, não foram comparadas entre si. Além disso, tais técnicas são tipicamente avaliadas usando uma métrica de regressão, enquanto uma de ranking seria mais adequada por representar melhor o resultado final apresentado aos usuários. A finalidade deste trabalho consiste em analisar e comparar os métodos de recomendação de revisões especializados assim como outros genéricos que não foram avaliados sob um ponto de vista personalizado e de ranking. Além disso, objetiva-se identificar as propriedades que tornam um método eficaz para o problema de recomendação de revisões. Usando um conjunto de dados do Ciao UK, os experimentos realizados consideram ajuste de parâmetros, métricas de regressão e de ranking, e comparação por meio de testes estatísticos. Os resultados experimentais indicam que as abordagens mais simples superam as mais complexas, e atributos observados assim como a avaliação sob uma perspectiva de ranking são caracterísitisticas importantes para esse problema. Todavia, as técnicas que realizam regressão de dados observados obtêm, supreendentemente, o melhor desempenho para ranking: uma Máquina de Vetor de Suporte para Regressão com kernel linear é a melhor abordagem, seguida pela Regressão Linear. Ambas ultrapassam estatisticamente métodos de aprendizado de ranking, sistemas de recomendação tradicionais e abordagens especializadas. Por fim, é proposta uma solução baseada no SVR que otimiza um erro de ranking em conjunto e que explora a relação entre autor e leitor, provendo uma melhoria pequena, porém significativa em relação à versão original.

Palavras-chave: Revisões *Online*, Utilidade Personalizada, Sistemas de Recomendação, Recomendação Top-n, Ranking.

Abstract

Reading online reviews before a purchase has become a customary action, as users rely on them to perform more grounded choices. Nevertheless, the increasing volume of reviews works as a barrier to their effectiveness so that many approaches try to predict reviews' global quality. The perceived helpfulness, however, may not be standardized to all users due to different backgrounds and preferences. Thus, recommending reviews in a personalized fashion is likely to be more accurate. Regarding this hypothesis, there are some personalized strategies for review recommendation that, however, are still not compared against each other. Also, such techniques are typically evaluated with a regression metric, whereas we argue that a ranking one is more adequate due to representing better the final result presented to the users. The goal of this work is to analyze and compare such specialized review recommenders as well as generic ones that have not been previously evaluated under a personalized and ranking viewpoint. We also aim at deriving insights about the properties of an effective predictor for the review recommendation problem. Using a dataset from Ciao UK, the experiments conducted here consider parameter tuning, both regression and ranking metrics, and comparison through statistical tests. Our experimental results indicate that simpler approaches overcome complex ones, and observed features as well as evaluation under a ranking perspective are important traits for this problem. Nonetheless, techniques performing regression on observed data obtain, surprisingly, the best performance for ranking: Support Vector Regression (SVR) with linear kernel is the best, followed by Linear Regression. Both statistically surpass learning to rank methods, traditional recommender systems and specialized approaches. Finally, we propose an SVR-based method that jointly optimizes a ranking loss and that exploits the relation between author and reader, providing a slight but significant improvement over the original version.

Palavras-chave: Online Reviews, Personalized Helpfulness, Recommender Systems, Top-n Recommendation, Ranking.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Online reviews platforms, such as Yelp[1], Amazon[2], TripAdvisor[3] and Ciao UK[4], play an important role on users' search for goods and services. To illustrate, a study conducted in the United States in 2014 points out that 88% of consumers have read online reviews before a purchase (vs. 85% in 2013) and 39% read them regularly (vs. 32% in 2013) [Anderson, 2014]. Besides, 72% of interviewees said that positive reviews increase their trust in a local business, and 88% trust online reviews as much as personal recommendations. The impact on sales was also measured quantitatively: increasing half star on Yelp raises businesses' sells by 19 percentage points [Anderson and Magruder, 2012]. Thus, online reviews also impact business owners.

For consumers, finding good reviews has become very difficult as too many are being produced. In such scenario, it is impossible to read every review whereas not all of them have the information readers are seeking. Thus, the great volume of reviews limits their effectiveness as a source of valuable information.

Reviews differ greatly in style and quality: some are detailed descriptions of positive and negative aspects, others superficial and subjective. Besides, the same review may be considered helpful by some and not helpful by others. In fact, the concept of *helpfulness* is not necessarily homogeneous for different users, as they have different backgrounds and preferences. In general terms, customers demand a technique for selecting useful reviews considering personal needs and preferences.

One way to meet such demand consists in defining a helpfulness index for each review and filtering the most suitable ones. Indeed, there are plenty of non-personalized solutions predicting a global helpfulness for each review, which is common to all read-

[1]**Yelp.** `http://www.yelp.com`
[2]**Amazon.** `http://www.amazon.com`
[3]**TripAdvisor.** `http://www.tripadvisor.com`
[4]**Ciao UK.** `http://www.ciao.co.uk`

ers [Lee and Choeh, 2014; Lu et al, 2010; O'Mahony and Smyth, 2009; Zhang and Tran, 2011]. Nonetheless, recent solutions demonstrate improvement by dealing with this problem in a *personalized* way, i.e., considering users' idiosyncrasy for helpfulness prediction [Moghaddam et al., 2012; Tang et al, 2013].

Other approaches may be used for processing the great number of online reviews: summarizing reviews' content [Hu and Liu, 2004; Zhan et al., 2009], allowing search of features inside reviews [Duan et al., 2013; Ganesan and Zhai, 2012] and recommending products based on such features [McAuley and Leskovec, 2013; Xu et al., 2012]. These options spare the need for reading reviews in their original format and may not be the best solution for a user demanding to deeply understand a product or service. Certain users might want to know exactly what others have perceived about a product and decide on their own if it suits their needs. Then, filtering original reviews is the desired solution in this case and is the focus of our work.

Next, we describe the context of a review platform in Section 1.1, discuss challenges in review exploitation by readers in Section 1.2, state the main contributions of our work in Section 1.3 and, finally, present text organization in Section 1.4.

1.1 Review Platform Scenario

A review-enabled application allows users to express their opinions through *reviews* about products, businesses, locations, news, movies and many others, generically referred to as *products*. An online review consists of a report available online created by a certain individual about his or her experience involving a product. Such platforms are impacting the behavior of consumers, who more often consult online reviews before a buying decision. As a consequence, users can make more informed and grounded choices and avoid frustrations after a purchase.

Such systems also provide a feedback for businesses' owners, allowing them to plan improvements upon that. Additionally, as Simonson and Rosen [2014] emphasize, products were previously sold mainly due to brand knowledge. Nowadays, a few consumers writing online reviews is enough for many people to know whether a product meet their needs, thus enabling the rise of unknown brands. This leads to a fairer market where quality and not only renown are considered.

There are several studies attempting to identify the motivations for a user to write a review [Christodoulides and Jevons, 2011; Dahlgren et al., 2015; Dellarocas and Narayan, 2006; Hennig-Thurau et al., 2004; Rensink, 2013]. They may want to add value to an online community, then they write to be a part of such community, to

exert power and to benefit others. They are also inspired to contribute due to their own need for consuming reviews; therefore, they want to do their part. Another reason lies in self-enhancement as users want to have good thoughts about themselves and their statuses [Dahlgren et al., 2015]. They may also feel frustrated with a purchase and write a review to release anxiety and even penalize the provider. On the other hand, a positive experience may create a gratitude desire for helping the company [Rensink, 2013]. As a drawback, there may be malicious users aiming at arbitrarily promoting or penalizing products. In short, different reasons lead users to generate their own experience report through reviews.

The elements of a review vary a little among different applications. In general, a user inserts a *rating* for a product under evaluation, between zero and five stars for instance, and composes a *text* justifying the rating and including pros and cons. In some cases, reviews may be enriched with media, such as images and videos. Moreover, a user is able to interact with others by evaluating reviews through a *helpfulness vote*. Such vote may be either a grade in a range, between zero and five for example, or a binary evaluation, i.e., positive or negative. The combination of all votes composes the *overall helpfulness* of a review, computed using some aggregating function. In some scenarios, inserting a *comment* about a review is also possible.

Thereafter, items have an *overall rating* in the platform (usually the calculated mean of ratings) and a popularity degree given by the *number of reviews*; while reviews have equivalently an *overall helpfulness* and popularity as the *number of votes*. For example, in Yelp and TripAdvisor, the overall helpfulness is the total number of positive votes; in Amazon, it is the ratio of positive by the total number of votes; and in Ciao UK, it is the average of vote values, which are in a range[5]. Users also have a profile, which may include the number of written reviews, the distribution of given ratings, the average helpfulness of written reviews, date of registration, among several others. In fact, many features may be derived from related reviews and votes. Some applications even allow the interaction among users through a social network whose links may either represent friendship, as in Yelp, or trust, as in Ciao UK.

An application allowing evaluation of users through reviews is, therefore, an environment full of useful information that may be obtained direct or indirectly. For that reason, reviews have a potential of developing *data-driven products*, applications that provide relevant information from complex automated analyses [Fu and Asorey, 2015].

[5]In Ciao UK, the arithmetic mean is not used, but a weighted mean considering users reputation (which are not publicly disclosed). Source: `http://www.ciao.co.uk/faq/rating-product-reviews,48#answer_5`

1.2 Open Challenges

Although consumers depend on reviews to make decisions, searching for suitable ones is not easy. The great adherence of users submitting reviews derives an intractable amount of information with high variability in style and quality. Indeed, Yelp has more than 90 million reviews[6]; and TripAdvisor, more than 290 million with an average of 55 reviews per business[7], both by the end of the third quarter of 2015. On the other hand, a survey indicates that 67% of users read up to 6 reviews; 85%, up to 10; 93%, up to 20; remaining only 7% who read more than 20 [Anderson, 2014]. Thus, whereas users are not willing to read many reviews, a lot of them are being produced such that extracting relevant information from these extensive data becomes imperative.

As a real example, Meyers[8] explored the difficulty in investigating the quality of popular products using Kindle Fire as example, which had 10,859 reviews on Amazon at the time of writing the article (on April, 2013)[9]. Considering an optimistic scenario, where reading a review demands approximately five seconds and only reviews of one and five starts are investigated, 7,208 in total: reading all would require 10 hours. This estimation motivates the current impossibility of scanning all reviews and the importance of filtering those more helpful to the end user.

To solve this problem, many review-enabled applications allow users to assess reviews through a helpfulness evaluation (e.g., from zero to five). Still, there are many reviews without evaluation and more being produced every day. In Amazon, for instance, only 10% of reviews have at least 10 helpfulness evaluations[10]. Hence, having all of them evaluated by a representative amount of users is practically impossible. Also, reviews with many positive votes are often ranked on top in a *rich-get-richer effect*, so that new ones are hardly considered useful [Moghaddam et al., 2012].

Several approaches [Liu et al., 2008; Lee and Choeh, 2014; Lu et al, 2010; Martin and Pu, 2014; O'Mahony and Smyth, 2009; Zhang and Tran, 2011] try to overcome the sparseness and the rich-get-richer effect in a non-personalized fashion. They predict a global helpfulness score by considering aspects such as author's reputation and textual statistics, for example, from part-of-speech tagging and sentiment analysis. Nonetheless, users have different backgrounds and preferences. For instance, a review about a photographic camera written by a professional photographer may be useful for another

[6] **Yelp – About Us.** `http://www.yelp.com/about`

[7] **TripAdvisor – Fact Sheet.** `http://www.tripadvisor.com/PressCenter-c4-Fact_Sheet.html`

[8] **How I Wish Amazon Reviews Worked, by Dr. Peter J. Meyers.** April 2013. `http://moz.com/blog/how-i-wish-amazon-reviews-worked`

[9] The same version of Kindle Fire contained 25,501 reviews in February, 2016.

[10] **A Statistical Analysis of 1.2 Million Amazon Reviews, by Max Woolf.** June 2014. `http://minimaxir.com/2014/06/reviewing-reviews`

expert, but not equally for a common user, who is more interested in basic resources and ease of use. Consequently, the helpfulness of a review is unlikely to be equally perceived by all readers.

Despite the efforts to design personalized recommendations in other domains (e.g., product recommendation [Balakrishnan and Chopra, 2012; Koren, 2008; Machanavajjhala et al., 2011]), only recent works consider *personalized* solutions for review recommendation [Moghaddam et al., 2012; Tang et al, 2013]. A personalized recommender system suggests new items (reviews, products, services, locations, etc.) for a certain user by predicting individual interest, on contrary of non-personalized recommendation based solely on general aspects of items, such as popularity.

Furthermore, although some of techniques for review recommendation exist, their experimental evaluations have not considered yet many existing techniques nor the best configuration of each under a unified statistical experimental project. Therefore, our goal is to compare existing approaches for review recommendation, both generic and specialized ones, divided in five paradigms: (*i*) mean-based predictors; (*ii*) general purpose regressors; (*iii*) learning to rank methods; (*iv*) general-purpose recommender systems; and (*v*) review recommender systems, the specialized approaches.

Our experiments differ from others in three ways. First, to the best of our knowledge, we are the first to compare a rich set of techniques for review recommendation, comprising of specialized solutions and other personalized competitive approaches. BETF [Moghaddam et al., 2012] and CAP [Tang et al, 2013], the specialized approaches, are not compared against each other in their original publications. Also, Linear Regression [Murphy, 2012] and Support Vector Regression (SVR) [Smola and Schölkopf, 2004] were considered for predicting review helpfulness [Moghaddam et al., 2012; Tang et al, 2013; Vasconcelos et al., 2015], but using *only* non-personalized features, whereas we apply personalized ones also. Moreover, the parameters and the versions of the baseline algorithms are not clear in existing evaluations. Overall, competitive approaches were not yet considered in their full potential, leaving unknown which the best solution is. We are also the first to apply learning to rank predictors and Regression-based Latent Factor Model (RLFM), a sophisticated recommender system, for review recommendation task.

Second, we compare the methods with statistical significance. This is accomplished by executing experiments in multiple runs: five different training and test sets, and three repetitions for non-deterministic methods due to stochastic variability. Then, our comparison provides a confidence of 95%. Specialized approaches are non-deterministic and, yet, variability and significance are not reported on the experiments previously conducted with them [Moghaddam et al., 2012; Tang et al, 2013].

Finally, we consider a ranking metric for evaluation, since the ultimate goal is to display reviews sorted decreasingly by helpfulness. Recommendation of reviews is intrinsically a top-n recommendation task [Cremonesi et al., 2010], whose purpose is to compose a selection of items, preferentially ranked, and should be evaluated as such. Evaluating recommender systems through ranking metrics is not a new proposal [Balakrishnan and Chopra, 2012; Christakopoulou and Banerjee, 2015; Cremonesi et al., 2010; Koren, 2008], whereas regression metrics are still common [Koren et al., 2009; Moghaddam et al., 2012; Tang et al, 2013]. However, such metrics suffer from the following drawbacks: they aim at *exact* prediction of evaluations, then disagreeing with the purpose of top-n recommender systems, for which a good *ordering* of items is preferred; and they consider equally important the prediction of all relevance levels, whereas higher ones should have priority as the end user is mainly interested on them [Christakopoulou and Banerjee, 2015]. Exact prediction is too restrictive for this problem and, when aiming at such goal, hardly a perfect model is built. The existing errors may cause an ordering that is not effective since this is not the objective. Indeed, absolute predictions are not even displayed to the end user, they are only used for ranking. By using a ranking-oriented definition and metric, the focus is only in what matters: a good *sorted selection* of items.

1.3 Main Contributions

Given the great volume of reviews and the diverse perception of helpfulness by users, the purpose of this work is to evaluate existing approaches and investigate key aspects for designing a **review recommender system** with a ranking goal. Specifically, we evaluate current solutions for review recommendation with a ranking perspective and statistical tests, highlight the properties of effective solutions, pursue extensions and define promising future work. Overall, our contributions are summarized as follows.

(i) **Evaluating techniques from multiple paradigms in a real dataset.** We compare a rich set of the techniques for review recommendation from five different paradigms. We describe such techniques in deep, allowing the detection of causes for experimental results. We consider a real dataset and perform investigations of helpfulness voting patterns to substantiate experiments and solutions. Our experimental design is *unified* and fills a gap from previous experiments regarding this problem. Some of the techniques whose source code were not found are implemented from scratch and are made publicly available.

(*ii*) **Modeling and evaluating review recommendation from a ranking perspective.** Previous techniques defined the problem as a rating prediction, whose goal was to predict absolute values of helpfulness. However, the final result presented to users is a ranking of reviews, then absolute predictions are unnecessary. Therefore, we focus on a ranking-oriented perspective of the problem and also contrast with commonly accepted regression goal.

(*iii*) **Identifying key design aspects and improving upon the state-of-the-art.** By comparing techniques from multiple paradigms and with very different designs, we are able to identify the properties that perform well for review recommendation problem. Additionally, we define new features and a new technique, Combined Ranking and SVR (CoRaSVR), and both slightly but significantly outperform the state-of-the-art of review recommendation.

1.4 Text Organization

The remainder of the work is organized as follows. Chapter 2 defines basic concepts and states the problem. Chapter 3 addresses related work. Chapter 4 presents the dataset used and investigates key properties for review recommendation task. Chapter 6.1 describes the techniques considered. Chapter 6 presents the statistical experimental design and the evaluation of techniques regarding parameter tuning, performance comparison, and extensions. Lastly, Chapter 7 addresses conclusions and future work.

Chapter 2

Concepts and Problem Definition

Research on recommender systems is relatively recent, comprising an individual area of study in the mid-1990's [Ricci et al, 2010]. Great attention was drawn to such research topic due to a Netflix competition, which awarded a million dollar for those who improved the performance of Netflix's movie recommendation in 10% [Koren et al., 2009].

The motivations behind a recommender system service provider lie in several aspects. Selling more, gaining user loyalty, improving user satisfaction and understanding user's needs are examples of providers' interests. Consumers also are driven to participate in order to find good items that otherwise would be difficult to encounter, to improve their profile and have more accurate suggestions, and to express ideas, help and influence others [Ricci et al, 2010]. In short, recommender systems may benefit both service providers and consumers.

In this chapter, we first introduce basic concepts related to recommender systems, their dynamics and types in Section 2.1. Then, we present the specific case of review recommendation and state the problem in Section 2.2. Finally, we address concluding remarks in Section 2.3.

2.1 Recommender Systems Basics

The initial development of recommender systems was built upon the observation that the majority of people rely on recommendations from peers and experts in daily decision-making [Ricci et al, 2010]. The simplest recommender technique suggests the most popular items to a user, in a non-personalized fashion. This practice is reasonable since, for an unknown user, the best solution lies in recommending for an average person. Indeed, the most popular items are more likely to be enjoyed than a random one.

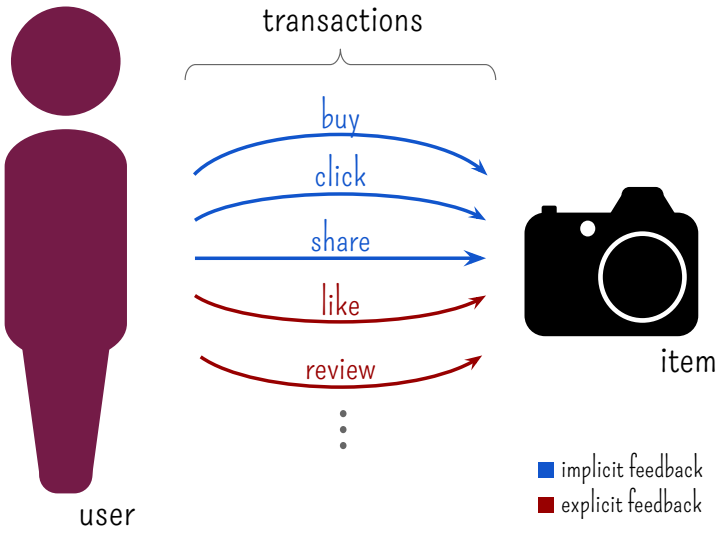


Figure 2.1: Illustration of the elements involved in a recommender system. A user demonstrates preferences to items through transactions, which may be performed as different actions.

Examples of recommendations based on popularity are the top 10 best-seller books, the most listened videos, and so on. In spite of that, the majority of recommendation approaches provides *personalized* suggestions, trying to surpass prediction based on popularity [Ricci et al, 2010]. In fact, the path for improving recommender systems is exactly towards overcoming the baseline popularity prediction through personalization, i.e., suggesting items, not necessarily popular, that the user like.

Figure 2.1 presents the three types of elements related to a recommendation problem: users, items and transactions [Ricci et al, 2010]. A *user* is an entity that recommendation results are intended for. His or her preferences regarding like and dislike of items are either individually entered or collected within the navigation.

An *item* is a general term indicating the object that the system suggests to the users. It may correspond to products to buy, places to visit and books to read, to name a few. Although in a review platform products and services might be items for recommendation, we instead focus on the recommendation of reviews as items.

At last, a *transaction* is an interaction performed under a certain context between a user and an item. It can be acquired explicitly, when users manually assert evaluations to items, or implicitly, derived from users' actions accomplished with other purposes.

The applicability of many recommender approaches is broad thanks to an abstraction of user, item and transaction that spans many scenarios. However, information regarding a particular environment, with specific entity types, might considerably improve performance, since such scenario may present special patterns and dynamics.

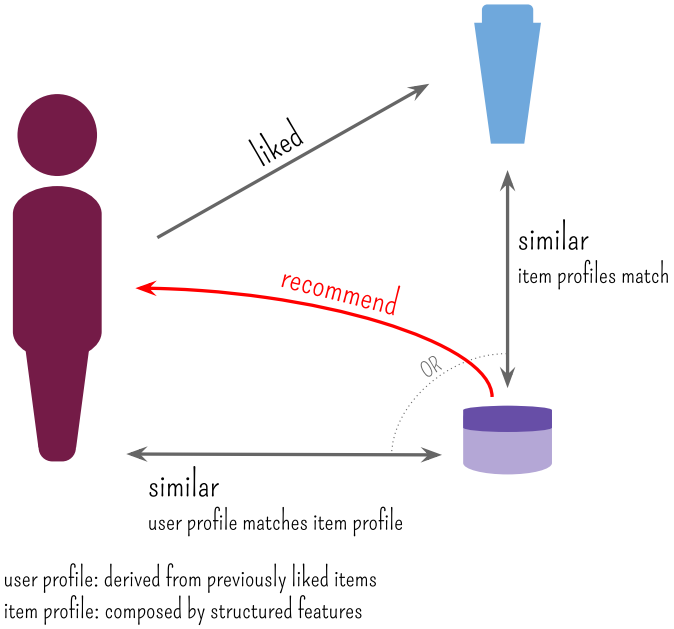


Figure 2.2: Representation of a content-based recommender system, which relies on user and/or item profiles and a similarity measure between them.

2.1.1 Types of Recommender Systems

Recommender systems are traditionally divided in two types: content-based and collaborative filtering [Lu et al., 2014]. A *content-based* recommender suggests items to a user based on the kind of items the user liked in the past, represented by observed features. In turn, a collaborative filtering method suggests items liked by similar users or similar items of the ones previously liked, such that the similarity is based on ratings. When both item profiles and collaborative information are taken into account, then a *hybrid* recommender system is composed which tends to improve performance by leveraging the drawbacks of each type [Ricci et al, 2010].

Specifically, a content-based recommender system suggests new items to a user based on the properties of the ones previously liked. The recommendation process is represented in Figure 2.2. Item profiles are composed by a structured set of observed features, for example, category, brand, material, and so on. Users are individually modeled through profiles by aggregating the preferences on previously evaluated items. Such profiles commonly model how much a user values each item feature and may be obtained using a heuristic or a learning method [Ricci et al, 2010]. A content-based system assumes that a user who liked certain items in the past will like similar items in the future. However, they lack diversity when users do not evaluate many different items and are not able to recommend for a person who has not evaluated anything.

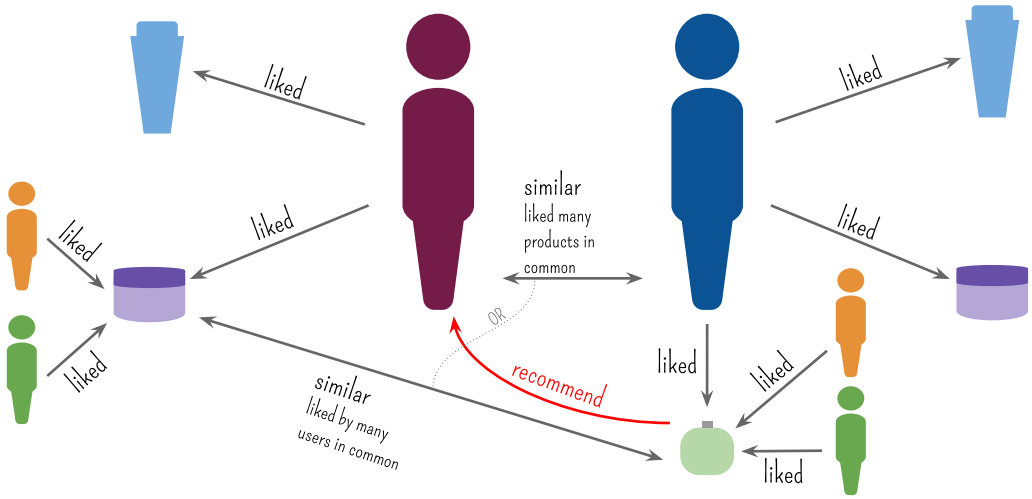


Figure 2.3: Representation of a recommender system applying a collaborative filtering approach, which uses collaborative information from similar users or items.

Building item profiles requires domain-expertise, although there are well-known guidelines, specially borrowed from other traditional research spheres [Ricci et al, 2010]. For example, derived from Information Retrieval, items composed by text, such as web pages, news and even reviews, may be represented using a vector space model. Cosine is the typical similarity measure in this scenario [Ricci et al, 2010]. Inspired by Artificial Intelligence, content-based systems may use past knowledge from user's evaluations to derive prediction rules for new items. For example, a classifier may be built for each user in order to predict if he will like or not an item [Ricci et al, 2010].

On an opposite track, collaborative filtering techniques rely on user-user or item-item similarity for recommendation, as Figure 2.3 outlines. A vector of given or received ratings represents users and/or items. Such abstraction is domain-independent and may be applied indistinctly to all scenarios, with any type of users, items and transactions. The basic assumption of these systems is that two users who had similar preferences in the past will share the same taste in the future; the same holds for two items that were liked by a representative share of common users.

Collaborative filtering is further specialized into memory-based or model-based solutions. A *memory-based* approach finds similar users or items relative to the user-item pair whose preference is to be predicted. The weighted average of similar users who rated the item (user-based) or the weighted average of similar items rated by the user (item-based) is used for prediction. They are also called *neighborhood-based* methods as they predict based on close entities [Ricci et al, 2010]. A *model-based* approach, on the other hand, transforms the matrix of ratings into a model. Past ratings are not used directly, but for building representations for users and items into the model.

There are other more fine-grained types of recommender system not described here. They are: demographic systems, relying on demographic profiles of users; knowledge based systems, which use specific domain knowledge of how items features meet user needs; and social recommender systems, based on information from friendships and other social networks [Ricci et al, 2010].

2.1.2 Types of Entities

Cold-start entities are a habitual challenge mainly in collaborative filtering solutions [Agarwal and Chen, 2009; Koren et al., 2009]. A *cold-start* refers to a user or an item whose past transactions are scarce or not available, i.e., they are new or unknown by the system, whereas the opposite is called *warm-start*. Under this circumstance, finding similar entities in collaborative filtering systems is difficult and cold-start users do not have a representative profile in content-based solutions.

The system may completely fail to provide recommendations for a cold-start. A hybrid technique rises as a great option for overcoming cold-starts by using the best solution under each scenario. For example, a content-based technique may be used for prediction of cold-start items and a collaborative filtering for warm-start.

2.1.3 Types of Feedback

User preferences may be obtained through explicit or implicit feedback expressed by users [Amatriain et al., 2009]. Explicit taste is inserted manually and restricted to cases when users feel motivated to contribute. Implicit ratings are derived from users behavior carried out with other intention, e.g., from a click, the duration of visualization of a web page, a purchase, a share with friends, and so on, as illustrated in Figure 2.1.

However, explicit ratings suffer from user inconsistencies named *natural noise* [Amatriain et al., 2009; Parra et al., 2011]. Distinct motivations may lead to different evaluation criteria, as well as insertion under distinct contexts, e.g., represented by time and location. For example, a user on vacation may tend to give higher ratings than after a stressful day of work. Thus, an effective method should be able to overcome noise.

On the other hand, implicit feedback is more difficult to relate with user preferences since it is derived indirectly, working as a heuristic. Nonetheless, in certain scenarios, it is the only source available due to user overhead in providing explicit opinion [Parra et al., 2011]. Implicit feedback is typically related to a positive interest, such as clicking into a web page or viewing it for long, whereas negative opinion is

hard to be derived [Hu et al., 2008]. Thus, combining both types of feedback is usually better for leveraging the drawbacks of each one.

2.1.4 Incorporating Biases

Typically, rating values are partially explained by users' and items' tendency of giving or receiving higher or lower evaluations, which is called bias [Ricci et al, 2010]. An entity bias is the average deviation of its ratings from the overall mean. A baseline predictor may be composed by only using bias variables, as follows:

$$\tau(u, i) = \mu + b_u + b_i, \tag{2.1}$$

where τ is the predictive function, u is a user, i is an item, μ is the mean rating, b_u is user bias, and b_i is item bias. The last two variables may be learned by an optimization method, such as stochastic gradient descent [Koren et al., 2009]. Alternatively, such variables may be computed statically as follows:

$$b_u = \frac{\sum_{i \in \mathcal{I}_u} (R_{ui} - \mu)}{|\mathcal{I}_u| + \lambda_1} \tag{2.2}$$

$$b_i = \frac{\sum_{u \in \mathcal{S}_i} (R_{ui} - \mu - b_u)}{|\mathcal{S}_i| + \lambda_2}, \tag{2.3}$$

where $\mathcal{I}_u$ is the set of items evaluated by user u, $\mathcal{S}_i$ is the set of users who evaluated item i, R_{ui} is the rating given by user u to item i, and λ_1 and λ_2 are regularization parameters [Ricci et al, 2010].

Using such baseline, the remaining ratings unexplained by bias, obtained by subtracting the bias baseline prediction, may be fitted using a second method. Thus, only the rating portion related to user-item interaction is adjusted in the second.

2.1.5 Recommendation Tasks

Recommender systems techniques typically rely on existing labeled data, the *training set*, which contains historical examples of truth labels for the level of interest of users to items. Using this set and assuming that past patterns are repeated in the future, a *predictor* is built as a function that, given an input user, returns a set of suggested items. Parameters related to predictors (e.g., coefficients, latent factors, weights) are typically adjusted to optimize prediction in the training set. Then, unseen data is used for evaluation of the method, designated the *test set*. Sometimes, a portion of

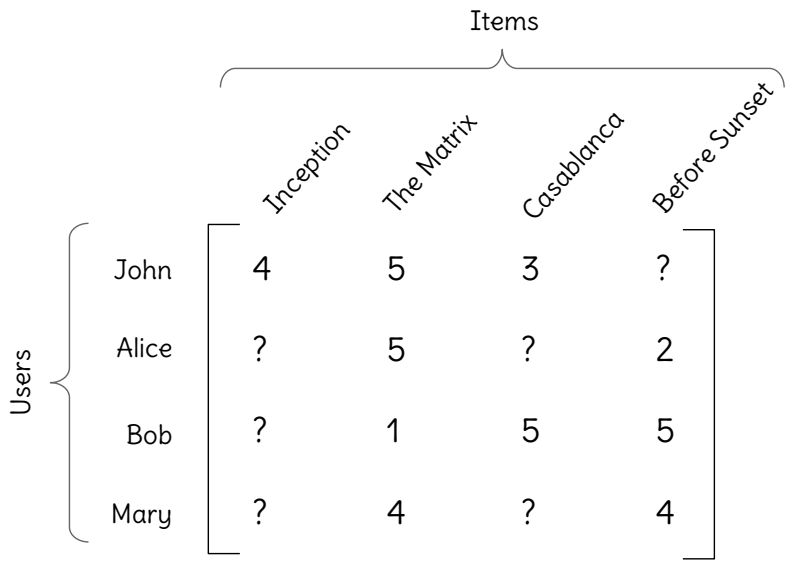


Figure 2.4: Example of a user-item matrix with movies as items. Observed ratings are numeric values, whereas unknown ones are represented by a question mark.

the training set is removed from the learning process and used for parameter tuning, named the *validation set*,.

A user-item matrix usually represents past ratings, which are in the training set. Figure 2.4 shows an example, with users represented in lines and items (movies in the example) in columns. To exemplify, user *Bob* gave a rating of 3 to item *Inception*. In general, if user i gives rating y to item j, then $R_{ij} = y$; if user i does not rate item j, then we represent $R_{ij} = ?$.

A recommendation problem may be defined according to two perspectives: (i) *rating prediction task*, aiming to predict the exact level of interest of a user to an item; or (ii) *top-n recommendation task*, aiming to select n items most preferred to a user [Deshpande and Karypis, 2004]. Although the majority of traditional strategies focuses on the former definition [Koren et al., 2009; McAuley and Leskovec, 2013; Moghaddam et al., 2012; Tang et al, 2013; Xu et al., 2012], the former has recently gained attention [Balakrishnan and Chopra, 2012; Bidart et al., 2014; Christakopoulou and Banerjee, 2015; Cremonesi et al., 2010; Koren, 2008], since many applications ultimately rank items using predicted ratings.

Specifically, the purpose of a rating prediction task is to fill missing values in the user-item matrix, that is, the cells in which $R_{ij} = ?$. Figure 2.5 shows an example of such predictions. Consequently, the values have to be in a close scale to true ratings, for example, from zero to five. In this task, the closer the predicted value to the true

Figure 2.5: Result obtained from a rating prediction recommender system, whose goal is to fill the missing values of the matrix. Predicted values, in red, are estimated in a close range of observed ratings.

Figure 2.6: Result obtained from a top-n recommender system. Predicted values, in red, are not necessarily in the same range of original ratings; instead, they represent relative indices. Then, a ranking of is obtained for each user.

one, the better. Then, evaluating the proximity of both values is necessary. This is accomplished by regression error metrics such as Root Mean Squared Error (RMSE) or Mean Absolute Error (MAE) [Hyndman and Koehler, 2006].

In turn, a top-n recommendation task does not focus on filling the matrix with exact values; instead, relevance values are used in any range just to allow the selection and ranking of the top-n items. Figure 2.6 demonstrates relevance estimation with a

simple example. Regression metrics do not make sense in this scenario, only classification metrics to evaluate the selection, such as precision and recall [Powers, 2007], and ranking ones to evaluate the selection and ranking simultaneously, such as Normalized Discounted Cumulative Gain at position p (NDCG@p) [Burges et al., 2005] and Mean Average Precision at p (MAP@p) [Yue et al., 2007].

The purpose of a recommender system is often to compose a ranked selection of the best items to a user. Consequently, optimality in a top-n task is more relevant to this end and, thus, we model review recommendation as such.

2.2 Review Recommendation Scenario

In a website of online reviews, a user may perform different roles. As an *author*, the user writes a *review* about a *product* (we use product as naming convention, but it may be any object in a review platform). As a *reader*, the user simply reads through existing reviews of a product. Then, as a *voter*, the user evaluates the helpfulness of existing reviews. Note that an individual may act as author and as voter. Both profiles may be used to gather more information regarding a user for recommendation: the current author may have a profile for previously acting as a voter, and the current voter as well for performing an author role before.

Specifically, a *review* is composed by a *text* (describing features about the product, e.g., experience of usage, pros and cons) and a *rating* (giving a grade to how much the author liked the product, e.g., 0 to 5 stars). Rating differs from *vote*, the grade given by a voter to a review.

Formally, this scenario considers a set of products $\mathcal{P} = \{p_1, p_2, ..., p_k\}$, a set of users $\mathcal{U} = \{u_1, u_2, ..., u_l\}$, a set of reviews $\mathcal{R} = \{r_1, r_2, ..., r_m\}$ and a set of helpfulness votes $\mathcal{H} = \{h_1, h_2, ..., h_n\}$. We define two functions to state relationships between elements in those sets: $\pi_r : \mathcal{R} \mapsto \mathcal{P} \times \mathcal{U}$ such that $\pi_r(r) = (p, a)$ states that review r was written by author a about product p; and $\pi_h : h \in \mathcal{H} \mapsto \mathcal{R} \times \mathcal{U}$ such that $\pi_h(h) = (r, v)$ defines that helpfulness vote h was given by voter v to review r. We emphasize that both author a and voter v belong to set $\mathcal{U}$.

2.2.1 Problem Statement

Optimally, a review recommender system provides a ranking of reviews in descending order of helpfulness for a given user and product, as shown in Figure 2.7. Thereafter, whenever a reader accesses a product, the most helpful reviews are on top, eliminating the burden to manually look for suitable ones.

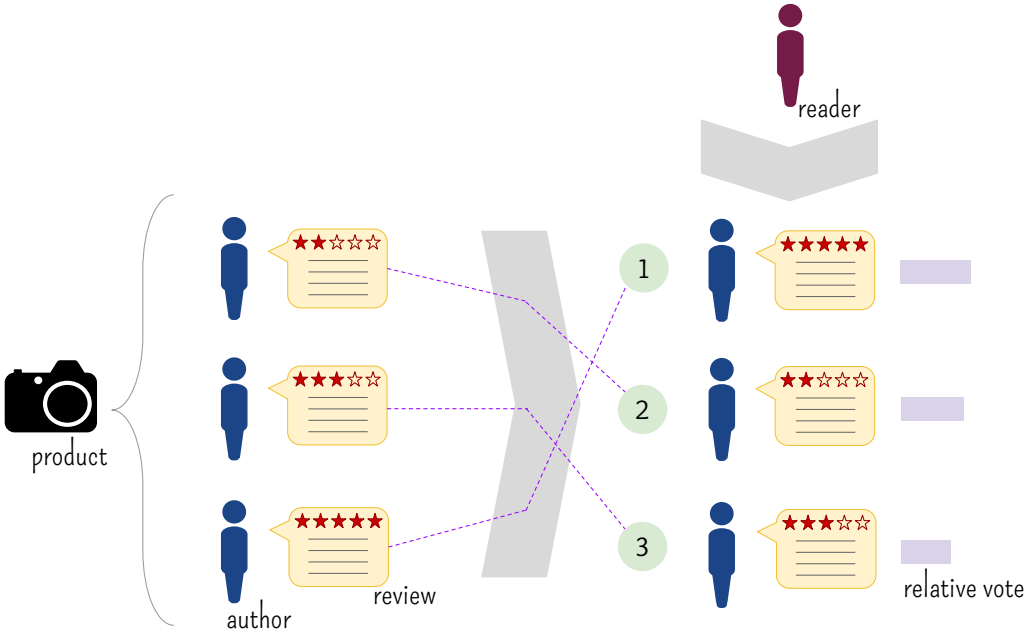


Figure 2.7: An overview of a review recommender system. Given a reader and a product (e.g., a camera) with reviews written by authors, the system retrieves a personalized ranking of reviews that matches the reader's interests.

The goal of a recommender system is to compose a *utility function*[1] [Adomavicius and Tuzhilin, 2005], restricted here for the specific case of review recommendation. Accordingly, the purpose of a *review recommender system* is to derive a utility function $\tau : \mathcal{U} \times \mathcal{R} \rightarrow \mathbb{R}$ such that, given an input (u, r), it measures the *relative helpfulness* of review r for a user u. Fitting τ requires a training set $\{\mathcal{P}^0, \mathcal{U}^0, \mathcal{R}^0, \mathcal{H}^0\} : \mathcal{P}^0 \in \mathcal{P},\ \mathcal{U}^0 \in \mathcal{U},\ \mathcal{R}^0 \in \mathcal{R},\ \mathcal{H}^0 \in \mathcal{H}$, unless a heuristic without a learning step is used. Users in the training set are further divided into those who acted as voters $\mathcal{V}^0$ and those who acted as authors $\mathcal{A}^0$, such that $\mathcal{V}^0 \cup \mathcal{A}^0 = \mathcal{U}^0$.

After the utility function is built, given a product p and a reader u, we may obtain a set of reviews associated to product p, defined as $\mathcal{R}_p = \{r \in \mathcal{R} \mid \exists a \in \mathcal{U} : \pi_r(r) = (p, a)\}$, and derive a ranking of reviews $r \in \mathcal{R}_p$ in decreasing order of $\tau(u, r)$. The utility function is optimal when rankings are generated in decreasing order of the ground-truth of helpfulness given by the users.

Stating review recommendation as a top-n task differs from most prior analyses of this problem. Past works focused on predicting absolute values for review helpfulness, i.e., they considered review recommendation as a rating prediction task [Moghaddam et al., 2012; Tang et al, 2013]. Nonetheless, helpfulness predictions are usually not

[1] We use the term *utility* to indicate the user interest in a generic recommendation task, and *helpfulness* to refer to the specific opinion of users about reviews.

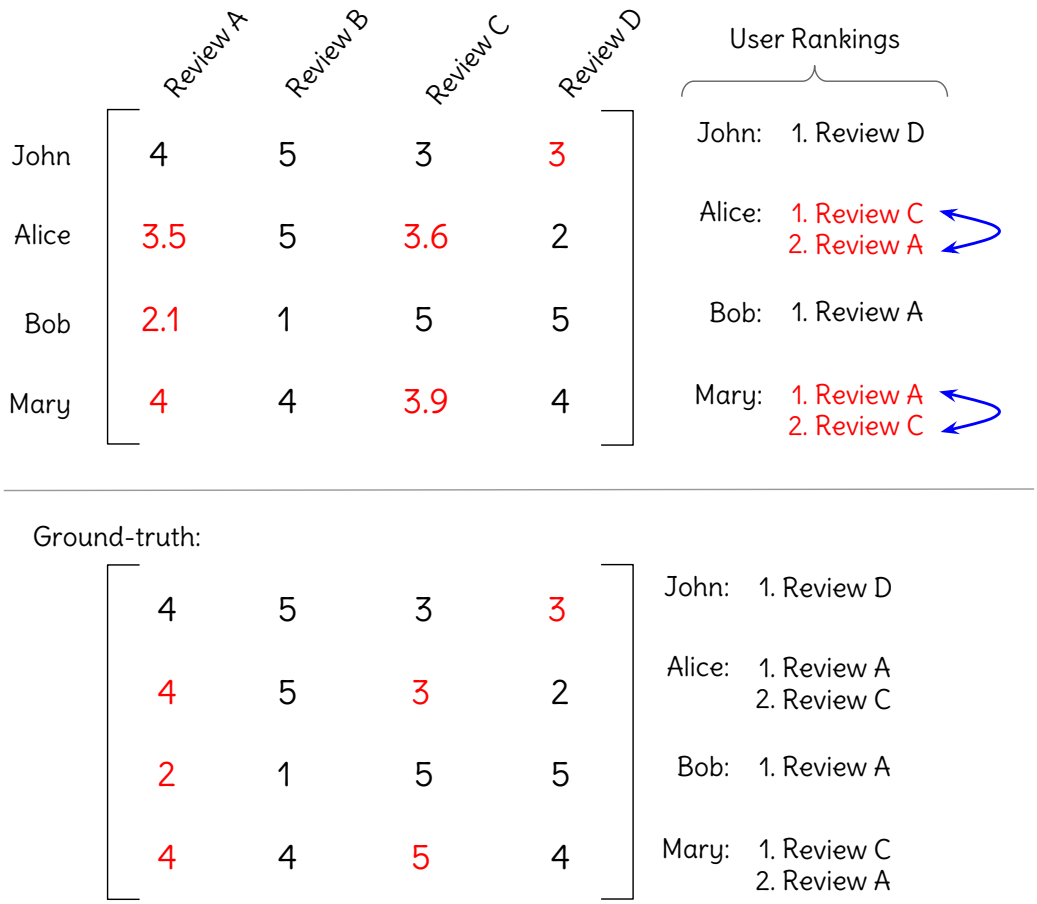


Figure 2.8: An example of a good regression, but poor ranking prediction. When regression error is minimized, many inversions in user rankings may occur.

presented to the user; thus, it is too restrictive to consider such option. Reviews are instead selected and ranked, only requiring to learn their relative positioning.

To elucidate the drawbacks of a rating prediction perspective for review recommendation, Figure 2.8 shows an example of a prediction with a good regression performance that incurs in bad ranking performance. A regression approach aims at exact prediction, so that the shorter the distance between the predicted value an the true one the better. A perfect regression would result in a perfect ranking, but such perfection is very hard achieve. Therefore, the existing errors may provide incorrect orderings.

In such example, we observe two types of failures: (i) medium regression error for all items, in case of user Alice, that caused an inversion, i.e., swapped pair of reviews; (ii) low regression error for one item with the cost of a high regression error for another, as represented by Mary's situation, promoting an inversion. In general terms, perfection in rating prediction task is hard to achieve. Thus, when trying to

optimize absolute values, an effort is spent into a useless task. Furthermore, by not focusing on relative positioning, low ranking performance may occur.

2.3 Concluding Remarks

Recommender Systems have broad applicability as anything may be recommended to a user: products, locations, services, friends, activities, news, and many others. We consider the specific recommendation of reviews, a not so typical task in a review platform, where more often the products described by reviews are recommended.

As such, we borrow several definitions and the overall context of recommender systems. They are mainly classified as content-based or as collaborative filtering and involve two types of users and items, cold and warm-start. Cold-start represents a challenge, especially for collaborative filtering approaches, which have no recommendation rule in this setting.

A recommender predictor is typically supervised, learning from a previous set of users' preferences. Such preferences may be gathered either implicitly or explicitly, the first suffering from inference errors and the second, from noise. We use a dataset with explicit ratings, thus an effective predictor for our task has to overcome such noise.

The research about Recommender Systems is slowly moving the perspective from rating-oriented to ranking-oriented. In fact, in many cases the goal is to produce a ranking of items, instead of an exact prediction of ratings. We adopt a top-n perspective in our work, evaluating all solutions using a ranking metric, besides a regression one.

The problem discussed in this work is, then, defined as ranking the reviews in decreasing order of helpfulness for a given reader and product. The product restricts the set of reviews in the ranking, while the user implies the right ordering.

Chapter 3

Related Work

Related work includes different strategies for dealing with information overload induced by reviews, existing approaches for ranking-oriented recommendation and analyses of the dynamics underlying a review application. Regarding the first topic (Section 3.1), there are proposals that predict the global helpfulness of reviews (Section 3.1.1). Such approaches consider the hypothesis that helpfulness is the same for all users, which may not hold in certain situations. On the other hand, there are also some personalized approaches of review recommendation (Section 3.1.2). Besides trying to predict a helpfulness index, there are other ways of extracting useful information from reviews, for instance, by composing a summary of the main features related to a product (Section 3.1.3). Extracted features may additionally be explored by allowing users to search for a suitable product (Section 3.1.4). A more typical recommendation problem in review platforms involves predicting the preferences of users to products (Section 3.1.5).

With respect to the remainder topics, research about recommender systems has recently focused on changing the problem definition to address the more usual purpose of ranking (Section 3.2), although not for the specific case of review recommendation. Finally, there are works analyzing review platforms' environment, investigating the process of writing and voting on reviews (Section 3.3).

3.1 Mechanisms for Processing Review Data

Figure 3.1 shows the different strategies of information extraction from reviews and addresses the trade-off between abstraction level and access to information. Given a target product, its reviews and user preferences, a *review recommendation* approach highlights the most important reviews, as a selection and/or a ranking. If a non-personalized recommendation is performed, user preferences are not considered. *Review summarization*

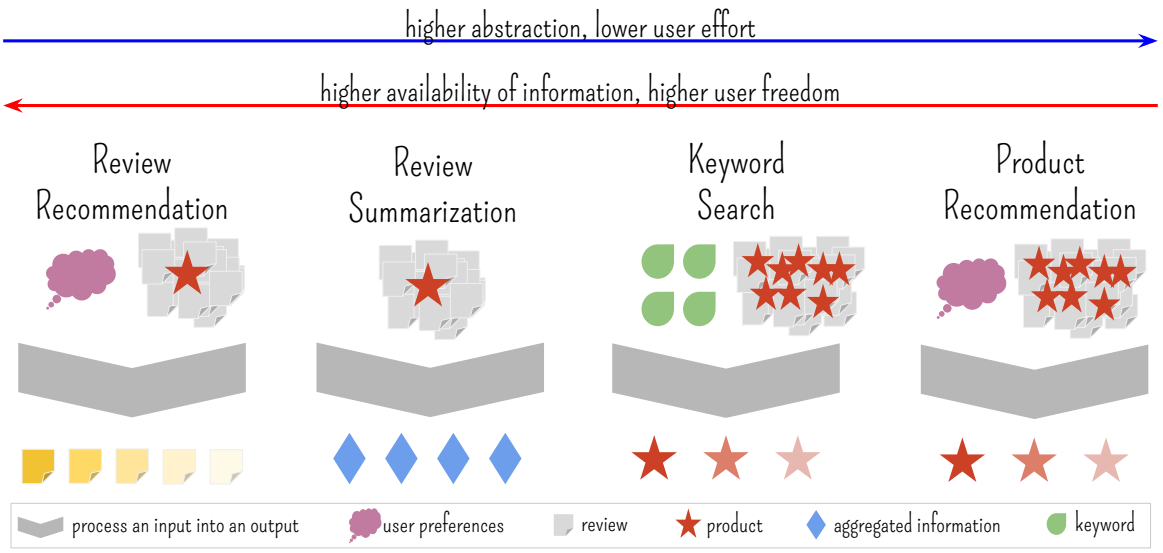


Figure 3.1: Different ways of processing review data.

receives as input a product with its reviews, and then derives aggregated information from them, i.e., a summary. *Keyword search* processes reviews from all products, possibly beforehand, and finds matching products, given a set of keywords. Finally, *product recommendation* applies user preferences to discover matching products, deriving a set of recommended ones. The last strategy typically uses just ratings, but review text may be used to augment information about users' opinion and product features. Although reading reviews requires a greater overhead, review recommendation may be useful for individuals investigating a product, especially if they already have one in mind and want to obtain further information.

3.1.1 Global Helpfulness Prediction

Many works propose a method of helpfulness prediction in a non-personalized fashion, using: regression, aiming at a specific real value [Kim et al, 2006; Lee and Choeh, 2014; Lu et al, 2010]; classification, by dividing helpfulness in non-ordinal categories [O'Mahony and Smyth, 2009; Zhang and Tran, 2011]; or comparison of the performance of both [Liu et al., 2008; Martin and Pu, 2014]. Initial works consider reviews as standalone documents and extract features from the text to derive the helpfulness prediction. Examples of features are: number of words, ratio of part-of-speech tags (such as nouns and adjectives), proportion of negative and positive words after sentiment analysis, subjectivity level, and readability. Subsequent works include other information considering that a user is the author of a review. Then, features are derived from author reputation and his place in the underlying social network, when

available. Number of reviews, average helpfulness received so far and the amount of third parties who trust him are some possible features for an author.

Under regression hypothesis, different algorithms are applied, such as Linear Regression [Lu et al, 2010], Support Vector Regression [Kim et al, 2006; Martin and Pu, 2014], Neural Networks [Lee and Choeh, 2014], Radial Basis Function [Liu et al., 2008], and Random Forest Regressor [Martin and Pu, 2014]. Classifiers are also explored using different methods: O'Mahony and Smyth [2009] investigate JRip, Decision Tree and Naive Bayes, being the first the best one in their experiments; and Zhang and Tran [2011] compare Naive Bayes, SVM and a new method based on Information Gain, such that the latter presents the best performance. All these works assume that there is a global helpfulness index, independent from a particular user.

3.1.2 Personalized Review Recommendation

Instead of using a global index, a personalized recommendation predicts reviews' helpfulness differently for each user. A few works [Moghaddam et al., 2012; Tang et al, 2013] consider user idiosyncrasy and obtain superior results than those disregarding it.

Moghaddam et al. [2012] address the need to recommend reviews and propose four increasingly sophisticated collaborative filtering approaches using latent factor models. Matrix Factorization (MF), a well-known model-based algorithm in Recommender System's field, is the first one and predicts helpfulness as the dot product of reader's and review's latent factors. The second version, Tensor Factorization (TF), exchanges review's dimensions by its underlying author and product, turning the rating matrix into a tensor to be factorized into reader, author and product latent matrices. The next improvement, Extended Tensor Factorization (ETF), consists in adding a new term to the objective formula such that ratings are explained by the inner product of author and product latent factors. Finally, they propose unBiased Extended Tensor Factorization (BETF), which accounts for reader and review biases in helpfulness value, and for author and product biases in rating value.

The four described methods are compared with mean-based predictors, Linear Regression, Support Vector Regression and User-Based Collaborative Filtering. The proposed approaches are better than all the others and BETF presents the best performance among them, using Root Mean Squared Error (RMSE). They use 90% of the data as training set and apply a random split, then not chronological. This setting may not be reproducible in practice due to requiring a large training size and disregarding time constraints. For instance, a lower ratio of cold-starts occurs, favoring the proposed methods. Besides, experiments are not reported with a confidence level.

In a similar personalized fashion, Tang et al [2013] propose Context-Aware Review Helpfulness Rating Prediction (CAP), a method based on latent variables, but also using observed information from social and content data. User behavior is divided into four cases: connecting with others (social network), writing reviews, evaluating reviews and evaluating products. Each behavior is a source of information for review helpfulness prediction: connected users tend to give higher grades among them; users with similar product evaluations also tend to give higher helpfulness votes one to the other; and author, reader and review properties impact the helpfulness. In this approach, the final helpfulness is given by a sum of latent factors, each one with a distribution centered on the regression of observed features. Regression coefficients and latent factors are fitted using a Monte Carlo Expectation Maximization algorithm.

CAP is compared with mean-based predictions, Linear Regression, Matrix Factorization and a joint Matrix Factorization and Neighborhood-Based Collaborative Filtering. CAP reveals a better performance using RMSE metric. The experiments are conducted by chronologically dividing 50% of the dataset for training set and 50% for test set. The experimental results, however, are not reported with statistical confidence. Additionally, CAP is not compared with the other specialized method, BETF.

3.1.3 Review Summarization

Composing reviews' summaries is another way of extracting useful information from reviews, which avoids manual investigation. Solutions of this type aim at concentrating all important data into a small piece of information, since many reviews repeat the same features about a product. Existing works compose summaries as: an inverted index from features to reviews, with sentiments attached [Hu and Liu, 2004; Zhan et al., 2009]; a cloud tag of frequent words and expressions [Wang et al., 2014]; and geographical visualization of polarity of features [Bjørkelund et al., 2012].

In general, works in this scope process reviews' texts performing *feature extraction* and then *sentiment analysis*, which associates a polarity to each feature. Such methods may benefit from a ranking of reviews per user: summaries may be obtained by prioritizing helpful reviews in a personalized manner.

3.1.4 Keyword Search over Reviews

Reviews provide an enormous source of objective and subjective features not available under a product's description. Besides, such features may be tagged with overall positive and negative sentiments derived from reviews. Then, users interested in a specific

set of features may perform keyword search for suitable products such that features are extracted from reviews' text. A product may be ranked, for example, according to the positive score of a desired property.

A few works exist in this scope [Duan et al., 2013; Ganesan and Zhai, 2012]. This class of methods has a higher abstraction level, since results are products presented to the user, who does not have to evaluate reviews or summaries. This solution is similar to a typical product recommendation but with contextual information represented by the desired set of features instead of user preferences.

3.1.5 Product Recommendation

A most typical recommendation problem in a review platform is recommending reviewed objects, avoiding completely the need for reading reviews, summaries or any additional information about them. Reviewed objects may be products (the generic name convention), services, locations, news, and many others. Such systems may also take advantage of reviews, for example, by gathering more information about users and products [D'Addio and Garcia Manzato, 2014; McAuley and Leskovec, 2013].

Product and review recommendation may be solved with similar approaches, represented by any abstract recommender system. Collaborative filtering stands out as a widely used solution, performing well in several applications [Koren et al., 2009]. Notably, Matrix Factorization, a model-based collaborative filtering method, has a central role in the winner recommender system of Netflix competition [Koren et al., 2009]. This method considers a set of latent factors for modeling users and items in the same space. Then, ratings are given by the inner product of a user's and a reader's factors. Nonetheless, the set of reviews is highly dynamic, thus constantly suffering from cold-start problem.

Hybrid techniques combine content-based and collaborative filtering advantages. For instance, Regression-based Latent Factor Models (RLFM) is a hybrid technique that uses latent factors, borrowed from collaborative filtering, that are centered on the regression of features, commonly adopted for content-based [Agarwal and Chen, 2009]. Not only item features are used, but also related to user and user-item interaction. This scenario works very well for cold-starts by predicting through regression on features.

3.2 Recommendation as a Ranking Task

Some recent solutions explore the potential of considering ranking optimization together with collaborative filtering, also known as *collaborative ranking*. Balakrishnan

and Chopra [2012] propose two pointwise and two pairwise models for learning a matrix factorization model for ranking. A pointwise model defines the loss function based on each rating, whereas a pairwise solution computes the loss function on pairs of ratings. The first pointwise approach consists of a two-stage model: (i) matrix factorization to learn latent factors that become feature vectors for each user and item; then, (ii) a regression model is built for this input. Another pointwise model is built by a neural network that simultaneously learns latent factors and regression weights. There are similarly two pairwise methods, one two-stage and another with a single stage; the only difference lies in the loss function, which is the cross entropy for the probabilities of an item to be ranked above another.

The Extended Collaborative Less-is-More Filtering (xCLiMF) [Shi et al., 2013] extends upon CLiMF [Shi et al., 2012] by allowing non-binary response. The Expected Reciprocal Ranking (ERR) is maximized, and ratings are represented by the inner product of user and item latent factors, like in Matrix Factorization. ERR considers a probabilistic framework for users' navigation based on a cascade model. xCLiMF presents better performance than CofiRank, which minimizes normalized Discounted Cumulative Gain (nDCG) [Weimer et al., 2007].

Christakopoulou and Banerjee [2015] provide three versions of algorithms that optimize ranking performance focusing on the error at the top of the list and also model prediction using Matrix Factorization. P-norm Push minimizes the p-norm of the vector composed by the heights of irrelevant items, consisting of the number of relevant ones below them. Infinite Push maximizes the infinite norm of the same array, corresponding to the maximum value. Reverse-Height Push minimizes the reverse height of relevant items, represented by the number of irrelevant items above them. However, all solutions use binary responses, not in a scale as we consider here.

In general, collaborative ranking relies heavily on latent factors by extending collaborative filtering. Consequently, they fail to predict for cold-starts and neglect the use of observed information for recommendation purpose.

3.3 Analyses of Reviews

Several works analyze review data by searching for patterns underlying the process of writing and evaluating reviews. Bakhshi et al. [2014] investigate whether exogenous factors (such as weather, demographics and time of the year) may influence ratings. If they do, an undesirable situation occurs since evaluations are supposed to consider endogenous factors only. Weather, for instance, is highly correlated to ratings, which

have lower values in the summer. Moreover, high population density and education level result in more reviews being produced in the area.

Danescu-Niculescu-Mizil et al. [2009] analyze reviews considering rating and helpfulness. They note that the closer a rating is to the average product rating, the higher the received helpfulness. By dividing the products according to rating variance, three patterns are identified: (i) for low variance, reviews with average rating have more helpfulness; (ii) for moderate variance, reviews are more useful if the rating is a little above the average; (iii) for high variance, the helpfulness distribution is bimodal, with one mode above and the other below the average, being the average a local minimum.

Jurafsky et al. [2014] show that reviews' text are biased towards positive emotions as they are above the mean obtained from the Google Books Corpus [Michel et al., 2011]. The authors also observe that reviews with low ratings often present a narrative about a trauma. Besides, low-cost restaurants are related to food addiction, whereas more expensive ones are associated to a complex language, composed by plenty and longer words, with a sensual perspective of food.

Interestingly, local contexts represented by other reviews may impact the perceived helpfulness. One would expect to observe a *rich-get-richer* effect on helpfulness values, in which case the higher a review ranking, the higher the votes it receives. Nonetheless, users tend to evaluate a review more positively if its current position is below the deserved one and negatively if above [Sipos et al., 2014]. Voting mechanism converges, i.e., the true helpfulness is reached with time. Besides polarity, voting participation is also analyzed: reviews in higher positions tend to receive more votes, which is indeed a *rich-get-richer* effect, but also reviews far from the deserved ranking.

3.4 Concluding Remarks

Most existing solutions for helpfulness prediction consider that users perceive the helpfulness in the same way. A few works [Moghaddam et al., 2012; Tang et al, 2013] show that personalization is more suitable, although they investigate for regression and the results do not indicate the significance level. Review recommendation approaches are still scarce and there is a great space left for improvement. We start to explore such space by assessing a rich set of approaches, from five paradigms, with statistical confidence and for a ranking task.

In spite of item recommendation and keyword-search being useful and promoting high level of abstraction to the user, in some cases, the user does want to investigate raw reviews. For example, there may be a situation where a consumer is interested in

a specific product and wants to obtain more information; or even wants to compare two products in detail. In such setting, given the great volume of reviews impossible to be fully read, the distinction of those more useful has an important role.

Summarizing reviews is a task that, similarly to review recommendation, allows users to obtain more information about an object of interest and may be used in a complementary manner. Challenges around this solution are even higher, as processing unstructured data – the reviews, selecting the most important information and composing a summary in some cohesive way are much farther from the current state-of-the-art. Other ways to processing review data includes keyword search by extracting features from reviews and direct product recommendation, possibly using reviews to augment available information.

A few solutions solve a recommendation problem from a ranking perspective, but some of them are restricted to binary relevance and they mostly only use latent features by extending collaborative filtering techniques. In a sparse scenario such as review recommendation, where reviews are being produced every day, relying on recommendations only for warm-start entities becomes impractical.

Analyses of dynamics underneath review sites show diverse patterns regarding conducts for writing and evaluating reviews. They mainly limit the reach of solutions relying on review data. For example, the influence of exogenous factors and context dependency make ratings and votes not completely reliable.

We focus on the specific case of review recommendation and evaluate specialized solutions that were not compared against each other nor with a ranking metric and statistical tests. We provide a much wealthier comparison of approaches, including five paradigms, than previously adopted for review recommendation [Moghaddam et al., 2012; Tang et al, 2013], and additionally with a ranking goal. We are the first one, for the best of our knowledge, to compare pairwise and listwise learning to rank solutions, as well as personalized regression and sophisticated recommender systems for review recommendation. By performing such experimental setting not applied so far, we identify the best solution and paradigm for ranking from a rich set, and elucidate the key design aspects for designing an effective solution.

Chapter 4

Helpfulness Voting Dynamics: A Case Study in Ciao UK

In this Chapter, we study the helpfulness voting dynamics in a dataset from Ciao UK. We start describing the platform (Section 4.1), presenting an overview of the dataset (Section 4.2) and then we investigate users participatory behavior (Section 4.3) and connection over a trust (Section network 4.4). Then, focusing on the purpose of review recommendation, we compare the universal and individual hypothesis for helpfulness (Section 4.5). In the following, we represent votes by a set of features and evaluate their importance for helpfulness prediction (Section 4.6). Finally, we present concluding remarks for all analyses (Section 4.7).

4.1 A Closer Look at Ciao UK

Ciao UK is a platform prevalent in Europe with a diverse set of reviewed items. There are 27 categories, including *Health*, *Computers*, *DVDs*, *Travel* and even *Ciao Café*, covering generic topics, from controversial debates to top 10 listings, for which users may provide their own opinion through a review.

Besides writing standard reviews and voting on them, users may also post videos reviewing a product, create questions to other users, and provide quick reviews with at most 150 words. A key advantage of this platform lies in publicly disclosing votes by users. This is a very rare feature, as this information is not available in the majority of applications, including Amazon, TripAdvisor and Yelp. Indeed, this is the *only* existing platform (to the best of our knowledge) containing each user's vote, required for learning and evaluating a personalized review recommender system.

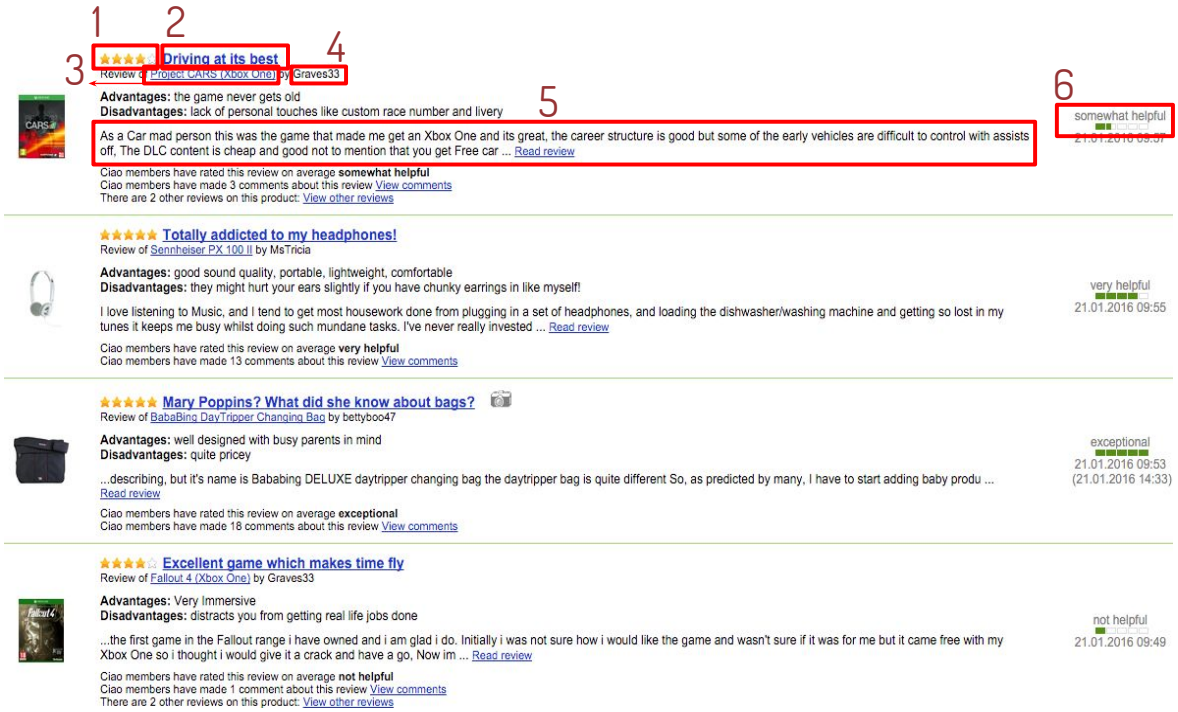


Figure 4.1: Example of reviews in the list of latest ones in Ciao UK. Label 1 indicates the rating; 2, the review title; 3, the product; 4, the author's login; 5, the review text; and 6, the overall helpfulness.

In most review platforms users post reviews without a monetary reward. On the other hand, Ciao UK allows earnings for each standard review written. The amount earned depends on the number and values of helpfulness votes received and two properties of a product: scarcity of available reviews and popularity[1].

Figure 4.1 shows a sample from the list of latest reviews (on January 21st, 2016), with the main elements related to a review: product, author, rating, text, and overall helpfulness. Each helpfulness value is associated with a label: 0 means off-topic; 1, not helpful; 2, somewhat helpful; 3, helpful; 4, very helpful; and 5, exceptional.

Figure 4.2 shows a review in detail, including author's basic reputation and the distribution of helpfulness given to the review. Personalized votes are publicly visible, as demonstrated by the login following each vote in the figure. Users do not always agree on the helpfulness, as shown by the distribution of vote values.

Detailed helpfulness votes are presented in Figure 4.3, for *helpful* and *somewhat helpful* levels. Optionally, a user may supply a comment in addition to a vote, typically justifying the helpfulness given. In this case, users complain about the lack of details, missing personal opinion and the format of the review. Not only users differ in the

[1]**Ciao UK: Earning Money.** `http://www.ciao.co.uk/faq/earning-money,12`

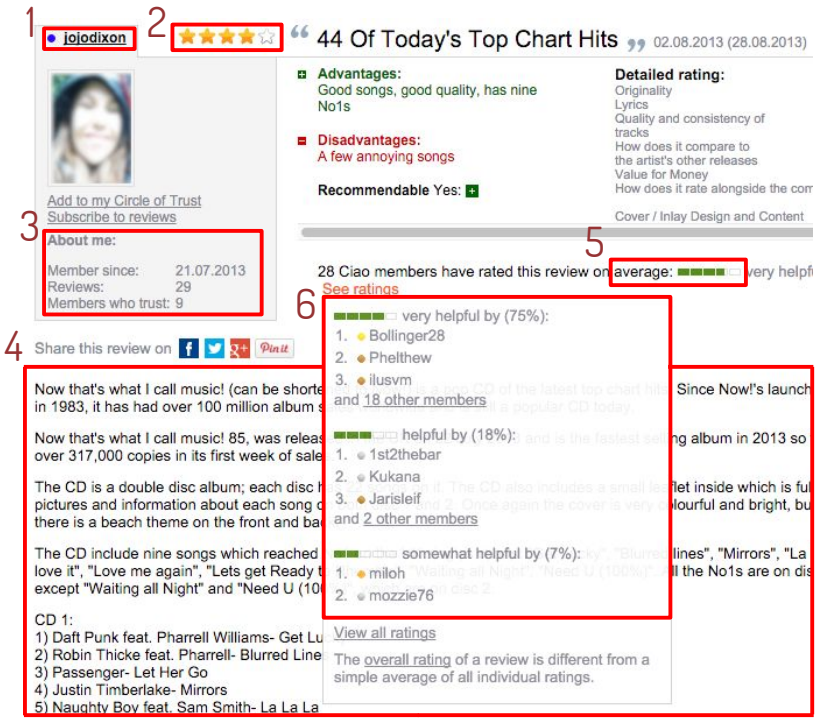


Figure 4.2: Fragment of a review page in Ciao UK, in which not all users agree on the review's helpfulness. Label 1 shows the author login; 2, the rating; 3, basic author reputation; 4, review text; 5, average helpfulness; and 6, distribution of votes.

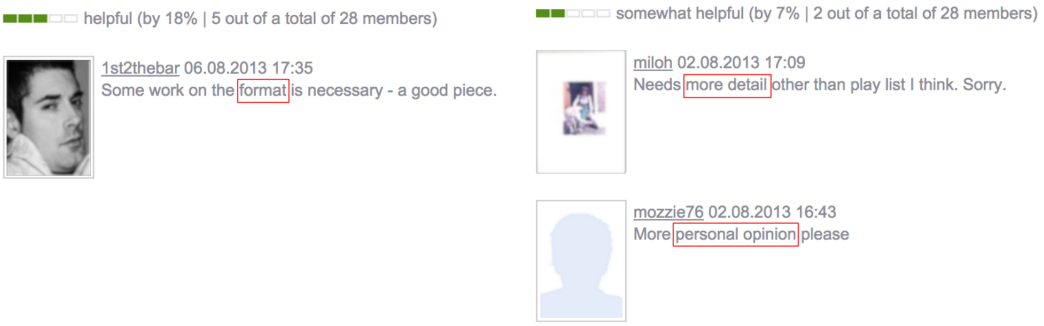


Figure 4.3: Example of a review's comments, which may be supplied with a vote. In this example, users complain about missing information in the review, a recurring reason for lower helpfulness votes.

perceived helpfulness, but also in what aspect they like or dislike in a review. Therefore, a global helpfulness index does *not* represent well this scenario.

Overall, Ciao UK is a very unorthodox review site that includes different categories, money rewarding, public helpfulness votes and specific dynamics. For instance, review payment might bias towards more helpful reviews. Also, publicly disclosed votes may intimidate true opinions, especially when the writer depends on them for monetary gain. Such scenario also allows that a group of users exchange many positive helpful-

Table 4.1: Statistics of the dataset.

	Original	**Filtered**
# Reviews	301,011	58,826
# Users	44,239	8,511
# Author	10,965	8,065
# Voters	43,590	1,737
# Authors & Voters	10,316	1,291
# Products	112,837	1,908
# Helpfulness Votes	8,870,598	457,697
Mean # Votes by Review	29.46	8.05
Reviews' Time Span	05.31.00 - 09.25.11	

ness votes, misleading the veracity of helpfulness evaluations. Although such situations may occur, we believe they do not heavily impact the experiments conducted in this work. Nonetheless, we leave such investigation for future work.

4.2 Dataset Overview

We use a publicly available dataset crawled from Ciao UK[2]. The crawling was performed through a breadth-first search starting with active users [Tang et al, 2013]. The date of the crawling is not available; all we know is that the last review was from 2011 and the work describing it was published in 2013.

To properly use this dataset, we disregard instances with empty fields, invalid rating values, invalid fields' format and texts with more than 40% of words outside WordNet dictionary[3] [Miller, 1995]. Also, since we adopt a ranking perspective for recommendation, we evaluate the performance of any technique using ranking metrics, thus requiring rankings of reasonable sizes. In our problem, each group of votes for the same reader-product pair provides a ranking, which corresponds to the reviews of the product evaluated by the given user. Figure 4.4 shows that most rankings are very small in the dataset considered. Indeed, 41.46% are composed by only one element, and the median value is two. Thus, we filtered the dataset to contain helpfulness votes associated to rankings with ten elements or more. The final dataset has around 5% of the original number of helpfulness votes, which covers around half million votes, being still a large sample. Table 4.1 has statistics of the original and filtered datasets.

[2]**Ciao UK Dataset.** `http://www.jiliang.xyz/Ciao.rar`
[3]WordNet is an English dictionary; thus we focus on English reviews.

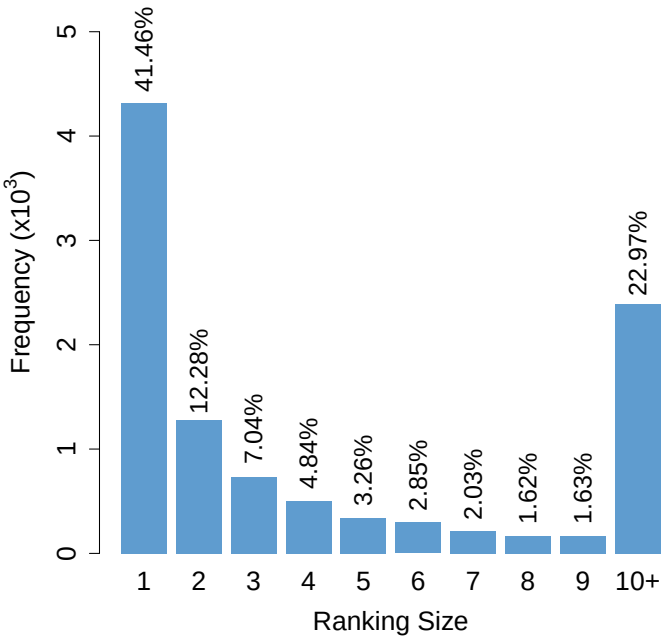
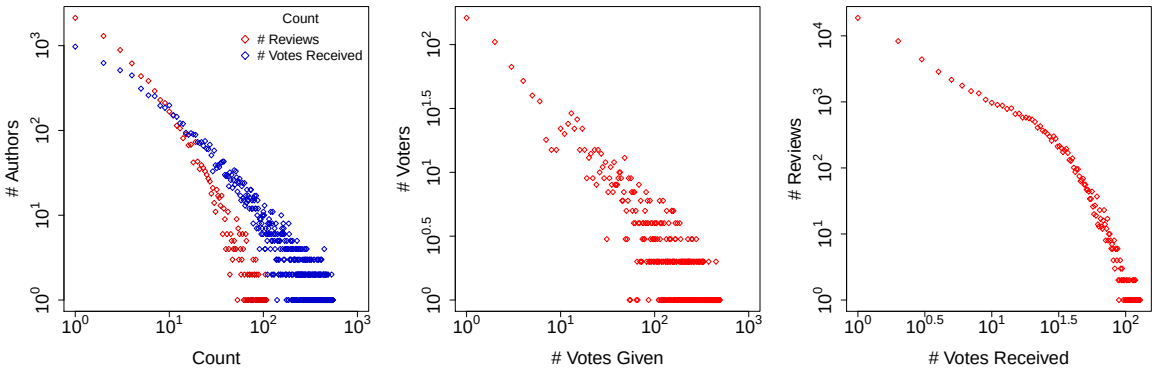


Figure 4.4: Distribution of ranking sizes in the original dataset.

Chronologically dividing the dataset into training and test sets is an important procedure, since otherwise future data may be used in the learning process. However, the dataset used does not contain votes' timestamps for performing such split. Hence, we use reviews' timestamps as an approximation for its votes' ones. Therefore, in a chronological split, votes from the same review lie in the same set – training or test – configuring all reviews as cold-starts. This situation severely penalizes collaborative filtering approaches and requires a workaround, as we further discuss in Section 5.7. Although this situation imposes certain restrictions, no other dataset with individual helpfulness votes by each user was found, neither a platform that makes this information publicly available.

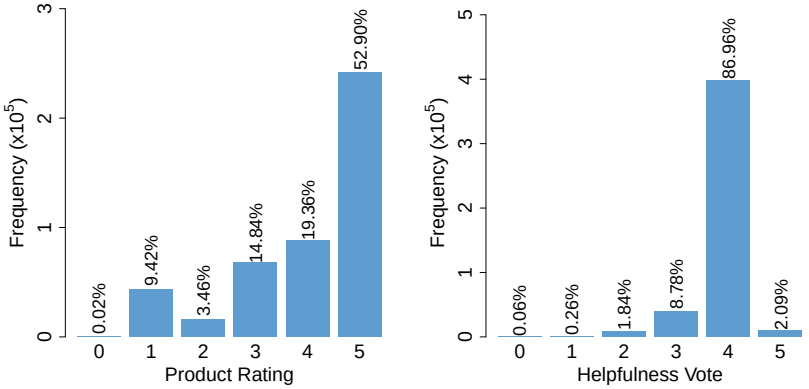
4.3 Participatory Behavior

In order to understand how users interact with the platform and their different profiles, we investigate patterns of activity and popularity. Figure 4.5a shows that many authors have written a few reviews, and a few authors have written many; the trend repeats for the number of votes received. A similar pattern is observed for the number of votes given by voters (Figure 4.5b) and the number of votes for a review (Figure 4.5c). All distributions are heavy-tailed, pointing to very different user and review profiles in



(a) **(b)** **(c)**

Figure 4.5: Distributions of activity and popularity, in log-log scale. Figure **(a)** contains the distribution of the number of reviews written and number of votes received per author. Figure **(b)** presents the number of votes given by users, as voters. Figure **(c)** shows the number of votes received per review.



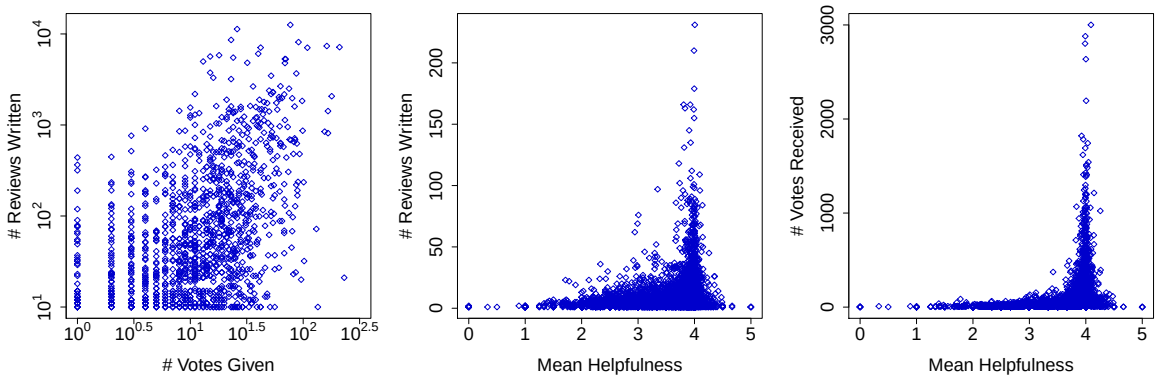
(a) **(b)**

Figure 4.6: Distributions of evaluation values: ratings as evaluations of products in Figure **(a)** and votes as evaluations of reviews in Figure **(b)**.

the platform, the majority of entities with a few related reviews or votes [Easley and Kleinberg, 2010]. In this context, recommender systems are relevant as, otherwise, the vast amount of unpopular reviews in the long tail[4] would not be easily discovered [Yin et al, 2012]. In turn, recommending to very different users, whose available information is of discrepant sizes, is a tough task. Thus, personalization would be advantageous, not only to consider different user preferences, but also different participatory profiles.

Figure 4.6 presents the distribution of users' evaluation as ratings and votes. Both distributions are very concentrated, with clear peaks around high ratings or votes. The latter peak is even more intense as over 85% of reviews have helpfulness vote equal to 4.

[4]Considering a plot with reviews sorted by popularity in x-axis and popularity in y-axis.



(a) Author vs. voter role **(b)** # Reviews vs. mean vote **(c)** # Votes vs. mean vote

Figure 4.7: Scatter plots correlating user participation and mean helpfulness.

The reason for such concentrated distribution may lie in users being more compelled to evaluate when they like a product or a review. In such scenario, any predictor outputting values close to 4 has a reasonably low regression error, which is another motivation for prioritizing ranking metrics. Since votes are publicly displayed in Ciao UK, users might feel intimidated to give low grades. Reviews with ratings 1 are more frequent than 2, a possible indication that people are also more motivated to vote if they extremely dislike a product than if they moderately dislike. Nonetheless, this does not occur for helpfulness votes; after all, the cost of buying a bad product is higher than the cost of reading a bad review. Another distinction between ratings and votes lies in the most frequent value: while it is 4 for votes, it is 5 for ratings, suggesting that users easily consider reviews as helpful, but hardly as perfect.

We further analyze authors' profiles by correlating their behavior as voter and as author. In such study, we use both Pearson correlation coefficient ρ and Spearman's rank correlation coefficient ρ_s. The first measures the strength of a linear correlation between two variables, whereas the second computes the correlation among rankings of two lists of observations. In turn, Spearman correlation indicates if the increase or decrease of one variable, by any proportion, is followed by a parallel growth or reduction of the other. Both coefficients occur in the range $[-1, 1]$, where -1 implies total negative correlation, 1 total positive correlation and 0, no correlation at all. The computation is performed using the following formulas:

$$\rho_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.1}$$

$$\rho_{s_{xy}} = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2 - 1)}, \tag{4.2}$$

where x and y are pairwise observations, d_i is the difference of ranking positions of i-th observation in x and y rankings, and n is the number of observations.

Figure 4.7a shows the scatter plot of the number of reviews written versus the number of votes given to an author, in a log-log scale. There is a weak to $\rho = 0.3751$, such that users tend to participate similarly in both scenarios; however, there is a reasonable variance of the strength of each role among users. Indeed, $\rho_s = 0.4455$, a moderate value. Such correlation indicates that, if a user participate more in one role, it tends to be more active in the other, but not necessarily in the same proportion.

Then, we explore how user activity and popularity correlates with helpfulness. We first consider the correlation of writing activity, represented by the number of reviews created, with the average helpfulness received by each author. The scatter plot of such comparison is in Figure 4.7b. To investigate if the increase of the number of votes coincides with higher helpfulness, we compute both correlation metrics, obtaining $\rho = 0.2300$ and $\rho_s = 0.2557$. When correlating average helpfulness with user popularity, measured as the number of votes received, we observe a lower value for Pearson and a greater one for Spearman, with $\rho = 0.2276$ and $\rho_s = 0.3266$. The scatter plot of this correlation is shown in Figure 4.7c. Thus, popularity is more important than activity for predicting relative helpfulness, although not linearly, as its peak around 4 is more prominent than for activity. Indeed, popularity is an aggregated form of readers' feedback while activity is not, thus working better for helpfulness prediction. Overall, there are underactive and unpopular users with a wide range of values for average helpfulness, in which case personalization may help distinguishing among them. However, it is clear that highly popular and active users typically have a high average helpfulness, around 4.

Although we use only one dataset, it is representative as it resembles the dynamics of other platforms. For instance, an analysis of a dataset from Epinions[5] includes a heavy tail distribution of users/reviews participation and popularity, and a highly unbalanced distribution of helpfulness [Moghaddam et al., 2012].

4.4 User Connections over a Trust Network

Ciao UK platform has an additional feature that allows users to interact through a special social network representing trust. Such network is directed; thus, when a user trusts another, the reciprocal is not necessarily true. Table 4.2 contains statistics from the trust network. Indeed, far from resembling a friendship network, whose edges are

[5]The link for such dataset is no longer available.

Table 4.2: Statistics of the Trust Network.

	Network	**Largest SCC**
# Nodes	18,998	9,629 (50.67%)
# Edges	145,528	126,867 (87.18%)
% Reciprocal Edges	34.21	40.72
Density	4.03×10^{-4}	1.37×10^{-3}
Average In/Out-Degree	7.66	13.18
Average Shortest Path Length[1]	4.75	4.52
Average Clustering Coefficient	0.12	0.16
Diameter	∞	15
# WCCs	1	1
# SCCs	8,928	1
Max # nodes in remaining SCCs	7	0

reciprocal by definition, there are only around one third of reciprocal edges. We observe a fair average in-degree and out-degree[6], whereas the density is low. Such values are increased for the largest Strongly Connected Component (SCC). The degrees are really concentrated, as the log-log scale distribution in Figure 4.8 shows. Surprisingly, the concentration is higher for out-degree, such that it is easier for a user to be highly active and trust many others than to be popular and trusted by a representative amount.

Approximately half of users are in the largest Strongly Connected Component, whereas the other half gathers in groups of at most seven nodes. Nonetheless, all of them are in the same Weakly Connected Component (WCC)[7]. The largest SCC is very connected, with higher density, higher clustering coefficient and diameter of 15. As we observe an engagement in using such network by some users, it may be a valuable source of information for recommendation, albeit not available in all cases.

An illustration of the largest SCC of the trust network is shown in Figure 4.9. Some users are highly trusted, represented by deep red and large nodes in the center, and gather many others around them. There are also two poles, using a force-directed layout[8] for visualization, with one more concentrated of high in-degree nodes, and the other with a more distributed in-degree across nodes. Finally, there are also some local influencers, i.e., with several trustors, represented by red nodes away from the center.

[6]Their average is the same, since the sum of in-degree equals the sum of out-degree.

[7]We believe, because of this and all the authors being in the trust network, that the crawling was performed through links of this network. The other possibility was to crawl reviews of users who voted in reviews of the already crawled users.

[8]Nodes suffer simultaneously attraction from connected nodes and repulsion from all nodes.

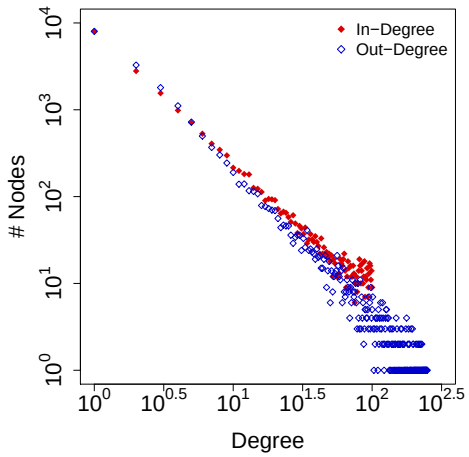


Figure 4.8: Distribution of in-degree and out-degree in the network, in log-log scale.

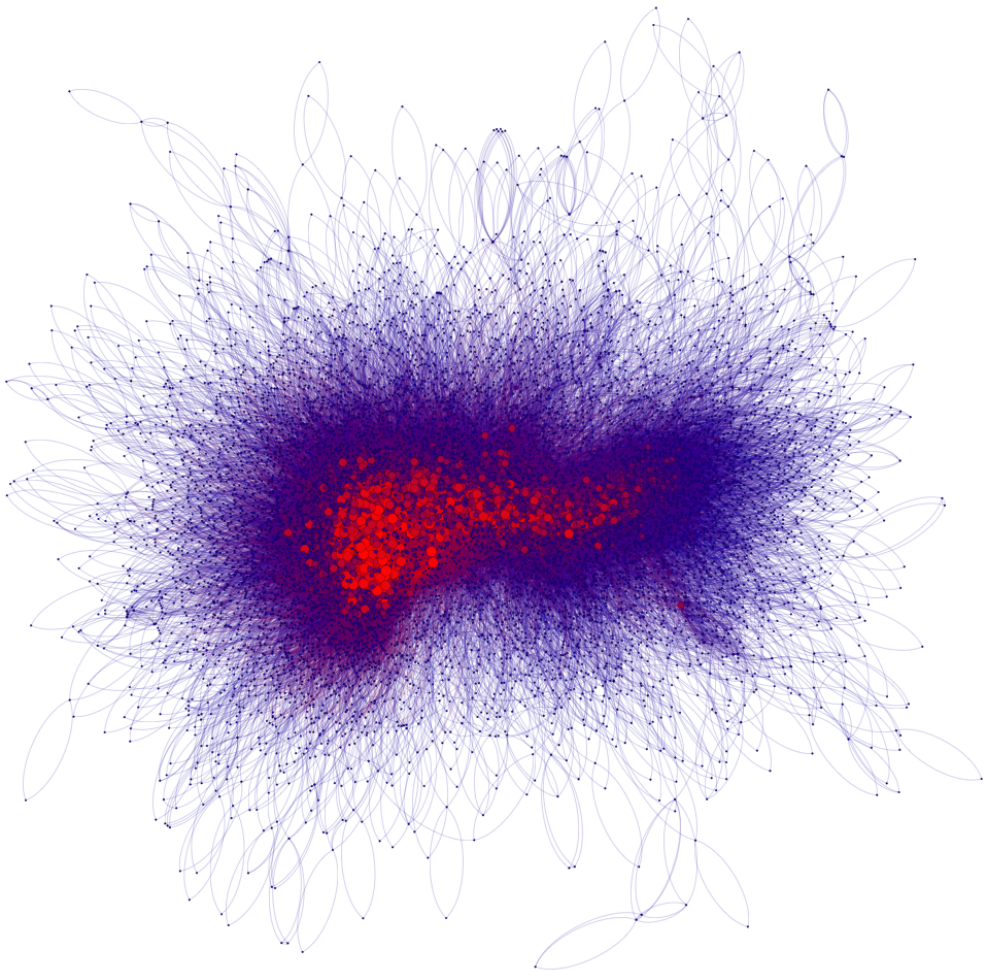
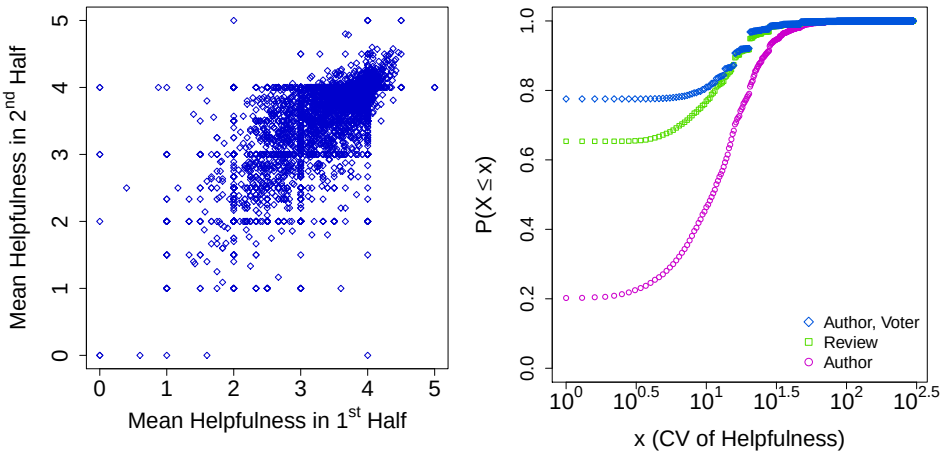


Figure 4.9: Subgraph of the trust network corresponding to the largest SCC.



(a) Mean vote in 1st vs. 2nd half **(b)** CDF of helpfulness CV

Figure 4.10: Investigation of author consistency and different entities conformity.

4.5 Review Helpfulness: Universal or Individual?

So far, we have claimed that review helpfulness is usually seen as a universal index, common to all users while, it should be an individual index, personalized to each user. In this section, we explore pieces of evidence of the extent of helpfulness explanation by a universal viewpoint and if there is indeed potential for a particular perspective.

First, we investigate the Author Consistency Hypothesis, which claims that different reviews of the same author tend to have the same quality and was validated in previous works [Lu et al, 2010]. We proceed as follows: the votes received by each user are divided chronologically in half, then the mean of each half is computed. We want to contrast if there is a tendency of both values being equal, so we use Pearson correlation. A high ρ is observed between the first and the second halves, equal to 0.7361 (while $\rho_s = 7052$ is weaker). This shows that, indeed, an author's votes tend to remain close in helpfulness. A scatter plot of the means of each author in the first and second halves is presented in Figure 4.10a. This also suggests that, when readers assess a review, they are mainly assessing the underlying author.

We further analyze the conformity of votes given to an author. Such conformity represents the degree of accordance among different readers for the helpfulness of an author. Thus, high conformity evidences that the author explains well the helpfulness value. In this analysis, we use the Coefficient of Variation (CV) [Jain, 1991], defined as the percentage of the standard deviation (σ) relative to the mean (μ):

$$CV = 100\,\frac{\sigma}{\mu} \tag{4.3}$$

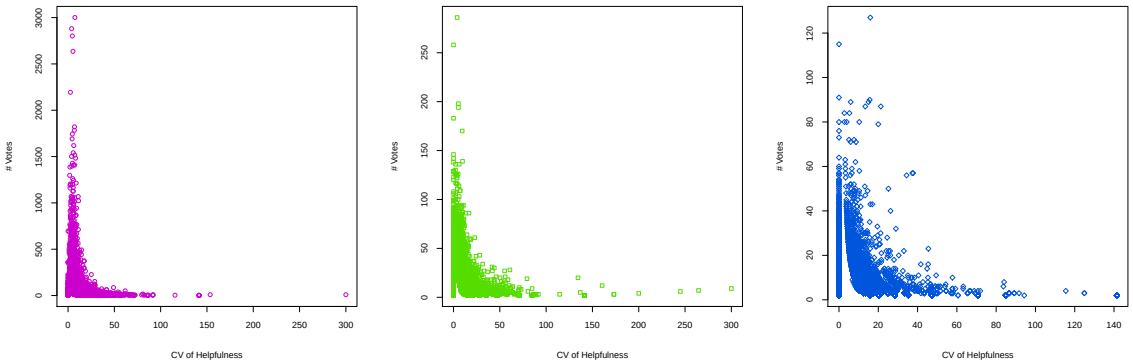
As the CV represents the level of variability, a low value indicates a high conformity. We divide the set of votes according to the corresponding author and compute the CV for each author's group. Besides conformity related to authors, we also evaluate such index related to reviews and author-reader pairs. Note that, when considering author and review groupings, a non-personalized conformity is evaluated. For instance, a high review conformity indicates that review's intrinsic characteristics, derived from its text, justify great part of helpfulness values. On the other hand, when evaluating for an author-reader pair, we consider a personalized conformity, as such groupings are specific to a reader[9]. Thus, we are assessing how the interaction between author and reader account for the observed helpfulness.

The Cumulative Distribution Function (CDF) for the CV related to authors, reviews and author-reader pairs is presented in Figure 4.10b. Around 20% of author votes, 65% of review votes and 78% of author-reader votes have zero variation. Although authors tend to be consistent, reviews are much more. Additionally, while a non-personalized solution explains great part of helpfulness of a review (as suggested by a low CV of votes belonging to a review) we observe a representative decrease of variation by considering each reader's opinion for an author.

We have further computed the confidence interval [Jain, 1991] for the mean CV for each entity, with 95% of confidence. The CV grouped by author presents an interval for the mean of 12.6628 ± 0.2908; grouped by review, of 5.2900 ± 0.1058; and grouped by author-voter, of 3.9624 ± 0.0651. No interval intercepts with another, which means that each entity conformity have significantly different means. Thus, the personalized author-reader conformity is significantly higher than non-personalized review and author, represented by lower mean of CV. Therefore, the author-reader pair, a personalized setting, explains helpfulness votes better than the review and the author, both non-personalized entities. This strongly motivates the pursuit of personalized methods for review helpfulness prediction as the next frontier for improvement.

Finally, Figure 4.11 correlates the CV of helpfulness with the number of votes. If higher CVs are correlated with higher number of votes, than author groupings may be penalized since they tend to be larger. However, we observe an opposite trend: higher number of votes typically have lower CV in all groupings. Following this trend, author-voter pairs should have higher CV since they usually have lower number of votes. Though, that is not the pattern observed: author-voter pairs have lower CV despite having lower votes. Then, such situation could be explained by other factors instead of the number of votes, probably the interaction between author and voter.

[9]It is not possible to assess review-reader conformity, as each reader can only give one vote to a review in most platforms, including Ciao UK.



(a) Author **(b)** Review **(c)** Author-voter

Figure 4.11: Scatter plot of CV versus number of votes for each grouping.

4.6 Characterization through Features

Some techniques considered in this work use features derived from the dataset for prediction. We use the set of features defined for CAP, a recently proposed review recommendation approach [Tang et al, 2013]. Table 4.3 briefly describes all features. Review's features are derived from review text, containing metrics such as length-based, part-of-speech tagging and sentiment statistics. Author's features capture their reputation and past behavior. These first two sets contain non-personalized information, i.e., they are the same for all readers, and were defined in an earlier work [Lu et al, 2010]. Reader's features include historical data of the target user of recommendation. Author-reader similarity's features capture similarity level between the author and the reader of a review. Author-reader connection's features measure the connection strength between an author and a reader, following the trend of social recommendation [Machanavajjhala et al., 2011] and the potential of the underlying trust network. Some of these features are not precisely defined in the original works; thus, we make some assumptions presented in Appendix A.

We evaluate the impact of features, first, by using Information Gain (InfoGain) [Yang and Pedersen, 1997]. This metric computes the entropy (i.e., impurity) change by splitting the data according to a feature value. In such calculation, the initial entropy on the whole data is computed and, then, the average entropy after splitting the data according to the value of a feature (for example, a group for feature A equals to 1, another for equals to 2 and so on). This impacts on how well a feature in isolation is able to distinguish between all helpfulness values.

Table 4.4 presents the top 10 features regarding information gain. The first five features are only non-personalized ones. In fact, there is a great part of review help-

Table 4.3: Features defined in the literature with a brief description. Each feature is derived from an entity or relationship, represented in fragment headers.

Feature	Description
	Review
r_num_tokens	Number of tokens, which are sequences of non-whitespace characters.
r_num_sents	Number of sentences.
r_uni_ratio	Unique words ratio.
r_avg_sent	Average sentence length.
r_cap_sent	Number of capitalized sentences.
r_noun_ratio	Ratio of tokens classified as nouns.
r_adj_ratio	Ratio of tokens classified as adjectives.
r_comp_ratio	Ratio of tokens classified as comparatives.
r_verb_ratio	Ratio of tokens classified as verbs.
r_adv_ratio	Ratio of tokens classified as adverbs.
r_fw_ratio	Ratio of tokens classified as foreign words.
r_sym_ratio	Ratio of tokens classified as symbols.
r_num_ratio	Ratio of tokens classified as numbers.
r_punct_ratio	Ratio of tokens classified as punctuation.
r_kl	KL divergence of review text relative to all reviews of the same product.
r_pos_ratio	Ratio of sentences tagged as positive.
r_neg_ratio	Ratio of sentences tagged as negative.
	Author
a_num_reviews	Number of reviews written by the author.
a_avg_rating	Average rating given by the author.
a_num_trustors	Number of users who trust in the author.
a_num_trustees	Number of users trusted by the author.
a_pagerank	Author's PageRank score in the trust network.
	Reader
v_avg_rat	Average rating given by the voter.
v_avg_rat_soc	Average rating of the social network.
v_avg_rat_sim	Average rating of similar users.
v_avg_help	Average vote given by the voter.
v_avg_help_tru	Average vote given by voter's trust network.
v_avg_help_sim	Average vote given by similar users.
v_num_trustors	Number of users who trust in the voter.
v_num_trustees	Number of users trusted by the voter.
v_pagerank	Voter's PageRank score in the trust network.
	Author-Reader Similarity
s_comm_rated	Number of products rated in common.
s_jacc_rated	Jaccard coefficient of ratings.
s_cos_ratings	Cosine similarity of ratings.
s_pear_ratings	Pearson correlation of ratings.
s_diff_avg_rat	Difference of author and voter average ratings.
s_diff_max_rat	Difference of author and voter maximum ratings.
s_diff_min_rat	Difference of author and voter minimum ratings.
	Author-Reader Connection
c_jacc_trustors	Jaccard coefficient of trustors of users.
c_jacc_trustees	Jaccard coefficient of trustees of users.
c_adam_trustees	Adamic-Adar score of common trustors.
c_adam_trustees	Adamic-Adar score of common trustees.
c_katz	Katz index of paths from voter to author.

Table 4.4: Top 10 features with higher InfoGain.

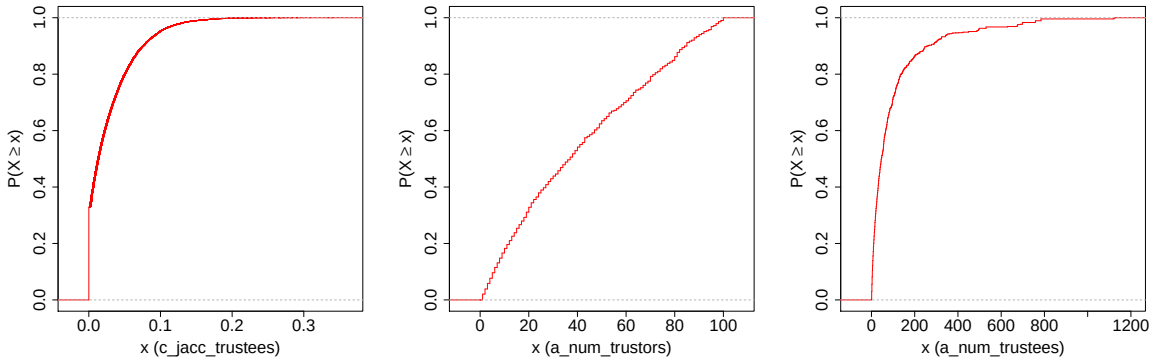
Position	Feature	InfoGain
1	a_pagerank	0.2189
2	r_num_tokens	0.1973
3	r_uni_ratio	0.1359
4	r_num_sents	0.1252
5	a_num_trustees	0.1145
6	v_avg_help	0.1128
7	v_pagerank	0.0922
8	v_avg_rat_soc	0.0922
9	v_avg_rat_sim	0.0911
10	v_avg_help_sim	0.0898

fulness that is explained by a global quality derived from review and author. However, the remainder features are all related to the voter. Thus, the same way recommender systems try to improve upon popularity prediction, incorporating personalization in solutions can surpass non-personalized ones.

Table 4.5: Top 10 features with higher absolute value of ρ_s. Features also in InfoGain Top 10 are followed by a red circle.

Position	Feature	ρ_s
1	r_num_tokens •	0.4099
2	r_num_sents •	0.3500
3	r_uni_ratio •	−0.3302
4	v_avg_help •	0.3134
5	a_num_trustees •	0.3115
6	a_num_trustors	0.2798
7	c_jacc_trustees	0.2380
8	c_adam_trustees	0.2200
9	c_adam_trustors	0.2164
10	a_num_reviews •	0.2018

We also assess the relative importance of different features by correlation ρ_s, defined in Section 4.3, with helpfulness. This metric is useful since we consider a top-n recommendation viewpoint. Table 4.5 shows that half of the features in ρ_s are also in the top 10 of InfoGain and there are only non-personalized ones until top 3. In general, no evaluated feature alone explains the helpfulness value fully. However, we observe that connection features share some relevance using this metric. In fact, 30%



(a) Jaccard of trustees **(b)** Authors' # trustors **(c)** Authors' # trustees

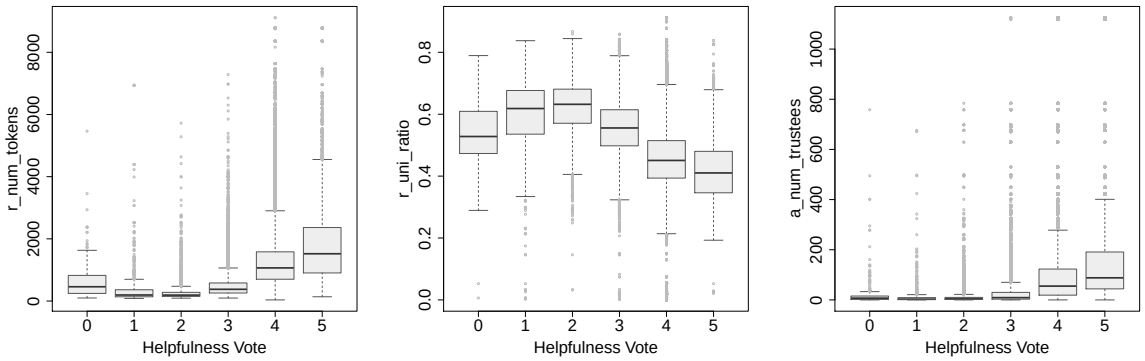
Figure 4.12: Distribution of Jaccard coefficient of trustees (Figure **(a)**), of authors' number of trustors (Figure **(b)**), and authors' number of trustees (Figure **(c)**).

of the times there is no overlap of trustees at all (Figure 4.12a), then *c_jacc_trustees* has considerable importance given that it is unavailable in many cases. Especially for ranking, connection features take a more representative place. The same importance is not observed for similarity features. Indeed, they are implicitly derived from user behavior while connection features are based on explicit expression of trust by users. Only *r_uni_ratio* is negatively correlated; probably many unique words may indicate a less readable text or with a lot of typos.

Surprisingly, the number of authors' trustees (their out-degree) is more correlated than the number of trustors (their in-degree). Figures 4.12c and 4.12b show that the number of trustees vary more widely and might distinguish authors better. As already indicated by the trust network analysis, users tend to have higher out-degree than in-degree. In such scenario, high participation by trusting many people is positively correlated with helpfulness, more than popularity for having many trustors.

We visually present the correlation with helpfulness for features that are both in InfoGain and ρ_s top 5 (except for *r_num_sents*, which presents a similar pattern of *r_num_tokens*). Figure 4.13a shows that, in general, the higher the number of tokens of a review, the higher the helpfulness vote of a review, as it is more likely to be a thorough description of the product. However, helpfulness votes equal to 1 or less do not follow such pattern. Thus, an extremely rejection is not very correlated with the size of the review. Figure 4.13b shows that helpfulness increases with the reduction of the ratio of unique words, but this trend is not observed for helpfulness equal to 1 or less. Finally, Figure 4.13c shows that the number of trustees is typically increases with helpfulness; but for ratings between 0 and 2, it has practically the same distribution.

Overall, we note that low helpfulness votes seems to follow a different pattern



(a) Number of tokens **(b)** Unique words ratio **(c)** Authors' # trustees

Figure 4.13: Box plots for features in both InfoGain and ρ_s top 5 (except the number of sentences), for each helpfulness level.

than medium and high ones, using the set of features specified. Such low evaluations may be related to very private and specific conditions. For example, an offensive phrase or recall of a bad memory may cause an extreme rejection by a reader. Such particular conditions are a challenge to review recommendation task. Nonetheless, there may be other unexplored features that explain low helpfulness cases better.

4.7 Concluding Remarks

We use a dataset crawled from Ciao UK for our analyses as no other platform (to the best of our knowledge) currently displays publicly the helpfulness votes, target of our prediction. As commonly occurs in many others, such dataset contains very different profiles of users and reviews related to participation and popularity. In such scenario, while recommending for users with very different sizes of associated historic data is hard, the purpose of review recommendation is strengthen by the volume of reviews with few evaluations and by distinct user profiles. A good recommender system may find good reviews in the long tail for a user that, otherwise, would hardly discover them, in a personalized way.

In a 0 to 5 scale, the high ratio of votes equal to 4 shows that users easily find reviews as helpful. Indeed, the author's collaboration is often viewed positively. Nonetheless, distinguishing helpfulness equal to 4 and 5 is still important, as well as both of them from reviews with helpfulness 3 or less. Authors tend to be consistent in quality across reviews, as their average helpfulness tends to be stable. The most popular and active authors in the platform are very often helpful, but not vice-versa. Predicting the helpfulness among unpopular authors remains a challenge.

Authors and reviews have little variation in vote values across readers, thus a great part of helpfulness is expressed by review and author quality. However, votes grouped by author and reader have an increased conformity, indicating that a personalized viewpoint implied by each reader explains better the helpfulness values.

The trust network underlying the platform is a promising source of information for review recommendation, as some users heavily use them. Indeed, connection features appear in Spearman correlation's top 10 features. Only non-personalized features are in the top 3 of both rankings of InfoGain and ρ_s. Thus, there is a great part of helpfulness votes that corresponds to content's intrinsic quality. Nonetheless, personalized features occur in the both feature rankings, an evidence that overcoming the current performance boundary may be accomplished by a personalized prediction.

Chapter 5

Multiple-Paradigm Techniques

We consider techniques from multiple paradigms in our comparison to assess the performance of review recommender systems under different assumptions. In this chapter, we present each paradigm with overall characteristics as well as specifics of each solution. By assessing very distinct approaches, we obtain a wealthy set of methods to analyze performance on review recommendation, not considered so far.

Specifically, specialized review recommendations, included in our comparison, were not compared against each other yet. Additionally, regressors were applied for this problem only in a non-personalized format, while we use a personalized viewpoint for all methods. Finally, we include learning to rank techniques, and a sophisticated recommender, Regression-based Latent Factor Models, both not previously evaluated for this task. By comparing such rich set of methods, we can derive the best predictor and identify key aspects that provide better performance for this task.

5.1 Paradigms

We evaluate methods from several paradigms with different levels of generality. Algorithms with well known competitive performance (e.g. LambdaMART) and widely used simple ones (as linear regression) are included in the comparison.

Each paradigm has a different generality level. We denote the group of methods of a given paradigm as a *class*. We describe some of them using naming conventions from their original scenario. Table 5.1 maps their original names to the specific one. The following classes are considered:

- *Mean-based Predictors* (MBP) compute mean statistics from the training set and use them for prediction (Section 5.2);

Table 5.1: Mapping of naming conventions from the original definition of LTR and GRS classes (generic) to the review recommendation scenario (specific).

	Generic	**Specific**
LTR	document	review
	query	reader-product pair
	relevance	vote
GRS	item	review
	user	reader
	rating	vote

- *Regressors* (REG) predict a real value, as close as possible to the response, for each set of features representing an instance (Section 5.3);
- *Learning to Rank Predictors* (LTR) determine a ranking of documents for a query given observed features (Section 5.4);
- *General-Purpose Recommender Systems* (GRS) predict a ranking of items by utility for each user (Section 5.5);
- *Review Recommender Systems* (RRS) obtain a ranking of reviews by helpfulness for a reader-product pair (Section 5.6).

Such classes are defined with increasing specialization, e.g., a REG method may be applied in place of a GRS technique, but not necessarily the other way around.

Finally, to consistently present the methods in a standardized way, Table 5.2 presents their shared symbols and definitions, using previous definitions from Section 2.2. In the following, we present the classes and their techniques.

5.2 Mean-based Predictors

Mean-Based Predictors are very simple: they compute mean values in the training set and use them as predictions. When the mean is not available (e.g., the training set does not have the votes of an author when using author's mean), any method uses the overall mean instead. There are three techniques in this class, as follows.

Overall Mean Vote (OM). This method predicts every helpfulness as the mean value of all votes in the training set. Thereby, the utility function is computed as:

$$\tau(u_i, r_j) = \frac{1}{|\mathcal{H}^0|} \sum_{h \in \mathcal{H}^0} y_h \tag{5.1}$$

Table 5.2: Symbols used for defining the methods.

y_h	True value of helpfulness for a vote $h \in \mathcal{H}$.
p_h	Predicted value of helpfulness for a vote $h \in \mathcal{H}$, i.e.: $p_h = \tau(u_i, r_j) : \pi_h(h) = (r_j, u_i)$
$\mathfrak{f}_{CLS}$	Function mapping a user u and a review r to the input expected by algorithms in class CLS. The domain is always $\mathcal{U} \times \mathcal{R}$, but the codomain is class-specific.
g	Logistic or sigmoid function, computed as: $g(x) = \frac{1}{1+e^{-x}}$
$\mathbf{v}^T\mathbf{w}$	Inner product of vectors $\mathbf{v}$ and $\mathbf{w}$ of dimension d, defined as: $\mathbf{v}^T\mathbf{w} = \sum_{i=1}^{d} \mathbf{v}_i\mathbf{w}_i$
$\mathbf{M}_{\cdot,c}$	Column vector composed by elements of all rows in column c of matrix $\mathbf{M}$.
$\mathbf{M}_{r,\cdot}$	Row vector composed by elements of all columns in row r of matrix $\mathbf{M}$.
$\mathbf{M}^{m\times n}$	Specifies that matrix $\mathbf{M}$ has m rows and n columns.
$\|\mathbf{v}\|$	The euclidean or L2-norm of vector $\mathbf{v}$ of dimension d, computed as: $\|\mathbf{v}\| = \sqrt{\sum_{i=1}^{d} \mathbf{v}_i^2}$
$\|\mathbf{M}\|_{Fro}$	The generalization of euclidean norm to matrices. Considering $\mathbf{M}^{m\times n}$, then: $\|\mathbf{M}\|_{Fro} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{M}_{ij}^2}$
D_e	Number of features of a given entity $e \in \{v, r, a, s, c\}$ denoting voter (reader), review, author, similarity and connection features, respectively.
$\{x_{1\ldots n}\}$	Set of variables x_i where i ranges from 1 to n, i.e.: $\{x_{1\ldots n}\} = \{x_1, x_2, \ldots, x_n\}$
$\mathbf{x}_e$	Vector of features associated to an entity or relation e, which may be: voter $v_i \in \mathcal{U}$, review $r_j \in \mathcal{R}$, author $a_k \in \mathcal{U}$, similarity s_{ik} or connection c_{ik} between voter $v_i \in \mathcal{U}$ and author $a_k \in \mathcal{U}$ (or a composition of the previous ones).
$\mathcal{H}^0_{v_i}$	Set of votes given by voter $v_j \in \mathcal{V}^0$, i.e.: $\mathcal{H}^0_{v_i} = \{h \in \mathcal{H}^0 \mid \exists r \in \mathcal{R}^0 : \pi_h(h) = (r, v_i)\}$
$\mathcal{H}^0_{r_j}$	Set of votes given to review $r_j \in \mathcal{R}^0$, i.e.: $\mathcal{H}^0_{r_j} = \{h \in \mathcal{H}^0 \mid \exists v \in \mathcal{V}^0 : \pi_h(h) = (r_j, v)\}$
$\mathcal{H}^0_{a_k}$	Set of votes given to reviews of author $a_k \in \mathcal{A}^0$, i.e.: $\mathcal{H}^0_{a_k} = \{h \in \mathcal{H}^0 \mid \exists r \in \mathcal{R}^0, v \in \mathcal{V}^0, p \in \mathcal{P}^0 : \pi_h(h) = (r, v) \wedge \pi_r(r) = (p, a_k)\}$
$\mathcal{H}^0_{v_i a_k}$	Set of votes given by voter $v_i \in \mathcal{V}^0$ to reviews of author $a_k \in \mathcal{A}^0$, i.e.: $\mathcal{H}^0_{s_{ik}} = \{h \in \mathcal{H}^0 \mid \exists r \in \mathcal{R}^0, p \in \mathcal{P}^0 : \pi_h(h) = (r, v_i) \wedge \pi_r(r) = (p, a_k)\}$

Note that the prediction does not depend on user u_i nor review r_j. Indeed, a constant value is predicted for all cases.

Author's Mean Vote (AM). For each author, AM computes the mean of votes associated to all reviews written by that author in the training set. Then, any vote associated to review r_i and user u_i is predicted as the mean of review's author, i.e.:

$$\tau(u_i, r_j) = \frac{1}{|\mathcal{H}^0_{a_k}|} \sum_{h \in \mathcal{H}^0_{a_k}} y_h, \qquad \exists p \in \mathcal{P}^0 : \pi_r(r_j) = (p, a_k) \tag{5.2}$$

In short, we restrict the set of votes as the ones given to the author of r_j and use the mean of such set as prediction.

Voter's Mean Vote (VM). For each voter, VM computes the mean of all votes given by that voter in the training set. Then, prediction is performed as:

$$\tau(u_i, r_j) = \frac{1}{|\mathcal{H}^0_{u_i}|} \sum_{h \in \mathcal{H}^0_{u_i}} y_h \tag{5.3}$$

In this case, the user u_i restricts the set of votes whose mean used for prediction. Methods in this class are mainly used for regression, as OM and VM provide equal predictions for reviews in the same ranking. Since rankings are composed by grouping reviews' votes for a reader-product pair, the voter (reader) mean and the overall mean are the same for all reviews in a ranking. AM is the only one that estimates differently for reviews in the same ranking.

5.3 Regressors

Regressors (REG) use a set of features (e.g.: number of characters of a review, number of reviews written by the author, average helpfulness given by the reader, etc.) to predict an approximation of the review's true helpfulness for a given user. Solutions of this type consider the ordinal nature of the output, opposed to classifiers. Preliminary experiments show that the latter presents worse performance then regressors counterparts; hence, they are not considered.

When used for ranking, REG is called *pointwise* due to learning scores for a query-document pair in isolation [Li, 2011]. Composing a review profile through features distinguishes this class as a *content-based recommender*. Moreover, as a common model is used for all users, who share coefficients, REG is also a *collaborative filtering* approach. Therefore, we apply methods in this class in a *hybrid* format[1].

[1]We did not find any reference for such classification of this class.

Using a regressor for review recommendation requires to apply a transformation to the input as $\mathfrak{f}_{REG} : \mathcal{U} \times \mathcal{R} \mapsto \mathbb{R}^n$, representing a mapping to a vector of n features. The utility function τ is derived with image close to the range of original votes, as algorithms in this class aim at absolute predictions. Three representatives of this class are considered in our experiments and described next.

Linear Regression (LR). This predictor learns a linear model to compute a vote given features related to a reader and a review. The predicted vote is a linear combination of these feature values added to a constant, the intercept:

$$\tau(u_i, r_j) = \mathbf{w}^T \mathbf{x}_{ij} + b, \qquad \mathbf{x}_{ij} = \mathfrak{f}_{REG}(u_i, r_j), \tag{5.4}$$

where $\mathbf{w}$ is a vector of coefficients and b is the intercept; both are adjusted in the learning process from past transactions. Different loss functions might be used for fitting coefficients. Typically, the least squares loss is applied with L_2 regularization [Ng, 2004], called *Ridge Regression* [Murphy, 2012], whose objective function is:

$$E_{LR} = \frac{1}{2} \sum_{h \in \mathcal{H}^0} (p_h - y_h)^2 + \frac{\beta}{2} \|\mathbf{w}\|^2, \tag{5.5}$$

where β is the *regularization factor* that rules the importance of the regularization term. Regularization is used to avoid *overfitting*, a phenomenon of overspecialization in the training set such that the fitted model represents not only the true pattern of the data, but also noise [Ng, 2004]. This situation results in bad performance in the test set. Lower coefficients tend to generate simpler models with a shorter gap of performance between training and test sets; thus, regularization is usually addressed by the norm of coefficients.

Support Vector Regression (SVR). SVR corresponds to the regressor version of Support Vector Machine classifier (SVM) [Vapnik, 1995; Vasconcelos et al., 2015]. A predictor is built by minimizing both model complexity (represented by the norm of the vector of coefficients) and absolute errors of data points outside a margin of tolerance ε. As a consequence, the model focuses on minimizing large errors only.

The trade-off between model complexity and intolerable errors is determined by parameter C, the *penalty factor*. The higher the value of C, the more errors are penalized and the more complex is the learned function. Representing nonlinear relations between features is possible by applying a kernel transformation.

When a linear kernel is used, the predictive function has the same format as LR, as presented in Equation 5.4. However, SVR does not consider errors below the ε tolerance while optimizing. The optimization goal is defined as follows:

$$
\begin{aligned}
&\textit{minimize} \quad E_{SVR} = C \sum_{h \in \mathcal{H}^0} (\xi_h + \xi_h^*) + \frac{1}{2}\|\mathbf{w}\|^2 \\
&\textit{subject to} \quad \forall\ h \in \mathcal{H}^0 : \\
&\qquad\qquad y_h - \mathbf{w}^T \mathbf{x}_{ij} - b \leq \varepsilon + \xi_h \\
&\qquad\qquad \mathbf{w}^T \mathbf{x}_{ij} + b - y_h \leq \varepsilon + \xi_h^* \\
&\qquad\qquad \xi_h, \xi_h^* \geq 0, \\
&\qquad\qquad \pi_h(h) = (u, r), \mathbf{x}_{ij} = \mathfrak{f}_{REG}(u, r),
\end{aligned}
\tag{5.6}
$$

where $\mathbf{w}$ and b are similar to LR, and ξ_h is the error of a lower prediction than the true value and ξ_h^*, of a greater prediction, both for vote h. Like LR, an L_2 regularization is traditionally applied for regularization; but differently, errors are considered as absolute deviations beyond ε, not squared. The trade-off between error and regularization costs is addressed by C instead of β, but replacing $C = \frac{1}{\beta}$ results in an equivalent definition. Figure 5.1 presents an example of a linear SVR model, representing the errors beyond the tolerance from above or below.

Gradient Boosting Regression Trees (GBRT). A regression algorithm with usual high performance for ranking is GBRT [Lucchese et al, 2015], also known as Multiple Additive Regression Trees (MART) [Friedman, 2001]. GBRT learns a non-linear function by combining several predictive functions of the same type (i.e., an ensemble) whose weighted sum of returned values provides the final prediction.

The model is built stage-wise by extending the current function through adding a new decision tree using as ground-truth the pseudoresponses [Burges, 2010]. Pseudoresponses are the negative of the loss function's derivative evaluated at each instance. Therefore, a model update simulates a gradient descent step, which changes the current function towards the negative of the gradient in order to reduce the error. In this

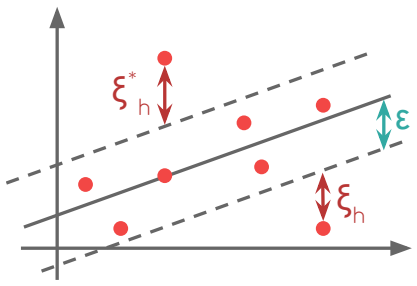


Figure 5.1: Illustration of a linear SVR model. The solid line represents the predictive model and dashed lines delimit tolerable deviations. Only instances whose prediction is beyond the tolerance are considered as errors.

case, an approximation of the gradient is computed through fitting a decision tree on pseudoresponses. The utility function has a format as follows:

$$\tau(u_i, r_j) = \sum_{t=1}^{T} \gamma_t F_t(\mathbf{x}_{ij}), \qquad \mathbf{x}_{ij} = \mathfrak{f}_{REG}(u_i, r_j), \tag{5.7}$$

where F_t is a decision tree built in the t-th iteration fitting on pseudoresponses, T is the number of decision trees and γ_t is the weight given for the result of tree F_t. The value of γ_t is typically chosen to minimize the loss function when the new tree is added, for instance, using Newton's method [Burges, 2010]. The value predicted after reaching a leaf node is, in most cases, just the average of training samples falling there.

5.4 Learning to Rank Predictors

Learning to Rank (LTR) approaches build a model to predict a ranking of documents given a query. Using features of a query-document pair, the model outputs a relevance of the document for the query. Relevance levels are estimated by considering only relative ordering and may even occur in a completely different but more advantageous range (e.g., less expensive to compute, or with a better predictive function upon features).

We evaluate two strategies: RankSVM [Chapelle and Keerthi, 2010] and LambdaMART [Burges, 2010]. They build predictive models using different strategies – pairwise and listwise, respectively – that represent a fair coverage of existing approaches. Such strategies differ in how the loss function is computed. A *pairwise* approach optimizes a loss defined over pairs of documents, typically as a classification problem for one being above or below the other. In turn, a *listwise* method tries to directly optimize a ranking loss function computed on all queries' rankings [Li, 2011]. Similarly to REG class, LTR techniques are applied with item profiles and common user weights, thus being *hybrid* recommenders[2].

An LTR predictor employs an input transformation of the form $\mathfrak{f}_{LTR} : \mathcal{U} \times \mathcal{R} \mapsto \mathcal{Q} \times \mathbb{R}^n$, where $\mathcal{Q}$ is a set of query identifiers and n is the number of features. The set $\mathcal{Q}$ is only used during the training phase for grouping reviews into rankings. Then, using only a feature vector, the predictor is able to output a relevance score. The utility function τ provides outputs not necessarily in the same range of votes, since this class aims at position correctness, not absolute predictions.

[2]Similarly as for REG class, we also did not find any reference for this classification.

RankSVM (RSVM). A representative of a pairwise learning to rank is RSVM, a version of SVM such that the number of inversions between documents in the same query is minimized [Chapelle and Keerthi, 2010; Canuto et al., 2013]. Thus, the problem becomes a classification task solved by SVM, such that input vectors are the differences of vectors from two instances. The two classes are: positive, when the first vector is ranked above; and negative, when it is below. The goal may be stated as:

$$
\begin{aligned}
&\textit{minimize} \;\; E_{RSVM} = C \sum_{h \in \mathcal{H}_P^0} \xi_{h,h'} + \frac{1}{2}\|\mathbf{w}\|^2 \\
&\textit{subject to} \;\; \forall\, h, h' \in \mathcal{H}_P^0 : \\
&\qquad y(\mathbf{w}^T(\mathbf{x} - \mathbf{x}')) \geq 1 - \xi_{h,h'} \\
&\qquad \xi_h \geq 0, \\
&\qquad \pi_h(h) = (u, r), \;\; \pi_h(h') = (u, r'), \\
&\qquad \pi_r(r) = (p, a), \;\; \pi_r(r') = (p, a'), \\
&\qquad \mathbf{x} = m_{REG}(u, r), \;\; \mathbf{x}' = m_{REG}(u, r'), \\
&\qquad y = +1 \leftrightarrow y_h > y_{h'}, \;\; y = -1 \leftrightarrow y_h < y_{h'},
\end{aligned}
\tag{5.8}
$$

where $\mathcal{H}_P^0$ is the set of pairs of votes in training set, restricted to votes in the same ranking. In such definition, h and h' are restricted to belong to the same voter u and corresponding reviews r and r' associated to the same product p. A strict ordering is assumed to assign classes; thus, pairs with equal relevance are disregarded. The hyperplane is defined as $\mathbf{w}^T(\mathbf{x} - \mathbf{x}') = 0$. No error occurs when positive examples are above the upper margin of the hyperplane $\mathbf{w}^T(\mathbf{x} - \mathbf{x}') \geq 1$ and negative examples are below the lower margin $\mathbf{w}^T(\mathbf{x}_i - \mathbf{x}') \leq -1$. An error such that $0 < \xi_{h,h'} \leq \frac{1}{\|\mathbf{w}\|^2}$ occurs

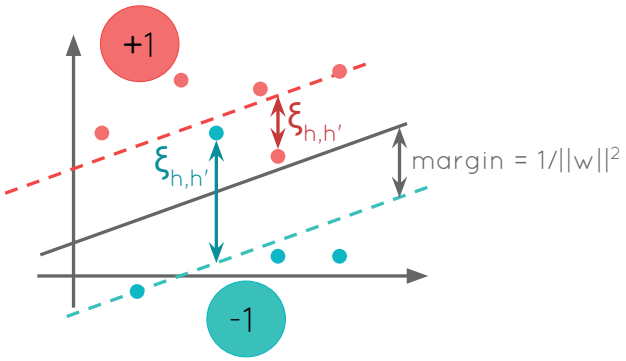


Figure 5.2: Illustration of a RSVM model. There are two classes, a positive and a negative, and each pair corresponds to a classified instance. Any pair beyond the correct margin provides an error equal to the distance to such margin.

for examples within the margin, whereas an error such that $\xi_{h,h'} > \frac{1}{\|\mathbf{w}\|^2}$ is derived for misplaced instances beyond the margin, which lie on the wrong class.

The vector $\mathbf{w}$ is normal to the hyperplane built separating the two classes such that the size of the margin is $\frac{1}{\|\mathbf{w}\|}$. Thus, by minimizing the norm of $\mathbf{w}$, the margin is maximized, making classes farther apart. If $\mathbf{w}^T(\mathbf{x} - \mathbf{x}') > 0$, then the model predicts that h is higher than h', and r is ranked above r'. We can similarly state that $\mathbf{w}^T\mathbf{x} > \mathbf{w}^T\mathbf{x}'$. Thus, the predicted relevance for each instance equals to $\mathbf{w}^T\mathbf{x}$, similar to Linear Regression only without the intercept. A kernel transformation can be used to compose nonlinear separating curves. Figure 5.2 presents a linear model of RSVM by dividing pairs into two classes, and shows the two types of errors.

LambdaMART (LMART). LMART is a combination of Gradient Boosting Regression Trees (GBRT) and LambdaRank [Burges, 2010] and has a central role in the winner technique of Yahoo! Learning to Rank Challenge [Canuto et al., 2013; Chapelle and Chang, 2011]. It empirically optimizes an arbitrary ranking function (e.g., the normalized Discounted Cumulative Gain at position p – nDCG@p). Such direct optimization over rankings distinguishes this technique as listwise [Balog and Ramampiaro, 2013]. Similarly to LambdaRank, a document relevance is updated based on the presence of inversions and proportionally to the score gain after swapping documents; however, it is not a pairwise technique as such updates are aggregated by rankings. Using these updates, an ensemble is built in a stage-wise fashion, as GBRT.

Let d and d' be a pair of documents such that d is more relevant, denoted as $d \succ d'$. This pair has an associated lambda variable $\lambda_{d,d'}$ that is proportional to the derivative of the probability of $d \succ d'$ obtained from the current predictive model. Lambda variables are also proportional to the variation of a ranking cost function when the pair is swapped. These variables are summed for all pairs involving a document d, such that a term is positive if d is above the other or negative if it is below. The overall λ_d of a document represents a force pushing it up or down the ranking. In LMART, such values are used as pseudoresponses and the trees are built similarly to GBRT, composing a similar predictive function. Opposed to LambdaRank, which updates all coefficients after evaluating a query, LambdaMART considers all queries jointly by dividing all documents over the tree leaves.

5.5 General-Purpose Recommender Systems

General-Purpose Recommender Systems (GRS) learn a model for suggesting items, preferentially in a personalized fashion. Fitting such model requires a set of previous

ratings given by users to items, composing the user-item matrix of ratings. Two representatives are addressed: Matrix Factorization [Koren et al., 2009], a model-based collaborative filtering technique, and RLFM [Agarwal and Chen, 2009], a hybrid method.

GRS require an identity transformation of the input as $\mathfrak{f}_{GRS} : \mathcal{U} \times \mathcal{R} \mapsto \mathcal{U} \times \mathcal{R}$. The utility function τ is obtained by minimize ordering error of item's ranking for a given user. Although we define this class as a top-n recommender system, the methods described here are originally defined to predict an exact value for item's utility, which is unnecessary and too restrictive for a ranking task.

Matrix Factorization (MF). This technique models user and item in a K-dimensional space of latent factors. The inner product of user and item latent factors estimates the rating given by a user to an item. The meanings of learned vectors' dimensions are unknown, but they represent properties of items and users that explain existing ratings. They might correspond, for instance, to category, style and even depth of characters [Koren et al., 2009]. Prediction is performed as:

$$\tau(u_i, r_j) = \mathbf{u}_i^T \mathbf{r}_j, \tag{5.9}$$

where $\mathbf{u}_i$ is the latent vector of user u_i, and $\mathbf{r}_j$ is the latent vector of review r_j, both with K dimensions. The dot product reflects the similarity between user and item.

MF may be solved by factorizing the matrix of ratings using Singular Value Decomposition (SVD). However, SVD is undefined when the matrix has missing values, so an imputation technique must be applied. This may not be the best solution as it increases the amount of data and may distort the model [Koren et al., 2009]. Instead, the problem may be stated as a generic optimization and solved using algorithms such as stochastic gradient descent or alternating least squares [Koren et al., 2009]. In this setting, regularization may be additionally introduced, such as an L_2 regularization for latent vectors, weighted by regularization factor β:

$$E_{MF} = \frac{1}{2} \sum_{h \in \mathcal{H}^0} (p_h - y_h)^2 + \frac{\beta}{2} \left(\sum_{u_i \in \mathcal{U}^0} \|\mathbf{u_i}\|^2 + \sum_{r_j \in \mathcal{R}^0} \|\mathbf{r_j}\|^2 \right) \tag{5.10}$$

The model may be adapted to add biases by jointly adjusting bias variables in the optimization algorithm [Ricci et al, 2010]. Thus, using a similar notation of Section 2.1.4, the prediction becomes:

$$\tau(u_i, r_j) = \mu + b_{u_i} + b_{r_j} + \mathbf{u}_i^T \mathbf{r}_j, \tag{5.11}$$

Table 5.3: Definition of variables used by RLFM.

K	Number of dimensions of latent vectors.
α_i	Scalar latent variable related to user $u_i \in \mathcal{V}^0$ following a gaussian distribution centered on the linear regression with coefficients $\mathbf{g}$ of user features: $$\alpha_i \sim \mathcal{N}(\mathbf{g}^T \mathbf{x}_{u_i}, \sigma_\alpha)$$
β_j	Scalar latent variable related to item $r_j, a_k : r_j \in \mathcal{R}^0, a_k \in \mathcal{A}^0$ following a gaussian distribution centered on the linear regression with coefficients $\mathbf{d}$ of item features: $$\beta_j \sim \mathcal{N}(\mathbf{d}^T \mathbf{x}_{r_j a_k}, \sigma_\beta)$$
$\mathbf{u}_i$	Latent vector of length K related to user $u_i \in \mathcal{V}^0$ following a multivariate gaussian distribution centered on K linear regressions, the k-th one with coefficients $\mathbf{G}_{\cdot,k}$, of user features: $$\mathbf{u}_i \sim \mathcal{MVN}(\mathbf{G}^T \mathbf{x}_{u_i}, \mathbf{A}_u)$$
$\mathbf{v}_j$	Latent vector of length K related to item $r_j, a_k : r_j \in \mathcal{R}^0, a_k \in \mathcal{A}^0$ following a multivariate gaussian distribution centered on K linear regressions, the k-th one with coefficients $\mathbf{D}_{\cdot,k}$, of item features: $$\mathbf{v}_j \sim \mathcal{MVN}(\mathbf{D}^T \mathbf{x}_{r_j a_k}, \mathbf{A}_v)$$
$\mathbf{x}_{ij}$	Vector of interaction features. In our specific case, it is the concatenation of $\mathbf{x}_{s_{ij}}$ and $\mathbf{x}_{c_{ij}}$.
$\mathbf{b}$	Linear regression coefficients of $\mathbf{x}_{ij}$.

In short, matrix factorization is a widely used and flexible method. It received many attention recently due to its fundamental role in the winner technique of the Netflix Prize Competition [Koren et al., 2009].

Regression-based Latent Factor Model (RLFM). As a representative of a hybrid technique, RLFM is a method that combines both the benefits of accuracy from collaborative filtering and the ability of recommending for item cold-starts from content-based approaches [Agarwal and Chen, 2009]. The rating given by user u_i to item r_j is predicted as follows:

$$\tau(u_i, r_j) = \mathbf{x}_{ij}^T \mathbf{b} + \alpha_i + \beta_j + \mathbf{u}_i^T \mathbf{v}_j, \tag{5.12}$$

where the variables are detailed in Table 5.3. Since RLFM is not specialized for review recommendation, author's features are not considered separately, but with review's ones composing item features. Regression term on interaction features, $\mathbf{x}_{ij}^T \mathbf{b}$, builds the base

prediction value. Variables α_i and β_j represent user and item biases, respectively, but tied with features through a linear regression. The inner product $\mathbf{u}_i^T \mathbf{v}_j$ is just like in Matrix Factorization, but centered on the regression of features in contrast. Whenever a cold-start is encountered, the regression of the corresponding features replaces the latent variable.

Latent variables and parameters are learned through a Monte Carlo Expectation Maximization (MCEM) procedure [Booth and Hobert, 1999]. A Expectation Maximization (EM) method learns parameters with incomplete data, represented by latent variables. This situation complicates directly maximization of log-likelihood as there is no closed form solution. EM adjusts parameter values iteratively and guarantees convergence to a local optimum of the log-likelihood [Do and Batzoglou, 2008].

Overall, the fitting process comprises a series of iterations representing refinements on the maximization of the log-likelihood. An iteration is divided in two phases: E-step computes the expectation of the log-likelihood on posterior distribution of latent factors, and M-step adjusts parameter by maximizing the expectation.

Specifically, latent factors and parameters are initialized before execution. Then, in E-step, the expectation is not in closed form due to latent vectors inner product term $\mathbf{u}_i^T \mathbf{v}_j$, with an unknown distribution. Gibbs Sampler, a Monte Carlo method, is applied to approximate the unknown distribution of latent factors conditioned on observed data and parameters [Murphy, 2012]. Afterwards, the empiric mean of latent factors provides an approximation of the expectation. Thus, the EM applied is a special version designated MCEM due to the presence of a Monte Carlo sampling. In this scenario, convergence to a local optimum is not guaranteed any longer, although it may still converge under regularity conditions, such as suitable number of iterations and stopping criterion [Booth and Hobert, 1999; Agarwal and Chen, 2009]. In M-step, parameters are adjusted through regression on empirical means, as this maximizes the expectation. Any off-the-shelf regression fitting procedure may be used.

5.6 Review Recommender Systems

Recent works have explored the hypothesis that users do not perceive helpfulness in the same manner; thus, relevant information for review recommendation is incorporated in a personalized fashion. We compare two methods: BETF [Moghaddam et al., 2012], a collaborative filtering technique[3], and CAP [Tang et al, 2013], a hybrid one. To the best

[3]BETF uses reviews' ratings as additional information. However, we do not consider this a content-based trait because they are not used for computing a similarity directly. Instead, BETF jointly fits two collaborative filtering recommender systems: one for rating and one for helpfulness.

of our knowledge, no previous work compares representatives of this class against each other. Also, experiments conducted with them do not provide statistical significance nor assessment using a ranking metric.

We present the representatives of this class in detail as their original papers lack specifics for a precise implementation. Specifically, Moghaddam et al. [2012] present the algorithm, the derivatives and the loss function for ETF, but not for BETF, which has a slightly different algorithm and loss. Tang et al [2013] do not include formulas related to most variables, the log-likelihood nor the algorithm. Therefore, we present a deeper description to fill the gap of definitions for such methods.

Review Recommender Systems (RRS) apply the identity transformation of the input as $\mathfrak{f}_{RRS} : \mathcal{U} \times \mathcal{R} \mapsto \mathcal{U} \times \mathcal{R}$. The utility function τ returns a value indicating a relative helpfulness and aiming at the lowest error for rankings of reviews. Similarly to GRS, techniques considered in this class were originally defined to compute absolute values, but they could calculate only a relative score instead.

UnBiased Extended Tensor Factorization (BETF). BETF is derived from Matrix Factorization with the following extensions: (*i*) the review dimension is replaced by two others, product and author, and the method becomes a Tensor Factorization; (*ii*) a new term is added to the objective function: sum of squared differences of ratings and the inner products of author and product latent factors; (*iii*) accounting for voter and review biases in helpfulness prediction, and for author and product biases in rating prediction [Moghaddam et al., 2012]. Then, the estimated helpfulness of review r_j to user u_i is computed as:

$$\tau(u_i, r_j) = g(\mu_r + \bar{h_{u_i}} + \bar{h_{r_j}} + \mathbf{s} \times_u \mathbf{u}_i \times_v \mathbf{v}_k \times_p \mathbf{p}_l), \qquad \pi_r(r_j) = (a_k, p_l), \tag{5.13}$$

where the variables are explained in Table 5.4. Bias variables are static, calculated in a preprocessing step. There are bias variables related to voter and review impacting vote values, and related to author and product impacting rating values (not used in prediction, only in fitting).

Tensor factorization is performed using Tucker model, which incorporates a central tensor [Shi et al., 2014]. To identify the factorized matrices cells generating each tensor cell, the tensor dimensions are named: u, v and p. Then, the tensor inner product is given by [Moghaddam et al., 2012]:

$$\mathbf{s} \times_u \mathbf{u}_i \times_v \mathbf{v}_k \times_p \mathbf{p}_l = \sum_{x=1}^{K} \sum_{y=1}^{K} \sum_{z=1}^{K} \mathbf{s}_{xyz} \mathbf{u}_{ix} \mathbf{v}_{ky} \mathbf{p}_{lz} \tag{5.14}$$

Table 5.4: Definition of variables used by BETF.

K	Number of dimensions of latent vectors.
μ_h	Overall mean of helpfulness in the training set.
μ_r	Overall mean of rating in the training set.
$\mathbf{s}$	Central tensor of shape $K \times K \times K$, representing dimensions' weights.
$\mathbf{u}_i$	Latent vector of length K related to voter $u_i \in \mathcal{V}^0$ following a gaussian distribution centered on a zero-vector: $\mathbf{u}_i \sim \mathcal{N}(0, \sigma_u^2 \mathbf{I})$
$\mathbf{v}_k$	Latent vector of length K related to author $a_k \in \mathcal{A}^0$ following a gaussian distribution centered on a zero-vector: $\mathbf{v}_k \sim \mathcal{N}(0, \sigma_v^2 \mathbf{I})$
$\mathbf{p}_l$	Latent vector of length K related to product $p_l \in \mathcal{P}^0$ following a gaussian distribution centered on a zero-vector: $\mathbf{p}_l \sim \mathcal{N}(0, \sigma_p^2 \mathbf{I})$
$\bar{h_{u_i}}$	Bias of voter $u_i \in \mathcal{V}^0$ for helpfulness, calculated as the average deviation of user u_i votes from the overall mean.
$\bar{h_{r_j}}$	Bias of review $r_j \in \mathcal{R}^0$ for helpfulness, calculated as the average deviation of review r_j votes from the sum of overall mean and corresponding voter bias.
$\bar{r_{a_k}}$	Bias of author $a_k \in \mathcal{A}^0$ for rating, calculated as the average deviation of user a_k ratings from the overall mean.
$\bar{r_{p_l}}$	Bias of product $p_l \in \mathcal{P}^0$, calculated as the average deviation of product p_l ratings from the sum of overall mean and corresponding author bias.

Latent variables are fitted to minimize the error [Moghaddam et al., 2012], composed of three terms: (*i*) sum of squared errors on vote prediction; (*ii*) sum of squared errors on rating prediction; and (*iii*) regularization of latent factors and central tensor. Note that two models are simultaneously fitted: one for helpfulness and the other for rating. Such error may be represented as[4]:

$$\begin{aligned} E_{BETF} = \frac{1}{2} \sum_{h \in \mathcal{H}^0} (y_h - p_h)^2 + \frac{1}{2} \sum_{r \in \mathcal{R}^0} (y_r - p_r)^2 + \frac{\beta}{2} \sum_{u_i \in \mathcal{V}^0} \|\mathbf{u}_i\|_{Fro}^2 \\ + \frac{\beta}{2} \sum_{a_k \in \mathcal{A}^0} \|\mathbf{v}_k\|_{Fro}^2 + \frac{\beta}{2} \sum_{p_l \in \mathcal{P}^0} \|\mathbf{p}_l\|_{Fro}^2 + \frac{\beta}{2} \|\mathbf{s}\|_{Fro}^2, \end{aligned} \tag{5.15}$$

[4]The original definition has one regularization factor for each type of latent factor, but all of them were applied with an equal value in the experiments. Therefore, we use a single regularization factor.

Table 5.5: Formulas used by BETF algorithm.

$\frac{\partial E^h_{BETF}}{\partial \mathbf{u}_i} = (p_h - y_h)g'(\hat{h})(\mathbf{s} \times_v \mathbf{v}_k \times_p \mathbf{p}_l) + \beta \mathbf{u}_i$
$\frac{\partial E^h_{BETF}}{\partial \mathbf{v}_k} = (p_h - y_h)g'(\hat{h})(\mathbf{s} \times_u \mathbf{u}_i \times_p \mathbf{p}_l) + (p_r - y_r)g'(\hat{r})\mathbf{p}_l + \beta \mathbf{v}_j$
$\frac{\partial E^h_{BETF}}{\partial \mathbf{p}_l} = (p_h - y_h)g'(\hat{h})(\mathbf{s} \times_u \mathbf{u}_i \times_v \mathbf{v}_k) + (p_r - y_r)g'(\hat{r})\mathbf{v}_k + \beta \mathbf{p}_l$
where :
$\hat{h} = p_h = \mu_h + \bar{h_{u_i}} + \bar{h_{r_j}} + \mathbf{s} \times_u \mathbf{u}_i \times_v \mathbf{v}_k \times_p \mathbf{p}_l$
$\hat{r} = \mu_r + \bar{r_{a_k}} + \bar{r_{p_l}} + \mathbf{v}_k^T \mathbf{p}_l$
$\pi_h = (r_j, u_i), \pi_r(r_j) = (p_l, a_k)$

where variables y_r and p_r represent the true rating and the predicted rating, respectively. The only purpose of fitting ratings is to improve the adjustment of $\mathbf{v}_k$ and $\mathbf{p}_l$ latent vectors by using additional data from ratings.

The fitting process uses a Stochastic Gradient Descent (SGD) algorithm to find a local minimum. SGD runs over several iterations and updates the unknown latent variables and coefficients towards the negative of the gradient of the loss function, which is the direction in which the loss reduces the most. The stochastic version considers the loss function for a single example, computing the derivative and adjusting parameters by observing only one instance. As the complete gradient is not computed, the result is stochastic, not necessarily following the direction of the real gradient. However, by doing updates more frequently, convergence is typically reached much faster. The larger the training set, the more costly to compute the complete derivative; thus SGD is an efficient approach for large scale. The update occurs as follows:

$$\phi_{t+1} = \phi_t - \alpha \frac{\partial E^i}{\partial \phi}(\phi_t), \tag{5.16}$$

where E^i is the cost related to a single example i, ϕ is any variable being adjusted, ϕ_t is the value of the variable after the t-th update and α is the learning rate which rules the magnitude of the step towards the gradient. Table 5.5 presents the partial derivatives used in BETF and Appendix B.3 contains their mathematical proofs.

Algorithm 1 represents the BETF procedure. Voters, authors and products are represented as indexes in the rating matrix and helpfulness tensor, received as input. Although the traditional stochastic gradients have terms for helpfulness error, rating error and regularization, the original proposal of BETF performs the updates differently. First, variables are updated considering only the terms related to helpfulness error for each vote, rating error for each review and then regularization for each variable. This way, helpfulness error is emphasized due to the greater cardinality of votes.

Algorithm 1: Algorithm of BETF Review Recommender System

Data: integer: K, max_iter; real: tol, α_0, β; tensor: $\mathbf{H}^{|\mathcal{V}^0|\times|\mathcal{A}^0|\times|\mathcal{P}^0|}$, $\mathbf{R}^{|\mathcal{A}^0|\times|\mathcal{P}^0|}$
Result: utility function τ

initialize $\mathbf{u}^{K\times 1}_{1\dots\mathcal{V}^0}$, $\mathbf{v}^{K\times 1}_{1\dots\mathcal{A}^0}$, $\mathbf{p}^{K\times 1}_{1\dots\mathcal{P}^0}$, $\mathbf{s}^{K\times K\times K}$ with small random values;
scale $\mathbf{H}_{ikl}$ and $\mathbf{R}_{kl}$ in $[0,1]$, $\forall\, i,j,k$;
$\mu_h \leftarrow mean(\{\mathbf{H}_{ikl} \mid i = 1\dots|\mathcal{V}^0|, k = 1\dots|\mathcal{A}^0|, l = 1\dots\mathcal{P}^0, \mathbf{H}_{ikl} \neq ?\})$;
$\mu_r \leftarrow mean(\{\mathbf{R}_{kl} \mid k = 1\dots|\mathcal{A}^0|, l = 1\dots|\mathcal{P}^0|, \mathbf{R}_{kl} \neq ?\})$;
for $i = 1$ **to** $|\mathcal{V}^0|$ **do** $\bar{h_{u_i}} \leftarrow \frac{1}{|\{k,l|\mathbf{H}_{ikl}\neq\ ?\}|}\sum_{k,l:\mathbf{H}_{ikl}\neq\ ?}(\mathbf{H}_{ikl} - \mu_h)$;
for $k,l : k = 1$ **to** $|\mathcal{A}^0|, l = 1$ **to** $|\mathcal{P}^0|$ **do** $\bar{h_{r_{kl}}} \leftarrow \frac{1}{|\{i|\mathbf{H}_{ikl}\neq\ ?\}|}\sum_{i:\mathbf{H}_{ikl}\neq\ ?}(\mathbf{H}_{ikl} - \mu_h - \bar{h_{u_i}})$;†
for $k = 1$ **to** $|\mathcal{A}|$ **do** $\bar{r_{a_k}} \leftarrow \frac{1}{|\{l|\mathbf{R}_{kl}\neq\ ?\}|}\sum_{l:\mathbf{R}_{kl}\neq\ ?}(\mathbf{R}_{kl} - \mu_r)$;
for $l = 1$ **to** $|\mathcal{P}_0|$ **do** $\bar{r_{p_l}} \leftarrow \frac{1}{|\{k|\mathbf{R}_{kl}\neq\ ?\}|}\sum_{k:\mathbf{R}_{kl}\neq\ ?}(\mathbf{R}_{kl} - \mu_r - \bar{r_{a_k}})$;
$t \leftarrow 1$; $\alpha \leftarrow \alpha_0$;
for $it = 1\dots max\_iter$ **do**
  $\alpha \leftarrow \frac{\alpha}{\sqrt{t}}$; $t \leftarrow t + 1$;
  foreach $(i,k,l) \mid \mathbf{H}_{ikl} \neq ?$ **do**
    $\hat{h}_{ikl} \leftarrow \mu_h + \bar{h_{u_i}} + \bar{h_{r_{kl}}} + s \times_u u_i \times_v v_k \times_p p_l$;
    $\mathbf{u}'_i \leftarrow \mathbf{u}_i - \alpha g'(\hat{h}_{ikl})(g(\hat{h}_{ikl}) - \mathbf{H}_{ikl})(\mathbf{s} \times_v \mathbf{v}_k \times_p \mathbf{p}_l)$;
    $\mathbf{a}'_k \leftarrow \mathbf{a}_k - \alpha g'(\hat{h}_{ikl})(g(\hat{h}_{ikl}) - \mathbf{H}_{ikl})(\mathbf{s} \times_u \mathbf{u}_i \times_p \mathbf{p}_l)$;
    $\mathbf{p}'_l \leftarrow \mathbf{p}_l - \alpha g'(\hat{h}_{ikl})(g(\hat{h}_{ikl}) - \mathbf{H}_{ikl})(\mathbf{s} \times_u \mathbf{u}_i \times_v \mathbf{v}_k)$;
    $\mathbf{s}' \leftarrow \mathbf{s} - \alpha g'(\hat{h}_{ikl})(g(\hat{h}_{ikl}) - \mathbf{H}_{ikl})(\mathbf{u}_i \otimes \mathbf{v}_k \otimes \mathbf{p}_l)$;
    $\mathbf{u}_i \leftarrow \mathbf{u}'_i$; $\mathbf{a}_k \leftarrow \mathbf{a}'_k$; $\mathbf{p}_l \leftarrow \mathbf{p}'_l$; $\mathbf{s} \leftarrow \mathbf{s}'$;
  end
  foreach $(k,l) \mid \mathbf{R}_{kl} \neq ?$ **do**
    $\hat{r}_{kl} \leftarrow \mu_r + \bar{r_{a_k}} + \bar{r_{p_l}} + \mathbf{a}_k^T \mathbf{p}_l$;
    $\mathbf{a}'_k \leftarrow \mathbf{a}_k - \alpha g'(\hat{r}_{kl})(g(\hat{r}_{kl}) - \mathbf{R}_{kl})\mathbf{p}_l$;
    $\mathbf{p}'_l \leftarrow \mathbf{p}_k - \alpha g'(\hat{r}_{kl})(g(\hat{r}_{kl}) - \mathbf{R}_{kl})\mathbf{a}_k$;
    $\mathbf{a}_k \leftarrow \mathbf{a}'_k$; $\mathbf{p}_l \leftarrow \mathbf{p}'_l$;
  end
  for $i = 1$ **to** $|\mathcal{V}^0|$ **do** $\mathbf{u}_i \leftarrow \mathbf{u}_i - \alpha\beta\mathbf{u}_i$;
  for $i = k$ **to** $|\mathcal{A}^0|$ **do** $\mathbf{v}_k \leftarrow \mathbf{v}_k - \alpha\beta\mathbf{v}_k$;
  for $i = k$ **to** $|\mathcal{P}^0|$ **do** $\mathbf{p}_l \leftarrow \mathbf{p}_l - \alpha\beta\mathbf{p}_l$;
  if *previous_loss - current_loss* $< tol$ **then**
    break;
  end
end
return $\tau(u_i, r_{kl}) = 5 \times g(\mu_h + \bar{h_{u_i}} + \bar{h_{a_j}} + \bar{h_{p_k}} + \mathbf{s} \times_u \mathbf{u}_i \times_v \mathbf{v}_j \times_p \mathbf{p}_k)$;

†We index reviews by author and product indices k and l, since we do not manipulate reviews explicitly. However, if we use function $\pi_r(r_j) = (a_k, p_l)$ to link the entities, then $r_{kl} = r_j$.

The complexity of the algorithm is $O(max\_iter \cdot K^3 \cdot |\mathcal{H}^0|)$. In practice, we do not iterate over all the tensor $\mathbf{H}$ checking for defined values; instead, we iterate over the training set $\mathcal{H}^0$. The main cost is related to fitting the votes due to $|\mathcal{H}^0| \geq |\mathcal{R}^0|$, as each review is related to one or more votes. Similarly, $|\mathcal{H}^0| \geq |\mathcal{V}^0|, |\mathcal{A}^0|, |\mathcal{P}^0|$, then such cost also surpasses initialization and regularization overheads. We note that the complexity is linear with the number of iterations and the size of the training set, in number of votes, but cubic with the number of dimensions of latent factors K. Thus, the increase of K severely penalizes the computational cost.

Table 5.6: Definition of variables used by CAP.

K	Number of dimensions of latent vectors.
h_{ij}	Helpfulness vote given by voter $u_i \in \mathcal{V}^0$ to review $r_j \in \mathcal{R}^0$. Its true value is denoted by $y_{h_{ij}}$ and predicted, $p_{h_{ij}} = \tau(u_i, r_j)$, related as: $$y_{h_{ij}} \sim \mathcal{N}(p_{h_{ij}}, \sigma_h^2)$$
α_i	Scalar latent variable related to user $u_i \in \mathcal{V}^0$ following a gaussian distribution centered on the linear regression with coefficients $\mathbf{g}$ of voter features, i.e.: $$\alpha_i \sim \mathcal{N}(\mathbf{g}^T \mathbf{x}_{u_i}, \sigma_\alpha^2)$$
β_j	Scalar latent variable related to review $r_j \in \mathcal{R}^0$ following a gaussian distribution centered on the linear regression with coefficients $\mathbf{d}$ of review features, i.e.: $$\beta_j \sim \mathcal{N}(\mathbf{d}^T \mathbf{x}_{r_j}, \sigma_\beta^2)$$
ξ_k	Scalar latent variable related to user $a_k \in \mathcal{A}^0$ following a gaussian distribution centered on the linear regression with coefficients $\mathbf{b}$ of author features, i.e.: $$\xi_k \sim \mathcal{N}(\mathbf{b}^T \mathbf{x}_{a_k}, \sigma_\xi^2)$$
$\mathbf{u}_i$	Latent vector of length K related to user $u_i \in \mathcal{V}^0$ following a gaussian distribution centered on K linear regressions, the k-th with coefficients $\mathbf{W}_{\cdot,k}$, of voter features, i.e.: $$\mathbf{u}_i \sim \mathcal{MVN}(\mathbf{W}^T \mathbf{x}_{u_i}, \mathbf{A}_u)$$
$\mathbf{v}_j$	Latent vector of length K related to review $a_j \in \mathcal{A}^0$ following a gaussian distribution centered on K linear regressions, the k-th with coefficients $\mathbf{V}_{\cdot,k}$, of review features, i.e.: $$\mathbf{v}_j \sim \mathcal{MVN}(\mathbf{V}^T \mathbf{x}_{r_j}, \mathbf{A}_v)$$
$\delta_1(i,k)$	Indicator function which is 1 if user i is similar (using cosine of ratings) to user k above the similarity average of i with all the other users, and 0 otherwise.
$\delta_2(i,k)$	Indicator function which is 1 if user i trusts user k in trust network of the platform, and 0 otherwise.
γ_i^k	Scalar latent variable related to voter $u_i \in \mathcal{V}^0$ and author $a_k \in \mathcal{A}^0$ following a gaussian distribution centered on the sigmoid function applied to the linear regression with coefficients $\mathbf{r}$ of author-voter similarity features, i.e.: $$\gamma_i^k \sim \mathcal{N}(g(\mathbf{r}^T \mathbf{x}_{s_{ik}}), \sigma_\gamma^2)$$
λ_i^k	Scalar latent variable related to voter $u_i \in \mathcal{V}^0$ and author $a_k \in \mathcal{A}^0$ following a gaussian distribution centered on the sigmoid function applied to the linear regression with coefficients $\mathbf{h}$ of author-voter connection features, i.e.: $$\lambda_i^k \sim \mathcal{N}(g(\mathbf{h}^T \mathbf{x}_{c_{ik}}), \sigma_\lambda^2)$$

Context-aware Review Helpfulness Rating Prediction (CAP). By incorporating content and social contexts into the recommendation model, CAP is a hybrid technique relying on both latent and observed features. Four user actions in the platform are addressed: connecting to other through the trust network, rating items,

Table 5.7: Definition of sets used by CAP.

$\mathcal{S}_i^0$	Set of similar users of $u_i \in \mathcal{V}^0$, i.e.: $\mathcal{S}_i^0 = \{a_k \in \mathcal{A}^0 \mid \delta_1(i,k) = 1\}$
$\mathcal{C}_i^0$	Set of trustees of $u_i \in \mathcal{V}^0$, i.e.: $\mathcal{C}_i^0 = \{a_k \in \mathcal{A}^0 \mid \delta_2(i,k) = 1\}$
$\mathcal{S}^0$	Set of all strong similarity pairs, defined as: $\mathcal{S}^0 = \{(u_i, a_k) \in \mathcal{V}^0 \times \mathcal{A}^0 \mid a_k \in \mathcal{S}_i^0\}$
$\mathcal{C}^0$	Set of all strong connection pairs, defined as: $\mathcal{C}^0 = \{(u_i, a_k) \in \mathcal{V}^0 \times \mathcal{A}^0 \mid a_k \in \mathcal{C}_i^0\}$

writing reviews, and evaluating helpfulness. Based on each action, a context is derived from a vote transaction that helps the prediction.

Overall, one content context and four social contexts are explored. The content context refers to the review text. Social contexts are related to: author context, derived from their reputation of writing reviews; rater context, obtained from past vote transactions; connection context, defined by the social network interaction of author and voter; and similarity context, derived from rating similarity of author and voter. Then, CAP extends Regression-Based Latent Factor Models (RLFM) to review recommendation by incorporating each such context [Tang et al, 2013]. The vote of user u_i to review r_j is predicted as:

$$\tau(u_i, r_j) = \mathbf{u}_i^T \mathbf{v}_j + \alpha_i + \beta_j + \xi_k + \delta_1(i,k)\gamma_i^k + \delta_2(i,k)\lambda_i^k, \tag{5.17}$$

whose variables are defined in Table 5.6. This prediction differs from RLFM's in four ways: (i) author and review variables are considered separately in β_j and ξ_k; (ii) author features are not used for computing $\mathbf{v}_j$ mean; (iii) dyadic features x_{ij} are split into two linear regressions with a logistic transformation; (iv) dyadic regressions are transformed by a sigmoid function and wrapped into latent variables γ_i^k and λ_i^k, and only added under certain conditions through indicator variables.

CAP applies MCEM to fit latent variables and parameters like RLFM, but differs on how parameters are adjusted in M-step as specific regression procedures are used. Ordinary Least Squares (OLS) [Murphy, 2012] fits regression parameters related to all latent variables, except γ_i^k and λ_i^k, which are solved by Newton-Raphson [Ypma, 1995] due to sigmoid transformation upon the regression.

The goal of an EM algorithm is to maximize the log-likelihood. In the special case of CAP algorithm, the log-likelihood is given by:

$$
\begin{aligned}
\ell_{\Omega,\Theta} = C &- \frac{1}{2}\sum_{u_i,r_j\in\mathcal{V}^0\times\mathcal{R}^0}\left(\frac{1}{\sigma_H^2}(y_{h_{ij}} - \mathbf{u}_i^T\mathbf{v_j} - \alpha_i - \beta_j - \xi_k - \delta_1(i,k)\gamma_i^k - \delta_2(i,k)\lambda_i^k)^2 + \log\sigma_h^2\right)\\
&- \frac{1}{2}\sum_{u_i\in\mathcal{V}^0}\left(\frac{(\alpha_i - \mathbf{d}^T\mathbf{x}_{u_i})^2}{\sigma_\alpha^2} + \log\sigma_\alpha^2 + (\mathbf{u}_i - \mathbf{W}^T\mathbf{x}_{u_i})^T\mathbf{A}_u^{-1}(\mathbf{u}_i - \mathbf{W}^T\mathbf{x}_{u_i}) + \log(det\mathbf{A}_u)\right)\\
&- \frac{1}{2}\sum_{r_j\in\mathcal{R}^0}\left(\frac{(\beta_j - \mathbf{g}^T\mathbf{x}_{r_j})^2}{\sigma_\beta^2} + \log\sigma_\beta^2 + (\mathbf{v}_j - \mathbf{V}^T\mathbf{x}_{r_j})^T\mathbf{A}_v^{-1}(\mathbf{v}_i - \mathbf{V}^T\mathbf{x}_{r_j}) + \log(det\mathbf{A}_v)\right)\\
&- \frac{1}{2}\sum_{a_k\in\mathcal{A}^0}\left(\log\sigma_\xi^2 + \frac{(\xi_k - \mathbf{b}^T\mathbf{x}_{a_k})^2}{\sigma_\xi^2}\right) - \frac{1}{2}\sum_{u_i,a_k\in\mathcal{S}^0}\left(\log\sigma_\gamma^2 + \frac{(\gamma_i^k - g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))^2}{\sigma_\gamma^2}\right)\\
&- \frac{1}{2}\sum_{u_i,a_k\in\mathcal{C}^0}\left(\log\sigma_\lambda^2 + \frac{(\lambda_i^k - g(\mathbf{h}^T\mathbf{x}_{c_{ik}}))^2}{\sigma_\lambda^2}\right),
\end{aligned}
\tag{5.18}
$$

where the derivation of such value is presented in Appendix B.1, the sets $\mathcal{C}^0$ and $\mathcal{S}^0$ are defined in Table 5.7, C is a constant that does not depend on latent variables Ω nor parameters Θ, defined as:

$$
\begin{aligned}
\Omega = &\{\alpha_{1\ldots|\mathcal{V}^0|}\}\cup\{\beta_{1\ldots|\mathcal{R}^0|}\}\cup\{\xi_{1\ldots|\mathcal{A}^0|}\}\cup\{\gamma_{1\ldots|\mathcal{V}^0|}^{1\ldots|\mathcal{S}_i^0|}\}\cup\{\lambda_{1\ldots|\mathcal{V}^0|}^{1\ldots|\mathcal{C}_i^0|}\}\\
&\cup\{\mathbf{u}_{1\ldots|\mathcal{V}^0|}^{K\times 1}\}\cup\{\mathbf{v}_{1\ldots|\mathcal{R}^0|}^{K\times 1}\}
\end{aligned}
\tag{5.19}
$$

$$
\begin{aligned}
\Theta = \{&\mathbf{g}^{D_v\times 1}, \mathbf{d}^{D_r\times 1}, \mathbf{b}^{D_a\times 1}, \mathbf{r}^{D_s\times 1}, \mathbf{h}^{D_c\times 1}, \mathbf{W}^{D_v\times K}, \mathbf{V}^{D_r\times K},\\
&\sigma_H^2, \sigma_\alpha^2, \sigma_\beta^2, \sigma_\eta^2, \sigma_\gamma^2, \sigma_\lambda^2, \mathbf{A}_u^{K\times K}, \mathbf{A}_v^{K\times K}\}
\end{aligned}
\tag{5.20}
$$

The likelihood is computed as follows. The response $y_{h_{ij}}$ depends on the latent variables. Then, the complete likelihood is given by the dot product of the probability of the response, given the variables, by the probability of the latent variables. As such variables are assumed to be independent, their probability is just the product of the probability of each one. All of them are assumed to follow a gaussian distribution and, after applying the log and performing several simplifications, Equation 5.19 is reached. An interaction variable, γ_i^k or λ_i^k, only impacts the helpfulness if the corresponding indicator variable, $\delta_1(i,k)$ or $\delta_2(i,k)$, is 1. Thus, all pairs with indices i,k such that $k \notin \mathcal{S}_i^0$ and $k \notin \mathcal{C}_i^0$ are outside the model.

In the E-step, as the product $\mathbf{u}_i^T\mathbf{v}_j$ corresponds to the inner product of two multivariate normal distributions and does not have a known density, the posterior distribution of latent factors given observed ratings and parameters $P(\Omega|\mathcal{H}^0,\Theta)$ is approximated by Gibbs Sampling, exactly the same as in RLFM. Using such method requires the values of the empiric mean and variance of the distribution of any variable

Table 5.8: Formulas used by CAP algorithm.

Formula	Similar
$E(\alpha_i\|\Omega-\{\alpha_i\}) = V(\alpha_i\|\Omega-\{\alpha_i\})\left(\frac{\mathbf{d}^T\mathbf{x}_{u_i}}{\sigma_\alpha^2}+\sum_{h\in\mathcal{H}_{u_i}}\frac{h_y-h_p+\alpha_i}{\sigma_H^2}\right)$ $V(\alpha_i\|\Omega-\{\alpha_i\}) = \left(\frac{1}{\sigma_\alpha^2}+\frac{\|\mathcal{H}_{u_i}\|}{\sigma_H^2}\right)^{-1}$	β_j, ξ_k
$E(\gamma_i^k\|\Omega-\{\gamma_i^k\}) = V(\gamma_i^k\|\Omega-\{\alpha_i\})\left(\frac{g(\mathbf{r}^T\mathbf{x}_{s_i^k})}{\sigma_\gamma^2}+\sum_{h\in\mathcal{H}_{s_{ik}}}\frac{h_y-h_p+\gamma_i^k}{\sigma_H^2}\right)$ $V(\gamma_i^k\|\Omega-\{\gamma_i^k\}) = \left(\frac{1}{\sigma_\gamma^2}+\frac{\|\mathcal{H}_{s_{ik}}\|}{\sigma_H^2}\right)^{-1}$	λ_i^k
$E(\mathbf{u}_i\|\Omega-\{\mathbf{u}_i\}) = V(\mathbf{u}_i\|\Omega-\{\mathbf{u}_i\})\left(\mathbf{A}_u^{-1}\mathbf{G}^T\mathbf{x}_{u_i}+\sum_{h\in\mathcal{H}_{u_i}}\frac{(h_y-h_p+\mathbf{u}_i^T\mathbf{v}_j)\mathbf{v_j}}{\sigma_H^2}\right)$ $V(\mathbf{u}_i\|\Omega-\mathbf{u}_i) = \left(\mathbf{A}_u^{-1}+\sum_{h\in\mathcal{H}_{u_i}}\frac{\mathbf{v}_j\mathbf{v}_j^T}{\sigma_H^2}\right)^{-1}$	$\mathbf{v}_j$
$\frac{\partial E^*}{\partial\mathbf{r}} = \frac{1}{\sigma_\gamma^2}\sum_{u_i,a_k\in S^0}\left(E^*_{\{\delta_{ij}^l\}}(\gamma_i^k)-g(\mathbf{r}^T\mathbf{x}_{s_{ik}})\right)g'(\mathbf{r}^T\mathbf{x}_{s_{ik}})\mathbf{x}_{s_{ik}}$ $\frac{\partial^2 E^*}{\partial\mathbf{r}\partial\mathbf{r}^T} = \frac{1}{\sigma_\gamma^2}\sum_{u_i,a_k\in S^0}\left((E^*_{\{\delta_{ij}^l\}}(\gamma_i^k)-g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))g''(\mathbf{r}^T\mathbf{x}_{s_{ik}})-(g'(\mathbf{r}^T\mathbf{x}_{s_{ik}}))^2\right)\mathbf{x}_{s_{ik}}\mathbf{x}_{s_{ik}}$	**h**

given all the others, i.e., for a variable ϕ, $E(\phi|\Omega-\{\phi\})$ and $V(\phi|\Omega-\{\phi\})$. Approximations for the mean is the point $\hat{\phi}$ where the derivative of the log-likelihood is zero $\ell'(\hat{\phi})=0$ and for the variance, the inverse of the second derivative applied to $\hat{\phi}$, i.e., $\ell''(\hat{\phi})^{-1}$ [Booth and Hobert, 1999]. The formulas for the mean and variance for each variable are given in Table 5.8. Variables with similar formulas are presented in column *Similar*, just requiring to replace corresponding variables, parameter and features.

In the M-step, the expectation is optimized towards a maximum by updating parameter values. As we approximate the expectation using a Gibbs Sampling, it is calculated by averaging the log-likelihood value across samples. Setting zero to derivatives of the approximated expectation is equivalent to solve an OLS regression. Let $\mathbf{w}$ be the vector of regression coefficients. Then, OLS is performed as:

$$\mathbf{w} = (\eta\mathbf{I}+\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{5.21}$$

Where $\mathbf{X}_{i,\cdot}$ contains the features and $\mathbf{x}_{u_i}$ contains the response, both from the i-th example. The matrix with η values in the diagonal is added to ease the inversion [Tang et al, 2013]. In the case of a vector such as $\mathbf{u}_i$, K regression problems are solved, one for each dimension. OLS is easily generalized for the vector case by turning $\mathbf{w}$ into a matrix $\mathbf{W}$ such that $\mathbf{W}_{\cdot,k}$ contains the regression coefficients of the k-th dimension and $\mathbf{y}$ also becomes a matrix $\mathbf{Y}=[\mathbf{y}_1\mathbf{y}_2...\mathbf{y}_K]$ of K response arrays.

However, parameters related to interaction variables fall under a different procedure due to the sigmoid transformation of the linear regression. Such parameters are

solved, instead, by Newton-Raphson method that approximates the root of a function [Ypma, 1995]. As derivative's root is a local optimum, the derivative is the main function. We also need the derivative of the main function, which is the second derivative of the expectation in this case. Such formulas are presented in Table 5.8 and are similar for both interaction parameters after performing the proper replacements.

The MCEM method applied for CAP is presented in Algorithm 2. Newton-Raphson function receives two parameters, besides the function and its derivative: max_iter_N and α_N, which represent the maximum number of iterations and the learning rate for fitting the parameter, respectively. The algorithm complexity is $O\left(num\_iter \cdot |\Omega|(num\_samples \cdot K + max\_iter_N \cdot D^3 + D \cdot K + |\mathcal{H}^0| \cdot K)\right)$, where D is the total number of features. All terms have a linear contribution of the number of iterations and the number of variables. Then, the first term corresponds to the sampling cost, which is the highest for array variables. As covariances are ignored, such cost is linear with K. The second term refers to Newton-Raphson cost. A Moore-Penrose pseudo-inverse is computed for each iteration of the method, whose cost is $O(D^3)$. Additionally, in each NR iteration, evaluating the function and its derivative cost $O(|\Omega| \cdot F)$ and $O(|\Omega| \cdot F^2$, respectively, where the second is higher. The third term is related to OLS and is represented by the cost of multiplying the matrix of features with the response matrix, in the array case. Typically, NR will be higher than OLS, but this depends on K. The fourth term corresponds to the update of σ_h^2, which computes all predictions, each one with a cost of $O(K)$ due to the dot product. Note that, differently than BETF (the other RRS approach), CAP's computational cost is linearly related to K and all the other variables, only cubic with D.

In the original publication of CAP, only the Monte Carlo mean and variance of α_i were presented [Tang et al, 2013]. The remainder means and variances and the derivatives of expectation with respect to r and h are all derived here. Additionally, we have observed that CAP is an extension upon RLFM, a fact not previously stated.

5.7 Implementation Specifics

The implementation of most approaches considers available source codes: all regressors – LR, SVR and GBRT – use Scikit-learn for Python [Pedregosa, 2011]; RSVM uses an implementation built upon SVMLight[5] [Joachims, 2006]; LambdaMART employs RankLib[6]; and Regression-Based Latent Factor Models (RLFM) uses the authors' avail-

[5]**Support Vector Machine for Ranking.** `http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`

[6]**The Lemur Project – RankLib.** `http://sourceforge.net/p/lemur/wiki/RankLib`

Algorithm 2: Algorithm of CAP Review Recommender System

Data: integer: K, $num\_samples$, num_iter, max_iter_N; real: η, α_N; sets: $\mathcal{V}^0, \mathcal{R}^0, \mathcal{A}^0, \mathcal{S}^0, \mathcal{C}^0, \mathcal{H}^0, \{\mathbf{x}_{u_{1\dots|\mathcal{V}^0|}}^{D_v \times 1}\}, \{\mathbf{x}_{r_{1\dots|\mathcal{R}^0|}}^{D_r \times 1}\}, \{\mathbf{x}_{a_{1\dots|\mathcal{A}^0|}}^{D_a \times 1}\}, \{\mathbf{x}_{s_{1\dots|\mathcal{V}^0|,1\dots|\mathcal{A}^0|}}^{D_s \times 1}\}, \{\mathbf{x}_{c_{1\dots|\mathcal{V}^0|,1\dots|\mathcal{A}^0|}}^{D_c \times 1}\}$

Result: utility function t

initialize Ω and Θ with small random values;
for $i = 1 \dots num\_iter$ **do**
  `// E-step`
  for $s = 1 \dots num\_samples$ **do**
    foreach $\phi \in \Omega$ **do**
      $\phi^{(s)} \leftarrow \mathcal{N}(E(\phi|\mathcal{L} - \{\phi\}), Var(\phi|\mathcal{L} - \{\phi\}))$;†
    end
  end
  foreach $\phi \in \Omega$ **do**
    $\phi^* \leftarrow mean(\{\phi^{(1)}, \dots, \phi^{(num\_samples)}\})$;
    $\phi^{*var} \leftarrow empiric\_var(\{\phi^{(1)}, \dots, \phi^{(num\_samples)}\})$;∓
  end
  `// M-step`
  foreach $\boldsymbol{\theta} \in \Theta$ **do**
    $\boldsymbol{\Phi} \leftarrow variables\_associated\_to(\boldsymbol{\theta})$;
    $\mathbf{Y} \leftarrow row\_bind([\phi^* for\ each\ \phi \in \boldsymbol{\Phi}])$;
    $\mathbf{X} \leftarrow row\_bind([features\_of(\phi)\ for\ each\ \phi \in \boldsymbol{\Phi}])$;
    if $\boldsymbol{\theta} \in \{r, h\}$ **then**
      $\boldsymbol{\theta} \leftarrow newton\_raphson(iter = num\_iter_N, learning\_rate = \alpha_N, function = \frac{\partial E}{\partial \boldsymbol{\theta}}(\boldsymbol{\Phi}), derivative = \frac{\partial^2 E}{\partial \boldsymbol{\theta}^2}(\boldsymbol{\Phi}))$
    else
      $\boldsymbol{\theta} \leftarrow (\eta \mathbf{I} + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$;
    end
    $\sigma^2 \leftarrow variance\_associated\_to(\boldsymbol{\theta})$;
    $\Sigma^* \leftarrow [\phi^*_{var} for\ each\ \phi \in \Phi]$;
    $n \leftarrow number\_of\_rows(\mathbf{X})$;
    if $\boldsymbol{\theta} \in \{\mathbf{W}, \mathbf{V}\}$ **then**
      $RSS \leftarrow \sum_{i=1}^{n}\sum_{j=1}^{K}(\boldsymbol{\theta}^T\mathbf{X}_{i,\cdot} - \mathbf{Y}_{ij})^2$;
      $\sigma^2 \leftarrow \frac{RSS + \sum_{i=1}^{n}\sum_{j=1}^{K}\Sigma^*_{ij}}{kn}$;
    else
      $RSS \leftarrow \sum_{i=1}^{n}(\boldsymbol{\theta}^t\mathbf{X}_{i,\cdot} - \mathbf{x}_{u_i})^2$;
      $\sigma^2 \leftarrow \frac{RSS + \sum_{i=1}^{n}\Sigma^*_i}{n}$;
    end
    $RSS_h \leftarrow \sum_{h \in \mathcal{H}^0}(p_h - y_h)^2$;
    $\sigma_h^2 \leftarrow \frac{RSS + \sum_{h \in \mathcal{H}^0} empiric\_var(\{p_h^{(1)}, \dots, p_h^{(num\_samples)}\})}{|\mathcal{H}^0|}$;
  end
end
return $\tau(u_i, r_j) = \mathbf{u}_i^T\mathbf{v}_j + \alpha_i + \beta_j + \xi_k + \delta_1(i,k)\gamma_i^k + \delta_2(i,k)\lambda_i^k$;‡

†If ϕ is a vector, then a multivariate normal distribution is used ($\mathcal{MVN}$).

∓ The empiric variance for the vector case is computed as the empiric variance of each dimension independently. Covariances are ignored.

‡When a variable ϕ is undefined for the input entity, it is replaced by the regression on features $\boldsymbol{\theta}^T\mathbf{x}$, where $\boldsymbol{\theta}$ is the weight parameter and $\mathbf{x}$ the feature vector associated. Also, a_k is such that $\exists p : \pi_r(r_j) = (p, a_k)$.

able source code[7]. The other algorithms – Matrix Factorization and the two methods in RRS (BETF and CAP) – were implemented from scratch and are publicly available[8]. Implementation details are shown next.

Preprocessing. All parameter values are scaled within $[0, 1]$, using maximum and minimum values in training set. We also perform mean imputation for feature values that were undefined in training set, such as average rating of a user with no written reviews and average helpfulness of a user who gave no vote.

Initial Values. Setting proper initial value is essential for good performance, both for latent variables and parameters, so we empirically determine the most suitable range of values. The best result for BETF occurs with initial values randomly in the range $(0, 1)$, since it applies logistic transformation. For CAP and MF, a smaller range is required for convergence and we use a random sample from $(10^{-10}, 10^{-8})$.

Bias Baseline. We account for bias as a preprocessing step for all techniques by subtracting the portion of the value explained by such bias baseline predictor. Then, the bias prediction is added back to the methods' prediction. We do not include bias for those techniques that already acknowledge it inside the model, specifically MF, RLFM, BETF and CAP. Biases are computed in a preprocessing step by using stochastic gradient descent method. Then, only the portion of helpfulness votes not explained by bias baseline prediction is fitted in each technique. Since our design has all reviews as cold-starts (because of chronological split, as discussed in Section 4.2), we replace review bias to a sum of author and product biases, inspired by BETF decomposition of review dimension into author and product [Moghaddam et al., 2012].

Cold-start Prediction. For collaborative filtering techniques, without any rule for cold-start prediction, we use the overall mean when a cold-start is found.

Matrix Factorization (MF). Since all reviews are cold-start, we model author of reviews as items under evaluation. Previous studies validated the Author Consistency Hypothesis [Lu et al, 2010] and we provide more evidence in Section 4.5. Additionally, we implement MF with the possibility of jointly bias fitting controlled by a user-defined parameter.

UnBiased Extended Tensor Factorization (BETF). As all reviews are cold-start in our design, we replaced review bias by a sum of biases from product and author, following a similar pattern of review dimension replacement conducted in this method.

[7]**Research Code for LFM.** `http://github.com/yahoo/Latent-Factor-Models`
[8]**Review Recom. Repository.** `http://github.com/lucianamaroun/review_recommendation`

Table 5.9: Summary of the techniques considered.

Method	Advantages	Disadvantages	Calculus of τ	Fitting
OM	Very simple and very fast	Not for ranking (constant), not personalized	Overall mean in the training set	Arithmetic mean of all votes
AM	Simple and fast, works for ranking	Not personalized	Authors' mean in the training set	Arithmetic mean of each author's votes
VM	Simple and fast, personalized	Not for ranking (constant)	Voters' mean in the training set	Arithmetic mean of each voter's votes
LR	Simple and fast	Over restrictive for ranking, only linear relation	Linear combination of features plus intercept	Cholesky, SVD, or iterative optimization[1]
SVR	Robust to noise, non-linear model through kernel	More costly, suffer with high dimensionality	Linear combination of features plus intercept	QP[2] on dual, or iterative optimization
GBRT	Unrestricted non-linear model, good for ranking in general	Tendency to overfit due to unrestricted format	Weighted sum of decision trees	Stage-wise fitting of decision trees on pseudoresponses
RSVM	Ranking goal, non-linear model through kernel	Cost of processing all pairs	Linear combination of features	QP on dual, or iterative optimization
LMART	Ranking goal, unrestricted non-linear model	Tendency to overfit due to unrestricted format	Weighted sum of decision trees' outputs	Stage-wise fitting of decision trees on lambda variables.
MF	Simple, no need of features	Not able to predict for complete cold-starts	Inner product of user and item latent factors plus bias variables and mean	SVD, or iterative optimization
RLFM	Predicts for cold and warm-starts	Need enough features, costly to sample	Sum of regression and latent variables	MCEM
BETF	No need of features, specialized for review recommendation	Not able to predict for cold-starts	Sum of mean, bias variables and tensor product of latent factors	Iterative optimization
CAP	Predicts for cold and warm-starts, specialized for review recommendation	Need enough features, costly to sample and compute NR	Sum of latent variables	MCEM

[1]Any procedure, such as stochastic gradient descent.
[2]Quadratic Programming.

5.8 Concluding Remarks

In this chapter, we defined a set of techniques from five classes of increasing specialization included in our experimental comparison. We described the overall characteristics of each class and how the original input is converted in order to be applied under the specific paradigm. We also presented the behavior of each technique, especially in terms of how the utility function is computed and the optimization objective, when it applies.

Table 5.9 summarizes of all methods considered in this work. We briefly discourse the advantages and disadvantages of each method, as well as the computation of the utility function and the fitting procedure.

As one of our goals is to understand how specialized solutions compare with other methods, we presented RRS approaches – BETF and CAP – in deep. Their original papers [Moghaddam et al., 2012; Tang et al, 2013] lack details required for implementation and analysis, such as precise algorithm and formulas, which we fully presented in this work. Despite being the state-of-the-art for review recommendation, they have not yet been compared in all of the following scenarios: (i) against each other; (ii) with statistical significance against other techniques; (iii) under a ranking perspective; and (iv) with the whole set of techniques presented here, all of them in a personalized setting (except mean-based predictors OM and AM). In the next section, we perform such comparisons, deriving the best overall approach and key aspects for an effective review recommender system.

Chapter 6

Experimental Analysis

In this chapter, we compare the considered techniques under a unified statistical design of experiments. Such design consists of evaluating all techniques after conducting parameter tuning, with both ranking and regression metrics and statistical significance (Section 6.1). We start the analysis by investigating the impact of parametrization (Section 6.2). Then, we compare all the solutions (Section 6.3) and draw some insights regarding review recommendation problem (Section 6.4). In the following, we delve deeper into the SVR predictor, the best performer, and explore extensions to improve this method for the defined task (Section 6.5). Finally, we present concluding remarks of our experimental analysis (Section 6.6).

6.1 Experimental Design

Given the large number of considered techniques, we choose to first group them by their class (e.g., REG, GRS), then evaluate different methods within the classes, and finally compare the best performer of each class. The experimental design is performed with: (i) optimized parameters, by evaluating performance in a validation set; (ii) ranking and regression metrics; and (iii) statistical significance, by considering variability.

With the purpose of assessing the effectiveness of existing methods under different scenarios, we execute them for different data splits. Five splits of the dataset are defined, with sliding windows over votes disposed chronologically. This allows to account for contextual variability, which consists of different behaviors of a technique due to distinct training and test sets. We also consider the stochastic nature of techniques by performing multiple runs for each split; then, we derive the average result for a split.

The splitting scheme works as follows: a window size is fixed with 60% of votes, shifting 10% of votes each time. Such size represents a compromise between overlap of

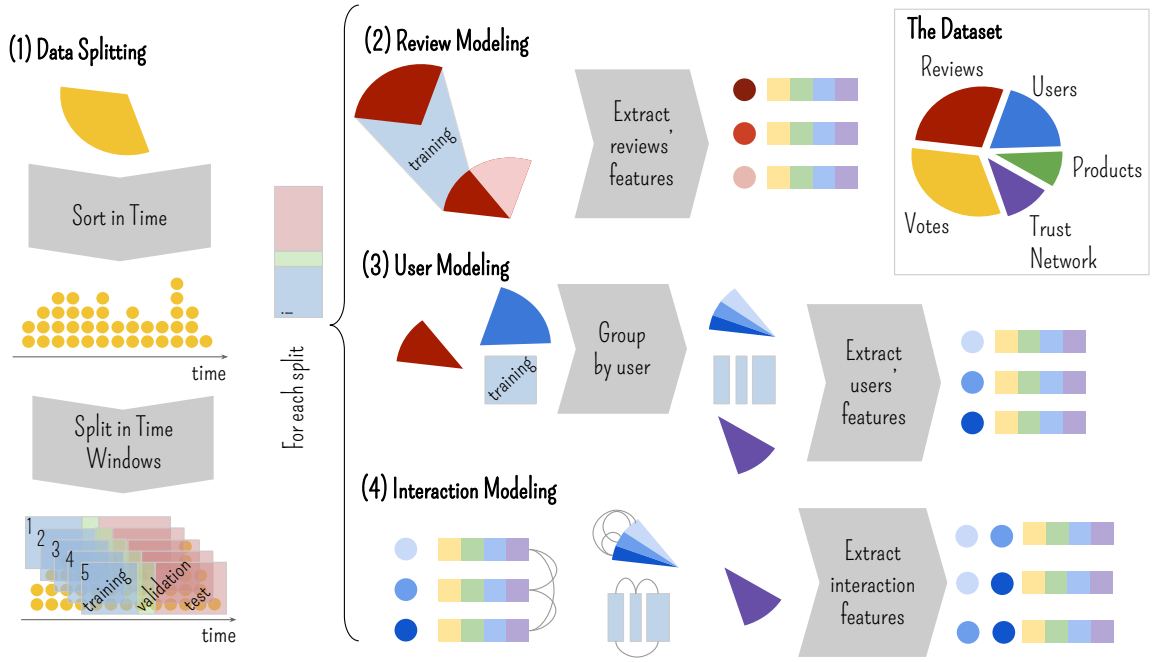


Figure 6.1: Preprocessing step for modeling the data to be used by the review recommender techniques. Votes are divided into splits and, for each split, features are extracted from reviews, users and interactions according to the training set available.

windows and amount of data for learning and evaluation. For each window, the first 40% represent the training set; the next 10%, the validation set; and the last 50%, the test set. Figure 6.1 shows the chronological split and the preprocessing step composing features from the training data for reviews, users and author-voter interactions. Only hybrid and content-based recommenders use such features, but all of them use the corresponding split's training set.

We use a larger test than training set because the former was further reduced by eliminating votes from voter-product pairs comprising less than 5 members. When dividing between training, validation and test sets, the rankings (which have originally at least 10 reviews) may have less than 5 in the test set. This filtering procedure is required for evaluating the methods using a ranking metric over the top 5 reviews. The final sizes of test sets are around 38% for all splits, very close to the training size.

We use a chronological opposed to a random split of the data for building training and test sets, differently of many prior studies [Lu et al, 2010; Moghaddam et al., 2012; Zhang and Tran, 2011]. A chronological split[1] regards temporal constraints when building the predictive model. Specifically, by doing such split, we avoid using future

[1] A solution may consider the hypothesis that users would give the same helpfulness vote regardless of the current time. Even so, if features that change over time are used, such as the number of written reviews, then a chronological split is required.

information to build the predictor (e.g.: feature values computed over data observed after the prediction time), which is unachievable in a real scenario, as already discussed in Section 4.2.

For every algorithm's evaluation, we compute the confidence interval (CI) with 95% of confidence (or likewise 5% of significance) relative to the results on different training-test splits. We compare any two solutions using a paired t-test [Jain, 1991], with the same confidence. In this test, the difference of scores in each split is calculated. Then, a hypothesis test is performed such that the null hypothesis states that the mean of differences is zero. If the such hypothesis is rejected with the desired level of confidence, the methods are significantly different. For such tests, we report the lowest possible significance, the p-value. As we adopt a 5% significance, p-values of 0.05 or lower yield significant results. To apply this test, we assume that performance results for a technique are normally distributed.

When performing parameter tuning, we evaluate different values for all parameters of LR, SVR, MF, RLFM, BETF and CAP, considering their original format. For the other methods, we focus on those that, reportedly, have larger impact on the results, leaving the others at their default values[2]. Note that methods in MBP class have no parameter.

The overall prediction process is represented in Figure 6.2. Initially, the specific input is generated as algorithms may differ in what data they expect. In case of a collaborative filtering approach, a user-item matrix is built without feature vectors, just using the identifiers of entities (users and reviews, possibly authors and products also) in the training set. For content-based approaches, the feature vectors of reviews, users and interactions are concatenated when belonging to the same vote.

After processing the input, the utility function is learned and applied to unseen data for each repetition. Validation and test sets follow the same format in the prediction process. The only difference lies in the purpose: while validation is intended for optimizing parameters, the performance on test set corresponds to the final performance measure of the algorithm and is not used for anything else.

In our evaluation, we consider three metrics: Root Mean Squared Error (RMSE), normalized Discounted Cumulative Gain at position p (nDCG@p) and Expected Reciprocal Rank (ERR). RMSE, a regression metric, is defined as follows:

$$RMSE(\mathcal{H}') = \sqrt{\frac{1}{|\mathcal{H}'|} \sum_{h \in \mathcal{H}'} (y_h - p_h)^2}, \tag{6.1}$$

[2]We did not evaluate different kernels for RSVM due to excessive computational cost.

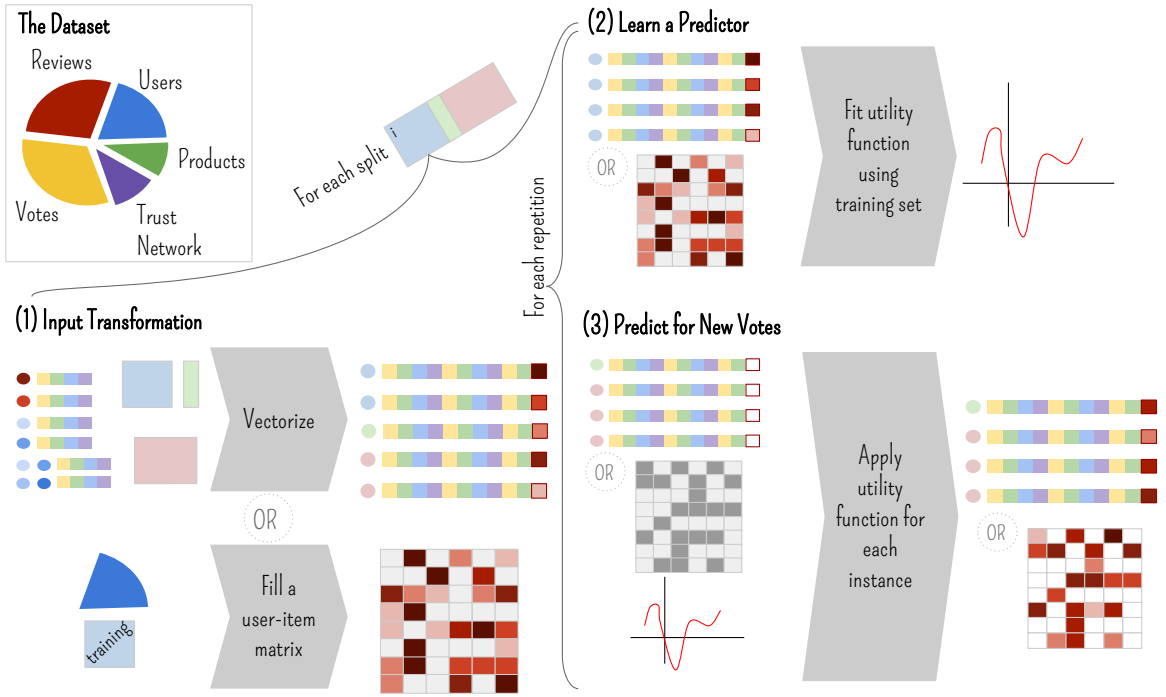


Figure 6.2: Prediction step for any algorithm. The input is composed by a set of feature vectors or user-item matrix. Then, the *utility function* is built and applied to unseen instances, in validation or test sets.

where $\mathcal{H}' \in \mathcal{H}$ is a set of votes whose prediction is under evaluation. Such metric is used in all previous evaluations of specialized methods [Moghaddam et al., 2012; Tang et al, 2013] and is a standard metric for recommender systems [Koren, 2008]. We use it mainly to contrast results under previous assumptions of review recommendation as a rating prediction task. However, the purpose of most recommendation tasks is to compose a ranking of items to a user; so, a ranking metric is more adequate [Balakrishnan and Chopra, 2012] and we also use nDCG@p, defined as:

$$\begin{aligned} DCG@p(\mathcal{O}_{u,p}) &= \sum_{i=1}^{p} \frac{2^{y_{u,r_i}} - 1}{log_2(i+1)}, \\ nDCG@p(\mathcal{O}_{u,p}) &= \frac{DCG@p(\mathcal{O}_{u,p})}{DCG@p(\mathcal{O}^*_{u,p})}, \end{aligned} \tag{6.2}$$

where $\mathcal{O}_{u,p}$ is a ranking of reviews for user u and product p, y_{u,r_i} is the true vote given by user u to the review ranked at position i in $\mathcal{O}_{u,p}$, and $\mathcal{O}^*_{u,p}$ is the ranking of reviews about product p sorted by decreasing true votes by user u (i.e., y_{u,r_i}), which is the best ranking possible. As nDCG@p is defined for a single ranking, our final score is the average nDCG@p of a set of rankings. This metric is beneficial due to its unrestricted

range of relevance values and its exponential decay, severely penalizing the score for lower positions [Anderson, 2014]. We apply such metric for $p = 1 \dots 5$, with a focus on $p = 5$, to address the cost of reading reviews.

Another ranking measure for a scale of relevance is Expected Reciprocal Rank (ERR) [Chapelle et al., 2009]. It considers user navigational behavior through a cascading model. For each element in a ranking, there is an associated probability P_k of a review at position k completely satisfying user needs:

$$P_k = \frac{2^{y_k} - 1}{2^{y_{max}}}, \tag{6.3}$$

where y_k is the helpfulness of a review at position k, y_{max} is the maximum possible relevance (5, in our case), and we use the gain function for estimating probabilities as performed in the original paper [Chapelle et al., 2009]. Thus, the ERR is defined as:

$$ERR(\mathcal{O}_{u,p}) = \sum_{i=1}^{|\mathcal{O}_{u,p}|} \frac{1}{i} P_i \prod_{j=1}^{i-1} (1 - P_j), \tag{6.4}$$

which is the sum of the reciprocal rank weighted by the probability of the user stopping at position i, $i = 1 \dots |\mathcal{O}_{u,p}$, thus being an expectation. Using a cascade model, each probability of stopping at one element is independent, thus allowing the product of probabilities to obtain the joint distribution [Chapelle et al., 2009]. Thus, the event of a user stopping at position i requires that he did not stop at positions from 1 to $i - 1$. We incorporate ERR to double check the performance under a ranking perspective, allowing to see if both metrics agree or not. As an advantage, ERR has a smooth stopping criterion, determined by a user probabilistic model. Similarly to nDCG@p, we evaluate several rankings by computing the mean.

We preferred not to use a binary relevance evaluation (for instance, Mean Average Precision at p – MAP@p) as such practice requires to divide votes into helpful or not helpful, then implying in information loss. Also, it is hard to explicitly state the cut point between the two classes: it may vary across users or it may be interpreted wrongly due to insufficient amount of training data. Finally, in settings where a very helpful review does not exists, it is still important to deliver the most relevant ones among the available set. For example, when no review with helpfulness 5 or 4 exists, differing 3 from the rest is still important.

Finally, the goal of a review recommender is to generate a total ordering of reviews available for each user-product pair. However, the dataset does not contain all reviews of a product with helpfulness votes from all users. Under this restriction, we only

Table 6.1: List of parameters with a description, evaluated range or set, and the best value for ranking (nDCG@5*) and regression (RMSE*).

Method	Symbol	Description	Range/Set	nDCG@5*	RMSE*
LR	β	Regularization factor	$(10^{-6}, 10^{6})$	10^{3}	10^{3}
	bias	Addition of bias	$\{true, false\}$	$false$	$false$
SVR	C	Penalty factor	$(10^{-4}, 10)$	10^{-3}	1
	kernel	Kernel function	$\{linear, poly2, poly3, rbf, sig\}$	$linear$	$linear$
	ε	Addition of bias	$(10^{-4}, 10)$	10^{-1}	10^{-1}
	bias	Addition of bias	$\{true, false\}$	$false$	$false$
GBRT	α	Learning rate	$(10^{-4}, 1)$	10^{-1}	10^{-1}
	T	# Trees	$(10, 10^{3})$	10^{2}	32
	D	Depth of trees	$(1, 5)$	4	2
	loss	Loss function	$\{ls, lad, huber, 0.5/0.9\text{-}quant\}$	ls	ls
	S	Subsample by tree	$(0.2, 1)$	1	0.6
	F	Max. features by tree	$(10, 43)$	43	10
	bias	Addition of bias	$\{true, false\}$	$false$	$false$
RSVM	C	Penalty factor	$(10^{-3}, 10)$	10^{-2}	–
	bias	Addition of bias	$\{true, false\}$	$true$	–
LMART	α	Learning rate	$(10^{-5}, 10^{-1})$	10^{-5}	–
	T	# Trees	$(10, 10^{3})$	10	–
	L	# Tree leaves	$(5, 25)$	5	–
	bias	Addition of bias	$\{true, false\}$	$true$	–
MF	K	# Latent dimensions	$(5, 50)$	25	50
	I	# Iterations	$(10, 50)$	10	50
	tol	Convergence tolerance	$(10^{-6}, 10^{-2})$	10^{-4}	10^{-4}
	α	Learning rate	$(10^{-5}, 1)$	10^{-1}	10^{-1}
	β	Regularization factor	$(10^{-4}, 1)$	10^{-4}	10^{-1}
	bias	Addition of bias	$\{true, false\}$	$true$	$true$
RLFM	K	# Latent dimensions	$(5, 50)$	15	25
	I	# Iterations	$(5, 25)$	20	25
	G	# Gibbs samples	$(10, 10^{3})$	30	30
	B	# Burn-in samples	$(0, 20)$	0	0
BETF	K	# Latent dimensions	$(3, 20)$	4	4
	I	# Iterations	$(5, 25)$	15	20
	tol	Convergence tolerance	$(10^{-6}, 10^{-2})$	10^{-4}	10^{-6}
	α	Initial learning rate	$(10^{-4}, 10)$	10^{-3}	1
	β	Regularization factor	$(10^{-4}, 10)$	1	10^{-3}
CAP	K	# Latent dimensions	$(3, 25)$	5	8
	I	# Iterations	$(5, 25)$	5	10
	G	# Gibbs samples	$(10, 10^{2})$	50	20
	I_N	# NR iterations	$(10, 50)$	10	10
	α	NR learning rate	$(10^{-5}, 10^{-1})$	10^{-2}	10^{-5}
	η	OLS constant	$(10^{-6}, 10^{-2})$	10^{-6}	10^{-5}

evaluate ordering of reviews with available votes for each user-product pair. This is a subset of the intended ranking, but whose relative positioning shall be equally good.

6.2 Parameter Tuning

The best parameter value is the one that minimizes either nDCG@5 or RMSE, according to the final goal. As we evaluate any method with both metrics to contrast them, we performed both tunings. Only LTR approaches are left out of RMSE evaluation

as they are not appropriate for regression: they predict relevance scores in any range, not necessarily the same of helpfulness. Table 6.1 lists all parameters evaluated, with a description, the range or set of values evaluated and the best value for nDCG@5 and RMSE. Each range has at least five values explored such that wide ranges (with more than two orders of magnitude) are explored exponentially (multiplying the previous value by a factor) and the remainder with equally spaced values.

Observing the parameters of REG class regarding nDCG@5, there is a tendency of best performance under simpler models: low $C = 10^{-2}$ and $kernel = linear$ for SVR, and high regularization $\beta = 10^3$ for LR. Additionally, in any method from this class, accounting for *bias* leads to worse performance. For GBRT, we note that the best loss function is least squares (ls), which penalizes high deviations more.

The optimal value for bias in LTR class is *true*, in opposite to REG. Better average performance for smaller values of T and L for LMART indicates that simpler models, with fewer rules, are more efficient.

For MF, jointly updating bias as additional variables is a good strategy. For both GRS methods, more iterations does not represent an improvement in accuracy. The same occurs for the number of Gibbs samples. No burn-in sample, disregarded in the beginning of sampling, was better than any. In general, such values provide a more stochastic model and less prone to overfitting.

Finally, the best parameters of techniques in RRS class also indicate that the simpler the model the better: we observe low to moderate values for K, I, G, and I_N. Moreover, a certain high regularization $\beta = 1$ is the best for BETF.

The distinction among optimal values for nDCG@5 and RMSE is an evidence of their different purposes. In general, when using RMSE criterion, more complex models are built, given by higher K, more iterations I, higher C, lower regularization β.

An additional detail regarding ranking and regression metrics is the optimal number of iterations. In iterative optimization – which is the case of MF, RLFM, BETF and CAP – a large number of iterations and a lower convergence tolerance do not always imply a better performance for nDCG@5. The reason for such unexpected behavior lies in different fitting and evaluation criteria. Here, the fitting goal of the enumerated techniques is minimizing a regression error, and performance is evaluated by considering a ranking metric. More iterations and lower convergence tolerance typically reduce the former, but not necessarily the latter as they are uncorrelated metrics. Even for a similar fitting and evaluation metric, a lower error in training does not always imply the same for test, as optimal RMSE parameters show, although they tend to be higher.

To our knowledge, such study of parametrization for review recommendation is new. Previous works including any of these methods for the same problem did not

Table 6.2: Evaluation of techniques for the three metrics considered. Parameters used for RMSE are different from those of ERR and nDCG@5. For ERR and nDCG@5, the higher the better and for RMSE, the lower the better.

Technique	nDCG@5	ERR	RMSE
OM	0.8808 ± 0.0265	0.6616 ± 0.0032	$\mathbf{0.4148 \pm 0.0489}$
AM	$\mathbf{0.8944 \pm 0.0278}$	$\mathbf{0.6762 \pm 0.0038}$	0.4394 ± 0.0624
VM	0.8808 ± 0.0265	0.6616 ± 0.0032	$\mathbf{0.4323 \pm 0.0681}$
LR	0.9269 ± 0.0178	0.6885 ± 0.0077	$\mathbf{0.3916 \pm 0.0463}*$
SVR	$\mathbf{0.9362 \pm 0.0132}^*$	$\mathbf{0.6983 \pm 0.0121}^*$	$\mathbf{0.3997 \pm 0.0524}*$
GBRT	0.9233 ± 0.0185	0.6857 ± 0.0060	$\mathbf{0.3621 \pm 0.0354}*$
RSVM	0.8790 ± 0.0279	0.6608 ± 0.0024	–
LMART	$\mathbf{0.8929 \pm 0.0277}$	$\mathbf{0.6726 \pm 0.0036}$	–
MF	0.8886 ± 0.0286	0.6703 ± 0.0023	$\mathbf{0.4370 \pm 0.0621}$
RLFM	$\mathbf{0.9229 \pm 0.0197}$	$\mathbf{0.6843 \pm 0.0053}$	$\mathbf{0.4167 \pm 0.0235}$
BETF	0.8869 ± 0.0281	0.6674 ± 0.0023	$\mathbf{0.4161 \pm 0.0509}$
CAP	$\mathbf{0.9213 \pm 0.0196}$	$\mathbf{0.6846 \pm 0.0060}$	0.4354 ± 0.0462

address such a parametric analysis [Moghaddam et al., 2012; Tang et al, 2013]. In fact, Tang et al [2013] only state that parameters were defined using cross-validation, and Moghaddam et al. [2012] perform sensitivity analysis of the number of latent dimensions K[3] [Moghaddam et al., 2012]. Furthermore, both works consider just RMSE metric.

6.3 Comparison of Techniques

We compare all techniques within their classes for nDCG@p, and then the best one of each class to identify the method with highest performance. As RMSE is widely used in the literature, we also quantify this metric using parameter settings optimized for it, except for methods in LTR. Table 6.2 presents average results with the corresponding 95% confidence intervals. The best of each class for a given metric is presented in bold, along with statistical ties. The overall best for a metric, together with statistical ties, has an asterisk suffix. Almost all ICs overlap; however, we compare any two methods using a paired t-test, much more powerful than an unpaired test and yielding many significant conclusions.

Table 6.3 shows the p-values of paired t-tests within classes for the performance of one method being significantly different than the other (thus, the null hypothesis states the equality of means for two methods). The better one is given by the higher

[3]Using what it seems to be the performance in test set.

Table 6.3: Result of paired t-test represented by p-values for each class regarding all metrics. The null hypothesis is represented by a pair of methods whose mean is tested on equality.

	p-value		
Hypothesis	**nDCG@5**	**ERR**	**RMSE**
OM = AM	< 0.0001	< 0.0001	0.0107
OM = VM	< 0.0001	< 0.0001	0.1188
AM = VM	< 0.0001	< 0.0001	0.2659
LR = SVR	0.0067	0.0054	0.0883
LR = GBRT	0.0104	0.0243	0.1847
SVR = GBRT	0.0040	0.0050	0.1064
RSVM = LMART	< 0.0001	< 0.0001	–
RLFM = MF	0.0005	0.0005	0.3095
CAP = BETF	0.0004	0.0003	0.0407

average performance in Table 6.2. We note that, in such tests, the same conclusions are obtained for nDCG@5 and ERR, while RMSE differs.

OM and VM do not to generate meaningful rankings of reviews. Their predictions are constant for a reader-product pair, thus, for all reviews in the same ranking. This is caused by OM predicting a constant value for all reviews and VM predicting a constant value to the same voter, both unique values for a given ranking. Thus, they do not change the ordering of reviews, composing a worst-case solution. On the other hand, AM provides different predictions for a ranking and performs significantly better. While OM and VM are not suited for ranking, they perform well for regression. OM, the simplest one, gives the best result for RMSE, statistically tied with VM and significantly outperforming AM with 95% of confidence.

All three methods in the REG class produce statistically different average of nDCG@5. Regarding RMSE, they are not statistically different though. Also, for nDCG@5, SVR is the best followed by LR and GBRT. For RMSE, when comparing average results, such an order is inverted.

For the remainder classes, LambdaMART has significantly the best performance for both ranking metrics within LTR class. In GRS, RLFM is significantly better; yet, when using RMSE, both techniques are statistically tied. Finally, in RRS class, CAP significantly outperforms BETF for nDCG@5 and ERR, whereas BETF is better considering RMSE. This illustrates the difference between ranking and regression goals. Also, collaborative filtering may work for the latter, but not quite for the former.

Table 6.4: Obtained p-values from paired t-test of best of classes for nDCG@5 by comparing a technique defined in the column with the one in the row.

	LR	RLFM	CAP	LMART	AM
SVR	0.0067	0.0053	0.0037	0.0012	0.0014
LR		0.0046	0.0013	0.0008	0.0010
RLFM			0.0219	0.0006	0.0008
CAP				0.0007	0.0010
LMART					0.0010

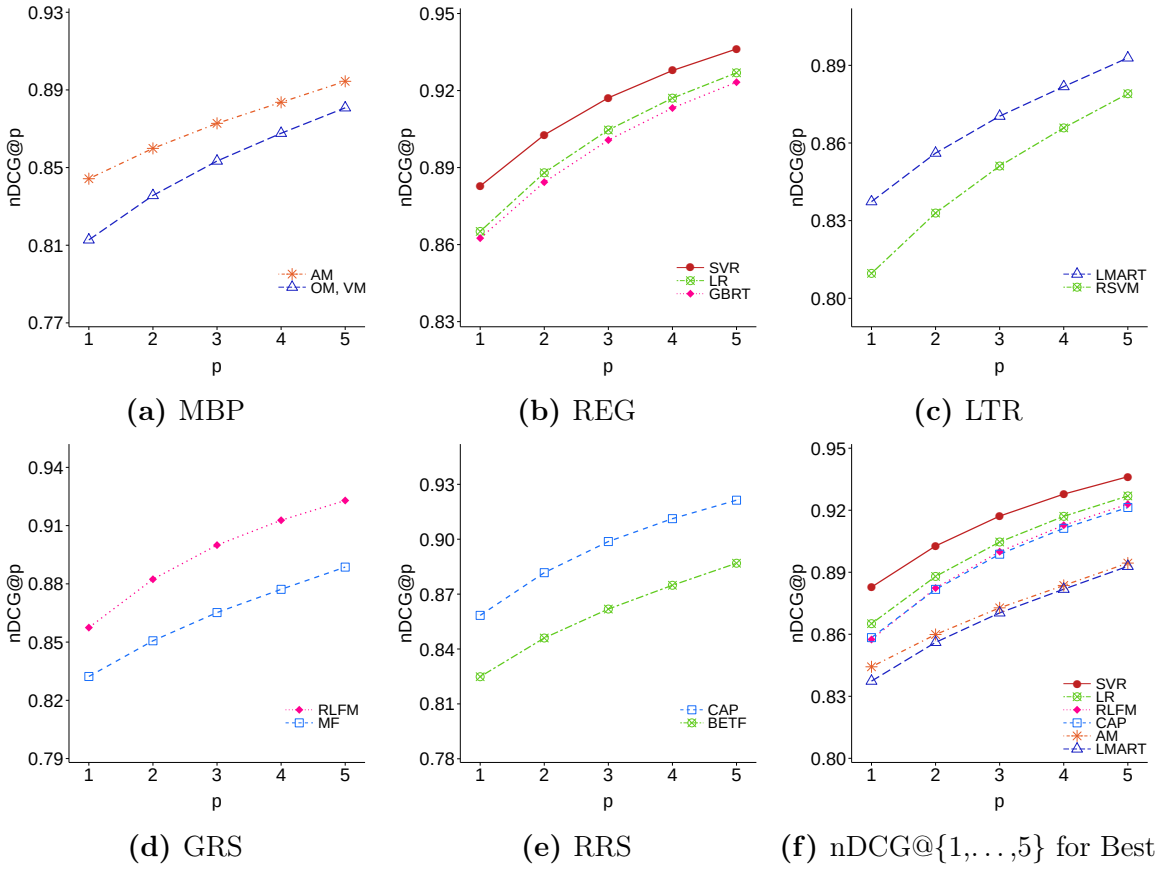
Table 6.5: Obtained p-values from paired t-test of best of classes for ERR by comparing a technique defined in the column with the one in the row.

	LR	RLFM	CAP	LMART	AM
SVR	0.0054	0.0051	0.0038	0.0013	0.0022
LR		0.0102	0.0040	0.00063	0.0022
RLFM			0.4824	0.0002	0.0017
CAP				0.0003	0.0017
LMART					0.0033

Table 6.6: Obtained p-values from paired t-test of best of classes for RMSE by comparing a technique defined in the column with the one in the row.

	OM	RLFM	BETF
GBRT	0.0210	0.0125	0.0227
OM		0.8994	0.1940
RLFM			0.9696

The p-values obtained from paired t-test of the best methods using nDCG@5, ERR and RMSE are shown in Tables 6.4, 6.5 and 6.6, respectively. SVR is significantly the overall best for both ranking metrics, followed by LR and both are statistically above all the others. RLFM significantly outperforms CAP using nDCG@5, but they are equivalent when considering ERR. LMART is worse than AM for both ranking metrics. The comparison is much more difficult regarding RMSE. In fact, GBRT outperforms all the other classes' representatives, which are statistically tied. The good performance of OM shows how such metric is vulnerable to high performance of naive predictors in the case of highly unbalanced distributions of responses.



(a) MBP **(b)** REG **(c)** LTR

(d) GRS **(e)** RRS **(f)** nDCG@{1,...,5} for Best

Figure 6.3: Comparison of techniques within classes in Figures **(a)**-**(e)** and of best of each class in Figure **(f)** using nDCG@p. The set of best algorithms includes LR, as it stands between SVR and the other classes in performance.

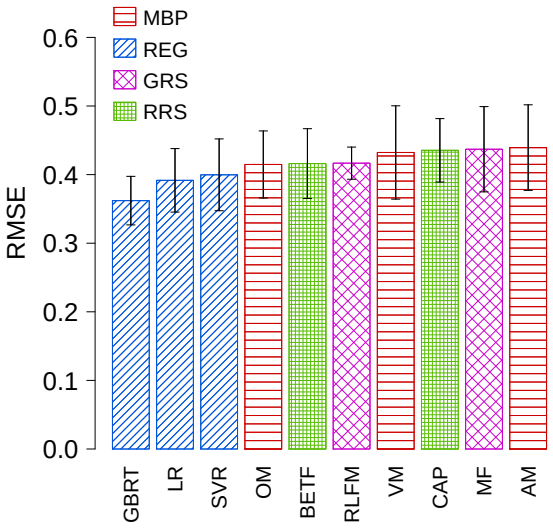
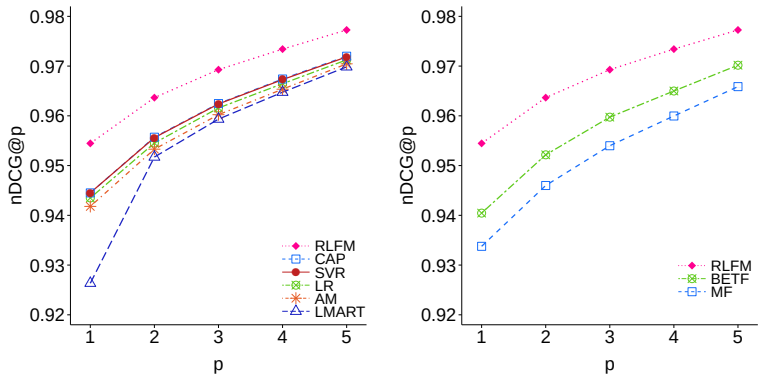


Figure 6.4: Comparison of all techniques but LTR regarding RMSE.



(a) Random split **(b)** Random split for CF

Figure 6.5: Comparison of techniques with nDCG@p using random splits. The best of each class is in Figure **(a)** and the best performer, RLFM, and collaborative filtering (CF) methods are in Figure **(b)**. RLFM is significantly the best.

Figures 6.3a to 6.3e show the average performance for classes using nDCG@p. The relative order of methods is consistent across values of p. 6.3f contrasts the nDCG@p of the best technique of each class, together with LR. Such method is the second best in REG, but above all the others, while comprising a very basic approach. Similarly, the performance is consistent with p, only CAP and RLFM are swapped by little for nDCG@1. Figure 6.4 presents visually the performance regarding RMSE.

We also compare the techniques by considering random splits of training and test sets, shown in Figure 6.5. Even though such split is often employed by previous works, we argue that it is not a valid approach, as discussed in Section 4.2. Figure 6.5a shows that this setting may lead to different conclusions: under a favorable context, with low rates of cold-start reviews, SVR is not the best approach, but RLFM.

As BETF and MF, both collaborative filtering approaches, are penalized by high rate of cold-starts, we also investigate their performance for such random splits. Although they improve in performance, they are still not the best approaches. Moreover, the performance of all methods are somewhat inflated by using random splits when compared to the results obtained with a chronological split, though not proportionally. Such results serve to show the importance of considering time constraints for a solid evaluation of recommendation methods.

Table 6.7: Ranking of best of classes according to each metric.

nDCG@5	SVR $\succ$ LR $\succ$ RLFM $\succ$ CAP $\succ$ AM $\succ$ LMART
ERR	SVR $\succ$ LR $\succ$ RLFM = CAP $\succ$ AM $\succ$ LMART
RMSE	GBRT $\succ$ OM = RLFM = BETF

Finally, Table 6.7 shows the ranking of classes representatives for each metric. We note that nDCG@5 and RMSE are uncorrelated metrics, as the relative performance of the methods change drastically among both. For example, unlike observed for nDCG@5, when RMSE is used as evaluation metric, we find that GBRT is the best method, while even naive MBP techniques have competitive performance. ERR and nDCG@5 tend to follow a same pattern, although there is a small variation, specifically whether RLFM and CAP are statistically different or not. Overall, nDCG@5 and ERR are very consistent, demonstrating that the ranking evaluation performed is cohesive.

6.4 Discussion of Results

We divide our discussion on the presented results into four main competitive properties of the techniques: ranking or regression objective, usage of observed or latent features, simple or sophisticated nature and absent or present bias. Then, we contrast our conclusions with previous results addressed in specialized baseline experiments.

6.4.1 Ranking versus Regression Goal

Although review recommendation is intrinsically a ranking problem, our results reveal that LTR methods are not the best performers for such task; their results are at best comparable to those produced by AM. They hardly learn good review orderings under sparse scenarios, i.e., when there are few evaluations of reviews for reader-product pairs. Also, a small portion of the features are related to user and review, which corresponds to query and document interaction, while such features are the most important for learning to rank [Chapelle and Chang, 2011]. In general, a few features penalize severely such learning process; for instance, in Yahoo Learning to Rank Challenge, more than 400 features are used.

Yet, probably the main problem with learning to rank solutions refers to the enormous amount of ties in pairs of reviews. In average, only 11.5% of all pairs in a given split have different relevance values for the filtered dataset used. Thus, from all data in training set, pairwise techniques, such as RankSVM, only use around one tenth as they disregarded tied pairs. Despite being listwise, LambdaMART compute the change in a performance metric by swapping pairs, but such action typically do not change the metric. Learning from a very small subset of the training set makes it difficult to derive a good predictor.

In spite of using a ranking goal for learning being not satisfactory, it is still important for evaluation as it is more closely related to the way results are presented

to the user. Indeed, ranking and regression are clearly uncorrelated problems for review recommendation, which is notable by very different results and conclusions addressed by nDCG@p and RMSE. Even naive predictors based on the mean perform well for RMSE, then making difficult to distinguish the methods with statistical significance. Since the ultimate goal is to provide an ordering of reviews to a reader, finding good predictors using ranking performance is a priority.

6.4.2 Observed versus Latent Features

Clearly, collaborative filtering techniques, very popular in recommender systems, do *not* perform as well for review recommendation. Sparseness is increased for reviews as recommendable items, since their volume is typically greater and their rate of increase is even higher. The cost of writing a review, indeed, is much lower than producing a new product. When there are many cold-starts, using observed features is crucial.

In our design, all reviews are cold-start. Nonetheless, Tang et al [2013] measure the rate of cold-starts in the original dataset, resulting in 44.72% in chronological sorting[4]. Still, almost half of votes to predict may not rely in latent factors, being mandatory to use features in order to overcome such sparseness. Additionally, Tang et al [2013] describe that CAP method with only latent vector inner product term is better than MF in warm-start case due to observed features, although they use RMSE.

Differently from MF, RLFM incorporates observed features into the model, then allowing prediction for cold-starts. Such behavior also results in better performance regarding nDCG@p. Therefore, considering that the review recommendation scenario is highly sparse, observed features are very important to complement historical behavior. This is also noted by the performance of BETF, a purely collaborative filtering approach that suffers severely from cold-starts. This method is outperformed by CAP using nDCG@p and ERR, although better in terms of RMSE. This suggests that observed features are especially important for ranking-oriented recommendation.

6.4.3 Simple versus Sophisticated Predictive Model

Author's reputation reveals important by AM performance for ranking, surpassing LMART. A very simple model may capture great part of observed values. Therefore, replacing the review as recommended item by the author seems to be a reasonable approach. Indeed, MF approximates the RMSE of RLFM by applying such a concept, being not significantly different.

[4]Tang et al [2013] use the same data and report a chronological split using vote timestamps. However, such information is not available in the dataset made publicly available.

Based on the experimental results, we observe that simpler models are usually more efficient than sophisticated ones, as suggested by high regularization factors, low penalty factors, few iterations for optimization and so on. Also, intrinsic simple models presented reasonably good performance, such as LR. Overall, a better generalization pattern is encountered in such setting and, then, predictors suffer less from overfitting. In fact, a linear model of features is as good as a kernel transformation for SVR.

Complex models from more generic classes, such as GBRT, RSVM and LMART, are behind in performance than simpler models. RLFM and CAP are the more specialized ones with better performance; they are better than their counterparts especially by using observed features. However, such capability is not enough to surpass SVR using nDCG@5, nor GBRT in terms of RMSE. Thereby, the specialized methods, BETF and CAP, are incapable of overcoming simpler methods in both metrics.

6.4.4 Absent versus Present Bias

Our results indicate that adding a bias factor, which typically improves recommendation results, is *not* a good practice for Regressors (REG), unlike for Learning-to-Rank (LTR) and General-Purpose Recommender Systems (GRS). This is likely due to the presence of many cold-starts, causing bias distributions in training and test to be discrepant. Then, test performance is degenerated, especially for absolute predictions.

On the other hand, for relative prediction in LTR techniques, bias improves the results. In this case, the model learned by removing bias effect and focusing on interaction overcame the discrepancies of biases in training and test sets. Additionally, voter average helpfulness is a feature that may replace bias in REG, but not in learning to rank as such feature is constant for a query. Finally, bias may improve performance for a predictor relying on few training samples (such as LTR does due to many tied pairs) by guessing right in some cases. But when added to an already robust model, the performance drops.

6.4.5 Contrasting with Previous Experiments

BETF was also compared with LR, SVR and MF in the original proposal paper, whose conclusion pointed towards the best performance of the first in terms of RMSE [Moghaddam et al., 2012]; yet, we conclude the opposite, except for MF. Our experiments are different due to four reasons: (i) we consider LR and SVR not only with non-personalized features, but also personalized; (ii) we evaluate using a chronological

split while the authors used a random and in a different dataset[5]; (iii) we use around the same amount of data for training and test, while the authors used 90% for training and the remainder for test; (iv) we compare solutions through a statistical test. Especially the larger training set benefits BETF as less cold starts occurs in test set.

We conclude from the results that, regarding RMSE, BETF is indeed better than MF (p-value equal to 0.0117); thus tensor modeling and simultaneously fitting ratings indeed improve over plain MF. However, SVR and LR are statistically better regarding RMSE (p-values equal to 0.0002 and 0.0010, respectively). Regarding ranking metrics, MF, LR and SVR have statistically higher performance than BETF under paired t-tests (the p-values are 0.0029, 0.0008 and 0.0005, respectively). MF is better for ranking probably due to its simplicity, as we note by several indication of better performance of simpler models for ranking. In our experiments, BETF suffers too much from cold-start, being the probable reason for being surpassed by SVR and LR in all evaluations.

Even though in the original paper [Zhang and Tran, 2011] CAP is compared with LR, only non-personalized features were used for such baseline. In their report, CAP outperformed LR in terms of RMSE. However, they did not address the statistical significance, besides neglecting the potential of using personalized features for LR. In our experiments, we found that LR is better both for ranking and regression, significantly through paired t-tests (p-values equal to 0.0013 and 0.0004, respectively).

CAP was previously claimed to be better than MF [Tang et al, 2013] in a work using the same dataset from Ciao UK, but without the filtering (ignoring invalid instances and requiring rankings with at least 10 reviews). In our experiments, CAP is statistically tied with MF using RMSE (the p-value is 0.8518). We use MF with bias and parameter tuning, procedures that are not clear in the original paper [Tang et al, 2013]. Differently from them, we apply MF with review replaced by author dimension (to overcome cold-starts) and we evaluate the performance in five scenarios with statistical tests. In turn, CAP is significantly better for ranking (the p-value is 0.0006).

We also observe that CAP produces results that are statistically worse than RLFM using nDCG@5 and statistically tied regarding ERR. Thus, the new model structure proposed in CAP is not significantly better. In fact, adding interaction features with plain regression is better than incorporating with logistic transformation and under certain constraints. In short, incorporating observed features reveals more relevant than changing the model structure. Such comparison with RLFM was not presented in the work proposing CAP [Tang et al, 2013], although the second is an extension upon the first.

[5]The authors use a dataset from Epinions, whose link for dowloading is currently broken.

Table 6.8: Prediction of reviews helpfulness for the considered example by the three best techniques, along with the ground-truth for helpfulness.

Review ID	Truth	SVR	LR	RLFM
177269	5	3.9217	3.9364	4.1653
7252	5	3.9196	3.9878	3.8722
82581	5	3.9190	3.9900	4.1733
11495	4	3.9187	3.9668	4.2678
70030	4	3.9184	4.0222	4.5724
71158	4	3.9183	3.9746	4.2137
143356	4	3.9180	4.0148	4.3899
130504	4	3.9175	3.9593	4.1021
171367	4	3.9170	3.9139	3.8908
74342	4	3.9167	3.9079	3.9972
5915	4	3.9161	3.8828	3.7378
26782	4	3.9149	3.8312	3.9288
18028	4	3.9148	3.8353	3.8629

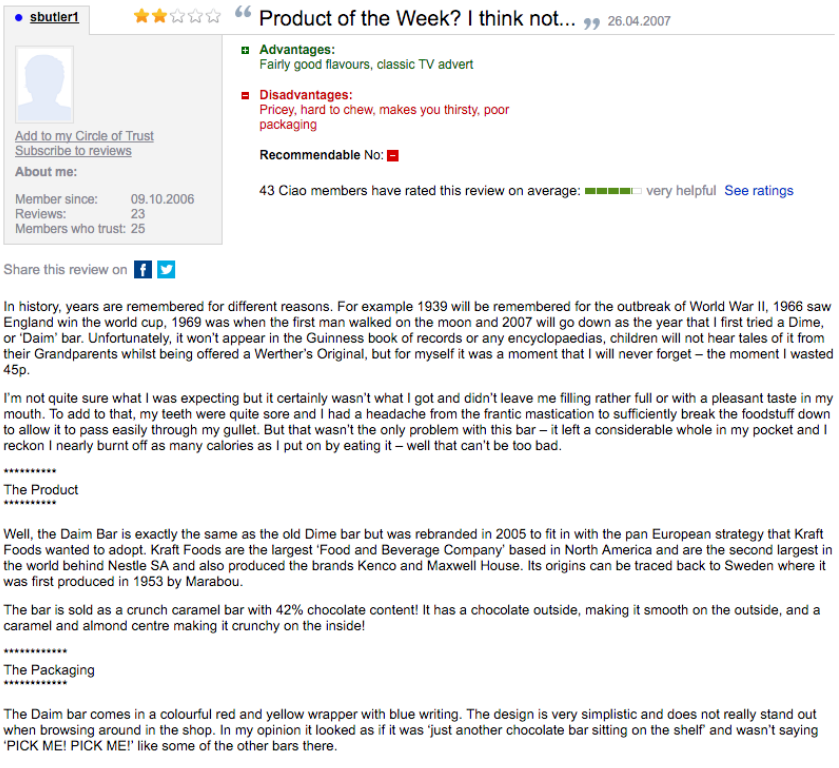
Table 6.9: Prediction of rankings by the three best techniques for the top 5. The ranking is obtained by sorting with predicted values and the corresponding true value of each position is presented.

Position	SVR	LR	RLFM
1	5	4	4
2	5	4	4
3	5	5	4
4	4	5	4
5	4	4	5

6.4.6 Illustrative Example

We now present an example of how SVR performs well for ranking, then contrasting with LR and RLFM, the second and the third best ones, respectively. We consider the product *Daim Bar* (a caramel bar covered with chocolate) and the reader with id *6044343*. This user configures as a cold-start, with only one known vote equal to 5.0, but engaged in the social network, with 92 trustors and 321 trustees. Table 6.8 lists all reviews evaluated by such user, along with the true helpfulness given by him and the prediction of SVR, LR and RLFM, the best ones for ranking.

In this example, we note that SVR predicts in a smaller range than the others, limiting regression performance, but capturing relative order better. Indeed, the ranking obtained by using SVR predictions is perfect, with nDCG@5 equals to 1.0, followed by LR with 0.754806 and RLFM with 0.754806. Contrasting with RMSE performance,



sbutler1

Product of the Week? I think not... 26.04.2007

Advantages:
Fairly good flavours, classic TV advert

Disadvantages:
Pricey, hard to chew, makes you thirsty, poor packaging

Recommendable No:

43 Ciao members have rated this review on average: very helpful See ratings

Add to my Circle of Trust
Subscribe to reviews

About me:

Member since: 09.10.2006
Reviews: 23
Members who trust: 25

Share this review on

In history, years are remembered for different reasons. For example 1939 will be remembered for the outbreak of World War II, 1966 saw England win the world cup, 1969 was when the first man walked on the moon and 2007 will go down as the year that I first tried a Dime, or 'Daim' bar. Unfortunately, it won't appear in the Guinness book of records or any encyclopaedias, children will not hear tales of it from their Grandparents whilst being offered a Werther's Original, but for myself it was a moment that I will never forget – the moment I wasted 45p.

I'm not quite sure what I was expecting but it certainly wasn't what I got and didn't leave me filling rather full or with a pleasant taste in my mouth. To add to that, my teeth were quite sore and I had a headache from the frantic mastication to sufficiently break the foodstuff down to allow it to pass easily through my gullet. But that wasn't the only problem with this bar – it left a considerable whole in my pocket and I reckon I nearly burnt off as many calories as I put on by eating it – well that can't be too bad.

****
The Product
****

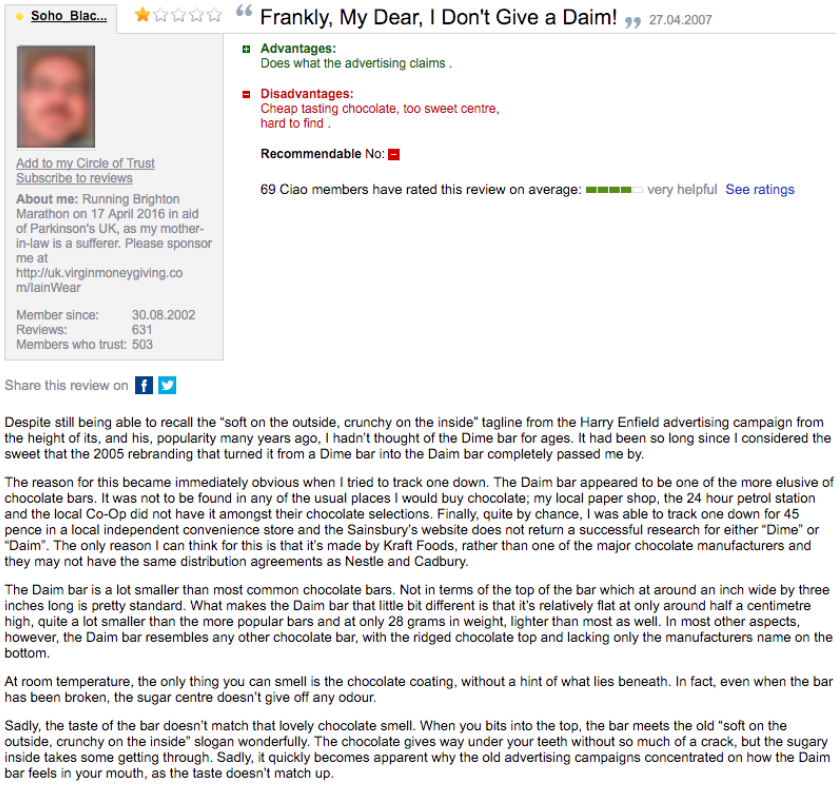
Well, the Daim Bar is exactly the same as the old Dime bar but was rebranded in 2005 to fit in with the pan European strategy that Kraft Foods wanted to adopt. Kraft Foods are the largest 'Food and Beverage Company' based in North America and are the second largest in the world behind Nestle SA and also produced the brands Kenco and Maxwell House. Its origins can be traced back to Sweden where it was first produced in 1953 by Marabou.

The bar is sold as a crunch caramel bar with 42% chocolate content! It has a chocolate outside, making it smooth on the outside, and a caramel and almond centre making it crunchy on the inside!

****
The Packaging
****

The Daim bar comes in a colourful red and yellow wrapper with blue writing. The design is very simplistic and does not really stand out when browsing around in the shop. In my opinion it looked as if it was 'just another chocolate bar sitting on the shelf' and wasn't saying 'PICK ME! PICK ME!' like some of the other bars there.

(a) Review with helpfulness equal to 5 (ID: 177269)



Soho_Blac...

Frankly, My Dear, I Don't Give a Daim! 27.04.2007

Advantages:
Does what the advertising claims .

Disadvantages:
Cheap tasting chocolate, too sweet centre, hard to find .

Recommendable No:

69 Ciao members have rated this review on average: very helpful See ratings

Add to my Circle of Trust
Subscribe to reviews

About me: Running Brighton Marathon on 17 April 2016 in aid of Parkinson's UK, as my mother-in-law is a sufferer. Please sponsor me at http://uk.virginmoneygiving.com/IainWear

Member since: 30.08.2002
Reviews: 631
Members who trust: 503

Share this review on

Despite still being able to recall the "soft on the outside, crunchy on the inside" tagline from the Harry Enfield advertising campaign from the height of its, and his, popularity many years ago, I hadn't thought of the Dime bar for ages. It had been so long since I considered the sweet that the 2005 rebranding that turned it from a Dime bar into the Daim bar completely passed me by.

The reason for this became immediately obvious when I tried to track one down. The Daim bar appeared to be one of the more elusive of chocolate bars. It was not to be found in any of the usual places I would buy chocolate; my local paper shop, the 24 hour petrol station and the local Co-Op did not have it amongst their chocolate selections. Finally, quite by chance, I was able to track one down for 45 pence in a local independent convenience store and the Sainsbury's website does not return a successful research for either "Dime" or "Daim". The only reason I can think for this is that it's made by Kraft Foods, rather than one of the major chocolate manufacturers and they may not have the same distribution agreements as Nestle and Cadbury.

The Daim bar is a lot smaller than most common chocolate bars. Not in terms of the top of the bar which at around an inch wide by three inches long is pretty standard. What makes the Daim bar that little bit different is that it's relatively flat at only around half a centimetre high, quite a lot smaller than the more popular bars and at only 28 grams in weight, lighter than most as well. In most other aspects, however, the Daim bar resembles any other chocolate bar, with the ridged chocolate top and lacking only the manufacturers name on the bottom.

At room temperature, the only thing you can smell is the chocolate coating, without a hint of what lies beneath. In fact, even when the bar has been broken, the sugar centre doesn't give off any odour.

Sadly, the taste of the bar doesn't match that lovely chocolate smell. When you bits into the top, the bar meets the old "soft on the outside, crunchy on the inside" slogan wonderfully. The chocolate gives way under your teeth without so much of a crack, but the sugary inside takes some getting through. Sadly, it quickly becomes apparent why the old advertising campaigns concentrated on how the Daim bar feels in your mouth, as the taste doesn't match up.

(b) Review with helpfulness equal to 4 (ID: 11495)

Figure 6.6: Review excerpts with different helpfulness for the reader and product.

SVR yields 0.5238441; LR, 0.5011651; and RLFM, 0.508706. Thus, LR is the best for regression, followed by RLFM and then SVR. Table 6.9 shows the true values at the top 5 by ordering according to the prediction of each technique.

Figure 6.6 contrasts a review with helpfulness equal to 5 with one equal to 4. We note that the former is more structured, divided into sections. It is also typically longer, with $2,267$ tokens against 846 of the review with helpfulness 4. We also note a quite ironic style of the review with helpfulness 5 that the reader probably enjoys. On the other hand, the review 4 contains much personal information regarding the difficult to find the product, and generally irrelevant information, like the size of the bar, which may be the cause for the reader's evaluation. This review receives a lower grade even though the author has a better reputation, with more written reviews and more trustors.

6.5 Inspecting the SVR Model

SVR is the best solution in our comparison of existing techniques, albeit being a regression model. By focusing only on higher errors, it is more robust than LR in finding a representative pattern and less susceptible to overfitting and to noise in training data. A perfect LR model (i.e., with no error) in training set implies in a perfect ranking, but such goal is hard to accomplish. A perfect SVR model, on the other hand, does not distinguish errors up to ε, thus not necessarily inducing a perfect ranking. For example, considering $\varepsilon = 0.1$, if the true values 3.50 and 3.60 are predicted as 3.55 and 3.52, respectively, no error occurs in SVR. However, their order is predicted wrongly.

Nonetheless, when regression responses are integer values, a *perfect* SVR model has a *perfect* ranking $\forall\ \varepsilon < 0.5$. Indeed, for such ε, each possible integer response has a margin of predicted values that does not overlap with another. For example, supposing $\varepsilon = 0.4$ and a *perfect* SVR model, a true value of 5 is predicted in range $[4.6, 5.4]$, of 4 in range $[3.6, 4.4]$, and so on. Note that, for any $\varepsilon < 0.5$, no overlapping of predictions for different integer values occurs in a perfect SVR model. Figure 6.7 shows such behavior.

A perfect LR is much more challenging to model all responses perfectly with a single generalization through features. By introducing an ε relaxation, SVR becomes much more robust and, in the case of *integer responses*, very close to a ranking purpose.

6.5.1 Investigating Features Impact

We further investigate how variations in the SVR model impact the result. Specifically, we investigate: (i) eliminating one group of features of the same source entity or relationship at a time; (ii) using feature selection criteria; and (iii) personalized against non-personalized feature sets.

Table 6.10 shows the performance comparison for a single set of features belonging to certain entity or interaction removed. Review features are the ones that impact the most. Indeed, content explains great part of helpfulness, even when it consists of simple textual statistics. Voter features are the second ones that reduces the average performance the most, but such difference is not significant. In fact, removing voter features is better in some cases and worse in others, probably working well only for certain users. Similarity also does not significantly worsen the results. This is an evidence that the similarity used, related to ratings given, does not represent well users' common interests. The remaining author and connection features impact the result in a similar way and significantly. This elucidates the importance of author's reputation and the interaction on the trust network, the latter a personalized trait, albeit the metric decreases only a little in average.

We also compare SVR with its non-personalized version, by considering only review and author features and using reviews' mean vote as responses, as Table 6.10 shows. The performance decreases significantly, then personalization explain the helpfulness, albeit the obtained gain is not high. For instance, the second best predictor, personalized LR, is significantly worse than non-personalized SVR (the p-value is

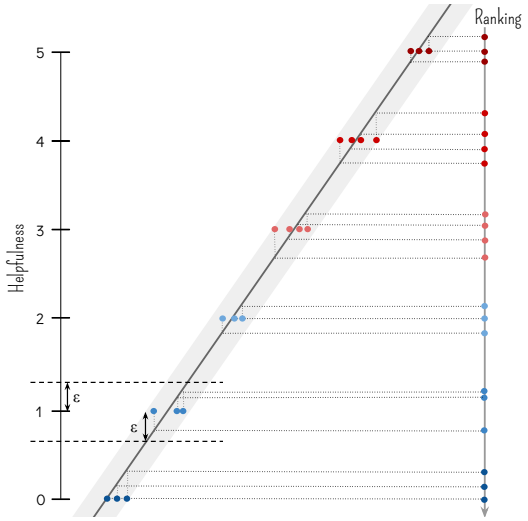


Figure 6.7: Ranking obtained from a perfect SVR model with $\varepsilon < 0.5$. For integer responses, absence of error in such model implies in absence of error in the ranking.

Table 6.10: SVR performance for one set of features removed at a time. A paired t-test is performed between a new version and the original, with all features, and p-values are shown. A red down arrow represents significant worsening, and a green equality sign, a statistical tie.

Feature Set	nDCG@5		p-value
All	0.9362 ± 0.0132		–
All \ Review	$\mathbf{0.9078 \pm 0.0256}$	▼	0.0040
All \ Author	$\mathbf{0.9356 \pm 0.0130}$	▼	0.0253
All \ Reader	0.9117 ± 0.0433	=	0.1300
All \ Similarity	0.9364 ± 0.0130	=	0.0973
All \ Connection	$\mathbf{0.9356 \pm 0.0130}$	▼	0.0056
All \ Personalized	$\mathbf{0.9319 \pm 0.0053}$	▼	0.0271

Table 6.11: SVR performance for a set of features defined by a feature selection criterion, InfoGain and ρ_s. A paired t-test is performed between SVR with all features and with the corresponding subset, then p-values are shown. A blue up arrow is presented for significant improvement, and a green equal sign for a statistical tie.

Feature Set	nDCG@5		p-value
All	0.9362 ± 0.0132		–
InfoGain Top 10	$\mathbf{0.9373 \pm 0.0130}$	▲	0.0009
ρ_s Top 10	$\mathbf{0.9374 \pm 0.0133}$	▲	0.0002
InfoGain Top 5	0.9174 ± 0.0304	=	0.0781
ρ_s Top 5	$\mathbf{0.9374 \pm 0.0131}$	▲	0.0002
ρ_s Top 4	0.9293 ± 0.0239	=	0.1655
ρ_s Top 3	0.9355 ± 0.0159	=	0.5554
ρ_s Top 2	0.8762 ± 0.1147	=	0.1829
ρ_s Top 1	0.9038 ± 0.1012	=	0.3793
ρ_s Top 10 \ 2nd	$\mathbf{0.9376 \pm 0.0133}$	▲	0.0004
ρ_s Top 5 \ 2nd	$\mathbf{0.9374 \pm 0.0131}$	▲	0.0007

0.0011). Thus, the personalization hypothesis is not strong for the considered features.

Table 6.11 shows that, surprisingly, removing features may improve the performance significantly, as represented by the performance using their top 10 features using InfoGain and ρ_s. We then proceed with increasing reduction, evaluating their top 5. Now, InfoGain has a lower performance in average, although not significantly, while ρ_s improves significantly. Evaluating the performance for top 1 to top 4 of ρ_s selection reveals that all of them are statistically tied with the original version. Top 1 presents a great reduction in average, but it is not statistically different when performing a paired

t-test. This indicates that, in some contexts, only the size of the review is enough for predicting their quality. Indeed, the higher the number of tokens, the higher the probability of the user finding the information he seeks.

However, the top 2 was worse than the top 1, probably due to multicollinearity of the number of tokens and sentences [Jain, 1991]. Thus, we investigate the two best subsets of features, ρ_s top 5 and top 10, without the second best feature. Such removal is represented by $\setminus\ 2^{nd}$, resulting in ρ_s *Top 10* $\setminus\ 2^{nd}$ as the overall best subset.

It is important to note that ρ_s Top 5 features, which is better than including all, contains only one personalized feature: the voter average helpfulness. Thus, from all personalized features, only user bias impact significantly. However, as personalization was suggested to be a good approach under other analysis, as presented in Section 4.5, new features are yet to be evaluated to fully understand its role. Moreover, such hypothesis may be important for a class of users, for instance, active ones.

6.5.2 Contrasting SVR Versions

Given the limited contribution of personalization using existing features and the possibility of leveraging ranking goal, we investigate extensions upon SVR attempting to obtain an improved version for review recommendation task. Table 6.13 shows the performance of each version and p-value of paired t-test evaluating their difference with original SVR. Such versions are:

- *Sel-SVR:* Version using the best subset of features found in Section 6.5.1, which corresponds to ρ_s Top 10 $\setminus\ 2^{\text{nd}}$.
- *Sel+New-SVR:* Version using the best subset of features along with new proposed interaction features described in Table 6.12.
- *Sel+New-CoRaSVR:* Stochastic gradient descent version of SVR with jointly ranking optimization and adaptations for focusing on the top of the ranking.

As the considered personalized features do not provide a strong benefit for this problem, we have investigated how the addition of simple interaction features, which are all personalized, impacts the result. By inserting 9 new features, we observed a significant improvement in SVR performance (Table 6.13). This indicates the great potential in exploring new features.

Furthermore, a joint optimization is evaluated by combining SVR loss, which is ε-insensitive, and cross-entropy loss for the relative position probability of pairs [Balakrishnan and Chopra, 2012], composing a Combined Ranking and SVR (CoRaSVR).

Table 6.12: New proposed features derived from voter-author interaction.

Feature	Description
c_vot_trust	Boolean variable which is 1 if the voter trusts the author and 0 otherwise.
c_aut_trust	Boolean variable which is 1 if the author trusts the voter and 0 otherwise.
c_close_from_vot	Inverse of shortest path length from voter to author; 0 if no path exists.
c_close_from_aut	Inverse of shortest path length from author to voter; 0 if no path exists.
s_diff_trustors	Number of author's trustors minus number of voter's.
s_diff_num_reviews	Number of author's written reviews minus voter's.
s_diff_pagerank	Author's PageRank score minus voter's.
s_diff_close	Author's closeness centrality minus voter's.
s_diff_eigen	Author's eigenvector centrality score minus voter's.

Table 6.13: Variations upon SVR. A blue up arrow shows significant improvement.

Technique	nDCG@5		p-value
SVR	0.9362 ± 0.0132		–
Sel-SVR	$\mathbf{0.9376 \pm 0.0133}$	▲	0.0004
Sel+New-SVR	$\mathbf{0.9381 \pm 0.0132}$	▲	0.0001
Sel+New-CoRaSVR	$\mathbf{0.9386 \pm 0.0127}$	▲	0.0007

On the one hand, SVR is improved by considering relative positioning through cross-entropy loss. On the other hand, pairwise learning to rank sparseness, due to many tied pairs, is overcome by regression's support.

By analyzing these results, it is remarkable how we observe statistical significance for small absolute improvements. In fact, a small change in average nDCG@5 may imply an improvement in several rankings. Also, this is not uncommon for a ranking task, as it was also reported in the Yahoo Learning to Rank Challenge [Chapelle and Chang, 2011]: the difference among the baseline, GBRT, and the winner was 0.00660 and among the first and second winners was 0.00075, regarding ERR.

Overall, selection of features, addition of new meaningful interaction features and jointly optimizing a ranking loss improve over plain SVR. Although such statistical improvements are small, they serve to show the unexplored potential in defining a better set of features and a more advantageous loss. We detail the significant best version, CoRaSVR, in the next Section.

6.5.3 Detailing CoRaSVR

CoRaSVR, a new method proposed as en extension upon SVR, performs a jointly ranking optimization along with SVR. A ranking loss is applied for the probability prediction of one review being ranked above another. Let r_i and r_j be a pair of

reviews and $r_i \succ r_j$ denote the event when r_i is ranked above r_j. The probability is predicted by applying the logistic function on the difference of feature vectors $\mathbf{x}_{r_i}$ and $\mathbf{x}_{r_j}$. Specifically, given ranking $\mathcal{O}_{u,p}$ related to a reader u and a product p, the probability of $r_i \succ r_j : r_i, r_j \in \mathcal{R}$ in such ranking is computed as [Balakrishnan and Chopra, 2012]:

$$g(p_{r_i} - p_{r_j}) = g(\mathbf{w}^T(\mathbf{x}_{r_i} - \mathbf{x}_{r_j})) = \frac{1}{1 + e^{-\mathbf{w}^T(\mathbf{x}_{r_i} - \mathbf{x}_{r_j})}} \tag{6.5}$$

The true values of such probabilities are either binary, indicating if one is ranked above the other, or obtained by scaling the difference of relevance scores of both reviews in $[0, 1]$ [Sculley, 2010]. Then, we apply the cross-entropy loss to such estimation of probabilities [Balakrishnan and Chopra, 2012], deriving a combination of a pairwise and a pointwise loss, as follows:

$$\begin{aligned} E_{CoRaSVR} = & \sum_{r_i, r_j \in \mathcal{O}_{u,p}} \gamma \left(y_{r_i} max\{|p_{r_i} - y_{r_i}| - \varepsilon, 0\} + y_{r_j} max\{|p_{r_j} - y_{r_j}| - \varepsilon, 0\} \right) \\ & + (1 - \gamma)\ y_{r_i, r_j} \left(log(1 + e^{p_{r_i} - p_{r_j}}) - y_{r_i, r_j}(p_{r_i} - p_{r_j}) \right) + \beta\ \|\mathbf{w}\|^2, \end{aligned} \tag{6.6}$$

where y_{r_k} is the true helpfulness (response) and $p_{r_k} = \mathbf{w}^T \mathbf{x}_{r_k}$ is the predicted helpfulness of review r_k in ranking $\mathcal{O}_{u,p}$; β is the regularization factor; γ regulates the trade-off between regression and ranking error [Sculley, 2010]; ε is SVR error tolerance; and y_{r_i, r_j} is the truth for probabilities. In short, the loss is defined over pairs and considers a pointwise error for both members and a pairwise loss for the pair. By simultaneously satisfying REG and LTR restrictions, CoRaSVR most generic purpose is for regression, then belonging in REG class. Sculley [2010] already addressed a combined ranking and regression approach in a generic format . We remark specifics of CoRaSVR's loss:

- *Scaling of probabilities' responses:* Probabilities' responses are computed as $y_{r_i, r_j} = \frac{2^{y_{r_i}} - 2^{y_{r_j}} + 31}{62}$, which scales individual responses exponentially and the difference, in range $[0, 1]$. Such individual exponential scaling helps distinguishing reviews for a ranking task, as already explored for pointwise and pairwise approaches [Balakrishnan and Chopra, 2012]. However, we did not include exponential scaling in regression loss as it decreases the performance in validation set. Besides, we scale the difference of responses in range $[0, 1]$ due to providing better results in validation set than a binary set.

- *Weighting error by true values:* We have further adapted the loss by weighting each error according to the true value. This way, we prioritize prediction of re-

views with higher helpfulness values and focus on the top, which contains the ones of main interest. Similarly, pairs with a higher gap in responses are emphasized in cross-entropy loss.

- *Ignoring zero helpfulness:* Finally, we have disregarded votes equal to zero for both regression and ranking error due to better performance in validation set. Such evaluations do not seem to follow the same pattern as the others, as noted in Section 4.6 and may mislead model fitting.

- *Considering only pairs with a response equal or greater to the second maximum in the ranking:* As described by Balakrishnan and Chopra [2012], accounting only for responses equal to the second maximum or greater in a ranking helps prioritizing prediction at the top. When a pair is composed by two responses below the second maximum, typically none is at the top of the ranking. Therefore, only pairs with at least one review among the highest helpfulness, specifically at least equal to the second maximum in the corresponding ranking, are considered in the optimization.

Furthermore, we apply stochastic gradient descent to learn the model, by using the following derivative for the error on a single instance r_i, r_j:

$$\begin{aligned}
\partial_{r_k} &= \begin{cases} \gamma\ y_{r_k}\mathbf{x}_{r_k} & p_{r_k} - y_{r_k} > \varepsilon \\ -\gamma\ y_{r_k}\mathbf{x}_{r_k} & p_{r_k} - y_{r_k} < -\varepsilon \\ 0 & |p_{r_k} - y_{r_k}| \leq \varepsilon, \end{cases} \\
\partial_{r_i,r_j} &= (1-\gamma)\ y_{r_i,r_j}\ (g(p_{r_i} - p_{r_j}) - y_{r_i,r_j})(\mathbf{x}_{r_i} - \mathbf{x}_{r_j}), \\
\frac{\partial E^{r_i,r_j}_{CoRaSVR}}{\partial \mathbf{w}} &= \partial_{r_i} + \partial_{r_j} + \partial_{r_i,r_j} + \beta\mathbf{w},
\end{aligned} \tag{6.7}$$

where the derivation is presented in Appendix B.4. The derivative at $|p_{r_k} - y_{r_k}| = \varepsilon$ is undefined, but we that it is zero. Then, the stochastic updates are performed in the same way as described for BETF in Section 5.6.

The optimal parameter values are computed in validation and are presented in Table 6.14. A higher weight is given to regression loss by parameter γ, thus showing its contribution in overcoming pairwise sparseness. Note that the number of interactions refers to each single instance update (opposed to BETF and MF, where one iteration goes through all data), as we sample the existing pairs due to their high cardinality. We do not perform early stopping under a convergence tolerance because the cost of

Table 6.14: List of parameters with a description, evaluated range or set, and the best value for ranking (nDCG@5*) for CoRaSVR.

Symbol	Description	Range/Set	nDCG@5*
I	# Iterations	$(10^5, 10^7)$	10^7
α	Initial learning rate	$(10^{-4}, 1)$	10^{-2}
β	Regularization factor	$(10^{-3}, 10)$	10^{-2}
γ	Ranking-regression trade-off	$(0, 1)$	0.8
ε	Regression tolerance on error	$(10^{-2}, 1)$	10^{-1}
p	Exponent of learning rate decay	$(0.15, 0.35)$	0.2
bias	Addition of bias	$\{true, false\}$	$false$

computing the current loss over all pairs is too high, since it is very costly to enumerate them all (they reach 2,241,239 pairs in one split).

6.6 Concluding Remarks

By performing a unified experimental evaluation with techniques from five different paradigms, we assess their performance and derive insights of key properties relevant for review recommendation problem with a ranking purpose. We remark that:

(*i*) Regression is significantly different than ranking, thus the latter should be prioritized for being aligned with the final purpose;

(*ii*) Chronological is not the same as random split between training and test set and the former should be the choice as it is a practical imposition;

(*iii*) Simpler models generalizes better the training set for test prediction, represented by linear predictors, a small subset of features, high regularization, few iterations and others;

(*iv*) Traditional good approaches in recommendation, such as collaborative filtering and addition of biases, are not good assumptions for a situation plenty of review cold-starts and with a ranking goal;

(*v*) SVR performs well for ranking under integer relevance scores by introducing ε relaxation;

(*vi*) SVR is significantly improved by selecting a subset of features through Spearman correlation (ρ_s), by addition of new interaction features and by jointly incorporating a ranking loss.

Chapter 7

Conclusion

Reviews are a source of information being increasingly used and produced by consumers. There is a large volume of reviews available, which is a challenge for readers seeking useful information. A common solution is to rank the reviews according to their global helpfulness, neglecting that users have different preferences, and thus perceive the helpfulness of a given review differently. Hence, recent works model the task as a *personalized* problem. In such context, we define review recommendation as a *top-n task*, borrowing concepts from recommender systems and following a recent trend of ranking-oriented modeling.

We used a dataset from Ciao UK to perform our experiments, which has the great advantage of containing personalized votes. Also, such dataset has an underlying trust network as an important source of information for recommendation. We observed a great discrepancy of types of users and reviews in terms of related information available. This situation motivates recommendation as a way of finding unpopular reviews in the long tail, but also represents a challenge to precisely recommend for very different users. We also noted that authors tend to have a consistent helpfulness across reviews; however, such stability increases when considering reviews' votes, and even more for votes related to an author-reader pair, a personalized criterion. This motivates seeking for personalized solutions as the next frontier for improvement.

A unified experimental design was performed to compare a rich and heterogeneous set of personalized solutions for review recommendation, from five paradigms, that were not previously compared against each other. Our experiments considered the best parametric configuration and RMSE, nDCG@p and ERR metrics. We concluded that personalization explains a portion of helpfulness value, being personalized SVR significantly the best approach, with 95% of confidence. Moreover, the simplicity of the model is advantageous, possibly suffering less from overfitting and producing a more

accurate generic model. Including observed features into the model is extremely important due to the sparseness of the problem, which has many cold-starts. Despite aiming at a ranking goal, pairwise and listwise learning to rank predictors suffer considerably from sparseness due to ties. Our evaluation indicated that a regression metric (e.g., RMSE) leads to significantly different conclusions compared to a ranking one (e.g., nDCG@p). However, the latter is more suitable as it represents well the final purpose of presenting a sorted list to a user. Finally, including new interaction features and jointly optimizing a ranking goal significantly improve SVR performance.

7.1 Contributions

The main contributions of our work are summarized as follows.

- **Extensive comparison of techniques from multiple paradigms for review recommendation using in a real dataset.** We compared a wealthy set of techniques, from different paradigms, for review recommendation in a real dataset from Ciao UK. Before diving into techniques and experiments, we have investigated the dataset and helpfulness voting dynamics, demonstrating, for instance, that personalization seems to be the right hypothesis for helpfulness prediction. Regarding the techniques evaluated, there is no previous work, to the best of our knowledge, that considers experiments with personalized LR, SVR, GBRT, RSVM, LMART and RLFM for such task. All considered solutions were compared in a unified and statistical experimental design, with parameter tuning. Specially, we evaluated specialized approaches for review recommendation, CAP and BETF, not yet compared. We studied such approaches, providing precise algorithms and formulas, which are not available in the original works. Specially, we identified that CAP is a extension upon RLFM and derived their formulas based on such finding for applying a MCEM algorithm. We have also implemented such approaches, with publicly available code. Overall, our comparison provided several insights about an effective ranking-oriented review recommender system. Specifically, good performance typically share the following aspects: (i) including observed features; (ii) building simpler models; and (iii) personalized prediction.

- **Assessment of review recommendation solutions under a ranking perspective, besides a regression one.** A recent trend consists in considering the recommendation problem as a ranking task. Indeed, the great majority of

solutions predict a real value for rating, but only to rank the results and present the top ones to the user. By considering the problem as a top-n task directly, the over restriction of regression is overcome and results tend to improve. Although learning to rank techniques were not the best performers in our comparison, we tuned and found a most suitable solution by using a ranking measure. This perspective yields the best final result presented to the user.

- **Exploitation of different versions of the best method – SVR.** We explored variations of SVR by applying feature selection, incorporating new features and simultaneously optimizing a ranking loss. We showed that only a set of 5 out of 43 features is enough for providing an effective predictor. Content features, derived from text, are the ones that impact the most. Furthermore, we defined new features to investigate the potential for improvement in personalization, obtaining significantly better results. We also defined a combined ranking and SVR method – CoRaSVR, which has an optimization objective composed of a ranking and a regression loss, yielding significant improvement.

7.2 Future Work

Future work includes exploring new observed features, specially personalized ones and related to review-user interaction, and taking more advantage of the ranking purpose. Additionally, considering the few limitations of Ciao UK dataset, experiment with a new one would be very advantageous. The model could also take into account the aging of reviews and other timely patterns that impact the helpfulness. Also, the model could specialize in different types of users. Finally, other metrics may be evaluated to better represent user's interest. We detail each future direction as follows.

- **Assessment using other datasets.** We only consider one dataset, as no other platform, to the best of our knowledge, has publicly available votes discriminated by each user. If another dataset becomes available with such information, evaluating under this new scenario will help understand the extent of the patterns observed in this work. Another possibility consists in deriving helpfulness opinion implicitly.

- **Further enhancements upon SVR.** We investigate new predictor versions built upon SVR by including new features and a ranking loss. Yet, many other features may be explored. For instance, more semantic features may be extracted from text (e.g.: topics) and be used for relating the reader with the author or

review in deeper aspects such as way of thinking and opinions. Besides, several other ranking losses may be composed, for instance, a listwise one.

- **Time dynamics included in the model.** An old review is likely not going to be as relevant as a new one; but that may depend on the product and category. Also, there may be underlying timing effects in users' behavior. Exploring such time aware model may provide effectiveness improvement.

- **Applying specific models for different types of users.** Personalization was not a strong hypothesis in our evaluation, as non-personalized SVR was statistically worse than SVR, but better than the second best. Therefore, an investigation of how personalization is important for different groups of users is the next step. Indeed, we show that users differ deeply in participatory behavior. For instance, the hypothesis that personalization works well only for active users may be verified.

- **Evaluation of other metrics.** When providing a set with most helpfulness reviews to a user, other characteristics may be important, such as diversity. The most distinct the features and opinions described in reviews, the wider knowledge a user would obtain; thus, allowing to perform a better decision. In general terms, other means of evaluating the results may be investigated to better account for user needs.

Bibliography

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749.

Agarwal, D. and Chen, B.-C. (2009). Regression-based latent factor models. In *Procs. of KDD*, Paris, France.

Amatriain, X., Pujol, J. M., and Oliver, N. (2009). I like it... i like it not: Evaluating user ratings noise in recommender systems. In *Procs. of UMAP*, Trento, Italy.

Anderson, M. (2014). Local consumer review survey 2014. `https://www.brightlocal.com/2014/07/01/local-consumer-review-survey-2014`.

Anderson, M. L. and Magruder, J. R. (2012). Learning from the crowd: Regression discontinuity estimates of the effects of an online review database. *Economic Journal*, 122(563):957–989.

Bakhshi, S., Kanuparthy, P., and Gilbert, E. (2014). Demographics, weather and online reviews: A study of restaurant recommendations. In *Procs. of WWW*, Seoul, Korea.

Balakrishnan, S. and Chopra, S. (2012). Collaborative ranking. In *Procs. of WSDM*, Seattle, WA, USA.

Balog, K. and Ramampiaro, H. (2013). Cumulative citation recommendation: Classification vs. ranking. In *Procs. of SIGIR*, Dublin, Ireland.

Bidart, R., Pereira, A., Almeida, J., and Lacerda, A. (2014). Where should i go? city recommendation based on user communities. In *Procs. of LA-WEB*, Ouro Preto, Minas Gerais, Brazil.

Bjørkelund, E., Burnett, T. H., and Nørvåg, K. (2012). A study of opinion mining and visualization of hotel reviews. In *Procs. of iiWAS*, Bali, Indonesia.

Booth, J. G. and Hobert, J. P. (1999). Maximizing generalized linear mixed model likelihoods with an automated monte carlo em algorithm. *JRRS: B*, 61(1):265–285.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *Procs. of ICML*, Bonn, Germany.

Burges, C. J. (2010). From RankNet to LambdaRank to LambdaMART: An overview. Technical report MSR-TR-2010-82, Microsoft Research.

Canuto et al., S. D. (2013). A comparative study of learning-to-rank techniques for tag recommendation. *JIDM*, 4(3):453–468.

Chapelle, O. and Chang, Y. (2011). Yahoo! learning to rank challenge overview. In *Procs. of the Yahoo! Learning to Rank Challenge - ICML*, Bellevue, WA, USA.

Chapelle, O. and Keerthi, S. S. (2010). Efficient algorithms for ranking with SVMs. *IR*, 13(3):201–215.

Chapelle, O., Metlzer, D., Zhang, Y., and Grinspan, P. (2009). Expected reciprocal rank for graded relevance. In *Procs. of CIKM*, Hong Kong, China.

Christakopoulou, K. and Banerjee, A. (2015). Collaborative ranking with a push at the top. In *Procs. of WWW*, Florence, Italy.

Christodoulides, G. and Jevons, C. (2011). The voice of the consumer speaks forcefully in brand identity: User-generated content forces smart marketers to listen. *Journal of Advertising Research*, 51(1):101–112.

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *Procs. of RecSys*, Barcelona, Spain.

D'Addio, R. and Garcia Manzato, M. (2014). A collaborative filtering approach based on user's reviews. In *Intelligent Systems (BRACIS), 2014 Brazilian Conference on*, pages 204–209.

Dahlgren, S., Johnson, A., and Liljenberg, C. (2015). Online reviews – what motivates you? a qualitative study of customers' motivation to write online reviews. Bachelor Thesis – Linnaeus University, Sweden.

Danescu-Niculescu-Mizil, C., Kossinets, G., Kleinberg, J., and Lee, L. (2009). How opinions are received by online communities: A case study on amazon.com helpfulness votes. In *Procs. of WWW*, Madrid, Spain.

Dellarocas, C. and Narayan, R. (2006). What motivates consumers to review a product online? a study of the product-specific antecedents of online movie reviews. In *Procs. of the WISE*, Evanston, IL, USA.

Deshpande, M. and Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177.

Do, C. B. and Batzoglou, S. (2008). What is the expectation maximization algorithm? *Nature biotechnology*, 26(8):897–900.

Duan, H., Zhai, C., Cheng, J., and Gattani, A. (2013). Supporting keyword search in product database: A probabilistic approach. *PVLDB*, 6(14):1786–1797.

Easley, D. and Kleinberg, J. (2010). *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press.

Esuli, A. and Sebastiani, F. (2006). SentiWordNet: A publicly available lexical resource for opinion mining. In *Procs. of LREC*, Genoa, Italy.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):pp. 1189–1232.

Fu, X. and Asorey, H. (2015). Data-driven product innovation. In *Procs. of KDD*, Sydney, NSW, Australia.

Ganesan, K. and Zhai, C. (2012). Opinion-based entity ranking. *Inf. Retr.*, 15(2):116–150.

Hennig-Thurau, T., Gwinner, K. P., Walsh, G., and Gremler, D. D. (2004). Electronic word-of-mouth via consumer-opinion platforms: What motivates consumers to articulate themselves on the internet? *Journal of Interactive Marketing*, 18(1):38–52. ISSN 1520-6653.

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Procs. of KDD*, Seattle, WA, USA.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *Procs. of ICDM*, Pisa, Italy.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688.

Jain, R. (1991). *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.* Wiley.

Joachims, T. (2006). Training linear SVMs in linear time. In *Procs. of KDD*, Philadelphia, PA, USA.

Jurafsky, D., Chahuneau, V., Routledge, B., and Smith, N. (2014). Narrative framing of consumer sentiment in online restaurant reviews. *First Monday*, 19(4).

Kim et al, S.-M. (2006). Automatically assessing review helpfulness. In *Procs. of EMNLP*, Sydney, Australia.

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Procs. of KDD*, Las Vegas, NV, USA.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Commun. ACM*, 42(8):30–37.

Lee, S. and Choeh, J. Y. (2014). Predicting the helpfulness of online reviews using multilayer perceptron neural networks. *Expert Syst. Appl.*, 41(6):3041–3046.

Li, H. (2011). A short introduction to learning to rank. *IEICE Transactions*, 94-D(10):1854–1862.

Liben-Nowell, D. and Kleinberg, J. (2007). The link-prediction problem for social networks. *JASIST*, 58(7):1019–1031.

Liu, Y., Huang, X., An, A., and Yu, X. (2008). Modeling and predicting the helpfulness of online reviews. In *Procs. of ICDM*, Miami, FL, USA.

Lu, W., Chen, S., Li, K., and Lakshmanan, L. V. S. (2014). Show me the money: Dynamic recommendations for revenue maximization. *PVLDB*, 7(14):1785–1796.

Lu et al, Y. (2010). Exploiting social context for review quality prediction. In *Procs. of WWW*, Raleigh, NC, USA.

Lucchese et al, C. (2015). Quickscorer: A fast algorithm to rank documents with additive ensembles of regression trees. In *Procs. of SIGIR*, Santiago, Chile.

Machanavajjhala, A., Korolova, A., and Sarma, A. D. (2011). Personalized social recommendations: Accurate or private. *PVLDB*, 4(7):440–450.

Martin, L. and Pu, P. (2014). Prediction of helpful reviews using emotions extraction. In *Procs. of AAAI*, Québec City, Canada.

McAuley, J. and Leskovec, J. (2013). Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Procs. of RecSys*, Hong Kong, China.

Michel, J.-B., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., Team, T. G. B., Pickett, J. P., Holberg, D., Clancy, D., Norvig, P., Orwant, J., Pinker, S., Nowak, M. A., and Aiden, E. L. (2011). Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

Miller, G. A. (1995). WordNet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Moghaddam, S., Jamali, M., and Ester, M. (2012). ETF: Extended Tensor Factorization model for personalizing prediction of review helpfulness. In *Procs. of the WSDM*, Seattle, WA, USA.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Ng, A. Y. (2004). Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Procs. of ICML*, Banff, Alberta, Canada.

O'Mahony, M. P. and Smyth, B. (2009). Learning to recommend helpful hotel reviews. In *Procs. of RecSys*, New York, NY, USA.

Parra, D., Karatzoglou, A., Yavuz, I., and Amatriain, X. (2011). Implicit feedback recommendation via implicit-to-explicit ordinal logistic regression mapping. In *Procs. of CARS*.

Pedregosa, F. e. a. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Powers, D. M. W. (2007). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Technical report SIE-07-001, School of Informatics and Engineering, Flinders University, Adelaide, Australia.

Rensink, J. (2013). What motivates people to write online reviews and which role does personality play? A study providing insights in the influence of seven motivations on the involvement to write positive and negative online reviews and how five personality traits play a role. Master Thesis – University of Twente, Netherlands.

Ricci et al, F. (2010). *Recommender Systems Handbook*. Springer-Verlag, 1st edition.

Sculley, D. (2010). Combined regression and ranking. In *Procs. of KDD*, Washington, DC, USA.

Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., and Hanjalic, A. (2013). xCLiMF: Optimizing expected reciprocal rank for data with multiple levels of relevance. In *Procs. of RecSys*, Hong Kong, China.

Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., and Hanjalic, A. (2012). Climf: Learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 139–146, New York, NY, USA. ACM.

Shi, Y., Larson, M., and Hanjalic, A. (2014). Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, 47(1):3:1–3:45.

Simonson, I. and Rosen, E. (2014). What Marketers Misunderstand About Online Reviews. *Harvard Business Review*.

Sipos, R., Ghosh, A., and Joachims, T. (2014). Was this review helpful to you?: It depends! context and voting patterns in online content. In *Procs. of WWW*, Seoul, Korea.

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

Tang et al, J. (2013). Context-aware review helpfulness rating prediction. In *Procs. of RecSys*, Hong Kong, China.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

Vasconcelos, M., Almeida, J. M., and Gonçalves, M. A. (2015). Predicting the popularity of micro-reviews: A foursquare case study. *Information Sciences*, 325:355–374.

Wang, J., Zhao, J., Guo, S., North, C., and Ramakrishnan, N. (2014). Recloud: Semantics-based word cloud visualization of user reviews. In *Procs. of GI*, Montreal, Québec, Canada.

Weimer, M., Karatzoglou, A., Le, Q. V., and Smola, A. J. (2007). COFI RANK – maximum margin matrix factorization for collaborative ranking. In *Procs. of NIPS*, Vancouver, British Columbia, Canada.

Xu, J., Zheng, X., and Ding, W. (2012). Personalized recommendation based on reviews and ratings alleviating the sparsity problem of collaborative filtering. In *Procs. of ICEBE*.

Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Procs. of ICML*, Nashville, TN, USA.

Yin et al, H. (2012). Challenging the long tail recommendation. *PVLDB*, 5(9):896–907.

Ypma, T. J. (1995). Historical development of the Newton-Raphson method. *SIAM Rev.*, 37(4):531–551.

Yue, Y., Finley, T., Radlinski, F., and Joachims, T. (2007). A support vector method for optimizing average precision. In *Procs. of SIGIR*, Amsterdam, The Netherlands.

Zhan, J., Loh, H. T., and Liu, Y. (2009). Gather customer concerns from online product reviews - a text summarization approach. *Expert Syst. Appl.*, 36(2):2107–2115.

Zhang, R. and Tran, T. (2011). An information gain-based approach for recommending useful product reviews. *Knowl. Inf. Syst.*, 26(3):419–434.

Appendix A

Assumptions about Features

For *r_comp_ratio* we consider comparative adverbs and adjectives; *r_uni_ratio* is considered as number of unique words divided by the total number of word occurrences; *r_avg_sent* uses sentences' length measured in tokens; *r_cap_ratio* assumes the capitalization of the first letter of the sentence; statistics of difference of author and voter ratings (*s_diff_avg_rat*, *s_diff_max_rat*, *s_diff_min_rat*) are calculated as the author's value minus the voter's one; adamic-adar features (*c_adam_trustees*, *c_adam_trustors*) considers neighbors in the undirected version of the trust graph; *c_katz* is calculated considering paths from voter to author in the directed graph and using $\beta = 0.005$, a standard value [Liben-Nowell and Kleinberg, 2007]; feature *r_kl* is evaluated removing stopwords and considering all reviews of the product in training set and additionally the review being modeled, when in test set; *r_pos_ratio* and *r_neg_ratio* are the ratio of words whose positive score in first synset of SentiWordNet [Esuli and Sebastiani, 2006] is higher than negative and negative is higher than positive, respectively, and positive and negative scores are exchanged if there is a preceding negative word in the same sentence; foreign words were identified by the absence in WordNet [Miller, 1995].

Appendix B

Mathematical Proofs

B.1 Complete Log-Likelihood for CAP

In this subsection, we derive the complete log-likelihood for the CAP system.

For this calculation, we use the formulas for normal distribution ($\mathcal{N}$) and multivariate normal distribution ($\mathcal{MVN}$) defined below.

$$P_{\mathcal{N}}(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{B.1}$$

$$P_{\mathcal{MVN}}(\mathbf{x}, \mu, \mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k det\mathbf{\Sigma}}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)} \tag{B.2}$$

Using these probability densities, the log-likelihood is given by the joint probability of observing the data ($y_{h_{ij}}$) and the latent variables Ω. This is equivalent to the factorization over the probabilistic graphical model, presented in Figure C.2. Thus,

$$\begin{aligned} L(\Omega, \Theta) = & \prod_{u_i, r_j \in \mathcal{V}^0 \times \mathcal{R}^0} P(y_{h_{ij}}|\Omega) \cdot \prod_{u_i \in \mathcal{V}^0} P(\alpha_i) \cdot \prod_{r_j \in \mathcal{R}^0} P(\beta_j) \cdot \prod_{a_k \in \mathcal{A}^0} P(\xi_k) \\ & \cdot \prod_{u_i, a_k \in \mathcal{S}^0} P(\gamma_i^k) \cdot \prod_{u_i, a_k \in \mathcal{C}^0} P(\lambda_i^k) \cdot \prod_{u_i \in \mathcal{V}^0} P(\mathbf{u}_i) \cdot \prod_{r_j \in \mathcal{A}^0} P(\mathbf{v}_j) \end{aligned} \tag{B.3}$$

All latent variables are considered to be independent of each other. Only $y_{h_{ij}}$ depends on all variables in Ω through the mean of distribution, $p_{h_{ij}}$. Then, we replace the probabilities using the probability density formula. For now on, we omit specific elements and sets in summations by using only indices i for u_i, j for r_j and k for a_k.

$$
\begin{aligned}
L(\Omega,\Theta) = &\prod_{i,j}\left(\frac{1}{\sigma_h\sqrt{2\pi}}e^{\frac{-(y_{h_{ij}}-p_{h_{ij}})^2}{2\sigma_h^2}}\right)\cdot\prod_{i}\left(\frac{1}{\sigma_\alpha\sqrt{2\pi}}e^{\frac{-(\alpha_i-\mathbf{d}^T\mathbf{x}_{u_i})^2}{2\sigma_\alpha^2}}\right)\\
&\cdot\prod_{j}\left(\frac{1}{\sigma_\beta\sqrt{2\pi}}e^{\frac{-(\beta_j-\mathbf{g}^T\mathbf{x}_{r_j})^2}{2\sigma_\beta^2}}\right)\cdot\prod_{k}\left(\frac{1}{\sigma_\xi\sqrt{2\pi}}e^{\frac{-(\xi_k-\mathbf{b}^T\mathbf{x}_{a_k})^2}{2\sigma_\xi^2}}\right)\\
&\cdot\prod_{i,k}\left(\frac{1}{\sigma_\gamma\sqrt{2\pi}}e^{\frac{-(\gamma_i^k-g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))^2}{2\sigma_\gamma^2}}\right)\cdot\prod_{i,k}\left(\frac{1}{\sigma_\lambda\sqrt{2\pi}}e^{\frac{-(\lambda_i^k-g(\mathbf{h}^T\mathbf{x}_{c_{ik}}))^2}{2\sigma_\lambda^2}}\right)\\
&\cdot\prod_{i}\left(\frac{1}{\sqrt{(2\pi)^K det\mathbf{A}_u}}e^{-\frac{1}{2}(\mathbf{u}_i-\mathbf{W}^T\mathbf{x}_{u_i})^T\mathbf{A}_u^{-1}(\mathbf{u}_i-\mathbf{W}^T\mathbf{x}_{u_i})}\right)\\
&\cdot\prod_{j}\left(\frac{1}{\sqrt{(2\pi)^K det\mathbf{A}_v}}e^{-\frac{1}{2}(\mathbf{v}_j-\mathbf{V}^T\mathbf{x}_{r_j})^T\mathbf{A}_v^{-1}(\mathbf{v}_i-\mathbf{V}^T\mathbf{x}_{r_j})}\right)
\end{aligned}
\tag{B.4}
$$

To obtain the log-likelihood $\ell(\Omega,\Theta)$, we take the neperian logarithm of this value using the property that the logarithm of a product of terms is the sum of the logarithm of each term.

$$
\begin{aligned}
\ell(\Omega,\Theta) = &\sum_{i,j}\left(\log\frac{1}{\sigma_h\sqrt{2\pi}}-\frac{(y_{h_{ij}}-p_{h_{ij}})^2}{2\sigma_h^2}\right)+\sum_{i}\left(\log\frac{1}{\sigma_\alpha\sqrt{2\pi}}-\frac{(\alpha_i-\mathbf{d}^T\mathbf{x}_{u_i})^2}{2\sigma_\alpha^2}\right)\\
&+\sum_{j}\left(\log\frac{1}{\sigma_\beta\sqrt{2\pi}}-\frac{(\beta_j-\mathbf{g}^T\mathbf{x}_{r_j})^2}{2\sigma_\beta^2}\right)+\sum_{k}\left(\log\frac{1}{\sigma_\xi\sqrt{2\pi}}-\frac{(\xi_k-\mathbf{b}^T\mathbf{x}_{a_k})^2}{2\sigma_\xi^2}\right)\\
&+\sum_{i,k}\left(\log\frac{1}{\sigma_\gamma\sqrt{2\pi}}-\frac{(\gamma_i^k-g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))^2}{2\sigma_\gamma^2}\right)\\
&+\sum_{i,k}\left(\log\frac{1}{\sigma_\lambda\sqrt{2\pi}}-\frac{(\lambda_i^k-g(\mathbf{h}^T\mathbf{x}_{c_{ik}}))^2}{2\sigma_\lambda^2}\right)\\
&+\sum_{i}\left(\log\frac{1}{\sqrt{(2\pi)^K det\mathbf{A}_u}}-\frac{1}{2}(\mathbf{u}_i-\mathbf{W}^T\mathbf{x}_{u_i})^T\mathbf{A}_u^{-1}(\mathbf{u}_i-\mathbf{W}^T\mathbf{x}_{u_i})\right)\\
&+\sum_{j}\left(\log\frac{1}{\sqrt{(2\pi)^K det\mathbf{A}_v}}-\frac{1}{2}(\mathbf{v}_j-\mathbf{V}^T\mathbf{x}_{r_j})^T\mathbf{A}_v^{-1}(\mathbf{v}_i-\mathbf{V}^T\mathbf{x}_{r_j})\right)
\end{aligned}
\tag{B.5}
$$

Simplifying further,

$$
\begin{aligned}
\ell(\Omega,\Theta) = &-\sum_{i,j}\left(\log\sigma_h + \frac{1}{2}\log 2\pi + \frac{(y_{h_{ij}} - p_{h_{ij}})^2}{2\sigma_h^2}\right)\\
&-\sum_{i}\left(\log\sigma_\alpha + \frac{1}{2}\log 2\pi + \frac{(\alpha_i - \mathbf{d}^T\mathbf{x}_{u_i})^2}{2\sigma_\alpha^2}\right)\\
&-\sum_{j}\left(\log\sigma_\beta + \frac{1}{2}\log 2\pi + \frac{(\beta_j - \mathbf{g}^T\mathbf{x}_{r_j})^2}{2\sigma_\beta^2}\right)\\
&-\sum_{k}\left(\log\sigma_\xi + \frac{1}{2}\log 2\pi + \frac{(\xi_k - \mathbf{b}^T\mathbf{x}_{a_k})^2}{2\sigma_\xi^2}\right)\\
&-\sum_{i,k}\left(\log\sigma_\gamma + \frac{1}{2}\log 2\pi + \frac{(\gamma_i^k - g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))^2}{2\sigma_\gamma^2}\right)\\
&-\sum_{i,k:k\in T_i}\left(\log\sigma_\lambda + \frac{1}{2}\log 2\pi + \frac{(\lambda_i^k - g(\mathbf{h}^T\mathbf{x}_{c_{ik}}))^2}{2\sigma_\lambda^2}\right)\\
&-\sum_{i}\left(\frac{K}{2}\log 2\pi + \frac{1}{2}\log det\mathbf{A}_u + \frac{1}{2}(\mathbf{u}_i - \mathbf{W}^T\mathbf{x}_{u_i})^T\mathbf{A}_u^{-1}(\mathbf{u}_i - \mathbf{W}^T\mathbf{x}_{u_i})\right)\\
&-\sum_{j}\left(\frac{K}{2}\log 2\pi + \frac{1}{2}\log det\mathbf{A}_v + \frac{1}{2}(\mathbf{v}_j - \mathbf{V}^T\mathbf{x}_{r_j})^T\mathbf{A}_v^{-1}(\mathbf{v}_i - \mathbf{V}^T\mathbf{x}_{r_j})\right)
\end{aligned}
\tag{B.6}
$$

Collapsing the terms which are constant across EM iterations, whose sum is represented by C, and replacing standard deviation by variance, we obtain the following:

$$
\begin{aligned}
\ell(\Omega,\Theta) = C &-\sum_{i,j}\frac{1}{2}\left(\log\sigma_h^2 + \frac{(y_{h_{ij}} - p_{h_{ij}})^2}{\sigma_h^2}\right)\\
&-\sum_{i}\frac{1}{2}\left(\log\sigma_\alpha^2 + \frac{(\alpha_i - \mathbf{d}^T\mathbf{x}_{u_i})^2}{\sigma_\alpha^2}\right) - \sum_{j}\frac{1}{2}\left(\log\sigma_\beta^2 + \frac{(\beta_j - \mathbf{g}^T\mathbf{x}_{r_j})^2}{\sigma_\beta^2}\right)\\
&-\sum_{k}\frac{1}{2}\left(\log\sigma_\xi^2 + \frac{(\xi_k - \mathbf{b}^T\mathbf{x}_{a_k})^2}{\sigma_\xi^2}\right) - \sum_{i,k}\frac{1}{2}\left(\log\sigma_\gamma^2 + \frac{(\gamma_i^k - g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))^2}{\sigma_\gamma^2}\right)\\
&-\sum_{i,k}\frac{1}{2}\left(\log\sigma_\lambda^2 + \frac{(\lambda_i^k - g(\mathbf{h}^T\mathbf{x}_{c_{ik}}))^2}{\sigma_\lambda^2}\right)\\
&-\sum_{i}\frac{1}{2}\left(\log det\mathbf{A}_u + (\mathbf{u}_i - \mathbf{W}^T\mathbf{x}_{u_i})^T\mathbf{A}_u^{-1}(\mathbf{u}_i - \mathbf{W}^T\mathbf{x}_{u_i})\right)\\
&-\sum_{j}\frac{1}{2}\left(\log det\mathbf{A}_v + (\mathbf{v}_j - \mathbf{V}^T\mathbf{x}_{r_j})^T\mathbf{A}_v^{-1}(\mathbf{v}_i - \mathbf{V}^T\mathbf{x}_{r_j})\right)
\end{aligned}
\tag{B.7}
$$

We can expand $p_{h_{ij}}$ in terms of latent variables, obtaining the following.

$$\begin{aligned}\ell(\Omega,\Theta) = C &- \frac{1}{2}\sum_{i,j}\left(\frac{1}{\sigma_h^2}(y_{h_{ij}} - \mathbf{u}_i^T\mathbf{v_j} - \alpha_i - \beta_j - \xi_k - \delta_1(i,k)\gamma_i^k - \delta_2(i,k)\lambda_i^k)^2 + \log\sigma_h^2\right)\\ &- \frac{1}{2}\sum_i\left(\frac{(\alpha_i - \mathbf{d}^T\mathbf{x}_{u_i})^2}{\sigma_\alpha^2} + \log\sigma_\alpha^2 + (\mathbf{u}_i - \mathbf{W}^T\mathbf{x}_{u_i})^T\mathbf{A}_u^{-1}(\mathbf{u}_i - \mathbf{W}^T\mathbf{x}_{u_i}) + \log(det\mathbf{A}_u)\right)\\ &- \frac{1}{2}\sum_j\left(\frac{(\beta_j - \mathbf{g}^T\mathbf{x}_{r_j})^2}{\sigma_\beta^2} + \log\sigma_\beta^2 + (\mathbf{v}_j - \mathbf{V}^T\mathbf{x}_{r_j})^T\mathbf{A}_v^{-1}(\mathbf{v}_i - \mathbf{V}^T\mathbf{x}_{r_j}) + \log(det\mathbf{A}_v) + \right)\\ &- \frac{1}{2}\sum_k\left(\log\sigma_\xi^2 + \frac{(\xi_k - \mathbf{b}^T\mathbf{x}_{a_k})^2}{\sigma_\xi^2}\right) - \frac{1}{2}\sum_{i,k}\left(\log\sigma_\gamma^2 + \frac{(\gamma_i^k - g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))^2}{\sigma_\gamma^2}\right)\\ &- \frac{1}{2}\sum_{i,k:k\in T_i}\left(\log\sigma_\lambda^2 + \frac{(\lambda_i^k - g(\mathbf{h}^T\mathbf{x}_{c_{ik}}))^2}{\sigma_\lambda^2}\right)\end{aligned} \tag{B.8}$$

We can observe that this is a modification of the formula presented for Regression-based Latent Factor Models [Agarwal and Chen, 2009] by incorporating new latent variables and dropping linear regression on dyadic features term ($\mathbf{x}_{ij}^T\mathbf{b}$). Using the definition of $E[\phi|Rest] = \hat{\phi}$ such that $\ell'(\hat{\phi}) = 0$ and $V[\phi|Rest] = \ell''(\hat{\phi})^{-1}$ [Tang et al, 2013], where ϕ is any variable, we have estimations of mean and variance, thus the full conditional probabilities, to be used in Gibbs Sampling.

B.2 Derivatives of Expectation of Log-Likelihood for CAP

We do not have a closed formula for the joint distribution of variables due to the product $\mathbf{u}_i^T\mathbf{v}_j$, that is, we do not know the distribution of a product of $\mathcal{MVN}s$. Then, this distribution is approximated through Gibbs Sampling. An empirical expectation (E^*) can be taken from this approximated distribution [Booth and Hobert, 1999], considering a sample of L elements.

$$\begin{aligned}E[X] &= \sum_x x\ p(x),\\ E^*[X] &= \frac{1}{L}\sum_{i=1}^{L} x_i,\end{aligned} \tag{B.9}$$

where x_i is the value of random variable X in i-th sample. We want the expected value of a function. Thus,

$$E[f(X)] = \sum_x f(x)\ p(x),$$
$$E^*[f(X)] = \frac{1}{L}\sum_{i=1}^{L} f(x_i)$$

After Gibbs Sampling step, we have L samples of the whole latent vector $\{\Omega\} = \{\Omega^{(1)}, \Omega^{(2)}, \ldots, \Omega^{(L)}\}$. Then, we may compute the probabilities for each occurred value or simply sum over the function f evaluated over each sample to obtain the empirical distribution of ℓ.

$$\begin{aligned} E^* &= \sum_{\{\Omega\}} p(\Omega|y_{ij}, \Theta)\ \ell(\Omega, \Theta) \\ &= \frac{1}{L}\sum_{l=1}^{L}\ \ell(\Omega^l, \Theta), \end{aligned} \tag{B.10}$$

where $\{\Omega\}$ is the set of all possible full latent vectors, Ω^l is the value of latent factor vector in l-th sample. In this case, the log-likelihood is a function of the latent variables, but, after the sample, we marginalize out of these values through the summation. We know how to calculate $\ell(\Omega, \Theta)$ because we have the values of the latent variables on each sample as well as the true value of the helpfulness ($y_{h_{ij}}$). Then, the result E^* is a function of only the parameters Θ, after the sample.

Since the value of E* is known, it is possible to derive it. As it is a summation, we can use the property of the derivative of a sum being the sum of derivatives. The formulas for all parameters can be easily found and set to zero in order to find an optimal value, which falls in OLS linear regression on empiric mean of each variable. However, parameters $\mathbf{r}$ and $\mathbf{h}$ are an exception, as we have a non-linear regression to solve due to the sigmoid transformation g. Instead, Newton-Raphson method can be used, which is a method for finding a root of a function. Since we want to maximize E^*, we find the root of its derivative through this method. Consequently, we have to calculate the first and second partial derivatives of E^* with respect to $\mathbf{r}$ and $\mathbf{h}$, the first is the function whose root is going to be estimated and the latter is its derivative. Only terms in $\ell(\Theta, \Omega)$ containing $\mathbf{r}$ are relevant and, first, we consider a single dimension $\mathbf{r}_j$.

$$
\begin{aligned}
\frac{\partial E^*}{\partial \mathbf{r}_j} &= \frac{\partial}{\partial \mathbf{r}_j}\frac{1}{L}\sum_{l=1}^{L} -\frac{1}{2}\sum_{i,k}\left(\frac{(\gamma_i^{kl} - g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))^2}{\sigma_\gamma^2}\right) \\
&= \frac{1}{L}\sum_{l=1}^{L} -\frac{1}{2}\frac{2}{\sigma_\gamma^2}\sum_{i,k}(\gamma_i^{kl} - g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))(-1)g'(\mathbf{r}^T\mathbf{x}_{s_{ik}})\mathbf{p}_{ij}^k \\
&= \frac{1}{\sigma_\gamma^2}\frac{1}{L}\sum_{l=1}^{L}\sum_{i,k}(\gamma_i^{kl} - g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))g'(\mathbf{r}^T\mathbf{x}_{s_{ik}})\mathbf{p}_{ij}^k \\
&= \frac{1}{\sigma_\gamma^2}\sum_{i,k}\frac{1}{L}\sum_{l=1}^{L}(\gamma_i^{kl} - g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))g'(\mathbf{r}^T\mathbf{x}_{s_{ik}})\mathbf{p}_{ij}^k \\
&= \frac{1}{\sigma_\gamma^2}g'(\mathbf{r}^T\mathbf{x}_{s_{ik}})\mathbf{p}_{ij}^k\sum_{i,k}\frac{1}{L}\sum_{l=1}^{L}\gamma_i^{kl} - \frac{1}{l}\sum_{\{\delta_{ij}^l\}} g(\mathbf{r}^T\mathbf{x}_{s_{ik}}) \\
&= \frac{1}{\sigma_\gamma^2}g'(\mathbf{r}^T\mathbf{x}_{s_{ik}})\mathbf{p}_{ij}^k\sum_{i,k}\left(E^*(\gamma_i^k) - \frac{1}{l}lg(\mathbf{r}^T\mathbf{x}_{s_{ik}})\right) \\
&= \frac{1}{\sigma_\gamma^2}\sum_{i,k}\left(E^*(\gamma_i^k) - g(\mathbf{r}^T\mathbf{x}_{s_{ik}})\right)g'(\mathbf{r}^T\mathbf{x}_{s_{ik}})\mathbf{p}_{ij}^k,
\end{aligned}
\tag{B.11}
$$

where $g'(\mathbf{r}^T\mathbf{x}_{s_{ik}})$ is $\frac{\partial g(\mathbf{r}^T\mathbf{x}_{s_{ik}})}{\partial(\mathbf{r}^T\mathbf{x}_{s_{ik}})}$. $E^*(\gamma_i^k)$ is the empiric mean of γ_i^k variable considering L samples. This solution is no different than a non-linear regression on the empiric mean of a variable by minimizing sum of squares. In this calculation, we use the Chain Rule of a composite function. In this scenario,

$$
\frac{\partial(\gamma_i^{kl} - g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))^2)}{\partial \mathbf{r}_j} = \frac{\partial(\gamma_i^{kl} - g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))^2}{\partial(\gamma_i^{kl} - g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))} \cdot \frac{\partial(\gamma_i^{kl} - g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))}{\partial(g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))} \cdot \frac{\partial(g(\mathbf{r}^T\mathbf{x}_{s_{ik}}))}{\partial(\mathbf{r}^T\mathbf{x}_{s_{ik}})} \cdot \frac{\partial(\mathbf{r}^T\mathbf{x}_{s_{ik}})}{\partial \mathbf{r}_j}
\tag{B.12}
$$

The vector derivative is just the extrapolation of this using the definition below.

$$
\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \left[\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_1}\ \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_2}\ \cdots\ \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_n}\right]
\tag{B.13}
$$

Then,

$$
\begin{aligned}
\frac{\partial E^*}{\partial \mathbf{r}} &= \left[\frac{\partial E^*}{\partial \mathbf{r}_1}\ \frac{\partial E^*}{\partial \mathbf{r}_2}\ \cdots\ \frac{\partial E^*}{\partial \mathbf{r}_n}\right] \\
&= \frac{1}{\sigma_\gamma^2}\sum_{i,k}\left(E^*_{\{\delta_{ij}^l\}}(\gamma_i^k) - g(\mathbf{r}^T\mathbf{x}_{s_{ik}})\right)g'(\mathbf{r}^T\mathbf{x}_{s_{ik}})\mathbf{p}_i^k
\end{aligned}
\tag{B.14}
$$

The term $\mathbf{x}_{s_{ik}}$ is a vector, which turns this result into a vector. The remainder

term is the same for all $\mathbf{r}_j$, which makes this formula easily extensible to vector case. The formula for $\mathbf{h}$ is similar.

In order to calculate the second order derivative by an index j' of $\mathbf{r}$, we have now a derivative of a vector by a scalar. This is defined as deriving each dimension j, which is $\frac{\partial E^*}{\mathbf{r}_j}$. Considering the dimensions j and j',

$$\begin{aligned}
\frac{\partial^2 E^*}{\partial \mathbf{r}_j \partial \mathbf{r}_{j'}} &= \frac{\partial}{\partial \mathbf{r}_{j'}} \frac{1}{\sigma_\gamma^2} \sum_{i,k} \left(E^*_{\{\delta^l_{ij}\}}(\gamma_i^k) - g(\mathbf{r}^T \mathbf{x}_{s_{ik}}) \right) g'(\mathbf{r}^T \mathbf{x}_{s_{ik}}) \mathbf{p}_{ij}^k \\
&= \frac{1}{\sigma_\gamma^2} \sum_{i,k} \left(\frac{\partial (E^*_{\{\delta^l_{ij}\}}(\gamma_i^k) - g(\mathbf{r}^T \mathbf{x}_{s_{ik}}))}{\partial \mathbf{r}_{j'}} g'(\mathbf{r}^T \mathbf{x}_{s_{ik}}) + (E^*_{\{\delta^l_{ij}\}}(\gamma_i^k) - g(\mathbf{r}^T \mathbf{x}_{s_{ik}})) \frac{\partial g'(\mathbf{r}^T \mathbf{x}_{s_{ik}})}{\partial \mathbf{r}_{j'}} \right) \mathbf{p}_{ij}^k \\
&= \frac{1}{\sigma_\gamma^2} \sum_{i,k} \left((-1) g'(\mathbf{r}^T \mathbf{x}_{s_{ik}}) \mathbf{p}_{ij'}^k g'(\mathbf{r}^T \mathbf{x}_{s_{ik}}) + (E^*_{\{\delta^l_{ij}\}}(\gamma_i^k) - g(\mathbf{r}^T \mathbf{x}_{s_{ik}})) g''(\mathbf{r}^T \mathbf{x}_{s_{ik}}) \mathbf{p}_{ij'}^k \right) \mathbf{p}_{ij}^k \\
&= \frac{1}{\sigma_\gamma^2} \sum_{i,k} \left((E^*_{\{\delta^l_{ij}\}}(\gamma_i^k) - g(\mathbf{r}^T \mathbf{x}_{s_{ik}})) g''(\mathbf{r}^T \mathbf{x}_{s_{ik}}) - (g'(\mathbf{r}^T \mathbf{x}_{s_{ik}}))^2 \right) \mathbf{p}_{ij}^k \mathbf{p}_{ij'}^k
\end{aligned} \tag{B.15}$$

We used the Product Rule in this differentiation. The first generalization consists of the second derivative in one dimension of the full vector.

$$\frac{\partial^2 E^*}{\partial \mathbf{r} \partial \mathbf{r}_{j'}} = \frac{1}{\sigma_\gamma^2} \sum_{i,k} \left((E^*_{\{\delta^l_{ij}\}}(\gamma_i^k) - g(\mathbf{r}^T \mathbf{x}_{s_{ik}})) g''(\mathbf{r}^T \mathbf{x}_{s_{ik}}) - (g'(\mathbf{r}^T \mathbf{x}_{s_{ik}}))^2 \right) \mathbf{p}_{ij'}^k \mathbf{x}_{s_{ik}} \tag{B.16}$$

Considering p_i^k to be a column vector, we have the first derivative as a column vector and the combination of columns compose a matrix. Each column of the matrix is multiplied by the corresponding $\mathbf{p}_{ij'}^k$.

$$\frac{\partial^2 E^*}{\partial \mathbf{r} \partial \mathbf{r}^T} = \frac{1}{\sigma_\gamma^2} \sum_{i,k} \left((E^*_{\{\delta^l_{ij}\}}(\gamma_i^k) - g(\mathbf{r}^T \mathbf{x}_{s_{ik}})) g''(\mathbf{r}^T \mathbf{x}_{s_{ik}}) - (g'(\mathbf{r}^T \mathbf{x}_{s_{ik}}))^2 \right) \mathbf{x}_{s_{ik}} \mathbf{x}_{s_{ik}} \tag{B.17}$$

Since we consider $\mathbf{x}_{s_{ik}}$ to be a column vector, the multiplication of a column vector by a row vector results in the desired value $\mathbf{p}_{ij}^k \mathbf{p}_{ij'}^k$ in position (j, j') of the second derivative matrix.

We just have to define, now, the values of the derivatives of the sigmoid function (f) and the first and second order derivatives are completely defined.

$$g'(x) = \frac{e^x}{(1+e^x)^2} \tag{B.18}$$

$$g''(x) = \frac{e^x (e^x - 1)}{(1+e^x)^3} \tag{B.19}$$

B.3 Derivatives of Loss Function for BETF

As stochastic gradient descent only regards the derivative of a single example, we derive such single-instance cost. We just consider the optimization formula only regarding a single instance h_{ikl} present in the training set.

$$\begin{aligned} E^{ikl}_{BETF} = &\frac{1}{2}(y_{h_{ikl}} - p_{h_{ikl}})^2 + \frac{1}{2}(y_{r_{kl}} - p_{r_{kl}})^2 + \frac{\lambda_u}{2}\left\|\mathbf{u}_i\right\|^2_{Fro} + \frac{\lambda_v}{2}\left\|\mathbf{v}_k\right\|^2_{Fro} \\ &+ \frac{\lambda_p}{2}\left\|\mathbf{p}_l\right\|^2_{Fro} + \frac{\lambda_s}{2}\left\|\mathbf{s}_{ikl}\right\|^2_{Fro} \end{aligned} \tag{B.20}$$

In order to apply stochastic gradient descent, we have to find the derivative of this expression related to $\mathbf{u}_i$, $\mathbf{v}_k$, $\mathbf{p}_l$ and $\mathbf{s}_{ikl}$. We compute first for $\mathbf{u}_i$.

$$\begin{aligned} \frac{\partial E^{ikl}_{BETF}}{\partial \mathbf{u}_i} &= \frac{1}{2}2(q_{ij} - g(\hat{q_{ikl}}))g'(\hat{q_{ikl}})\frac{\partial h_{ikl}}{\partial \mathbf{u}_i} + \frac{\lambda_u}{2}2\mathbf{u}_i \\ &= (q_{ij} - g(\hat{q_{ikl}}))g'(\hat{q_{ikl}})(\mathbf{s} \times_v \mathbf{v}_j \times_p \mathbf{p}_k) + \lambda_u \mathbf{u}_i, \end{aligned} \tag{B.21}$$

where we calculate $\frac{\partial h_{ikl}}{\mathbf{u}_i}$ as follows.

$$\begin{aligned} \frac{\partial h_{ikl}}{\mathbf{u}_{id}} &= \frac{\partial(\sum_{x=1}^{K}\mathbf{u}_{ix}\sum_{y=1}^{K}\sum_{z=1}^{K}\mathbf{s}_{xyz}\mathbf{v}_{jy}\mathbf{p}_{kz})}{\partial \mathbf{u}_{id}} \\ &= \frac{\partial(\mathbf{u}_{id}\sum_{y=1}^{K}\sum_{z=1}^{K}\mathbf{s}_{dyz}\mathbf{v}_{jy}\mathbf{p}_{kz})}{\partial \mathbf{u}_{id}} \\ &= \sum_{y=1}^{K}\sum_{z=1}^{K}\mathbf{s}_{dyz}\mathbf{v}_{jy}\mathbf{p}_{kz} \\ &= \sum_{z=1}^{K}\sum_{y=1}^{K}\mathbf{s}_{dyz}\mathbf{v}_{jy}\mathbf{p}_{kz} \\ &= \sum_{z=1}^{K}\mathbf{p}_{kz}\sum_{y=1}^{K}\mathbf{s}_{dyz}\mathbf{v}_{jy} \\ &= \sum_{z=1}^{K}\mathbf{p}_{kz}(\mathbf{s} \times_v \mathbf{v}_j)_{dz} \\ &= (\mathbf{s} \times_v \mathbf{v}_j \times_p \mathbf{p}_k)_d \\ \frac{\partial h_{ikl}}{\mathbf{u}_i} &= \mathbf{s} \times_v \mathbf{v}_j \times_p \mathbf{p}_k \end{aligned} \tag{B.22}$$

Which is a $(K, 1)$ vector. For $\mathbf{v}_j$ and $\mathbf{p}_k$, we also have terms related to the matrix of ratings.

$$\frac{\partial E^{ikl}_{BETF}}{\partial \mathbf{v}_j} = (q_{ij} - g(\hat{q_{ikl}}))g'(\hat{q_{ikl}})(\mathbf{s} \times_u \mathbf{u}_i \times_p \mathbf{p}_k) + (o_{jk} - g(\hat{o_{jk}}))g'(\hat{o_{jk}})\mathbf{p}_k + \lambda_v \mathbf{v}_j \tag{B.23}$$

$$\frac{\partial E^{ikl}_{BETF}}{\partial \mathbf{p}_k} = (q_{ij} - g(\hat{q_{ikl}}))g'(\hat{q_{ikl}})(\mathbf{s} \times_u \mathbf{u}_i \times_v \mathbf{v}_j) + (o_{jk} - g(\hat{o_{jk}}))g'(\hat{o_{jk}})\mathbf{v}_j + \lambda_p \mathbf{p}_k \tag{B.24}$$

$$\tag{B.25}$$

In the algorithm proposed, differently from typical stochastic gradient descent, each derivative term is updated separately, avoiding multiple updates regarding the same related entities. For example, the second term of the last derivative is related to author and product pair, which repeats for all votes given to the review written by the author to the product. The same happens with regularization terms, which would be updated once for each helpfulness vote, in each iteration. Instead, the proposed algorithm update terms from different sources separately: first, regarding voter error; second, regarding rating error; and finally, regarding regularization. This way, vote error is prioritized over rating error and regularization. Although we present derivative terms together, they are updated separately.

Given the definition $\hat{h}_{ikl} = \mu_h + \bar{h_{u_i}} + \bar{h_{r_j}} + \mathbf{s} \times_u \mathbf{u}_i \times_v \mathbf{v}_k \times_p \mathbf{p}_l$, we have the following:

$$\frac{\partial h_{ikl}}{\mathbf{s}_{xyz}} = \mathbf{u}_{ix}\mathbf{v}_{jy}\mathbf{p}_{kz} \tag{B.26}$$

So, each dimension is multiplied by the corresponding dimension in $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{p}$. Thus, we can extended to the whole tensor case.

$$\frac{\partial h_{ikl}}{\mathbf{s}} = \mathbf{u}_i \otimes \mathbf{v}_j \otimes \mathbf{p}_k \tag{B.27}$$

Where $\otimes$ is the outer product of tensors resulting in a new one with the sum of both dimensions [Moghaddam et al., 2012]. Consequently, we have the following derivative regarding the objective function.

$$\frac{\partial E^{ikl}_{BETF}}{\partial \mathbf{s}} = (q_{ij} - g(\hat{q_{ikl}}))g'(\hat{q_{ikl}})(\mathbf{u}_i \otimes \mathbf{v}_j \otimes \mathbf{p}_k) + \lambda_s \mathbf{s} \tag{B.28}$$

B.4 Derivatives of Loss Function for CoRaSVR

We first focus in the derivative of the regression portion of $E_{CoRaSVR}$, represented by ∂_{r_k}, related to a review r_k:

$$\begin{aligned}
\partial_{r_k} &= \frac{\partial\left(\gamma\ y_{r_k} max\{|p_{r_k} - y_{r_k}| - \varepsilon), 0\}\right)}{\partial \mathbf{w}} \\
&= \begin{cases} \frac{\partial\left(\gamma\ y_{r_k}(|p_{r_k} - y_{r_k}| - \varepsilon)\right)}{\partial \mathbf{w}} & |p_{r_k} - y_{r_k}| > \varepsilon \\ \frac{\partial\left(\gamma\ y_{r_k} 0\right)}{\partial \mathbf{w}} & |p_{r_k} - y_{r_k}| \leq \varepsilon \end{cases} \\
&= \begin{cases} \frac{\partial\left(\gamma\ y_{r_k}((p_{r_k} - y_{r_k}) - \varepsilon)\right)}{\partial \mathbf{w}} & p_{r_k} - y_{r_k} > \varepsilon \\ \frac{\partial\left(\gamma\ y_{r_k}((y_{r_k} - p_{r_k}) - \varepsilon)\right)}{\partial \mathbf{w}} & p_{r_k} - y_{r_k} < -\varepsilon \\ \frac{\partial\left(\gamma\ y_{r_k} 0\right)}{\partial \mathbf{w}} & |p_{r_k} - y_{r_k}| \leq \varepsilon \end{cases} \\
&= \begin{cases} \gamma\ y_{r_k}\mathbf{x}_{r_k} & p_{r_k} - y_{r_k} > \varepsilon \\ -\gamma\ y_{r_k}\mathbf{x}_{r_k} & p_{r_k} - y_{r_k} < -\varepsilon \\ 0 & |p_{r_k} - y_{r_k}| \leq \varepsilon \end{cases}
\end{aligned} \tag{B.29}$$

Note that the derivative of constants y_{r_k} and ε are 0, and the derivative of p_{r_k} is $\mathbf{x}_{r_k}$ as $p_{r_k} = \mathbf{w}^T\mathbf{x}_{r_k}$.

We now derive the term related to cross entropy loss for pairwise probability, which we denote ∂_{r_i,r_j}, related to a pair of reviews r_i, r_j.

$$\begin{aligned}
\partial_{r_i,r_j} &= \frac{\partial\left((1-\gamma)\ y_{r_i,r_j}\left(\log(1 + e^{p_{r_i} - p_{r_j}}) - y_{r_i,r_j}(p_{r_i} - p_{r_j})\right)\right)}{\partial \mathbf{w}} \\
&= (1-\gamma)\ y_{r_i,r_j}\frac{1}{e^{p_{r_i} - p_{r_j}}}\frac{\partial(1 + e^{p_{r_i} - p_{r_j}})}{\partial \mathbf{w}} - y_{r_i,r_j}(\mathbf{x}_{r_i} - \mathbf{x}_{r_j}) \\
&= (1-\gamma)\ y_{r_i,r_j}\frac{1}{e^{p_{r_i} - p_{r_j}}}e^{p_{r_i} - p_{r_j}}(\mathbf{x}_{r_i} - \mathbf{x}_{r_j}) - y_{r_i,r_j}(\mathbf{x}_{r_i} - \mathbf{x}_{r_j}) \\
&= (1-\gamma)\ y_{r_i,r_j}\left(\frac{1}{e^{-(p_{r_i} - p_{r_j})}} - y_{r_i,r_j}\right)(\mathbf{x}_{r_i} - \mathbf{x}_{r_j}) \\
&= (1-\gamma)\ y_{r_i,r_j}\left(g(p_{r_i} - p_{r_j}) - y_{r_i,r_j}\right)(\mathbf{x}_{r_i} - \mathbf{x}_{r_j})
\end{aligned} \tag{B.30}$$

where $\frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}} = g(x)$.

Appendix C

Probabilistic Graphical Models of Review Recommender Systems

The probabilistic graphical model (PGM) for BETF and CAP are presented in this Chapter. Such models help understanding the relationships among latent and observed variables of each method. It also allows to obtain the likelihood by factorizing over the PGM.

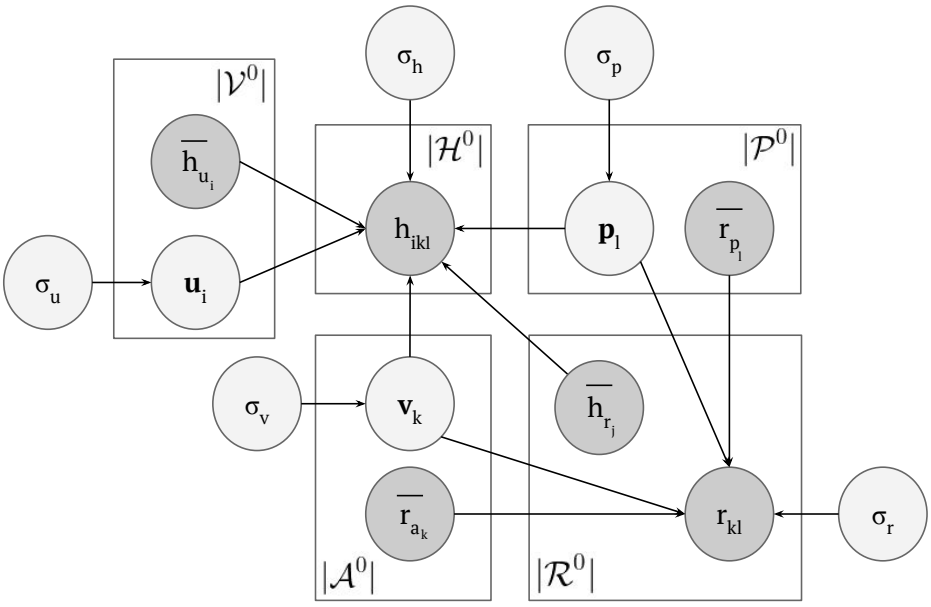


Figure C.1: PGM for BETF. Rectangles represent the cardinality of the given variable. Observed variables are represented by dark circles and latent, by light ones. We use h_{ij} as $y_{h_{ij}}$.

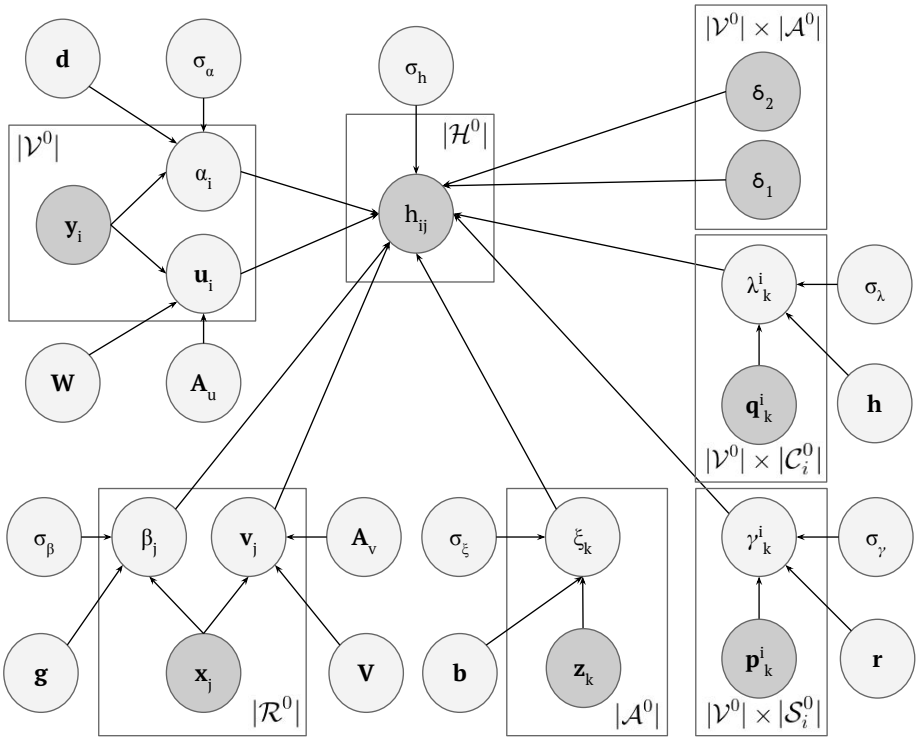


Figure C.2: PGM for CAP. The notation is the similar to Figure C.1.