REVENUE OPTIMIZATION AND CUSTOMER TARGETING IN DAILY-DEALS SITES

ANISIO MENDES LACERDA

REVENUE OPTIMIZATION AND CUSTOMER TARGETING IN DAILY-DEALS SITES

Thesis presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais. Departamento de Ciência da Computação. in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Nivio Ziviani
Co-Advisor: Adriano Veloso

Belo Horizonte
December 20, 2013

Lacerda, Anisio Mendes

L131r Revenue Optimization and Customer Targeting in Daily-Deals Sites / Anisio Mendes Lacerda. — Belo Horizonte, 2013
xvi, 95 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.
Orientador: Nivio Ziviani
Coorientador: Adriano Veloso

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

CDU 519.6*73 (043)

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

FOLHA DE APROVAÇÃO

Revenue optimization and customer targeting in daily-deals sites

ANÍSIO MENDES LACERDA

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

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

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

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

PROF. LEANDRO BALBY MARINHO
Departamento de Sistemas e Computação – UFCG

PROF. RICARDO BAEZA -Yates
YAHOO - Research Barcelona

PROF. WAGNER MEIRA JÚNIOR
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 20 de dezembro de 2013.

À Gisele, o amor da minha vida.

Aos meus pais e irmã que foram fundamentais para que este sonho se tornasse realidade.

Aos amigos, por sempre estarem ao meu lado.

...

Agradecimentos

A minha querida esposa Gisele pelo amor e carinho incondicionais. Obrigado Gisele por todo o apoio nos momentos difíceis durante os quais tive vontade de desistir de tudo. Gisele sem seu amor nada disso seria possível, obrigado por compartilhar sua vida comigo.

Aos meus amados pais Maria Nilza Mendes Lacerda e Sebastião Duarte Lacerda pelo amor, preocupação e todos os sacrifícios realizados no passado que me permitiram chegar até aqui.

A minha querida irmã Pollyana Mendes Lacerda, a melhor irmã do mundo, pelo carinho e apoio durante todos os momentos dessa caminhada.

A minha querida tia Marina Duarte Lacerda, que esteve junto de mim desde o início, compartilhando os momentos felizes e ajudando nos momentos difíceis.

A minha prima Ana Maria Mafra, que me ajudou muito quando cheguei a Belo Horizonte. Sem sua ajuda tudo teria sido muito mais difícil.

Ao Prof. Nivio Ziviani pelo excelente trabalho de orientação e pelo exemplo de dedicação incansável aos trabalhos de professor, pesquisador e empreendedor. Sua enorme experiência, tanto acadêmica quanto profissional, foi fundamental para a conclusão deste trabalho. Sua preocupação com a qualidade e a excelência do trabalho são marcas na minha formação e que levarei por toda a minha vida.

Ao Prof. Adriano Veloso pelas conversas e com quem aprendi técnicas de *machine learning* que foram fundamentais para a realização deste trabalho de tese.

Agradecer a todas as pessoas que me ajudaram é uma tarefa penosa pois sempre corro o risco de omitir alguém. Ainda assim, não poderia deixar de agradecer à grandes amigos que fiz nesta caminhada: Álvaro, Bárbara, Charles, Claudine, Davi, Denílson, Fabiano, Humberto, Pável, Thierson, Tupy, Wallace, Rickson, Marco Túlio, Cristiano, Hendrickson, Rafael, Emanuel, Rocha, Delboni, Marcelino, Fred, B3, Bezerro, Fernando, Bola, Rodrygo, Erickson, Modesto, Louback, Vítor, Thiago, Itamar, Cristiano, Thales, Leonardo, Sabir.

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

Abstract

Daily-deals sites (DDSs), such as Groupon and LivingSocial, attract millions of customers looking for products and services at significantly reduced prices. The challenge of DDSs is to find the best match between deals and customers while generating as much revenue as possible. Hence, successful DDSs need to maximize revenue and increase customer satisfaction. These two problems are investigated in this thesis.

The revenue of DDSs depend how much they charge merchants for each deal sold and on the number of coupons sold for each deal (aka. deal size). The percentage charged is a business decision, but the deal size can be predicted. This thesis introduces a new model for deal size prediction that considers both intra-market competition and market interplay among deals. The deal size prediction method offers gains in precision ranging from 8.18% to 17.67% in comparison with state-of-the-art methods.

Maximizing revenue depends on deal sizes and on customers, which most of the time are reached through email marketing. If customers receive uninteresting, non-personalized emails on a daily-basis, they stop paying attention to emails and treat them as spam. When dealing with customer satisfaction, we tackled two problems: (i) how to reduce the volume of emails sent daily without reducing revenue and (ii) how to personalize emails.

For reducing the percentage of emails sent, we propose criteria to sort customers according to their purchase probability and to apply a muti-armed bandit algorithm to choose which criterion is more appropriate to select the user that should receive the email. Experiments showed that reducing the number of emails sent by 40% does not affect the number clicks on deals advertised in the emails.

For the task of email personalization, we model the problem so as to obtain user feedback on deals of the day as soon as possible. We start emailing deals recommended by current methods and wait for user feedback. Having feedback, the remaining users receive better recommendations by reranking current recommendation lists using feedback. Experiments show that our algorithms offer gains in precision ranging from 7.9% to 34.0% in comparison with state-of-the-art recommendation algorithms.

Contents

Chapter 1

Introduction

Couponing is one of the oldest, most effective, and widely used tools to promote sales [Collard *et al.*, 2001, Belch *et al.*, 2008]. The world's first coupon was created by Coca-Cola in 1887. The ticket, mailed to homes throughout the United States of America and strategically placed in magazines offered potential customers a free glass of the year-old drink. From 1894 to 1913, approximately 8.5 million coupons were redeemed nationwide, marking the first coupon campaign a success [Donnelly, 2012].

Nowadays, the online daily deal industry has been experiencing this kind of success. Daily-Deals Sites (DDSs) provide online discount coupons for diverse products and services, including restaurant tickets, theater entries, haircuts, etc., and have been established in more than 50 countries. A DDS operates as a mediator, where its administrator works with local merchants (sellers) to negotiate a deal (product/service) that is sold to customers (buyers). Just as an example of their huge success, it is known that around 60% of all online shoppers subscribe to DDSs in the USA [Dholakia, 2012], and they spent $3.6 billion on these websites in 2012, an increase of nearly 87% over 2011 [Freed and Berg, 2012].

Historically, local business have reached customers through a variety of methods, including online advertising, direct mail, newspapers, the yellow pages, radio, television, and promotions. However, these channels are not always available or may be too costly for most local businesses. Hence, DDSs became a new and important group-buying alternative for both local business and customers.

While merchants are mainly interested in locally disseminating their brand in order to increase revenue at low costs, potential customers are seeking discounted prices for products and services (aka. deals). Hence, the challenge of DDSs is to find and offer the best deals from merchants that suit customer's interest (relevance), while generating as much revenue as possible. In other words, successful DDSs need to *maximize revenue* and *increase customer satisfaction*. These two problems are investigated in this thesis.

From an economic perspective, the profitability of DDSs depends directly on two factors: (i) the commission associated with a deal, and (ii) the number of deals that are actually sold, typically referred as the deal size. While commissions are business-dependent decisions, an estimation of the number of deals to be sold is a particularly important information for building successful deals catalogs (i.e. the set of deals active during a pre-defined period). If the deal size can be predicted, the DDS administrator can simulate catalogs that maximize the profitability of the DDS.

While being of paramount importance to the success of DDSs, accurately predicting deal sizes is surrounded by challenges. First, the catalog of deals is usually available for a limited time frame, which on average varies from 4 to 5 days [Byers *et al.*, 2012a]. Second, deals may compete among themselves for customer preference, resulting in hesitation and possibly hurting the corresponding sales as well as the overall revenue. Finally, the attractiveness of a deal cannot be assessed in isolation, but rather it is relative to other deals also shown in the catalog.

Despite these challenges, existing solutions for the deal size prediction problem [Byers *et al.*, 2012a, Lappas and Terzi, 2012] produce global predictors that often neglect the existence of complex interactions between deals, such as competition for customer preference. Existing works also assume that characteristics of deals are equally important. Hence, in the first part of this thesis we propose to model the deal size prediction problem as a machine learning regression task. The solution is based on the concept of deal markets, i.e., sets of deals that are likely to attract the attention of similar customers.

Having an effective catalog of deals, the next step to increase revenue is to advertise deals. A variety of channels are used to suggest deals to customers, including mobile apps, websites, online affiliates, advertising networks, and social networks. However, the primary approach is yet to send email featuring the deals of the day [Freed and Berg, 2012]. Considering that each email sent contains a small subset of all available deals, selecting deals that are relevant to specific customers is essential to guarantee customer satisfaction. The second problem tackled in this thesis goes into this direction, and studies which deals should be recommended, when and to whom.

Although there is a variety of recommendation algorithms available, the daily-deals scenario has characteristics that make the problem particularly challenging. First, most of the customers of a DDS are sporadic bargain hunters, and thus preference data is extremely sparse and noisy. Moreover, deals have a short life period, making data extremely volatile. Finally, customer taste and interest may undergo temporal drifts as new deals appear. State-of-the-art collaborative-filtering recommendation methods explore the overlap in customers past consumption behavior and assume that the cat-

alog is almost static. Such characteristics make it hard for existing recommendation algorithms to improve on their daily-deals recommendations.

1.1 Research Outline and Questions

Two broad questions motivate the research of this thesis: (i) *Can we select a set of deals that maximizes the revenue of a DDS?*, and (ii) *Can we improve customer satisfaction when advertising offers by email?* Individual components towards solving these problems already exist (see Section 2.1 and Section 2.2 for an overview). However, other aspects, such as how to take advantage of the available deals to learn a predictor for the number of coupons sold for each deal, or which customers are more receptive to deal recommendations, have not yet been investigated.

In this thesis, we aim to close some of these gaps, contributing to the long-term goal of a complete integration of merchants and customers throughout a DDS. In Figure 1.1, we illustrate the problems investigated in this thesis and how they relate to the key players in the DDS ecosystem.

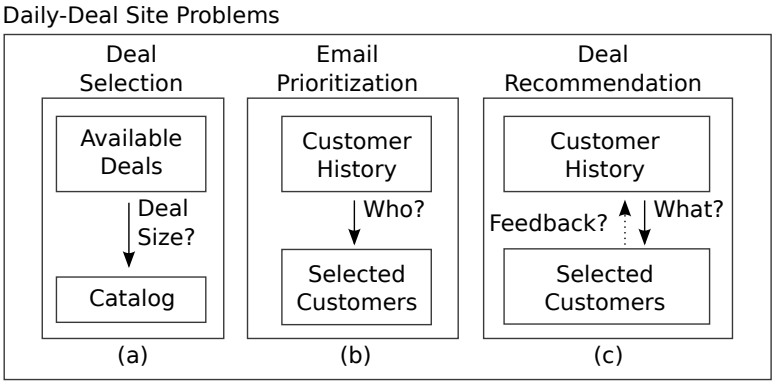


Figure 1.1: Overview of the DDS problems tackled in this thesis.

In Figure 1.1(a), we show the deal selection problem, which consists in deciding, in a daily basis, which of the available deals will form the catalog of the day. Our proposed solution to this problem consists in predicting the number of coupons likely to be sold for each deal in the catalog – a task referred, from now on, as deal size prediction. We study the existence of complex interactions among deals when predicting deal size, raising the following research questions related to the Deal Selection Problem:

RQ1 How can we predict deal size considering Market Interplay and Intra-Market Competition?

In DDS, a Market is represented by a group of deals offering products or services that attract the interest of a similar set of customers, such as "Restaurants" or "Hair Services". Market Interplay identifies deals that belong to different markets but can positively influence the sales of each other. For example, catalogs featuring deals on eye ware and services of an eye exam may influence customers in buying both deals simultaneously. Market competition, in turn, happens inside the same market, when two merchants selling a coupon for the same or similar products have a negative impact on the sales of one (in this case, it will depend on the budget of the customer and the prices of the deals) or both, as customers might find it difficult to pick one and end up buying none.

It is hard to tackle these problems when predicting the deal size without taking the market into account . Hence, our strategy is to separate deals into markets. Note that markets here differ from the predefined categories of products defined by the DDSs. When a merchant wants to feature a deal, he needs to classify the deal within a category, even if it is "Other". Markets can be seen as a more flexible set of categories.

RQ2 How can we identify Markets?

We propose a method to separate deals into markets assuming that deals within the same market are likely to be described using similar terms. For instance, offers associated with "Gyms" may be described using terms such as "class", "fit", and "body", while offers associated with "Hair Services" may be describe using terms such as "hair", "salon", and "look".

Given that, we first assess the descriptive and discriminative power of deals textual content by using heuristics previously published in the context of indexing Web pages and classifying social media. We then use a probabilistic topic model, namely *Latent Dirichlet Allocation* (LDA), to discover markets.

Given that we are able to find meaningful markets of deals, we then focus on modeling the whole deal size prediction problem as a set of sub-problems. In particular, we propose to build a different prediction model for each market by addressing the following question:

RQ3 How Markets Identification help to model Deals Interplay and Competition?

After identifying markets, we proposed to use multiple specialized *Support Vector Regressor* (SVR) predictors, where each predictor is dedicated to a market. Note that,

the specialized SVR predictor outcomes the deal size of an individual deal. To couple with the complex interactions among deals, we propose an expectation-maximization algorithm approach to model market interplay and intra-market competition while minimizing the prediction error.

Figure 1.1(b) illustrates the second problem investigated in this thesis is related to Email Prioritization. In particular, we deal with customer satisfaction by focusing on the emails sent to customers alerting them to products and services with big discounts for a limited time. In this scenario, we deal with two different problems: (i) which and how many customers should receive the alerts and (ii) which deals should compose the emails. From these two problems, the following research question arouse:

RQ4 How can we identify customers with a high probability of clicking on email deals?

This research question is related to the customer prioritization problem (Fig. 1.1 (b)). The customers with high probability of clicking offers should be the first to receive emails. There are many evidence that can give hints of whether a customer will click the offer or not, and they can be used as sorting criteria to email delivery. First, we identify some of these evidence, including the number of past purchases, days since last purchase, customer similarity with the active catalog, and sort customers according to them. Second, we model the task of determining the best sorting criterion as a reinforcement learning problem. Specifically, we use Multi-Armed Bandits (MAB), which is a special class of sequential optimization problems derived from the more general paradigm of Reinforcement Learning [Sutton and Barto, 1998]. Looking at the results obtained with the algorithm, we can choose the best trade-off between the number of emails sent and number of customer clicks.

Fig. 1.1(c) illustrates the Deal Recommendation problem, which is related to the content of each email. Regarding this problem we raise the following research questions:

RQ5 How can we suggest more appealing deals?

State of the art collaborative filtering algorithms do not perform well in DDS scenarios due to the three challenges previously mentioned: (i) most of the customers of a DDS are sporadic bargain hunters, and thus past preference data is extremely sparse and noisy; (ii) DDSs work with daily catalogs that are very dynamic, and hence data is extremely volatile and scarce; (iii) customer taste and interest may undergo temporal drifts.

In this thesis, we propose new methods to address these challenges based on a simple idea: *get customer feedback on the active catalog as soon as possible.*

RQ6 How can we get customer feedback about current catalog?

In order to collect feedback from customers, we model the relationship among customers as a co-purchase graph, where nodes represent customers and edges a common deal purchased in the past. Based on the network, we use centrality metrics to sort customers that are more likely to provide feedback and share similar tastes with other customers.

The recommendation process is modeled as a sequential optimization problem, and again we use the Multi-Armed Bandit setting to sequentially run the recommendations. However, instead of modeling the arms as sorting metrics, we model the arms as customers. Customers are ordered according to a single criterion, which initially could have been any of those previously tested. However, one important part of the method is to augment the network to rerank deals. Hence, the probability of clicking the email has to be as high as the "influence" of the customer in the network, better represented by centrality metrics.

1.2 Thesis Contributions

All contributions of this thesis are strengthened by the fact that all experiments were performed in real-world datasets of DDSs. The main contributions of this thesis are:

Contributions in Deal Selection:

- An extensive analysis of the textual features of three datasets from world known DDS, namely, (i) Groupon[1], (ii) Living Social[2] , and (iii) Peixe Urbano[3].
- A content-based strategy to separate deals into markets that exploit the structure of each deal.
- A method for deal size prediction that takes into account both market interplay and intra-market competition.
- A systematic set of experiments showing that our solutions offer gains in precision ranging from 8.18% to 17.67% when compared against existing solutions.

[1]http://www.groupon.com
[2]http://www.livingsocial.com
[3]http://www.peixeurbano.com

Contributions in Email Prioritization:

- A framework for the assessment of the click rate on emails at a sequential personalized email recommender system.
- An empirical evaluation of the trade-off between the number of emails sent to customers and the corresponding click rate.

Contributions in Deal Recommendation:

- A characterization of a real co-purchase network, in which the nodes represent customers and edges connect customers that bought at least one common coupon in the past.
- Two sets of distinct criteria to provide an order to send personalized emails to customers.
- Two exploration-exploitation strategies for the problem of selecting the most relevant offers to put within emails by exploiting feedback from customers.
- An empirical evaluation of our algorithms using real data obtained from a large daily deals website show gains in precision ranging from 7.9% to 34.0% in contrast to state-of-the-art recommendation algorithms.

Time Performance:

- In deal selection, the key steps of our solution are computed off-line.
- In email prioritization, we train the recommender off-line and compute the deal recommendations online.
- In deal recommendation, the most costly steps are computed off-line (i.e., sorting customers and training the recommender) and only the email recomendations are built online.

1.3 Thesis Outline

The remainder of this thesis is organized as follows:

- Chapter 2 is divided in two parts, and presents background and related work on both Daily-Deals Sites and Recommender Systems. The first part introduces

basic concepts on the DDS business model and a review of the relevant academic literature. This discussion is followed by a definition of the deal size prediction problem and previous work on the matter. Next, a characterization of a real world DDS, namely Peixe Urbano, is performed. This dataset is used throughout this thesis. The second part presents the main tasks related to Recommender Systems and a taxonomy based on the nature of data used to build the recommendation. It also describes how the recommendation problem can be modeled as a Multi-Armed Bandit problem (MAB). In particular, related work on MAB recommendation algorithms is discussed.

- Chapter 3 shows the method proposed to tackle the deal size prediction task. In particular, we discuss the descriptive and discriminative power of deal textual features, and present other features associated with deals, and also discuss the approach proposed to separate deals into markets. Furthermore, we detail our solution to infer deal size that is modeled as a machine learning regression task. Our solution takes into account both the market interplay and the intra-market competition when predicting the size of a deal. We finish this chapter by presenting the performance of our method in contrast to state-of-the-art methods.

- Chapter 4 shows a framework to identify the best order to send personalized emails to customer to maximize the click through rate. We first present a set of intuitive criteria and then detail our MAB-based framework to sequentially decide the best criteria. Finally, we present a detailed experimental analysis of our solution.

- Chapter 5 presents our approach to improve deals recommendations sent to customers by exploiting preference feedback. We start discussing exploration-exploitation strategies. Next, we present our solution to split customers in the sequential recommendation problem into two groups, i.e., those that will participate during the exploration and exploitation phases. We also discuss how the recommendations sent are changed according to customer feedback provided in the exploration phase. We finish this chapter presenting a discussion on the performance of our approach in contrast to state-of-the-art recommender systems.

- Chapter 6 presents a summary of our contributions and points out possible directions of future research.

Chapter 2

Background and Related Work

This chapter is divided into two parts. The first part presents the Daily-Deal ecosystem by detailing each player, namely, Daily-Deal Site, Merchant, and Customer, and how they interact with each other. It also reviews the literature on DDS, emphasizing the deal size prediction problem, which refers to estimating the number of coupons sold by a deal. The first part finishes with a description of a real world DDS purchase dataset, i.e., Peixe Urbano, used in the experiments throughout this thesis. The second part is related to Recommender Systems, and shows a taxonomy based on the nature of data used to build the recommendation. It also describes how the recommendation problem can be modeled as a Multi-Armed Bandit (MAB) problem and reviews the literature on MAB recommendation algorithms.

2.1 Daily-Deals Sites

The daily-deals business[1] is a group-buying concept based on a synergistic view: helping local-business to advertise their products and services while providing customers significantly discounted offers (aka. deals). The daily-deals concept has been around for almost a decade, and started gaining popularity in July 2004 with the launching of Woot.com.[2] Woot originally presented one offer a day, which remained active until either it was sold out or 24 hours elapsed. In early 2010, Woot was acquired by Amazon and its model changed in a way that, if a product sold out fast enough, a new product replaced it immediately.

[1] In the literature, daily-deals are also referred as collective buying, group buying, deal-of-the-day, one deal a day, flash sales, and online discount vouchers.

[2] http://www.woot.com

Following this trend, in 2008 Groupon entered the market and became the fastest online company to reach a billion-dollar valuation [Donnelly, 2012]. LivingSocial was launched in 2009, and nowadays the two companies are the top leaders in the daily-deals business in the US. In Brazil, Peixe Urbano has been launched in 2010 and is the leader in the daily-deals segment in the country.

Most of daily-deals sites (DDSs) work directly with local merchants and online retailers to offer products and services with significantly discounted prices when compared to recommended retail prices. Typically, a minimum and maximum number of deals are made available, and initially offered for 24 hours. DDSs are currently changing this strategy and increasingly offering longer deal buying periods to increase sales performance, besides allowing more than one deal to run in a single market at the same time (i.e., two merchants offering the same or very similar products at the same time). Common deals negotiated at DDSs include, for example, health and fitness, restaurants and bar, salons and spas related products.

In this model, instead of purchasing an offer directly from the retailer, customers purchase the product or service from the DDS, which retains customer data. Once the minimum number of deals has been sold, i.e. the deal tipping point has been reached, the DDS charges the customers' credit card and the deal is sent as an electronic voucher redeemable at the retailer or service provider. The promotional value of the vouchers has an expiration date, and after that period the value can be discounted from the deal original price or is lost, depending on the country policy.

An online coupon is often used as a key marketing and advertising technique. In traditional internet advertising, which includes sponsored search, contextual ads and branding ads, the ad network will charge advertisers even for a few number of conversions (for a survey on Internet Advertising see Evans [2008]). In contrast with this pricing model, DDSs receive their payment only upon satisfying the minimum number of conversions before the deal expiry (i.e. when the deal reaches the tipping point). As a consequence, if the deal does not tip, the DDS receives no payment and misses the opportunity of advertising other offers from the same merchant, losing revenue. Otherwise, if the deal exceeds the minimum limit, the DDS receives a payment that is given by the product of the number of conversions and fee per transaction, which is similar to the computational advertising model.

In Figure 2.1, we present a schematic view of the Daily-Deals Ecosystem. There are three main players:

- *Daily-Deal Site* provides a mechanism that enables merchants to promote their products or services to customers. The DDS acts as an intermediary, selecting

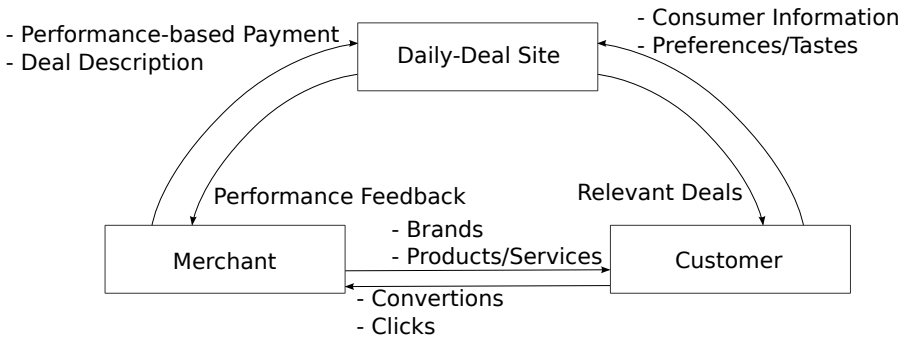


Figure 2.1: The Daily-Deal Ecosystem.

the most expected profitable offers from local businesses and sending them to customers using a variety of delivery channels, such as email, websites, social networks, affiliate programs, and internet advertising. The revenue of the DDS is proportional to the number of coupons sold and the fee associated with each deal.

- *Merchant* requires spaces to place its discounted offers on delivery channels of DDSs. The potential advantage of offering deals on DDSs is large-scale advertising. Discounted goods can benefit merchants through advertising, by exposing them to new customers and allowing an online "buzz" [Byers *et al.*, 2012b]. The merchant is charged only if his deal tips, i.e., if the minimum number of deals is sold.

- *Customer* creates a profile in a DSS, containing his/her geographical location, email account and possibly topics of interest, which are used to build his/her preferences. This profile, the current catalog, the geographical information, among other factors, may serve as key evidence to determine which deals a customer will receive. Note that the exact method is not unique and may vary from one DDS to another.

2.1.1 Research Problems in DDSs

A literature review on problems related to DDSs shows that most works focus on analyzing their business model and its advantages and drawbacks to the three main players, namely, DDSs, merchants, and customers. Works discussing computational problems related to DDSs are much more scarce, mainly due to lack of data. Although Groupon and LivingSocial have Application Programming Interfaces (API) to access data, all researchers can get is what is available on the website. Richer information, such as the purchase history, are not publicly available. For this reason, most relevant

research questions about DDSs computation problems need to be simplified due to lack of data. One of the main advantages of this work is that, as we have access to the purchase history, a richer and more sophisticated analysis of DDSs problems can be performed.

Having said that, this section starts discussing two computational problems that have been investigated in this domain, namely revenue maximization and the propagation effect of deals in social networks. Regarding revenue maximization, deal size prediction is the main problem discussed, and it is the first step to create a deal catalog. It corresponds to the first problem studied in this thesis, and is discussed in detail in Section 2.1.2. In short, given a set of candidate deals, when one wants to create a single-day catalog, deal-size is used to help in the deal selection problem. When various days and the deals interactions among these days are considered, deal size prediction helps to solve the deal scheduling problem [Lappas and Terzi, 2012].

Concerning the propagation effect of deals in social networks, Byers *et al.* [2012a] assessed the impact that a deal has on the merchant's subsequent ratings in social review sites such as Yelp. In this first work, the authors observed that a negative side effect for merchants offering deals at Groupon was that, on average, their Yelp ratings decline significantly.

In a subsequent work [Byers *et al.*, 2012c], they revise their conclusions and suggested that reviews from Groupon customers are lower on average because they correspond to real, unbiased customers. In contrast, the general body of reviews on Yelp contain a fraction of reviews from biased and even potentially fake sources.

Park and Chung [2012], in turn, studied how deals propagate through the Twitter microblogging service. They concluded that daily-deal sharing on Twitter happens most often in the morning and around the middle of the week, and the deals offered in multiple locations tend to be shared more frequently. They also provided evidence that sharing deals in Twitter improves the deals sales performance, and they named this phenomenon eWOM (Electronic Word Of Mouth).

Apart from the computational problems, as already mentioned, a variety of works focus on the DDS business model, discussing their effectiveness, identifying possible drawbacks, and suggesting ways to improve its sustainability. They also study effective marketing strategies and customer behavior. Although these works are not the main focus of this thesis, understanding the nature of DDSs may give us insights to create solutions to the computational problems discussed later.

From a business perspective, Byers *et al.* [2011] examined the business model of Groupon and presented evidence that it is behaving strategically to optimize deal offerings, giving customers incentives other than price to make a purchase, e.g., deal features, limited inventory, and deal duration.

As this industry matures, Byers *et al.* [2012b] raised questions about the health of DDSs business model, focusing on understanding the sustainability and long-term outlook of DDSs. They concluded that using customer demographic information as evidence to better target consumers to deals and offering more appropriate incentives for salespeople are crucial for the health of business.

Regarding the merchant, one line of study investigates and identifies the pros and cons of using DDSs as marketing channels. In Kumar and Rajan [2012], the authors studied two aspects of using coupons as a marketing strategy: (i) whether they provide short- and long-term profitability, and (ii) whether they bring new customers for the business. They concluded that coupons yield profits for the business and also discussed that coupons ensure customer acquisition in specific business categories. In Farahat *et al.* [2012], in turn, the authors were interested in understanding the factors that make merchants repeatedly run deals over time.

An empirical analysis of the experience of businesses that used Groupon to promote their goods was carried out on Dholakia [2010]. They conducted a survey-based study of 150 businesses that had been run Groupon promotions between June 2009 and August 2010. They reported that the promotion was profitable for 66% and unprofitable for 32% of respondents. They also presented evidence that the treatment given by merchant's employees to customers is the most important factor for the success of a Groupon promotion.

Finally, a distinct line of work focuses on understanding customers acceptance of DDSs and customer loyalty. The authors in Krasnova *et al.* [2013] studied customers behavior on DDSs. In particular, they addressed the determinants of customer loyalty. Their main finding was that customer loyalty is mainly driven by, as expected, monetary incentives, i.e., the opportunity to gain better quality and save money. However, they also stated that the customer loyalty can be promoted at the presence of benefits associated to deals, such as novel and exclusive goods.

In Li and Wu [2013], the authors investigated shopping behavior in DDS. In particular, they identified the exposure of the total number of vouchers sold in real-time as a key factor that influences new buyers to make a purchase. They also suggested that messages about deals sent throughout social network websites, such as Facebook and Twitter, generate a word-of-mouth effect that influence purchase decision.

2.1.2 Predicting Deal Size

The deal size prediction problem refers to inferring the expected number of coupons sold by a deal. The problem was first proposed in Byers *et al.* [2012a] and then extended in Lappas and Terzi [2012], Ye *et al.* [2012].

Among other contributions, such as considering how "soft incentives" affect the revenue of Groupon and LivingSocial, Byers *et al.* [2012a] studied the predictability of the deals size using deals features. In particular, they modeled the deal size prediction problem as a linear combination of deals features and used ordinary least squares regression to fit their model. The main difference between Byers *et al.* [2012a] and our model is that their predictor is learned from the whole catalog with no distinction of business markets. Hence, they do not take into account market interplay and competition. Our approach employs multiple predictors, where each predictor is specialized to predict the size of deals belonging to a given market. In a second phase, we combine these individual models. We used their model as a baseline of comparison in our experiments.

In Lappas and Terzi [2012], the authors used a regression method to estimate the deal size. They also learn a specific model per market and employ the following features: (i) deal location (i.e., city or area for which the deal is available), (ii) business reputation (i.e., global traffic rank and website's reputation from Alexa[3]), (iii) the price of the deal, (iv) the price after the discount, (v) the tipping point of the deal, (vi) seasonality (i.e., period of the year when the offer is made available). Further, the authors proposed to extract terms from deal description and used a hierarchical clustering algorithm that is a combination of LDA [Blei *et al.*, 2003] and a flat clustering algorithm based on Kullback-Liebler distance [Pinto *et al.*, 2007] to separate deals into markets. Different from their work, we analyzed the deal structure to determine the best set of descriptive/discriminative terms and to assess a proper weighting scheme for these terms. This work is also used as a baseline in our experiments.

Ye *et al.* [2012] presented an investigation of the group purchasing behavior of daily deals in Groupon and LivingSocial, and introduced a predictive dynamic model of deal size prediction. Their focus was inferring the popularity of deals as a function of time, i.e., they were interested on sequential prediction of deal size by assuming that the deals are available to the predictor as streaming data. A key difference from our deal size predictor is that we infer deal size at the moment that deals are included in the catalog. As expected, we measure our prediction error at the moment the deal expires, by computing the difference between real deal size and predicted deal size.

2.1.3 Peixe Urbano: Understanding the dataset

The experiments were performed using a real-world dataset obtained from Peixe Urbano, the largest DDS in Brazil. The dataset comprises a sample of a 2-month period

[3]http://www.alexa.com

with 455 deals. Although we did have access to a bigger dataset, a preliminary study showed that purchases tend to follow a pattern within a given month. The results of these experiments cannot be shared due to confidentiality, but the sample gave us flexibility to perform a much larger set of studies. In the method that use this dataset, 31 days were used for training and the next 30 for test.

However, a dataset larger than the one described above was used in one experiment: deal size prediction. For this experiment, two public datasets were already available for the task. In order to make Peixe Urbano comparable to these two, we increased the time period for one year (2012). In this second version of the dataset, we have a sample that contains 4,309 deals.

In Figure 2.2, we illustrate why DDS scenarios present a big challenge to current recommendation algorithms. We present, for each testing day, the normalized volume of new deals and On average, 30% of deals arrive to the catalog every day, and remain valid, on average, for 4 days.

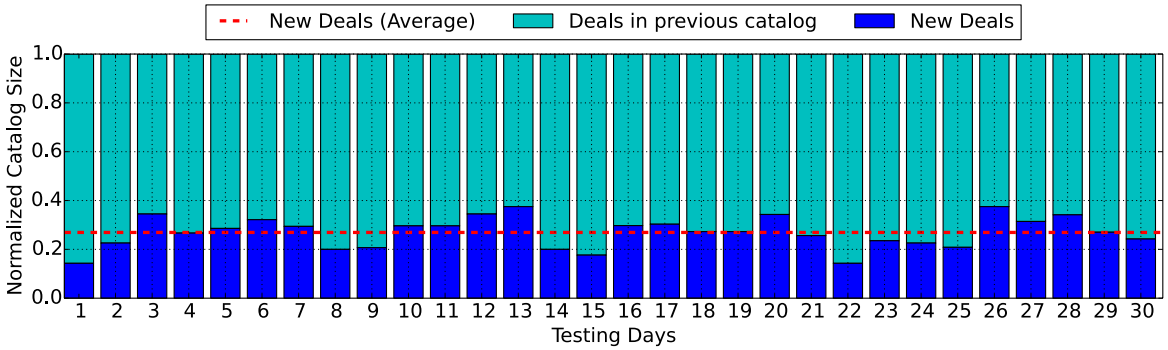


Figure 2.2: Dynamic nature of the catalog: number of old and new deals for each test day.

In Figure 2.3, we show the Cumulative Distribution Function (CDF) of purchases. We see that more than 50% of the customers purchased only one deal, while around 80% of the customers purchased at most 2 deals. On average, each customer purchases 1.36 deals, making data extremely sparse if compared with more studied recommendation scenarios, such as NetFlix (with 208 ratings per user) and MovieLens (with 166 ratings per user).

From Peixe Urbano we obtained data using an API that was made available to us. Textual features are composed of terms in Portuguese, and the 4,309 deals were expressed using 31,163 distinct terms. In Chapter 3, we present an extensive analysis of the descriptive and discriminative power of textual features present in deals for Peixe Urbano used in the experiments that validates our deal size prediction method.

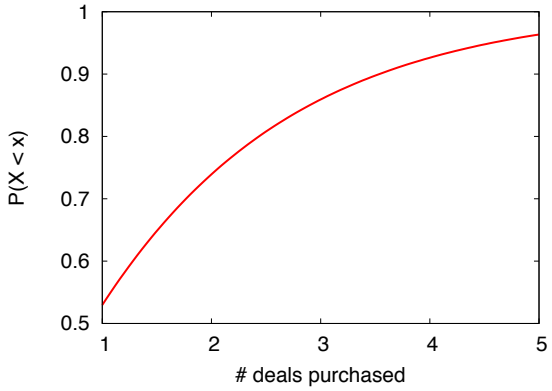


Figure 2.3: (Color online) Cumulative Distribution Function.

2.2 Recommender Systems

The rapid growth to the content available on the Web and the emergence of e-commerce has led to the development of recommender systems [Resnick and Varian, 1997]. A recommender system is a personalized information filtering technology that pro-actively suggests items to customers based on explicitly declared preferences, e.g., ratings, or implicitly observed actions, e.g., purchase history. Recently, recommender systems have been used in a variety of applications, such as to suggest movies [Choi *et al.*, 2012], TV programs [Barragáns-Martínez *et al.*, 2010], or music [Aizenberg *et al.*, 2012] a customer will find enjoyable; and to identify news of interest [De Francisci Morales *et al.*, 2012], or products to buy [Schafer *et al.*, 1999].

In this thesis, we are interested on a recommendation task that uses email as a delivery channel to suggest a list of relevant deals (items) to target users (customers). Formally, the recommendation function $\delta : \mathcal{C} \times \mathcal{D} \rightarrow \mathfrak{R}$ maps pairs of customer/deal to real numbers indicating the relevance of this deal to the customer, where $\mathcal{U}$ is the set of customers, and $\mathcal{D}$ is the set of deals.

The success of email recommendation has been already experienced at Groupon. In September 2013, they reported that direct email is the delivery responsible for almost 40% of all transactions in their largest market, i.e., North America [Groupon, 2013]. They also reported that email personalization has driven a nearly 30% lift in the North America email purchase rate [Groupon, Inc, 2012], and in markets with high deal density, such as Chicago, personalized emails have a 50% higher purchase rate [Groupon, 2012].

In Figure 2.4, we present an overview of the recommendation process in terms of inputs and outputs. As the input, we may have a rating matrix that contains the preferences of customers for items as a rating number in a given scale, e.g., stars in the

$[1, 5]$ scale. Note that if we have only implicit feedback, such as the purchase history, we use a binary matrix, where 1 entries refer to items previously bought and 0 entries to items not bought. A second type of entry is related to items content. Specifically, given a set of features $F = \{f_1, f_2, \ldots, f_n\}$, each item is represented by F. As an example, the items could be represented by the words that describe them, and the F set would be the words vocabulary.

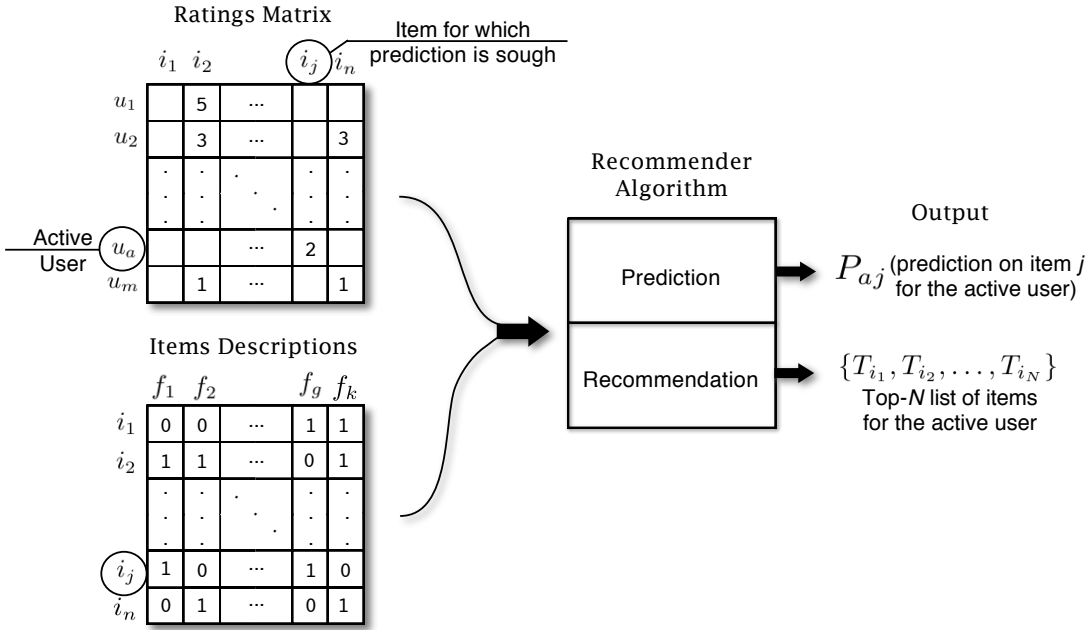


Figure 2.4: The recommendation process.

Regarding the output of Recommender Systems, they are used to either predict whether a particular customer will like a particular item (*prediction problem*) or to identify a set of N items that will be of interest to a certain customer (top-N *recommendation problem*) [Deshpande and Karypis, 2004]. The focus of this thesis is on a particular class of $top-N$ recommendation algorithms that build the recommendation model by analyzing the similarities between the various customers using their purchase history, as detailed in the next section.

2.2.1 Categories of Recommendation Algorithms

Regardless of the recommendation task being considered, recommendation algorithms can be classified as non-personalized and personalized. The most common and simple non-personalized estimator is the Most Popular (aka. TopPop – Top Popular) algorithm, which suggests the most popular items to any customer, irrespective to his or her tastes. Typically, this algorithm serves as baseline to more sophisticated personalized algorithms.

Personalized recommender systems algorithms are classified in one of three categories [Adomavicius and Kwon, 2009]: (i) content-based, (ii) collaborative-based, and (iii) hybrid approaches. In the following sections we detailed each category considering their pros and cons in the DDS domain.

2.2.1.1 Content-based Approaches

In content-based recommender systems, the customer receives items similar to the ones he preferred in the past. In particular, content-based methods estimate the utility $u(a, i)$ of an item i for target customer a using the utilities $u(a, i_k)$ assigned by customer a to items $i_k \in I$ that are "similar" to item i. In the context of DDSs, for instance, in order to suggest deals to customer a, the content-based recommender system tries to understand the commonalities among the deals customer a has purchased in the past (specific merchants, demographic information, terms in deals description, etc.). Hence, only the deals that have a high degree of similarity to customer's tastes are suggested.

Content-based algorithms are not suited to our scenario. In our application, namely personalized email recommendation, there is a high percentage of customers for which we have a poor profile, i.e., almost no information from past purchases. This happens for two reasons. First, as presented in Dholakia [2010], almost 80% of customers in DDSs are new. Second, customers are able to register on DDS to receive recommendations presenting just their email address and, again, in this case we have no information about their tastes. Another weakness of content-based approaches is over-specialization [Adomavicius and Tuzhilin, 2005, Barragáns-Martínez *et al.*, 2010]. Since most of the customers in DDS bought just one deal in the past, content-based approaches would restrict customers to receive recommendations of deals similar to those few already experienced.

2.2.1.2 Collaborative-filtering Approaches

The basic assumption of collaborative filtering approaches is that customers who have similar preferences in the past are likely to have similar preferences in the future. Typically, Collaborative-Filtering (CF) techniques involve matching the preferences of the target customer with the preferences of similar customers (nearest neighbors) to produce suggestions of items not yet seen by the target customer [Adomavicius and Tuzhilin, 2005]. Note that here there is no need of customer profiles (e.g. location, age, etc) nor contents metadata (description, price, etc), making CF domain independent strategies.

Traditionally, CF systems consider explicit feedback from customers as their preferences, e.g., the number of stars a customer gives to a movie [Bennett and Lanning, 2007]. This type of data is unavailable in DDS scenarios, and we focus here on implicit feedback strategies. Implicit feedback CF strategies have been proposed in variety of scenarios [Oard *et al.*, 1998, Kelly and Teevan, 2003, Rendle *et al.*, 2009, Davidsson and Moritz, 2011]. We look into details on the works of Hu *et al.* [2008] and Shi *et al.* [2012], given that both strategies are representative of the state-of-the-art algorithms when only implicit feedback is available.

The Weighted Regularized Matrix Factorization (WRMF) algorithm is a matrix factorization model for item suggestion based on implicit feedback data [Hu *et al.*, 2008]. WRMF is formulated as a regularized Least-Squares problem, in which a weighting matrix is used to differentiate the contributions of observed interactions (i.e., positive feedback) and unobserved ones. Shi *et al.* [2012], in turn, proposed Collaborative Less-is-More Filtering (CLiMF). The algorithm aims at optimizing Reciprocal Rank (RR) for top-N recommendation in domains with binary relevance data. In these domains, the interaction between customers and items is stored using a single bit, i.e., a bit "1" when the interaction exists and "0" otherwise.

Regarding CF strategies for recommendation in DDSs, the authors in Ebrahimi *et al.* [2012] claimed to have addressed the problem. However, as they did not have access to a real purchase dataset, they simulate a daily-deal dataset using a subset of the Yelp Academic Dataset [Yelp, 2012]. In particular, they selected businesses from Yelp that have run a deal on Groupon, excluding all other business. Since Yelp customers provide ratings to business, they use these ratings as ground-truth data and information from profile of Yelp customers to enrich a traditional CF strategy [Herlocker and Konstan, 2002]. Note that explicit feedback like ratings are unavailable in all DDSs considered in this thesis, namely, Groupon, LivingSocial, and Peixe Urbano.

The authors exploit context information about the customer's product or service preferences to predict daily-deal categories relevant to customers as follows. First, they identify similarities among customers using the Pearson Correlation Coefficient, and consider the value 0.7 as the threshold that defines positive correlation between customers. Second, their algorithm aggregates the ratings to product or service categories given by similar customers to the one who will receive the recommendation. This information is used to predict the rating of the corresponding product or service category according to this customer. Their algorithm recommends categories with predicted ratings equal to or greater than 4. Other algorithms for DDS recommendation are described in Section 2.2.2, as they use the multi-armed bandit strategies explained in the next section.

The first difference between our recommendation strategy and Ebrahimi *et al.* [2012] is that they are interested in suggesting categories of daily-deals to customers. Another difference is that they model the recommendation task as a rating prediction problem, whereas in our work we are interested on building a ranked list of deals. Finally, they assume context-based features, e.g. deal category, which are unavailable in our dataset.

A particular problem that CF approaches are unable to handle is the arrival of new items, i.e., new items are regularly added to recommender systems. Since CF algorithms rely on customer's preferences to make suggestions, the system will not be able to recommend an item until it is purchased by a substantial number of customers. This problem is aggravated in DDS recommendation because, behind the fact that new items are daily added to catalog, a considerable portion of items are also removed from catalog on a daily basis.

2.2.1.3 Hybrid Approaches

In Table 2.1, we present a characterization of hybrid methods, as proposed by Burke [2002a]. A variety of recommendation algorithms use a hybrid approach by combining collaborative filtering and content-based methods, which helps to avoid certain limitations of content-based and collaborative systems [Barragáns-Martínez *et al.*, 2010, Burke, 2002b]. Besides the aforementioned characterization of hybrid methods, Burke [2002a] also presented empirical comparison with pure strategies, i.e., CF and content-based strategies. These strategies include refining recommendations from other algorithms or switching from one recommender to another according to some criteria. Adomavicius and Tuzhilin [2005] also discussed different ways to combine the aforementioned techniques into a hybrid recommender system.

2.2.2 Recommendation as a Multi-Armed Bandit Problem

A multi-armed bandit (MAB) algorithm is a sequential decision making process under uncertainty. Bandit problems involve taking a decision in each timestep t, e.g., deciding which customer should receive an email or which news article should be recommended to a new website visitor. MABs have been studied by statisticians for a long time [Robbins, 1952]. Recently, this type of algorithms have been used in recommendation scenarios such as news articles [Li *et al.*, 2010a], advertisement [Chakrabarti *et al.*, 2008], and movies [Zhao *et al.*, 2013]. The following sections present an overview of MAB literature (Section 2.2.2.1), the MAB algorithms relevant to our work (Section 2.2.2.2), and MAB recommendation algorithms (Section 2.2.2.3).

Table 2.1: Hybridization Strategies for Recommender Systems

Hybridization Strategy	Description
Cascade	Final recommender refines the recommendations given by another.
Weighted	The scores of many recommenders are combined together to produce a single recommendation.
Switching	The algorithm switches between recommender methods depending on the current situation.
Mixed	Recommendations from a variety of distinct recommenders are presented at the same time.
Feature combination	Features from different recommendation data sources are combined into a single algorithm.
Feature augmentation	Output from one algorithm is used as an input feature to another.
Meta-level	The model learned by on recommender is used as input to another.

2.2.2.1 The Multi-Armed Bandit Problem

Multi-armed bandit problem is a special class of sequential optimization problems derived from the more general paradigm of Reinforcement Learning [Sutton and Barto, 1998] and were introduced in the seminal paper of Robbins [1952]. The paradigm of Reinforcement Learning (RL) is situated in between supervised learning and unsupervised learning, and deals with learning in sequential decision making problems where there is limited feedback [Kaelbling *et al.*, 1996].

A key aspect that differentiates RL from other machine learning paradigms is that the learning process is performed as a trial and error strategy [Sutton and Barto, 1998]. Hence, RL should not be seen as a specific class of learning methods, but rather as a learning *problem* or a *paradigm*. In Figure 2.5, we illustrate the basic cycle of Reinforcement Learning paradigm. This class of algorithms allows an *agent* to interact with an unknown environment over a series of timesteps, observing the current *state* of the environment, taking *actions*, and receiving as feedback only a *scalar* reward signal.

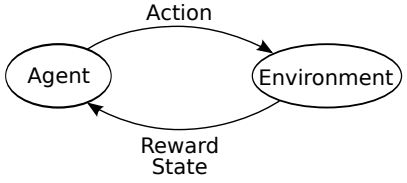


Figure 2.5: The Reinforcement Learning Paradigm.

In a variety of complex domains, RL is the only feasible way to train a program to perform at high levels. For instance, if we want to build a machine that learns to play chess. It would be unfeasible to use a supervised learner for two reasons: (i) first, the cost to have a teacher that take us through many games and point out the best move for each position is prohibited, (ii) second, in many situations, there is no such thing as the best move; how good is a movement depends on the moves that follow. The goal of the agent in this context is to perform actions that maximize *cumulative reward*, i.e., the reward signal in the long run.

In this thesis we are particularly interested in the Multi-Armed Bandit Problem (MAB) [Robbins, 1952]. Historically, the name "bandit" comes from an imaginary gambler playing a K slot machine in a casino. The gambler may pull an arm from any of the slot machines. Each time an arm is pulled, a random reward, independent from any previous rewards, is returned. The arms are also assumed to be independent from each other. The gambler objective is to accumulate as much reward as possible in the long run.

Formally, a multi-armed bandit setting is described as: Let K refer to the number of arms that the decision-maker can pull. At each time step t, the agent pulls arm i_t and an associated reward r_{i_t} is returned. The rewards are drawn from an unknown distribution of arm i_t. The process repeats for a horizon $T > 0$ until it ends. The decision-maker objective is to maximize the sum of rewards.

For instance, let us consider the problem of determining the best order to send personalized recommendations to customers modeled as a MAB. Here, we have K possible distinct criteria to order customers (arms) and we want to discover the best one. In order to send an mail, the MAB algorithm needs to choose a criteria. For each email sent, an associated positive reward is returned if there is a click, or no reward is returned if the customer ignores the email. The process repeats for a maximum horizon equals to the total number of customers. Finally, the objective is to maximize the click through rate on recommendations.

As already explained, a multi-armed bandit algorithm, also called a multi-armed policy, is a strategy that determines the sequence of arms chosen in order to deliver a maximal reward at each time step t, and, ultimately, maximizes the cumulative reward. However, MABs search for the best trade-off between learning about the options (exploration) and using the current knowledge to make the best choice (exploitation). In order to maximize the cumulative reward in the long run, the key challenge in bandit problems is the need for balancing exploration and exploitation, also known as the *Exploration/Exploitation Dilemma*.

2.2.2.2 MAB Algorithms

Multi-Armed Bandit setting has been used to tackle the exploration-exploitation problem in machine learning and statistics [Berry and Fristedt, 1995, Jun, 2004, Audibert *et al.*, 2009, Bubeck and Cesa-Bianchi, 2012]. Traditionally, the focus of MAB research is on context-*free* algorithms, which model situations where no side information is considered and the reward depends only on the action taken. In contrast, contextual multi-armed bandit algorithms are strategies that take *context* (side information) into account to choose an action from a set of possible actions to maximize the total rewards of chosen actions. In this section we focus on context-free algorithms.

A simple and widely used algorithm for MAB problem is the ϵ-greedy algorithm [Thompson, 1963], which receives a parameter ϵ responsible for balancing the exploration and exploitation phases. At each time-step $t = 1, 2, \ldots$, the algorithm follows a greedy approach (i.e., it chooses the arm with the highest estimate) with probability $(1 - \epsilon)$, and it chooses a random arm with probability ϵ. More formally, given the initial empirical means $\hat{\mu_1}(0), \ldots, \hat{\mu_K}(0)$,

$$p_i(t+1) = \begin{cases} 1 - \epsilon + \epsilon/K, & \text{if } \arg\max_{j=1,\ldots,K} \hat{\mu}_j(t) \\ \epsilon/K, & \text{otherwise} \end{cases}$$

where $p_i(t)$ stands for the probability of selecting arm i in round t. Cesa-Bianchi and Fischer [1998] presented several variants of ϵ-greedy in which the value of ϵ decreases over time. In an earlier empirical investigation, Vermorel and Mohri [2005] did not find any practical advantages in varying ϵ. Therefore, in the methods proposed here we consider only fixed values of ϵ.

A special scenario of MAB is the budget-limited multi-armed bandit problem, first presented in Guha and Munagala [2007] and then generalized in Tran-Thanh *et al.* [2010]. It assumes that arms actions are costly, and constrained by a fixed-budget. Since the budget is predefined, the problems' duration is finite, and the ideal exploitation is not obtained by pulling the optimal arm repeatedly, but combinations of arms that maximize the reward within the budget. Because of that, the reward for all arms must be estimated, as any of them can appear in the optimal combination.

A common strategy to tackle the budget-limited multi-armed scenario is the ϵ-first [Tran-Thanh *et al.*, 2010]. In this strategy, the first ϵ budget B is dedicated to exploration, and the remaining $1-\epsilon$ to exploitation. Hence, exploration and exploitation are done in separate phases, i.e., only when exploration finishes exploitation starts. The optimal solution of the budged learning problem is NP-HARD even when at most one play is allowed per arm [Madani *et al.*, 2004], and when the cost to play an arm is unitary [Goel *et al.*, 2006].

Overall, greedy MAB algorithms typically perform well in a variety of applications [Sutton and Barto, 1998, Vermorel and Mohri, 2005, Kuleshov and Precup, 2010] when the running time horizon is finite. A drawback of the $\epsilon-$greedy algorithm is that when it explores it chooses equally among all criteria. There is no explicit preference for specific choices, and the worst-appearing criterion or the next-to-best are equally likely to be explored. Hence, other MAB techniques, which also focus on theoretical optimal-convergence guarantees, were introduced in Auer *et al.* [2002].

The authors presented a class of MAB algorithms called *upper confidence bound* (UCB) that are a simpler and more elegant implementation of the idea of optimism in the face of uncertainty, proposed by Lai and Robbins [1985]. There are many algorithms that implements UCB-based strategies. For instance, in Auer *et al.* [2002], the authors presented the UCB1 and UCB2 algorithms that assumes arbitrary set of reward distributions.

In this thesis, we use the UCB–Normal algorithm proposed in Auer *et al.* [2002], which assumes normally distributed reward distributions with unknown mean and variance. From now on, we refer to it just as UCB. The algorithm chooses criterion c_i if, until time step t, the corresponding arm has been played less then $\lceil 8 \log t \rceil$ times. Otherwise, (i.e., if all arms have been played more than $\lceil 8 \log t \rceil$ times) it chooses arm c_i that maximizes:

$$\mathcal{R}_t(c_i) + \sqrt{16 \times \frac{q_{c_i} - k_{c_i} \times \mathcal{R}_t^2(c_i)}{k_{c_i} - 1} \times \frac{\log(t-1)}{k_{c_i}}} \tag{2.1}$$

where q_{c_i} is the sum of squared rewards obtained from criterion/arm c_i, and k_{c_i} is the number of times criterion c_i has been chosen so far. A drawback of these approaches is that when the number of arms is big, exploration becomes difficult.

2.2.2.3 MAB Recommendation

Traditional recommender systems, such as those based on collaborative-filtering, content-based, and hybrid techniques, provide relevant suggestions by leveraging customer's taste as demonstrated by their previous applications (see Section 2.2.1). However, in many Web-based scenarios, e.g., daily-deals, datasets suffer frequent changes, with items popularity also changing over time. Furthermore, a large number of customers do not offer or disclose personal information neither past consumption [Li *et al.*, 2010a]. These issues make traditional recommender systems difficult to apply, as shown in previous literature [Chu and Park, 2009].

Hence, it is indispensable to learn good matches between customer interest and available items when one or both of them are new. However, acquiring such information

may be costly and reduce customer satisfaction in the short term, which leads to two possible competing goals: gathering information to provide a good match between customer and items, and maximizing customer satisfaction in the long run. This problem is indeed an instance of the previously mentioned exploration-exploitation dilemma.

The previous section presented three well-known MAB algorithms to tackle this problem. They started to be applied recently in the recommendation scenario, but in most cases under a new paradigm, named context-aware MABs. Context-aware multi-armed bandits[4] have been applied to a variety of problems, including Web content optimization [Li *et al.*, 2010a], Web documents ranking [Radlinski *et al.*, 2008], online advertisement [Chakrabarti *et al.*, 2008, Li *et al.*, 2010b], and movie recommender systems [Bouneffouf *et al.*, 2012].

Most MAB algorithms applied to recommendation follow this paradigm, and model the recommendation task as a MAB where the items to be recommended are the arms and the customers, represented as feature vectors, are the context. In particular, the customer (context) is revealed at each time step. For instance, the news recommender systems proposed in Li *et al.* [2010a] encode customer context as a combined vector of demographic information, geographic features, and behavioral categories, and encode news articles by a vector of categories including URL and editor categories.

In this same direction, the authors in Chakrabarti *et al.* [2008] described a contextually advertising system that combines ad relevance with both historical impression and click information using a logistic regression model. In Li *et al.* [2010b], the authors presented another work in contextual advertising systems focusing on strategies to balance the long-term impacts of exploration and exploitation on the system performance. Mahajan *et al.* [2012], in turn, proposed a UCB-based contextual bandit algorithm to estimate average rating for comments.

One of the few works that use "context-free" MAB for recommendation is Zhao *et al.* [2013]. They investigated the customer cold-start problem, and were also interested in understanding how the preferences of old customers change over time. The authors addressed the recommendation problem under the matrix factorization framework as a pure collaborative filtering algorithm. They did not consider content features either from customer or item, i.e., they did not explore any contextual information. Note that at first the problem we solve here could be seen as an item-cold start problem. However, the main difference is that our items frequently appear but also frequently disappear, and this short life time needs to be considered by the method.

[4]In the literature, context-aware bandits are sometimes called contextual bandits, associative bandits, bandits with side information, bandits with covariate, and associative reinforcement learning.

Chapter 3

Predicting Deal Size

In this chapter, we propose a method to improve the effectiveness of the deal size prediction task in order to help DDS administrators to better select the catalog of deals, maximizing profitability.

As already mentioned, there are two main challenges in this task. First, the catalog is usually available for a limited time frame, which may vary, on average, from 4 to 5 days [Byers *et al.*, 2012a]. This may compromise the amount of historical data that is available for learning predictors, harming the effectiveness of algorithms such as *Support Vector Regressor* (SVR) [Drucker *et al.*, 1997, Basak *et al.*, 2007]). Second, deals may compete among themselves for customer preference, resulting in hesitation and possibly hurting the corresponding sales as well as the overall revenue.

Despite these challenges, existing solutions to deal size prediction [Byers *et al.*, 2012a, Lappas and Terzi, 2012] often neglect the existence of complex interactions between deals (see Section 2.1.2 for a literature review on these methods). The attractiveness of a deal cannot be assessed in isolation, but rather it is given in relative terms when compared to other deals that are also shown in the catalog [Ariely, 2008]. The main difference between the method proposed here and previous work is that it takes these interactions into account, i.e., when predicting deal size we consider not only the isolated deal, but all the other deal options in the current catalog.

The solution proposed relies heavily on the existence of different markets. A market is defined as a set of deals that are likely to attract the interest of a similar group of customers. According to this definition, deals within the same market may compete for preference of the same customers. On the other hand, deals in different markets might help each other boost their sales. In our algorithm, we propose to consider the aforementioned situations at the same time to improve the efficacy of deal size prediction.

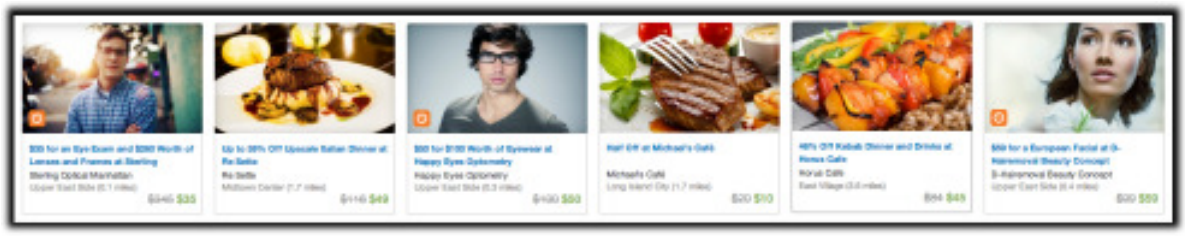


Figure 3.1: A catalog of deals that are shown to potential customers. Some deals may compete for the preference of the customers, while other deals may leverage the interest to other deals.

In Figure 3.1, we present an illustrative example of the existence of competition and complementarity between deals. Consider a commonly observed case in which a customer is looking for discounts in restaurants and has a limited budget of $50. Considering the catalog illustrated, he/she would chose between an Italian dinner or a Kebab, but he/she is unlikely to increase his/her budget in order to buy both deals. An even worse possible outcome may happen when the similarity between the two services is so high that causes hesitation, and the customer quits the DDS without buying any of the competing deals. As a result, apparently good deals may be less effective than expected. Turning our attention to complex deal interplay, consider a customer that buys a coupon for an eye exam. He/She will probably need to buy an eyewear too, and having both deals in the same catalog might mean a double sale.

In Figure 3.2, we present our method to tackle the deal size prediction problem. Given a catalog, we first create representations of deals based on terms describing them, as detailed in Section 3.1.1. Next, the deals serve as input for a Market Identification process, details in Section 3.2. Note that DDS market specialists manually separate deals into categories (markets), such as restaurants, entertainment, and fitness. This is a costly and labor-intensive process which assigns deals to predetermined categories. However, these categories may be excessively broad, not reflecting our definition of market. For this reason, we proposed an automatic approach to automatically separate deals into markets, assuming that deals in a specific market are likely to be described using a similar vocabulary.

The markets are created using a topic identification method, namely *Latent Dirichlet Allocation* (LDA) [Blei *et al.*, 2003]. Having markets, the deal size prediction phase starts, as detailed in Section 3.3. For each market we produce an SVR deal size predictor which receives as input features such as coupon price, discount, starting day, etc, and outputs the estimated number of coupons sold for the corresponding deal. Partitioning the full set of deals into smaller subsets (i.e., markets) drastically decreases

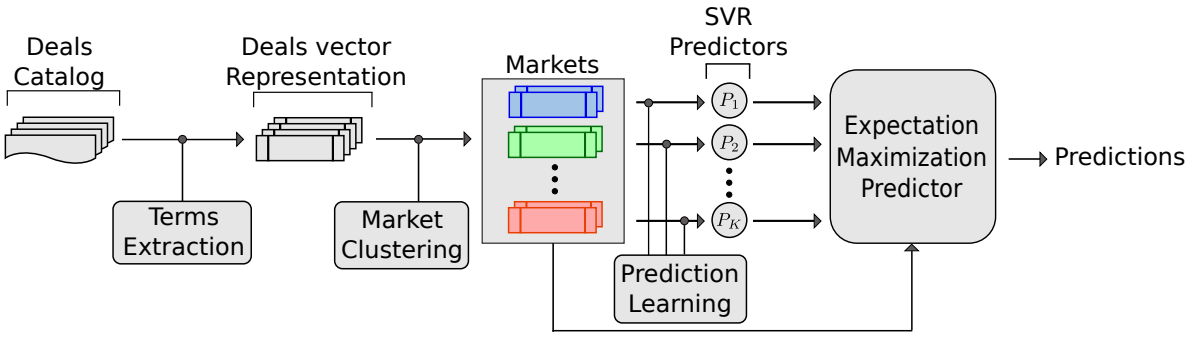


Figure 3.2: An overview of the proposed approach for deal size prediction.

the variance associated with features and deal sizes. As a result, the amount of training resources needed to build effective SVR predictors is also decreased, mitigating problems due to lack of historical data.

Having deal sizes predicted for specific markets, competition between deals within the same market is taken into account by normalizing the representativeness of the market (i.e., the expected number of coupons sold considering all deals in the market) by the number of competing deals. Therefore, the size of each deal is re-scaled in order to reflect the impact due to the competition for customer preference. More specifically, the normalizing factor associated with each deal is obtained using an Expectation-Maximization approach which minimizes the prediction error by considering (i) features associated with the target deal we are predicting the size, and (ii) features associated with the other deals in the same market.

3.1 Data and Feature Engineering

In this section, we start by describing the three datasets used in our experiments to assess the performance of deal prediction methods. Namely, we detail Groupon, LivingSocial, and Peixe Urbano datasets. We also present textual and non-textual features available on the datasets used.

3.1.1 Dataset Description

Our goal is to evaluate our solutions to deal size prediction on ground-truth data. We expended significant time, effort, and resources to obtain high quality data from daily deals sites, which shall be made available at publication time (except Peixe Urbano dataset). We look at data from three major DDSs: Groupon (GP), LivingSocial (LS), and Peixe Urbano (PU).

For Groupon and LivingSocial we used their respective APIs to expand two existing public datasets [Byers *et al.*, 2012a]. These datasets were crawled during different time periods. For Groupon, the crawling process started on Jan-3rd-2011 and finished on Jul-3rd-2011, resulting in 16,409 deals. For LivingSocial, the crawling process started on Mar-21st-2011 and finished on Jul-3rd-2011, resulting in 2,610 deals. These original datasets were expanded by collecting the textual content of all deals.

The new datasets include, apart from the original meta-data such as coupon price, coupon discounted price, starting day of the week, among others, features that will help to separate deals into markets, such as merchant's name, title, highlight, and description. The textual features are composed of terms in English. The vocabulary contains 119,525 distinct terms for Groupon and 19,102 distinct terms for LivingSocial. The third dataset is Peixe Urbano, already described in Section 2.1.3. We used the larger Peixe Urbano dataset, which has 4,309 deals described by 31,163 distinct terms.

As detailed in the following sections, Peixe Urbano does not contain any information about deals categories, such as Restaurant, Food or Beauty. On the other hand, Groupon deals are grouped into 12 categories and LivingSocial into 11 categories. The distribution of deals among categories is highly skewed in both cases, as detailed later.

3.1.2 Textual Features Associated with Markets

The first step of the method proposed in this chapter is to separate deals into markets. We start by describing a deal according to a set of terms associated with it, as we believe that deals within the same market are likely to be described using similar terms. We represent a deal using the following features:

- Merchant's name: terms appearing in the name of the merchant associated with deal d are used to represent d.
- Title of the deal: terms appearing in the title of deal d are used to represent d.
- Highlight of the deal: terms appearing in the highlight summary of deal d are used to represent d.
- Description of the deal: terms appearing in the description of deal d are used to represent d.
- Bag-of-words: all textual features are taken as a single document, thus, the same terms in different features are represented by the same element in the vector.

Table 3.1: Number of distinct terms within each feature for Groupon (GP), LivingSocial (LS), and Peixe Urbano (PU).

	Average			Maximum		
	GP	LS	PU	GP	LS	PU
Name	2.63	2.44	2.38	10	8	7
Title	8.63	5.41	11.44	23	13	27
Highlight	11.67	–	52.56	37	–	139
Description	132.02	57.24	93.49	351	101	196
BoW	135.75	58.06	123.79	361	103	244
Concatenation	154.65	65.23	161.37	381	111	329

- Concatenation: all textual features are concatenated in a single vector, and hence the same terms in different features are considered different elements in the vector.

In Table 3.1, we show the average and maximum number of distinct terms in each feature of the deal after eliminating stop-words and performing stemming. For english datasets, i.e. Groupon and LivingSocial, we used the Porter algorithm [Porter, 1980] to stem words. For the portuguese dataset, i.e. Peixe Urbano, we used the stemming algorithm proposed in Orengo and Huyck [2001]. As expected, we used different stopwords lists for each language. Typically, the Groupon and Peixe Urbano datasets contain more terms on average than LivingSocial dataset. We believe that the difference is related to the minimum number of characters that each website requires from merchants to describe a deal. We evaluate to which extent these features can separate markets by analyzing their relative quality using metrics of descriptive and discriminative power presented in Fernandes *et al.* [2007] and Figueiredo *et al.* [2009].

Assessing the Descriptive Power of Feature

The assessment of the descriptive power of each feature is based on a heuristic metric called *Average Feature Spread*, or simply *AFS*, which is computed as follows. Let $\mathcal{D}$ be the collection of deals, and s be a feature associated with a specific deal $d \in \mathcal{D}$ (e.g., merchant's name, title, description or highlight). A term t must appear in at least one feature $s \in d$. The *term spread*, denoted by $TS(t,d)$, is a function that returns the number of features of d in which the term t appears, and is defined as:

$$TS(t,d) = \sum_{s \in d} i, \text{ where } i = \begin{cases} 1 & t \in s \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

The intuition behind $TS(t,d)$ is that the larger the number of features containing term t, the more term t represents deal d. Next, we specify the *Feature Spread* of a feature s associated with a deal d, $FS(s,d)$, as the average term spread across all terms in feature s. Assuming $|s|$ as the number of distinct terms in feature s, we have:

$$FS(s,d) = \sum_{t \epsilon s} \frac{TS(t,d)}{|s|} \tag{3.2}$$

This heuristic asserts how much feature s represents deal d, thus being an estimate of the descriptive power of feature s regarding an arbitrary deal d. Hence, the overall descriptive power of feature s in the $\mathcal{D}$ is given as the average feature spread when considering all deals in $\mathcal{D}$. This metric is called *Average Feature Spread* of feature s, or simply $AFS(s)$, and is given as:

$$AFS(s) = \frac{\sum_{d \epsilon \mathcal{D}} FS(s,d)}{|\mathcal{D}|} \tag{3.3}$$

Assessing the Discriminative Power of a Feature

The assessment of the discriminative power of each feature is based on a heuristic metric called *Average Inverse Feature Frequency* ($AIFF$). The $AIFF$ metric is a variation of the IDF metric [Baeza-Yates and Ribeiro-Neto, 2011], which is called *Inverse Feature Frequency* (IFF). Given a feature s occurring in $|\mathcal{D}|$ deals, and a term t that appears in at least one deal in $\mathcal{D}$, the $IFF(t,s)$ is given as:

$$IFF(t,s) = \log\left(\frac{|\mathcal{D}|}{freq(t,s)}\right) \tag{3.4}$$

where $freq(t,s)$ refers to the number of deals in $\mathcal{D}$ in which the term t appears in feature s.

The IFF metric estimates how much information holds the appearance of a given term in a given feature. The intuition is that terms appearing too frequently in a feature are poor to discriminate deals. Hence, we define the *Average Inverse Feature Frequency* of feature s, or simply $AIFF(s)$, as:

$$AIFF(s) = \frac{\sum_{t \epsilon s} IFF(t,s)}{|V_s|} \tag{3.5}$$

where $|V_s|$ is the number of distinct terms appearing in feature s when all deals in $\mathcal{D}$ are considered.

In Table 3.2, we show *AFS* and *AIFF* values associated with each of the four textual features considered. As observed, merchant's name is the most descriptive and discriminative feature, followed by title of the deal. Surprisingly, the description of the deal is the less descriptive feature.

Table 3.2: *AFS* and *AIFF* values associated with each feature for Groupon (GP), LivingSocial (LS), and Peixe Urbano (PU).

	AFS			AIFF		
	GP	LS	PU	GP	LS	PU
Name	3.11	2.38	3.48	9.06	7.28	7.49
Title	2.46	1.93	2.55	8.80	7.09	7.05
Highlight	1.84	–	1.68	8.49	–	6.93
Description	1.14	0.95	0.35	8.92	6.66	6.85
BoW	2.14	1.86	2.28	8.91	6.67	6.81
Concatenation	2.01	0.88	2.14	8.86	6.80	6.97

3.1.3 Features Associated with Deal Size

The second part of the proposed method is to perform deal size prediction. This section describes the features available for that, namely:

- the price of the deal,
- the price after the discount,
- the tipping point of the deal (i.e., minimum number of coupons that must be sold to enable the deal), (Not available for Peixe Urbano dataset.)
- a boolean value that indicates if the deal is running for multiple days or not,
- a boolean value that indicates if the deal is featured in the site or not,
- a boolean value that indicates if the deal inventory is limited or not,
- the day of the week in which the deal started,
- the category of the deal,
- the city in which the corresponding merchant is located.

Our basic assumption is that the values of such features strongly depend on the market that the corresponding deal is associated with. Assuming the category of the deal as being a coarse approximation for their markets, in Tables 3.3 and 3.4, we show number of deals and price variation according to Groupon and LivingSocial categories (recall that Peixe Urbano does not have categories). The last line contains the statistics for all deals, i.e., ignoring categories.

In Table 3.3, we present statistics for all 12 Groupon categories, which contain 16,409 deals. We see that the distribution of deals among categories is highly skewed. The "Nightlife" category is the smallest category in terms of number of deals with 272, whereas the category "Restaurants" is the largest with 2,893 deals. When analyzing minimum, average, and maximum price statistics among categories, we conclude that they are very different from one category to another. For instance, the average price among deals in "Restaurants" is US$ 32.51, whereas the average price of deals related to traveling products is much bigger, US$ 402.31. For all categories, we see that there is a large gap between the minimum price and the maximum price of deals. When considering the maximum price, the "Nightlife" and "Education" categories have the smallest minimum price.

Table 3.3: Price variation associated with Groupon categories.

Category	# Deals	Min. Price	Avg. Price	Max. Price
Restaurants	2,893	4.00	32.51	3000.00
Food & Drink	1,165	4.00	44.41	2044.00
Arts & Entert.	2,932	4.00	99.98	3649.00
Health & Fitness	2,108	8.00	264.39	6000.00
Nightlife	272	8.00	39.66	350.00
Automotive	376	9.00	127.65	3199.00
Prof. Services	626	10.00	190.34	5300.00
Travel	404	10.00	402.31	4950.00
Shopping	2,249	10.00	89.18	2699.00
Home Services	364	10.00	233.66	2500.00
Beauty & Spas	2,468	10.00	201.35	5825.00
Education	552	10.00	147.73	950.00
All	16,409	4.00	135.50	6000.00

In Table 3.4, we present statistics for all 10 categories referring to LivingSocial dataset, which contains 2,610 deals. As previously discussed for Groupon dataset, we also see that the distribution of deals among categories is highly skewed for LivingSocial dataset. The "Education" category is the smallest in terms of number of deals with

30, whereas the category "Restaurants" is the largest with 642 deals. Different from Groupon, for LivingSocial the minimum price for a specific category is far from the average minimum price. For instance, "Escapes" category have a minimum price equals to 115.00, while the average price for all deals is just 6.00. Another conclusion that can be outlined here is that the maximum price statistic among categories is very different from one category to another. For instance, the maximum price of "Specialty Food" is 125.00, while the maximum price for "Entertainment" category is $9,606.00$.

Table 3.4: Price variation associated with LivingSocial categories.

Category	# Deals	Min. Price	Avg. Price	Max. Price
Specialty Food	139	6.00	22.47	125.00
Entertainment	642	7.00	121.21	9606.00
Retail	158	10.00	61.21	709.00
Restaurant	425	10.00	33.47	350.00
Services	306	12.00	138.35	750.00
Beauty	463	15.00	219.18	2200.00
Health	336	20.00	256.86	5400.00
Education	28	30.00	106.71	180.00
Getaway	30	45.00	420.86	2060.00
Escapes	83	115.00	923.56	6960.00
All	2,610	6.00	164.16	9606.00

3.2 Identifying Deals Markets

As already discussed, discovering meaningful markets from textual features associated with deals is an important step for subsequent deal size prediction. In the previous section we investigated if terms associated with a deal could be used to separate markets, and we know they can. In this section, we show how those features can be used as input for a LDA based topic-identification strategy to separate deals into markets, as detailed in Section 3.2.1.

We define a market as a set of deals that are likely to attract the interest of a similar group of customers. Therefore, given that customers have a limited budget, deals belonging to the same market are more likely to compete for customer preference. A clear strategy to market identification would be to analysed the purchase history, in order to separate deals that were purchased by a similar set of customers. Unfortunately, such historic data is very limited due to the scarceness of recurrent or regular

customers, that is, typical DDS customers are bargain hunters and perform sporadic purchases.

Another possible strategy would be to interpret deal category as a market, such as Restaurant or Travel. However, as already discussed earlier, this approach has many drawbacks, such as deal categories being fixed and unbalanced, and not even existing for Peixe Urbano. Experiments showed that using categories instead of markets for deal size prediction leads to poorer results.

Therefore, we propose an alternative strategy to market identification by employing Latent Dirichlet Allocation, or simply LDA [Blei *et al.*, 2003], to elicit latent markets from textual features associated with deals. The proposed strategy is described in Section 3.2.2.

3.2.1 Deals Representation

The inputs for the LDA method used to separate deals into markets are deal vectors. Given a specific feature s, we represent an arbitrary deal d as a vector in a n–dimensional space $\{t_1, t_2, \ldots, t_n\}$, where n is the number of distinct terms appearing in feature s. The six features introduced in Section 3.1.2 are used to represent a deal. Further, in order to assess to how extent terms and deals are related to each other, we also consider and evaluate different term weighting schemes:

- *TS*: the weight assigned to term t is given as $TS(t, d)$, which is the spread of t in deal d, and captures its descriptive power.
- $TS \times IFF$: the weight assigned to term t is given as the product $TS(t, d) \times IFF(t, s)$, and simultaneously captures descriptive and discriminative powers of t.
- *TF*: the weight assigned to term t is given as $freq(t, s)$, which returns the number of times t appears in feature s.
- $TF \times IFF$: the weight assigned to term t is given as the product $freq(t, s) \times IFF(t, s)$.

3.2.2 Identifying Latent Markets

This section shows how we identify markets using Latent Dirichlet Allocation (LDA), a method for latent topic identification. LDA is an appropriate method for this task because we assume that deals belonging to the same market are likely to be described

using similar terms. When a merchant writes a deal, he has certain terms in mind. For instance, if he runs a Chinese restaurant business, possible terms used to describe a deal may include but are not limited to "restaurant", "food", and, more specifically, "Chinese food". On the other hand, if the merchant runs a gym, he may describe a deal for discounts in a gym plan using terms such as "class", "fit", and "body". Before showing how we used LDA, we give a brief description of the method.

Probabilistic topic models, such as LDA, are a suite of algorithms whose aim is to discover the hidden thematic structure, i.e. topics, in a large collection of documents [Blei *et al.*, 2003]. The aim of LDA is to identify latent topics in document collections. In our scenario, documents are deals and we interpret a topic as a market. Formally, a probabilistic topic model captures this intuition in a statistical framework, which allows examining a set of documents and discovering, based on the occurrence of their words, the topics itself and the distribution of topics among the documents.

LDA allows documents to have a mixture of topics. A topic is defined as a distribution over a fixed vocabulary. For instance, the mentioned *sports* topic has words about sports with high probability and the *economy* has words about economy with high probability. LDA assumes that these topics are specified before any data has been generated. Hence, for each document in the collection, it generates the words in a two-stage process:

1. Randomly chooses a distribution over topics
2. For each word in the document
    - Randomly chooses a topic from the distribution over topics in step 1.
    - Randomly chooses a word from the corresponding distribution over the vocabulary.

In order to describe LDA from a probabilistic modeling perspective, we start by defining the following notation. Topics are represented as $\beta_{1:K}$, where each β_k is a distribution over a vocabulary. For each document $d \in D$, LDA associates a vector θ_d to represent the topic proportions in d, where $\theta_{d,k}$ is the topic proportion for topic k in document d. We also define the vector z_d, for each document d, where $z_{d,n}$ is the topic assignment for the nth word in document d. Finally, the observed words for documents d are vectors w_d, where $w_{d,n}$ is the nth word in document d.

In Figure 3.3, we use this notation to present the probabilistic assumptions behind LDA using a graphical model. We use a plate notation to describe LDA as proposed in Blei *et al.* [2003]. Plates are represented using rectangles and indicate replicated

variables, for instance, K topics and D documents. Traditionally, graphical models use unshaded nodes to refer to hidden variables, such as topics, and topic proportions. On the other hand, observed variables, such as document words, are represented using shaded nodes. An edge from variable A to variable B means that variable B is conditional dependent on variable A. For instance, LDA model states that the distribution of observed words within a document is dependent on how much each topic is represented by a document.

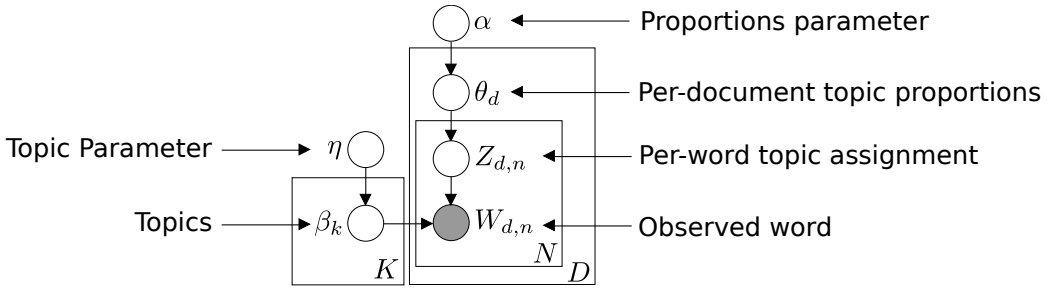


Figure 3.3: Diagram of the LDA graphical model. Each node is a random variable. The unshaded nodes refer to the hidden variables, i.e., the topics, assignments, and topic proportions. The shaded nodes refer to the observed nodes, i.e. the words of documents. The rectangles are "plates" and indicate replicated variables. Hence, the N plate represents the set of words within documents, the D plate represents the collection of documents within the collection, and the K plate represents the set of given topics.

In order to derive the joint distribution of the hidden and observed variables already mentioned, we can use the dependencies between these variables as presented in Figure 3.3. In the following, we present the joint probability distribution of all variables proposed in LDA model:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d) (\prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n}))$$

LDA was already used to solve several problems, including matrix factorization [Agarwal and Chen, 2010], influential user identification [Weng *et al.*, 2010], tag recommendation [Krestel *et al.*, 2009], and word sense disambiguation [Boyd-Graber *et al.*, 2010]. In our particular case, identifying market using the LDA strategy has a series of advantages, such as the robustness to cold start since it is not based on historic data.

In our model, a deal can be described as a mixture of different topics that are related to terms picked by the merchant, and these topics reflect the merchant's view of her business market. Given this assumption, we interpret latent topics discovered

by an LDA algorithm as being markets. More specifically, each deal d in the catalog of the day is associated with a k–dimensional market distribution $\theta_{d,m}$, which encodes the fraction of terms in d that relates to market m (where k is an input parameter and gives the number of markets to be identified). In other words, our LDA analysis outputs the topic/market proportions $\theta_{d,m}$ for each deal in the catalog of the day. For simplicity, we assume that deal d belongs to the market for which the highest proportion $\theta_{d,m}$ is observed.

In Table 3.5, we show markets identified from the Groupon dataset using this LDA formulation. Recall that we interpret each topic as a market. As we can see, LDA finds meaningful markets. Since a market is a probability distribution over the entire vocabulary, we also present the top ten words related to each market. Another conclusion here is that the top ten words of each topic often are semantically related among each other.

Table 3.5: Top ten terms from each of $k = 5$ markets in the Groupon dataset. Such markets were identified from the highlight of the deal, and using $TS \times IFF$ as the weighting scheme. Each column is labeled with an "interpretation" of the corresponding market. All markets are easily interpretable.

"Gym"	"Hair Salon"	"Sports"	"Dentistry"	"Ice Cream"
class	salon	camping	dental	tour
fit	hair	week	teeth	chocolate
body	look	sport	care	cake
train	services	day	value	sweet
workout	cut	academia	whitening	ice
boot	haircut	value	smile	cream
gym	treatment	football	white	adventure
month	make	young	today	paintball
art	lock	kick	day	guide
session	conditioner	soccer	dentistry	indoor

3.3 Predicting Deal Size

This section presents the proposed deal prediction model based on markets. First, we employ multiple SVR predictors, where each SVR predictor is specialized to predict the size of deals belonging to a given market. Then, we employ an iterative Expectation-Maximization procedure in order to exploit competition among markets.

3.3.1 Specialized SVR

Support Vector Regression [Drucker *et al.*, 1997], or simply SVR, is an established non-linear regression technique that has been applied successfully to a variety of predictive problems. In order to apply SVR to deal size prediction, we represent deals as follows. Let $\mathcal{D} = \{\mathcal{D}_{m_1}, \mathcal{D}_{m_2}, \ldots, \mathcal{D}_{m_k}\}$ be the collection of all past deals, and each $\mathcal{D}_{m_i} = \{d_1, d_2, \ldots, d_q\}$ be a partition of $\mathcal{D}$ composed of all q deals that belong to market m_i. Further, each deal d_j is represented as a feature-vector, where the constituent features are those discussed in Section 3.3.

The training-set that is used to build an SVR predictor to market m_i is composed of deal/size pairs of the form $\{(d_1, s_1), (d_2, s_2), \ldots, (d_q, s_q)\}$, i.e., each pair is composed of a deal $d_j \in \mathcal{D}_{m_i}$ and its corresponding size s_j. The specialized SVR predictor is a proper combination of features that estimate size s_j for an arbitrary deal $d_j \in \mathcal{D}_{m_i}$. More specifically, for each deal d_j, an estimating function takes the form $f(d_j) = \langle w, \Phi(d_j)\rangle + b$, where $w \subset \Re^n$, $b \subset \Re$, and Φ denotes a non-linear transformation from $\Re^n$ to high-dimensional space. The SVR objective is to find the minimum value of w and b by solving the following regularized optimization problem:

$$\min_{w} \frac{1}{2} w^T w + C \sum \xi_\epsilon(w; d_j, s_j). \tag{3.6}$$

$$\xi_\epsilon(w; d_j, s_j) = \max(|w^T d_j - s_j| - \epsilon, 0)^2 \tag{3.7}$$

where $C > 0$ is the regularization parameter, and (3.7) is the ϵ-insensitive loss function associated with (d_j, s_j). The parameter ϵ is given so that the loss is zero whether $|w^T d_j - s_j| \geq \epsilon$ [Smola and Schölkopf, 2004]. The intuition here is that we do not care about errors as long as they are less than ϵ, but any deviation larger than this are not accepted. We use a radial basis function (RBF) as the transformation function Φ, and all other parameters were chosen by using cross-validation within training set [Mitchell, 1997].

3.3.2 Exploiting Deals Interactions – The CPMB Model

The specialized SVR predictor does not take into account competition that may exist between deals in the catalog. For instance, deals in the catalog of the day, denoted as S, may compete with each other. As a result, SVR predictions may become over-estimated. In order to model competition, we first partition the catalog into k markets, so that $S = \{S_{m_1}, S_{m_2}, \ldots, S_{m_k}\}$. All deals $q \in S$ must be assigned to one of these markets, and we say that S_{d_j} is the market for which deal d_j is assigned to (i.e.,

$d_j \in S_{m_i}$ implies that $S_{m_i} = S_{d_j}$). Given the SVR prediction $f(q)$ for all deals $q \in S$, the estimated size of deal $d_j \in S$ is given as a combination of two factors – the SVR prediction for d_j, and an average factor that encompasses all deals within the same market:

$$\sigma(d_j) = \alpha \times \frac{\sum_{\forall q \in S_{d_j}} f(q) \times \rho(S_{d_j})}{|S_{d_j}|} + (1-\alpha) \times f(d_j) \tag{3.8}$$

where $\alpha = 0.5$ in order to weigh equally both factors, and $\rho(S_{d_j})$ is an unknown parameter which re-scales the representativeness of market S_{d_j}. The values of $\rho(S_i)$ are initialized as the mean number of deals that belongs to S_i in training data. We employ an Expectation-Maximization procedure using the training-set in order to find the value for $\rho(S_{d_j})$ that minimizes the loss function:

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum (y_i - \hat{y}_i)^2} \tag{3.9}$$

where n is the number of predictions performed, and y_i and $\hat{y}_i$ are respectively, the actual and the predicted deal sizes. We name this prediction model as CBPM, standing for Competitive Business Market Predictor, and we evaluate its effectiveness in the next section.

3.4 Experimental Results

The experiments performed in this section use the collected datasets introduced in Section 3.1.1 following the Interleaved Test-Then-Train methodology [Bifet *et al.*, 2010], where each deal in the catalog on day t is evaluated, and then included into the historical data that becomes available at day $t+1$. Our intention is to mimic as close as possible the production system of a DDS (i.e., we are considering that a DDS predicts deal size of each available offer in the beginning of the day, and overnight they incorporate previous data into the training-set, in order to predict the deal sizes in the next day).

We evaluate the effectiveness of our models using the Root Mean Squared Error (RMSE) [Salkind, 2010]. As shown in Equation 3.9, $\hat{y}_i$ is the predicted value of i-th sample, while y_i is the corresponding true value. Finally, n is the number of instances in the dataset for which the prediction is performed. The results reported correspond to an average over all days. Significance tests were performed ($p<0.05$) and the best results for each dataset are shown in bold.

As already mentioned in Section 2.1.2, we compare our model in contrast to the following baselines:

- Global Predictor (GLPR) [Byers *et al.*, 2012a] – The predictor is learned from the whole training-set, ignoring the existence of markets. The size of an arbitrary deal q is given as:

$$\sigma(q) = 2^{\beta_0 + \sum \beta_i \times f_i} \tag{3.10}$$

  where features f_i correspond to the ones discussed in Section 3.1.3, and the corresponding weight β_i is found using the Ordinary Least Square algorithm [Mitchell, 1997].

- One Predictor per Business Market (OPBM) [Lappas and Terzi, 2012] – The predictor is learned using an SVM-based regression method proposed by Shevade *et al.* [2000], and employing the following features: (i) deal location (i.e., city or area for which the deal is available), (ii) business reputation (i.e., global traffic rank and website's reputation from Alexa[1]), (iii) the price of the deal, (iv) the price after the discount, (v) the tipping point of the deal, (vi) seasonality (i.e., period of the year when the offer is made available). Further, the authors proposed to extract terms from deal description and used a hierarchical clustering algorithm that is a combination of LDA [Blei *et al.*, 2003] and a flat clustering algorithm based on Kullback-Liebler distance to separate deals into markets [Pinto *et al.*, 2007].

3.4.1 Weighting Schemes, Deal Representation, and Markets

This section examines the impact of different weighting schemes on the effectiveness of deal size prediction. We start presenting the results of each dataset separately, and then, we discuss and compare results among datasets.

In Figure 3.4, we present the performance of our approach in contrast to baselines already mentioned when considering Groupon. The error is presented as a function of the number of markets. Each line within the graph refers to a specific weighting scheme as described in Section 3.2.1. We see that the performance of baselines is independent of the number of markets. We first discuss the deal representation as using terms from Merchant's Name, Title, and Concatenation features. Here, all weighting schemes present a decrease in error as the number of markets increase, and the best performance is given by *TS* and *TF* schemes. Now, we consider the Highlight, Description, and BoW

[1]http://www.alexa.com

schemes. When using Highlight, the $TS \times IFF$ is better than $TF \times IFF$ scheme when considering less than 50 markets. However, after 50 markets, the opposite happens, i.e., the $TF \times IFF$ scheme present a better performance than $TS \times IFF$ scheme. A similar behavior happens in both Description and BoW deal representations.

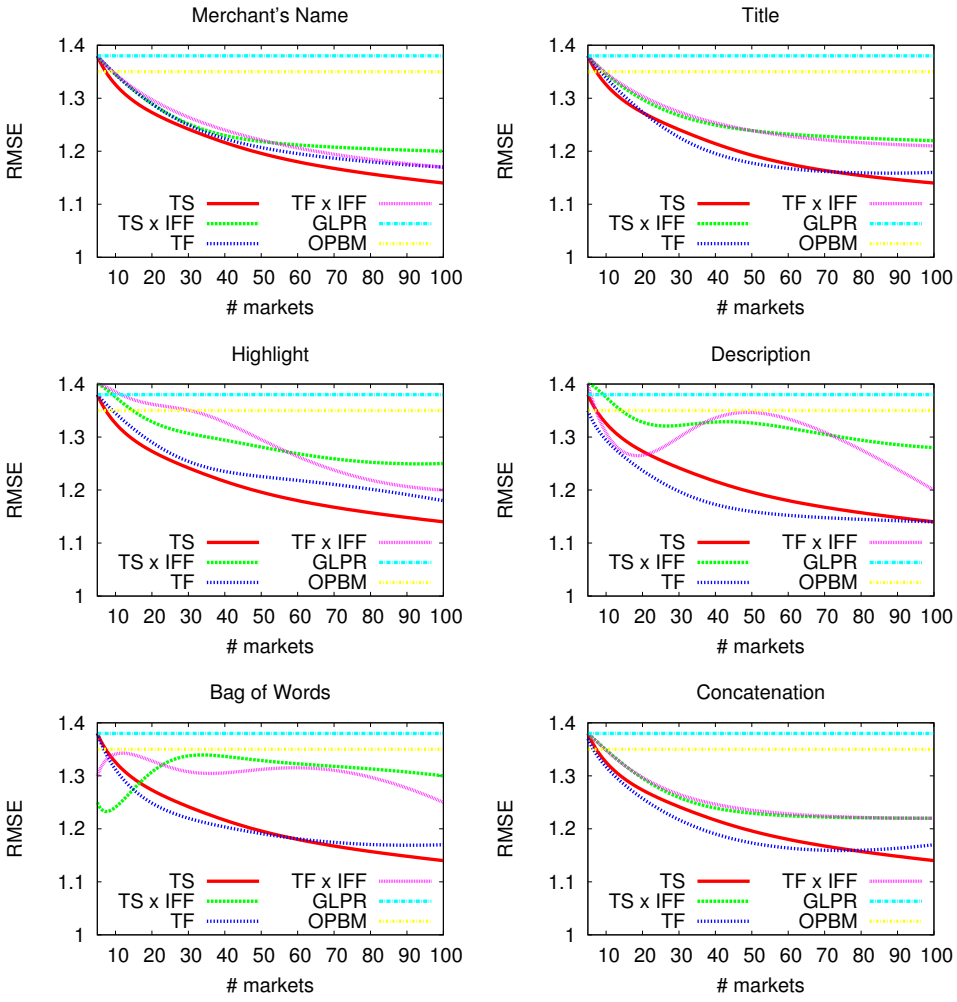


Figure 3.4: RMSE numbers for Groupon with varying number of markets.

In Figure 3.5, we analyze the RMSE values for LivingSocial as a function of the number of markets. Note that the deals from LivingSocial dataset do not present the highlight feature. We have here a similar behavior as presented in Groupon, i.e., the best values refers to TS and TF schemes. Different from Groupon, in some scenarios our performance is worst than those from baselines. For instance, when considering terms from Merchant's Name and the $TS \times IFF$ as weighting scheme, the performance of baseline GLPR for more than 80 markets is better. We also can see a similar behavior for Title, however, with more than 60 markets. Another difference from Groupon is that when we increase the number of markets the performance gains are smaller than

in the previous dataset. In general, we have the best performance when considering less than 30 markets with almost all deal representations.

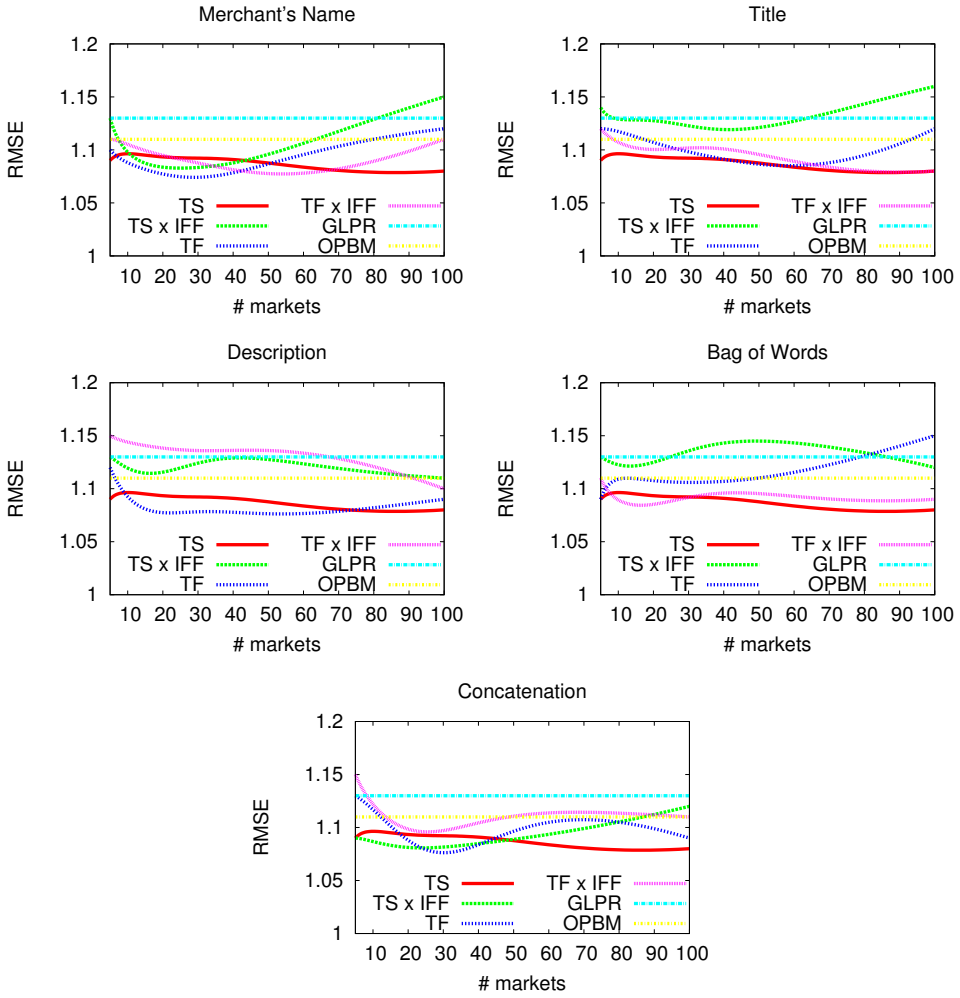


Figure 3.5: RMSE numbers for LivingSocial with varying number of markets.

In Figure 3.6, we present the performance of our method in Peixe Urbano. As the case of Groupon, we have here all 6 features, i.e., Merchant's Name, Title, Highlight, Description, BoW, and Concatenation. For small values of number of markets, usually less than 10 markets, we present a performance worst than our baselines. We have a special scenario for description, in which we have gains over the baseline when considering more than 60 markets. Typically, our errors decrease as the number of markets increase. As we already discussed for Groupon and LivingSocial, we have the best performance when using the weighting schemes *TS* and *TF*, independently of the feature used.

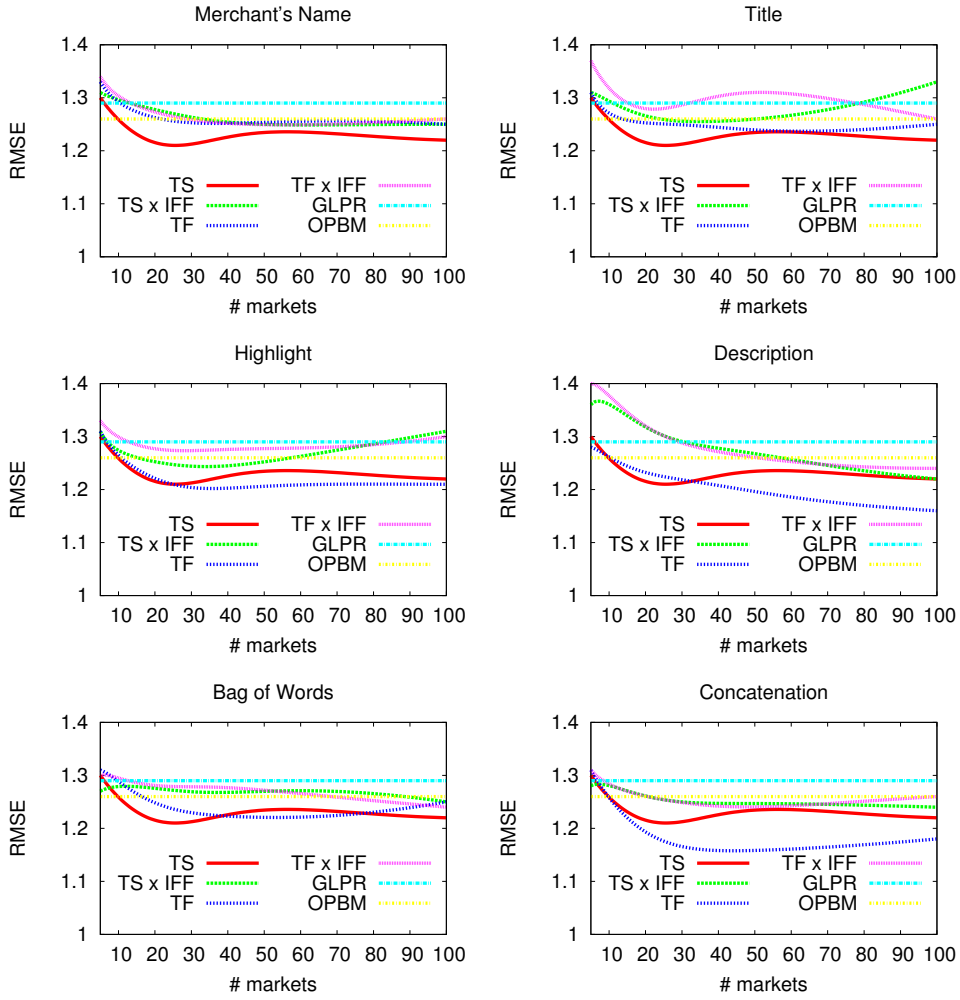


Figure 3.6: RMSE numbers for Peixe Urbano with varying number of markets.

We now consider all datasets and analyze the impact that different ways of representing the deals have on the prediction performance. Results obtained using Description are, usually, the best ones in all datasets. Note that this is consistent with our textual analysis shown in Tables 3.1 and 3.2. Hence, the larger amount of content clearly favors Description as source of data when compared to other textual features. The content size may also explain the good performance of description given that we separate deals into markets using LDA, which is known to work better in the presence of more co-occurrence between terms [Blei *et al.*, 2003].

Considering the other two features, Title usually outperforms Highlight in Groupon. In spite of the smaller amount of content present on Title, it shows bigger *AFS* (descriptive power) and *AIFF* (discriminative power) than Highlight, which turns out to be more dominant factors for prediction accuracy. On the other hand, Title presents a similar performance compared to Highlight in Peixe Urbano, independently

of their difference in amount of content. Hence, other aspects than quality-related ones (e.g., amount of content, discriminative power, and descriptive power) that should be taken into account to assess deal content, are subject to future work. Note that, Highlight is unavailable in Living Social.

Regarding the results achieved using both feature combination strategies, i.e. Concatenation and BoW, we found that Concatenation is less sensitive to the number of markets in Groupon and, on the other hand, BoW is less sensitive when considering Living Social. When considering Peixe Urbano, both combinations are sensitive to the number of markets. However, in most of our results both strategies presents better performance when compared to the baselines.

Analyzing the comparative performance of our methods when considering all three datasets, we show that, for any give feature (or feature combination), the results in Living Social are much better than in the other datasets. For instance, when considering Title, the values in RMSE ranges from 1.14 to 1.043 in Living Social, from 1.355 to 1.15 in Groupon, and from 1.355 to 1.16 in Peixe Urbano. Since we take into account textual features to separate deals into markets, it is also important to note that we are able to improve prediction accuracy on both English (Groupon and Living Social) and Portuguese (Peixe Urbano) datasets. In other words, our model correctly predicts deal size independently of the language used.

3.4.2 Overall Effectiveness

In Table 3.6, we present the RMSE numbers of our method and baselines. We conduct deal size prediction experiments combining each term weighting scheme with each textual feature (and feature combination) deal model. Given the results shown in Figures 3.4, 3.5, and 3.6, the overall conclusion is that our proposed model presents better effectiveness, in terms of RMSE, than both baselines. When comparing our model against the two baselines, we reach gains of 17.67%, 9.60%, and 11.77%, in Groupon, Living Social, and Peixe Urbano, respectively.

3.4.3 Local vs Global Predictors

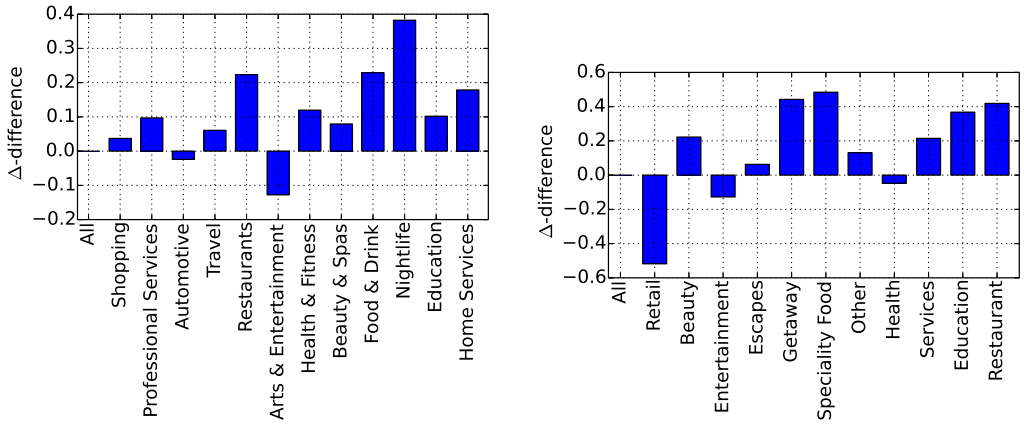
We also investigate the effect on prediction effectiveness when training a specific predictor for each market, rather than using a single predictor. Specifically, we define Δ-difference as the difference, in terms of RMSE, between a single, global predictor (for all markets) and multiple, specific predictors for each market. Hence, a positive difference means that using a market-specific predictor is better than using a single

Table 3.6: RMSE numbers for our model CBMP and for the baselines: GLPR and OPBM. For each dataset, we present our best result considering all combinations of weighting scheme, feature space, and number of markets. For all three datasets the best results were obtained when using *TF* as weighting scheme and representing deals by concatenating the terms. For Groupon (GP) the best number of markets was 50, whereas for Living Social (LS) and Peixe Urbano (PU) the best number was 30.

	CPMB	GLPR	Gain (%)	OPBM	Gain (%)
GP	**1.1332**	1.3764	17.67	1.3544	16.33
LS	**1.0203**	1.1287	9.60	1.1112	8.18
PU	**1.1430**	1.2956	11.77	1.2575	9.11

predictor for all markets, and, obviously, a negative difference means the opposite. Here, we use the deals categories as markets.

In Figure 3.7, we present the average results over all deals for Groupon and LivingSocial datasets.[2] For both datasets, we have that most of the observed differences are positive, thus, it is usually better using a market-specific predictor than a single predictor for all markets. The negative results are probably a consequence of the distribution of deals among categories, which is very skewed, as showed in Tables 3.3 and 3.4.



(a) Groupon (b) Living Social

Figure 3.7: Market-specific prediction model vs. global model.

[2]Since category information is unavailable for Peixe Urbano, we leave this dataset out.

3.5 Time Performance

In this section we discuss the time performance of our proposed approach for deal size prediction, and analyze the four main steps of the process illustrated in Figure 3.2 on Page 29. The first step, extracting terms from deal textual content, is linearly dependent on the number of deals n and terms t describing a deal, i.e., O(nt). The second step, clustering deals into markets, uses Latent Dirichlet Allocation (LDA) to find clusters. The running time of the LDA method is O(no), where o is the number of topics [Sontag and Roy, 2011]. The third step refers to training a Support Vector Regressor (SVR) for each market. We used the implementation available in the Scikit-learn package.[3] The time complexity of the SVR method is $O(fn)$, where f is the number of features. Finally, the last step of our approach is to re-scale the SVR predictions using our Expectation-Maximization (EM) algorithm. This step is linearly dependent on the number of deals and the maximum number of iterations that EM is allowed to execute. Thus, the running time of the algorithm is $O(ni)$, where i is the maximum number of iterations. All steps previously described are computed off-line, and the total running time is $O(fn)$.

3.6 Final Remarks

This chapter proposed the CBMP model, a new model to predict the size of a deal, i.e., the number of coupons that are expected to sell. This task is crucial to DDS administrators in order to help them simulating different sets of catalogs that maximize the DDS profitability. Our prediction model takes into account competition among deals while performing predictions, being this the main reason for its superiority when compared against state-of-the-art baselines. These interactions may lead to: (i) deals that compete for customer preference (possibly decreasing their sizes), or (ii) deals that complement other deals (possibly leveraging their sizes). To address the interaction among deals, we introduce the concept of market, which is defined as a set of deals that attract the interest of customers with similar preferences. We first propose a content-based method to identify markets by exploring deal structure. Additionally, we present an expectation-maximization algorithm that models market interplay and intra-market competition by minimizing the error in prediction. By conducting experiments on three real datasets, we obtained gains that range from 8.18% to 17.67% over previously proposed methods.

[3] http://www.scikit-learn.org

Chapter 4

Prioritizing Emails with Multi-Armed Bandits

In this chapter, we study the email prioritization task, i.e., how can we select a subset of all customers to send email that are more likely to click on email offers. In Figure 4.1, we present the abuse-unsubscribe, open, and click rates of emails sent by DDSs and average number from other ecommerce sites.

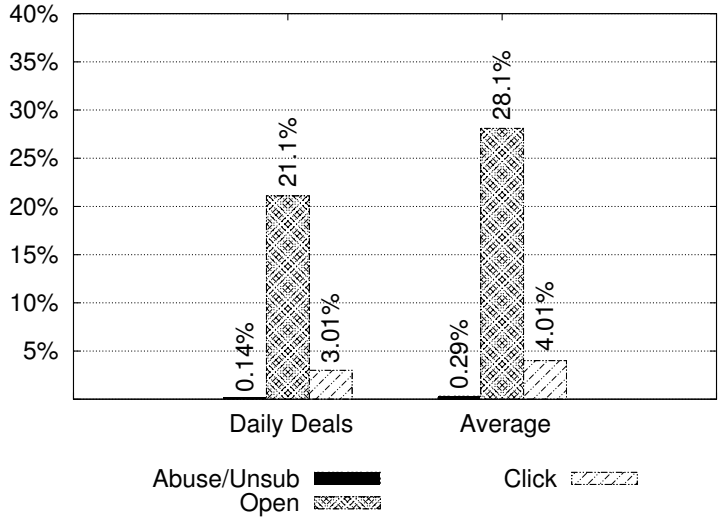


Figure 4.1: Abuse/unsubscribe, open, and click rates. Data obtained from MailChimp website.

We can see from Figure 4.1 that, although the number of emails sent by daily deals websites is possibly far in excess, the abuse-unsubscribe rate associated with DDSs is very low. However, as emails are sent in a daily-basis, customers do not give these alerts immediate attention. This is reflected in the open and click rates. Open

and click rates are significantly lower for daily deals websites, and this may be a result of the large number of (unnecessary) emails sent to customers. Therefore, a possible strategy to increase click rates of daily deals websites is to decrease the number of alerts sent to customers, in a way that they receive only useful alerts and are more likely to click on them.

The main goal of the method proposed in this chapter is to increase the perceived value associated with daily deals alerts by predicting the customers that will actually open and click the offers in the corresponding email. We propose a method that ranks customers in a way that those appearing first in the ranking are more likely to open and click the email. The lower in the ranking, the lower the probability of a customer receiving the email. This strategy contributes to more sustainable and effective email marketing.

There are many evidence or features that can give hints of whether a customer will click the email or not. For instance, a customer who has bought the same deals as other customers may be interested in alerts on other deals already purchased by these customers. Or, a customer who has bought a deal recently is less likely to purchase another deal in the near future. Also, certain customers may be interested only on buying deals falling within a given price range (or within a discount range). Thus, we may sort customers in different ways using different criterion.

Selecting the best sorting criterion is a challenging task, as it may vary as a function of the deals available in the catalog of the day. Hence, it is necessary to continuously explore all possible criteria, performing an explicit trial-and-error search for the best criterion to apply at each moment (i.e., we need to constantly monitor the performance of each criterion, in order to select the one providing the highest reward at the moment).

We model this task as a reinforcement learning problem Sutton and Barto [1998] in which the goal is to accumulate rewards from a payoff distribution with unknown parameters that are learned sequentially. We employ multi-armed bandit algorithms (e.g., ϵ−greedy [Thompson, 1963], and upper confidence bound [Auer *et al.*, 2002]) as a way to map customers to criteria, so as to maximize the click rate by selecting the best criterion to apply at each time step. In, Figure 4.2, we present the method proposed to solve the sequential task of sending emails.

Note that the multi-armed bandit scenario is the most appropriate here, since at a given time t, we have no information about which criterion is the best to sort customers. Specially in online scenarios, the MAB approach allows us to learn, according to the current conditions of the dataset, the best criteria to apply testing all of them concomitantly.

In Figure 4.2, we show that each arm represents a sorting criterion and at each time step t the MAB decides which arm should be used to sort customers. The top customer according to that criterion is selected to receive an email. According to the selected customer, a state-of-the-art recommendation algorithm generates deal recommendations, and the top-5 deals are sent by email. Every time a customer clicks a deal in the email sent the system receives a reward, which helps the MAB algorithm choosing the criterion which should be used to choose the next customer. The sorting criteria (i.e., the arms) are detailed Section 4.1.

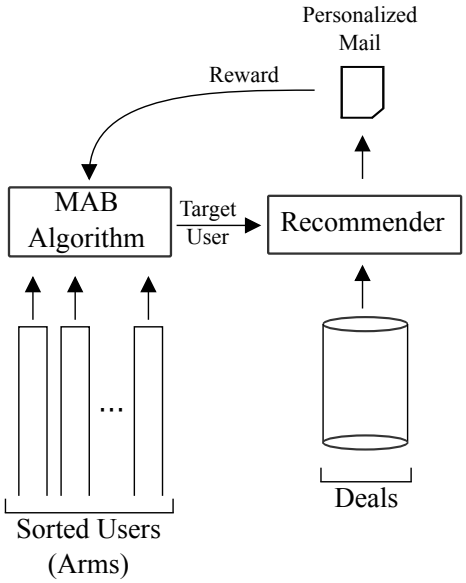


Figure 4.2: Overall representation of the sequential personalized email recommendation.

4.1 Criteria for Sorting Customers

We exploit two types of sources of evidence that (intuitively) suggest those customers that seem more likely to click the email: aggregates statistics and past history and catalog portfolio, as explained next.

4.1.1 Aggregated Statistics

Statistics derived from previous customer activities may be used as evidence of receiving clicks:

- Number of purchases already performed (c_1) – customers that purchase deals frequently are more likely to click (and to buy) again than customers that perform sporadic purchases. Thus, the likelihood of receiving a click from an arbitrary customer x is simply given as:

$$c_1(x) = \#\text{ purchases performed by } x \tag{4.1}$$

- Days since the last purchase (c_2) – customers that have bought a deal recently are less likely to click (and to buy) again in the near future. Thus, the likelihood of receiving a click from an arbitrary customer x is simply given as:

$$c_2(x) = \#\text{ days since last purchase performed by } x \tag{4.2}$$

- Savings (c_3) – daily-deals websites are intended to provide discounts on products/services to customers. Hence, the sum of money that a customer save when using a daily-deals website (i.e. his/her savings), can be interpreted as a proxy to the likelihood of being interested on new discounts. From market research this group of customers is known as "heavy customers" and merchandising efforts are focused on them [Wansink and Park, 2000]. The likelihood of receiving a click from an arbitrary customer x is given as:

$$c_3(x) = \sum_{d \in D_x} \gamma_d - \phi_d \tag{4.3}$$

  where D_x is the set of deals previously bought by customer x, γ_d is the original price of deal d, and ϕ_d is the discounted price of deal d.

4.1.2 Past History and Catalog Portfolio

Customer past history and the possible interactions with deals composing the current catalog may also be used as evidence of receiving clicks:

- Average price (c_4), average discount (c_5), and average discount percentage (c_6) – given the previous purchases of customer x and the average price associated with these purchases, $\mu(x)$, the likelihood of receiving a click from x is given as:

$$c_4(x) = \frac{1}{|\mu(x) - y|} \text{ st. } \forall z \in \mathcal{C} : |\mu(x) - z| \geq |\mu(x) - y| \tag{4.4}$$

where $\mathcal{C}$ is the current catalog, and y and z are variables representing the prices associated with the deals in $\mathcal{C}$. Similarly, we may also take into account the discount and the discount percentage associated with deals in the current catalog $\mathcal{C}$.

- Customer preference/taste (c_7) – given the past history associated with customer x, we employ state-of-the-art recommendation algorithms, such as WRMF (see Section 2.2.1), to estimates the likelihood of a specific deal being relevant to x. More specifically, a score is associated with each deal in the current catalog $\mathcal{C}$, and indicates the relevance of the deal to customer x. In this context, we denote the average score associated with customer x as $\mu(x)$, and the corresponding variance as $\sigma(x)$. Intuitively, high values of $\mu(x)$, in addition to high values of $\sigma(x)$, suggest that some deals in $\mathcal{C}$ are indeed relevant to x. In contrast, low values of $\mu(x)$ with low values of $\sigma(x)$, indicate that deals in $\mathcal{C}$ are unlikely to be relevant to customer x, thus decreasing the likelihood of receiving a click from x, which is given as:

$$c_7(x) = \mu(x) - \sigma(x) \tag{4.5}$$

4.2 Multi-Armed Bandit Algorithms

In our context, the multi-armed bandit problem is defined as follows. Each time step we want to send an alert, we must choose one of the criteria discussed in the last section. The chosen criterion is used to sort the customers in order to decide who is more likely to click the email featuring the deals. We assume that a reward is acquired after deciding the customer that will receive the email, depending whether the customer actually clicked the email or not. The objective is to choose a sequence of criteria that accumulates as much reward as possible, while considering a fixed number of steps (i.e., the number of customers to whom we will send the email).

We also assume that at each time step t, we have access to an estimate of the reward associated with each criterion. Such reward estimate, denoted as $\mathcal{R}_t(c_i)$, is given as the average reward accumulated so far, that is:

$$\mathcal{R}_t(c_i) = \frac{r_1 + r_2 + \ldots + r_k}{k} \tag{4.6}$$

where criterion c_i has been chosen k times so far, and r_j is the reward obtained at each time c_i was the chosen criterion, and is defined as:

$$r_j = \begin{cases} 1/\eta & \text{if the customer clicked the email} \\ 0 & \text{otherwise} \end{cases}$$

where η is the number of purchases in current testing day. Hence, the cumulative reward for each testing day is 1.0.

The greedy algorithm is to always select the criterion associated with the highest reward estimate at each time step. We say that such algorithm fully exploits the current reward estimates. However, as time passes these estimates become more and more uncertain. If, despite not being the best guess at the moment, a different criterion is chosen, then we say that we are exploring other options because this would improve the reward estimates associated with the other possible criteria. Thus, exploration is related to the "pursuit of information" in order to decrease the uncertainty in the reward estimates, while exploitation is related to maximize the immediate reward by performing greedy choices. There is a trade-off between exploration and exploitation, and multi-armed bandit algorithms differ among themselves in the way they balance exploration and exploitation.

We used two algorithms to prioritize emails: the ϵ-first and the Upper Confidence Bound Algorithms, both described in Section 2.2.2.2. These algorithms were compared to prioritizing the emails using each criterion isolated, without the MAB strategy.

This is because not a lot of works have discussed this problem in the literature. Similar versions of the problem were already addressed, but usually from a customer perspective for filtering junk messages. For example, Yoo *et al.* [2009] were the first address the personalized email prioritization problem, which consists in predicting the importance of email messages. In other words, they were interested in providing a ranked list of emails sorted by an importance score, which is a customer-dependent metric.

4.3 Experimental Evaluation

In this section we empirically analyze the performance of multi-armed bandit algorithms for the sequential personalized email recommendation for Peixe Urbano. We used the 2-months version, as detailed in Section 2.1.3.

We are interested in evaluating how fast the algorithms decide the best criterion to apply at each time step. Our basic evaluation measure is the cumulative reward, as we want to maximize the number of clicks received (i.e., reward) while minimizing the number of emails sent. Here we send emails to only a fraction of the customers. The fraction of the customers that receives emails is computed using all users from training data. Then, we calculate the fraction of clicks that were indeed captured (note that some clicks may be missed, since we are not sending emails to all customers). This

number is accumulated on a daily-basis, so that we have the cumulative reward up to day t. Due to data sensitivity, the values of reward reported in this section were normalized by the total number of email clicks for a given day, so that the maximum reward for a day is equals to 1.

Finally, we assume that all clicks received up to day t are incorporated into the historical data available at day t+1 (i.e., past history of the customers), and can be used as input to the WRMF recommendation algorithm, which sorts the deals according to the target customer and send the top-5 by email.

4.3.1 The Independence of Sorting Criteria

This first experiment shows how similar are the rankings generated by different sorting criteria. This is important because MAB algorithms assume that the arms are independent from one another. We use a rank correlation metric, namely Kendall Tau, to measure the extent to which the order of the observations (or customers) in a ranking A differs from the order of the observations in B, as defined in Equation 4.7. Given two pairs ($a_1 \in A$, $b_1 \in B$) and ($a_2 \in A$, $b_2 \in B$), they are considered concordant if $a_1 < b_1$ and $a_2 < b_2$, and discordant otherwise. As all pairs are compared among themselves, the total number of pairs is $\frac{n \times (n-1)}{2}$, where n is the number of observations.

$$\tau = \frac{|concordant\ pairs| - |discordant\ pairs|}{|total\ number\ of\ pairs|} \tag{4.7}$$

If all pairs are concordant, $\tau = 1.0$, that is, customers appear exactly in the same order in both A and B. If the pairs are all discordant, $\tau = -1.0$, that is, customers are in exactly the opposite order. If there are equal numbers of concordant and discordant pairs then $\tau = 0.0$ and there is no relationship between the rankings. In Table 4.1, we show all the values of τ are very close to 0, showing that the rankings generated are not correlated. As previously discussed, this result attests the utility of the several considered centrality rankings as arms within our MAB approach.

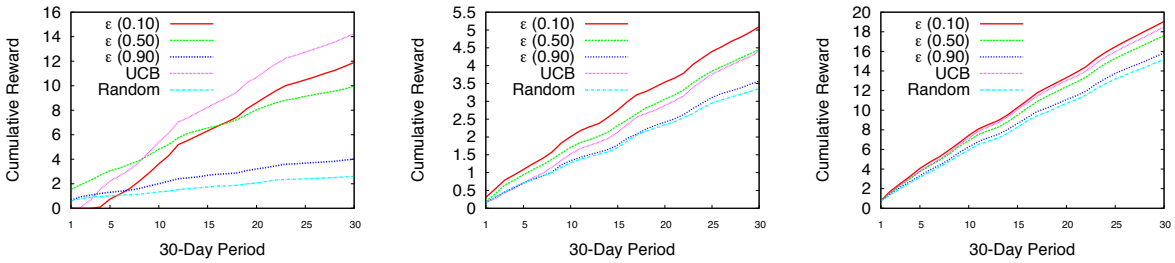
4.3.2 Recommendation Experiments

In Figure 4.3, we present the upper bound and the actual performance of multi-armed bandit algorithms. Particularly, in Figure 4.3 (a), we present the upper bound performance of the algorithms, and the fraction of emails sent is 10%. Further, we assume that one of the criteria is perfect, putting in the top of the ranking customers that clicked in the deals. The remaining 6 criteria, on the other hand, are fool and ineffective, putting in the top of the rank customers that did not click in the deals. For

Table 4.1: Kendall Tau considering rankings Generated by different Centrality Measures

	Less Recent	Savings	Avg. Price	Avg. Discount	Avg. Disc. Percentage	Customer Preference
Total Purchases	0.032	0.070	0.025	-0.018	-0.021	0.014
Less Recent	1.000	0.042	-0.013	0.012	-0.022	0.008
Savings		1.000	0.033	0.016	-0.011	0.010
Avg. Price			1.000	0.025	0.018	0.012
Avg. Discount				1.000	-0.012	0.009
Avg. Disc. Percentage					1.000	-0.016

this experiment we evaluate UCB and ϵ−greedy algorithms (with ϵ varying from 0.10 to 1.00). The objective of this experiment is to assess whether the algorithms are able to select the best criterion.



(a) (b) (c)

Figure 4.3: (a) Upper bound performance for 10% of emails sent (b) Actual performance for 10% of emails sent. (c) Actual performance for 50% of emails sent.

The UCB algorithm is the best performer. This happens because UCB do not randomly choose the criterion. Instead, it prefers the criterion that is closest to the best one. The ϵ−greedy algorithm, on the other hand, needs to randomly explore other options (criteria), and in this case it often chooses one of the poor criteria. Further, the best value for ϵ may vary with the number of emails sent.

In Figures 4.3 (b) and (c), we show the actual performance of the algorithms, i.e., for this experiment we used the criteria discussed in Section 4.1. The results were different from the previous experiment. Specifically, 0.1−greedy algorithm was the best performer, followed by 0.5−greedy and UCB algorithms. Note that the performance

of the algorithms tend to become closer as more emails are sent, since in this case less clicks are missed. Finally, the performance increases with the number of emails sent.

In Table 4.2, we show the cumulative reward associated with the criteria discussed in Section 4.1 when they are used independently, and compares these number with those obtained when sending emails using two versions of ϵ-first strategy with three different parameters and the UCB. These numbers refer to a period of 30 days. Note that sending all emails leads to the total reward (i.e., 30.00), since no clicks are missed no matter the sorting criterion used. By decreasing the number of emails sent, we can compare the cumulative reward obtained by each criterion.

Table 4.2: Cumulative reward associated with each criterion as a function of the number of emails sent.

Fraction of emails sent	c_1	c_2	c_3	c_4	c_5	c_6	c_7	$\epsilon(0.1)$	$\epsilon(0.5)$	$\epsilon(0.9)$	UCB
10%	1.77	4.72	3.41	2.63	2.15	1.90	4.65	5.01	4.43	3.56	4.55
50%	8.18	19.42	18.13	13.60	13.01	15.04	17.89	21.58	20.30	18.82	20.45
90%	24.04	28.54	27.70	27.30	27.23	27.10	27.82	28.57	28.02	27.33	28.36
100%	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.0	30.00	30.00	30.00

As observed, c_2 and c_7 are among the best overall criteria, while c_1 and c_5 are among the worst (although this may vary depending on the number of emails sent). Note that the MAB algorithms are always superior to the isolated criterion when the fraction of emails sent is low, namely 10% and 50%. When these fraction of emails sent increases to almost all customers, the differences decrease, but ϵ-first with parameter 0.1 still obtains more rewards than the isolated criterion.

In Figure 4.4, we show the fraction of times a specific criterion was chosen using UCB algorithm for each testing day. Here, the fraction of emails sent is 50%. The three best criteria that are chosen more frequently are c_2, c_7, and c_3, which are the best criteria according to Table 4.2. The UCB algorithm stats the process by assessing the expected reward and the confidence of this assessment. Thus, for the first days of test, it tries all arms. Once the UCB algorithm define high confidence on expected rewards for each arm, it tends to ignore inefficient arms and privileges those that return higher rewards. For instance, we can see in Figure 4.4 that UCB algorithm almost ignores arms c_1, c_4, c_5, and c_6 in the last testing days.

In Figure 4.5, we show the fraction of times a specific criterion was chosen, arm usage, by the $\epsilon-$greedy algorithm as a function of testing days. As in Figure 4.4, we also consider here the fraction of emails sent is 50%. We present the arm usage for different values of ϵ. In Figures 4.5 (a), (b), and (c), we use $\epsilon = 0.10$, $\epsilon = 0.50$, $\epsilon = 0.90$, respectively. For low values of ϵ the $\epsilon-$greedy algorithm rarely explores other criteria, focusing most of the time on the best-performing criterion. As a result, the two criteria

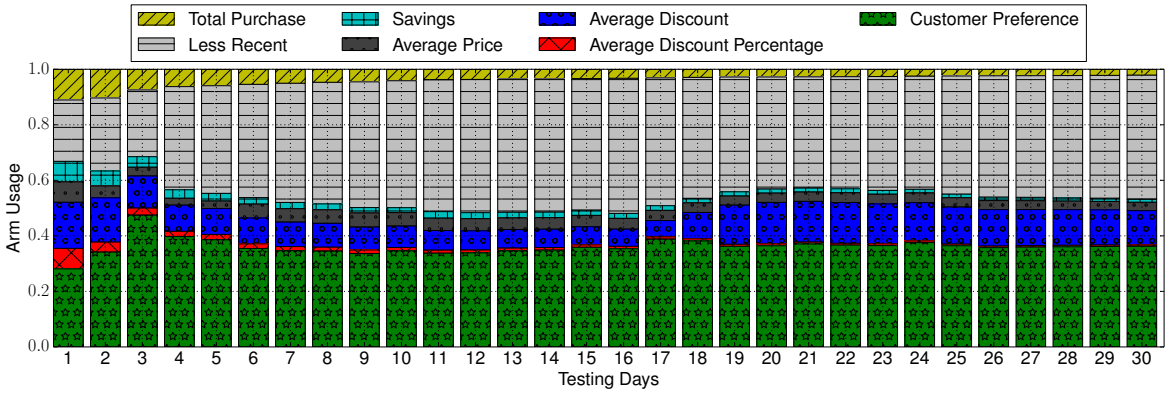


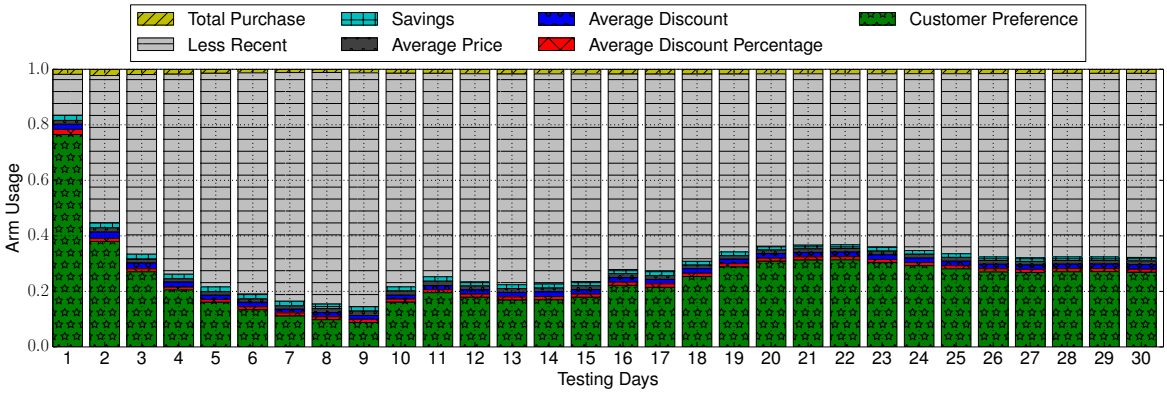
Figure 4.4: Fraction of times that a specific criterion (i.e., arm) was chosen using the UCB algorithm.

that are chosen more frequently are c_2 and c_7 (see Fig 4.5 (a)). Note that this is the same behavior as UCB algorithm that we previously discussed. As time passes, the algorithm converges to an efficient combination of these two criteria. Increasing the value of ϵ enables other criteria to be explored (see Fig 4.5 (b)). In particular, for higher values of ϵ all criteria tend to be used in equal proportion, since in this case the best criterion is being chosen with a low probability(see Fig 4.5 (c)).

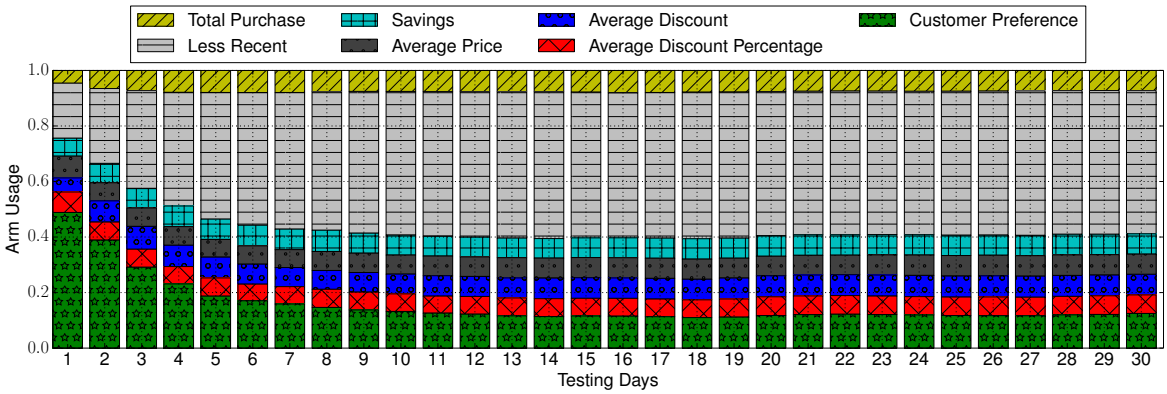
In Figure 4.6, we present the cumulative reward in terms of the fraction of emails sent. We concern here the trade-off between maximizing the cumulative reward while minimizing the number of emails sent to the customers. For this experiment, we assume that the upper bound performance is given by always choosing a perfect arm (i.e., a criterion that always lead to a click). Observe that we cannot send less than 40% of the emails without compromising reward. The performance of 1.00−greedy algorithm lies exactly in the diagonal since this algorithm randomly explores all criteria. The best performer is the 0.10-greedy algorithm, which enables a 10% reduction in terms of the number of emails sent, without compromising reward. If a small reduction in terms of reward is allowed (e.g., less than 10%), then we may avoid sending about 30-40% of the emails.

4.4 Time Performance

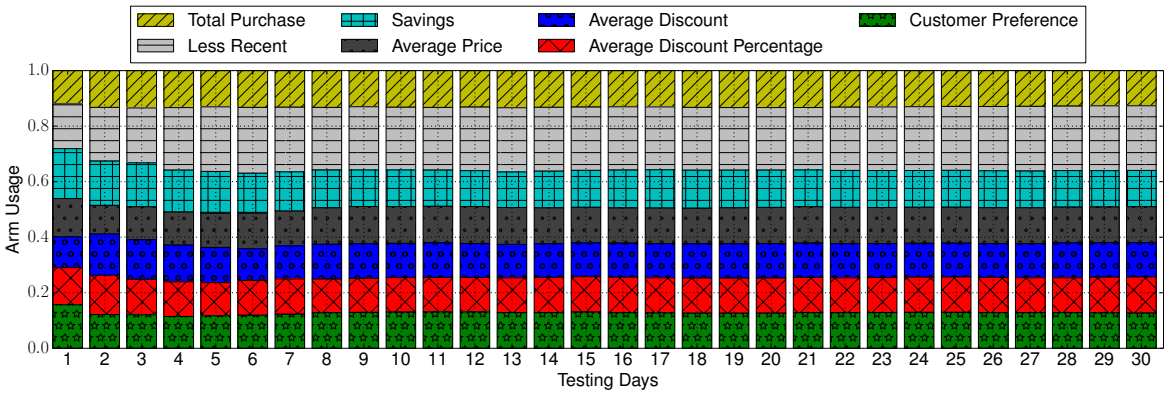
In this section we discuss the time performance of our proposed approach for email prioritization, and analyze the three steps of the process presented in Figure 4.2 on Page 51. All steps depend on the number of customers c and, unless stated otherwise, are calculated off-line. The first step concerns a sorting algorithm, which runs in $O(c \log c)$. The MAB algorithm, presented in the second step, is linear in c. The third



(a)



(b)



(c)

Figure 4.5: Fraction of times that a specific criterion (i.e., arm) was chosen using the ϵ–greedy algorithm. (a) $\epsilon = 0.10$. (b) $\epsilon = 0.50$. (c) $\epsilon = 0.90$.

step, which refers to the recommender, has two phases. First, we train the recommender off-line and then make recommendations online. Training the recommender depends on c and the number of non-zero observations N in the customer/deal matrix. We

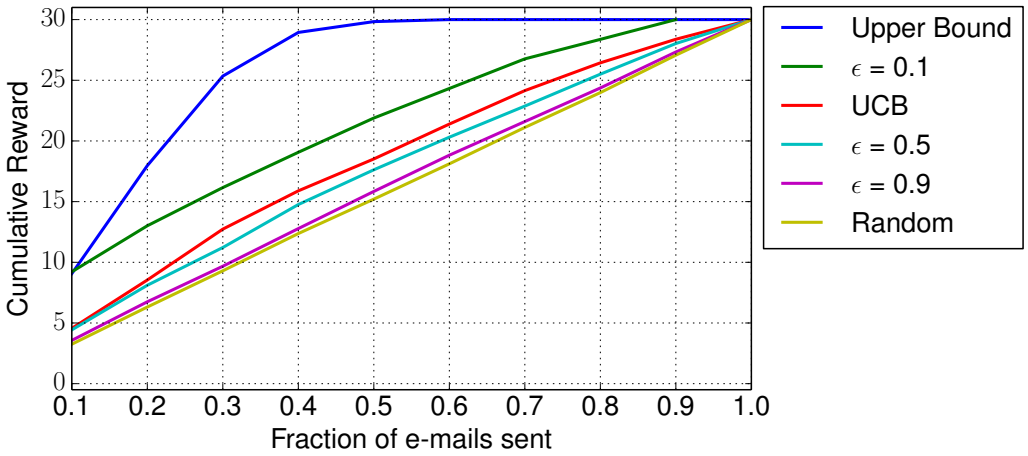


Figure 4.6: Trade-off between number of emails sent and cumulative reward.

used the WRMF algorithm available in the MyMediaLite package.[1] The running time of the WRMF algorithm is $O(c + N)$ [Hu *et al.*, 2008]. The online part of the method depends on the number s of customers selected to receive emails, and is $O(s)$, where $s \leq c$. The overall running times of the off-line and online methods are $O(c + N)$ and $O(s)$, respectively.

4.5 Final Remarks

This chapter discussed the important problem of adding value to daily-deals recommendation. Strategies to daily-deals recommendation usually relies on email marketing, and the perceived value of these emails are seriously compromised given the large number of emails sent to customers. We proposed an alternative strategy that sends emails featuring the daily-deals only to those customers that are more likely to click.

We introduce several criteria that can be used to sort customers based on aggregated statistics and past history. The best criterion at a given time step is not known in advance and may vary with time. We employ two well-known multi-armed bandit algorithms in order to chose the best-performing criterion at each time step, and evaluate their performance on real data.

The best algorithm deals efficiently with the trade-off between maximizing the cumulative reward and minimizing the number of emails sent to customers. More specifically, for the Peixe Urbano dataset, it enables a 10% reduction in terms of the number of emails sent without affecting reward. If a small reduction in terms of reward is allowed (i.e., 10%), then we may avoid sending about 30-40% of the emails.

[1] http://www.mymedialite.net

Chapter 5

Improving Daily Deals Recommendation with Suggestion Feedback

DDSs work with daily catalogs of deals, and their main advertising strategy is still email [Freed and Berg, 2012]. Every day they send registered customers featured deals that, if non-personalized, are commonly viewed as spam. In order to provide more relevant and appealing deals recommendations, current recommender systems need to learn to address three main challenges. First, most of the customers of a DDS are sporadic bargain hunters, and thus past preference data is extremely sparse and noisy. Second, deals have a short life period, and hence data is extremely volatile and scarce. Finally, customer taste and interest may undergo temporal drifts.

This chapter presents new methods to address these challenges based on a simple idea: *get customer feedback on active catalogs as soon as possible*. In the proposed methods, customer feedback is obtained by using a Multi-Armed Bandit (MAB) algorithm, which is a special class of sequential optimization problems derived from the more general paradigm of Reinforcement Learning [Sutton and Barto, 1998, Kaelbling *et al.*, 1996]. In a MAB setting, an agent interacts with the environment and, for each time slot, it is required to take one from a set of possible options (arms). For each choice, the environment returns a reward (positive or negative) to the agent. MAB searches for the best trade-off between learning about the options (exploration) and using the current knowledge to make the best choice (exploitation). The final goal of MAB algorithms is to maximize the cumulative reward in the long run, and the key challenge in bandit problems is the need for balancing the exploration and exploitation.

The proposed method is based on a greedy exploration and exploitation strategy [Kaelbling *et al.*, 1996], which decides which arm to pull, i.e., which customers should give/receive feedback and when. The idea is to ask for feedback to those customers that are likely to provide it, and then exploit the gathered feedback with customers that are likely to benefit from it. In this scenario, each customer is an arm and a positive reward is given if the customer clicks an offer sent by email and provides feedback. Considering this strategy, two big questions need to be answered: how to identify which customers should belong to the exploration and exploitation groups, and how to improve current recommended deals with customer feedback.

In order to divide customers into two groups, we first modeled the problem using a co-purchase graph that evolves as customers purchase deals. In the co-purchase graph, each node represents a customer and an edge is created if the corresponding customers purchase the same deal. Intuitively, a deal becomes more likely to be relevant to a particular customer if he/she is close to customers that have also purchased this deal. Given this scenario, customers become important to the exploration phase according to their purchase probability and centrality in the graph, i.e, the more connected they are, the more they buy and the more customers they can influence.

We use a set of complex network metrics [Albert and Barabási, 2002, Boccaletti *et al.*, 2006] that reflect these desired properties to sort customers. Based on that we choose which customers should give and which customers should receive feedback. After customer feedback is obtained in the exploration phase, rankings of traditional collaborative filtering algorithms are reordered to reflect the new feedback.

5.1 Explore-then-Exploit Recommendation

This section describes how we modeled our recommendation approach as an explore-then-exploit process. The entire recommendation process is repeated daily and the sequential steps are:

Sorting Customers We start the process by sorting customers that (i) are more likely to provide feedback, and (ii) share similar tastes with many others (i.e., their feedback is likely to benefit other cutomers).

Splitting Strategy Once we sort customers, we are now interested on splitting customers into two distinct sets: (i) customers that we ask feedback on the current catalog, and (ii) customers that will receive improved recommendations based on the previously gathered feedback.

Exploration Once we determine the set of customers that will provide feedback on the current catalog, in this step we send recommendations to this set and store their click feedback.

Exploitation Once we gather feedback on the current catalog, in this step we use this feedback to send improved recommendations to exploitation customers.

In the following sections we detail each step of our approach.

5.1.1 Sorting Customers

This step starts by building a co-purchase network as illustrated by a small example in Figure 5.1. The co-purchase network is built from purchase historical data. The network is represented by an unweighted undirected graph, where each node represents a customer and each edge connecting two customers represents a deal purchased by both customers.

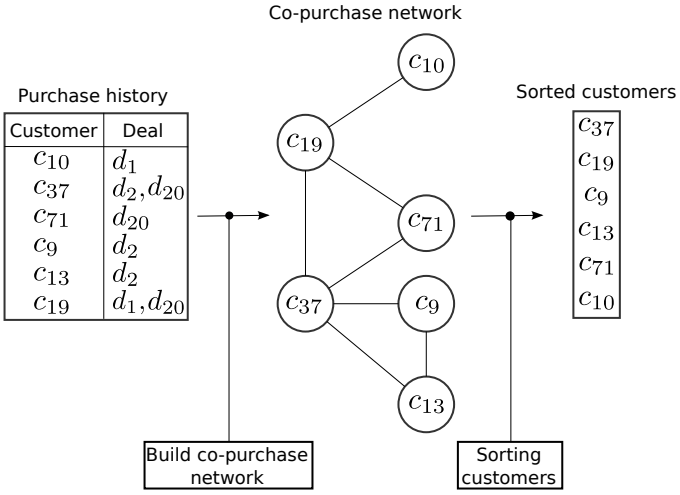


Figure 5.1: Sorting customers according to their purchase history data.

In Figure 5.1, we show six purchase records involving six customers and three deals. For instance, customer c_{37} bought deals d_2 and d_{20}. This purchase history data leads to the co-purchase network shown in Figure 5.1. As an example, customers c_{10} and c_{19} bought deal d_1, thus they are connected in the co-purchase network. Here, we sort customers according their degree in the co-purchase network.

Since we have the co-purchase network, we are able to sort customers. As sorting criteria we want that two properties be satisfied, i.e., we want separate customers that (i) are more likely to provide feedback (i.e., to purchase a deal), and (ii) share

similar tastes with many other customers (i.e., their feedback is likely to benefit other customers).

These two properties are more evident in customers that are central in the co-purchase network, where central means customers that have higher score in a given network centrality metric as described in the following. We employ five representative network centrality metrics [Freeman, 1979, Bonacich, 1987, Borgatti, 2005] to sort customers:

- The *Degree* of a customer is defined as the number of other customers having similar taste to her, that is, the number of customers she is directly connected to in the co-purchase network. This metric may be related to the probability of purchase (the higher the degree, the higher the odds of a purchase).

- The *Betweenness* of a customer, which is also called socio-centric betweenness centrality, is computed as the fraction of shortest paths between all pairs of customers that pass through the customer of interest. As expected, customers that have a high probability of occurring on a randomly chosen shortest path between two customers are said to have high betweenness centrality. A formal definition of the betweenness centrality b_u of target customer u is given by the summation of the number of geodesic paths between any two nodes s and t through u, normalized by the total number of geodesic paths between s and t, given by:

$$b_u = \sum_{s \neq v \neq t} \frac{\theta_u^{st}}{\theta^{st}}, \tag{5.1}$$

  where θ^{st} is the total number of shortest paths from node s to node t and θ_u^{st} is the number of those paths that pass through u.

- The *Eigenvector* of a customer is defined in a circular manner. In particular, the eigenvector centrality of a node is proportional to the sum of the centrality values of all its neighbors. In our co-purchase network, a central node is characterized by her connectivity to other important nodes. In this case, a central customer corresponds to a well-connected node and has a dominant influence on the surrounding sub-graph. We specify the eigenvector centrality in terms of the adjacency matrix of our co-purchase network. Let v_i be the i_{th} element of vector v, representing the centrality of node i, where $N(i)$ is the set of neighbors of node i and let A be the $n \times n$ adjacency matrix of the undirected co-purchase network. In matrix notation we have:

$$Ae = \lambda e, \tag{5.2}$$

where e is an eigenvector of A, and λ is its associated eigenvalue. Typically the largest eigenvalue is the preferred one.

- The *PageRank* of a customer is defined recursively and depends on the value of the *PageRank* of her neighbors. Formally, the *PageRank* of a customer u_t is given by:

$$u_t = \alpha \sum_j A_{tj} \frac{u_t}{k_j^{out}} + \beta, \tag{5.3}$$

where k_j^{out} is the out-degree of node j. This normalization is done to obtain a stochastic matrix (either all the columns or all the rows sum to one).

- The *Clustering Coefficient* of a customer denotes how close its neighbors are to forming a clique (i.e., a complete graph). Formally, it is defined as the ratio between the total number of connections among the neighbors of target customer u_t and the total number of possible connections between her neighbors, which is given by:

$$C_{u_t} = \frac{L}{\binom{n}{2}}, \tag{5.4}$$

where L is the number of actual links between the neighbors of u_t, and n stands for the number of neighbors of u_t.

5.1.2 Splitting Customers

Having a sorted list of customers obtained in the previous step, we separate them into exploration and exploitation sets. The process of separating customers into exploration and exploitation sets may employ one of the strategies discussed in the following paragraphs.

In Figure 5.2, we illustrate the two proposed splitting strategies following the same previously presented small example. The input of both strategies is showed in Figure 5.2(a) and refers to the array of customers sorted by their degree in the co-purchase network. The two strategies are Full Explore and k-Way Merge.

The Full Explore strategy sort customers following the order imposed by the corresponding centrality metric. In this case, most central customers (wrt. a specific centrality metric) receive email messages first, and are thus fully explored. This is the strategy initially proposed by Lacerda *et al.* [2013], and is used as a baseline in this work. We refer to this strategy as FE (Full Explore).

Since we have the sorted customers given by the Full Explore strategy, we split them following an ϵ-first strategy [Tran-Thanh *et al.*, 2010], where a pure exploration

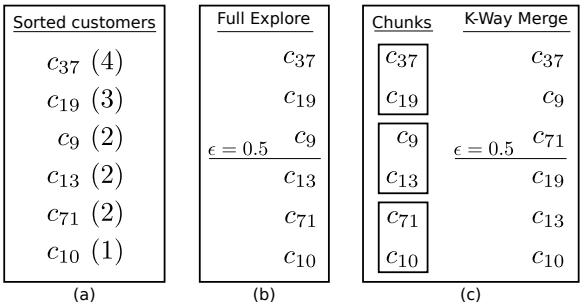


Figure 5.2: Strategies for balancing the exploration and exploitation sets. Fully exploring central customers, or merging them into exploration and exploitation. In this example, we use $k = 2$ (i.e. the size of each chunk) in the k-Way Merge strategy.

phase is followed by a pure exploitation phase. In this case, given a total of N customers, the exploration phase comprises $\epsilon \times N$ customers, while the exploitation phase includes the remaining $(1 - \epsilon) \times N$ customers.

In Figure 5.2(b), we present the final split returned by the Full Explore strategy. In particular, we consider $\epsilon = 0.5$, i.e., we equally split customers between the exploration and exploitation sets. Hence, the exploration set is given by customers c_{37}, c_{19}, and c_9 and the exploitation set is given by customers c_{13}, c_{71}, and c_{10}. The Full Explore strategy considers the customers sorted by the original sorting criterion (e.g., Degree) and split them according to the given ϵ value.

The strategy of fully exploring central customers has a limitation. As the co-purchase network is expected to have a very low diameter, central customers have high probability of being connected. For instance, if we explore the feedback given by the two most central customers in the network, the probability of both having a high number of common neighbors is very high. In this case, exploring both customers might be ineffective since their feedback is likely to be redundant. In fact, feedback from only one of these customers would be enough, as it benefits a set of customers very similar to the one influenced by the feedback of the other customer. In addition, central customers are likely to benefit the most during exploitation.

To avoid the possible redundant feedback we propose to alternate central customers into exploration and exploitation sets using a k-Way Merge algorithm. We refer to our algorithm as KWM (k-Way Merge) for short. Specifically, customers are arranged into chunks, so that some chunks are composed by central customers while other chunks contain more peripheral customers. These chunks are then merged into a single list of customers by alternating them, ensuring that both exploration and exploitation sets contain central customers. The k-Way Merge strategy also use the ϵ-first strategy to split customers between the exploration and exploitation sets.

In Figure 5.2(c), we present the two phases of the k-Way Merge strategy. First we consider that the size of each chunck is 2, i.e., $k = 2$. Hence, we have three chunks, namely, (i) c_{37}, c_{19}, (ii) c_9, c_{13}, and (iii) c_{71}, c_{10}. On the right side of Figure 5.2(c), we present the resulting merge of these chunks. Here, we also consider $\epsilon = 0.5$ and split the final merged rank half for exploration and half for exploitation. Hence, the k-Way Merge strategy returns customers c_{37}, c_9, and c_{71} for exploration and customers c_{19}, c_{13}, and c_{10} for exploitation. In this example, we avoid a possible redundant feedback from customers c_{37} and c_{19} because they are now in different sets.

5.1.3 Exploration

Since we split customers into exploration and exploitation sets, in this section we detail how we proceed to recommend deals for exploration set of customers. As already mentioned, during exploration step we intent to gather feedback from customers to improve the performance of recommendation for exploitation customers.

The Exploration phase consists of obtaining feedback from customers with regard to the deals of the day. Customers in the exploration set receive recommendations using the baseline recommender algorithm alone, i.e., no feedback is considered, featuring the current deals in the catalog. After that, the algorithm waits for feedback from these customers. Having gathered feedback from customers in the exploration set, the co-purchase network is updated and the exploitation phase starts.

In Figure 5.3 we illustrate the exploration phase of our approach. Here we are considering the Full Explore strategy to select customers to exploration set. The customers selected for exploration are c_{37}, c_{19}, and c_9. For each timestep t, where $1 \leq t \leq 3$, we select the target customer c_t to send recommendations and capture their feedback f_t. The feedback is used to update the co-purchase network. For instance, in Figure 5.3, we create two edges between customers (c_{10}, c_{71}) and (c_{10}, c_9).

5.1.4 Exploitation

Having send recommendations for all customers in exploration set, we start the Exploitation phase of our approach. In the Exploitation phase, all customers receive a message featuring a list of deals. Given the ranked list of deals initially produced by the baseline recommendation algorithm for a particular customer, we incorporate the feedback obtained in the co-purchase network in order to further personalize the ranking toward the potential interests of this customer. Specifically, given a target customer, deals in the current catalog that were purchased by her neighbors (in the

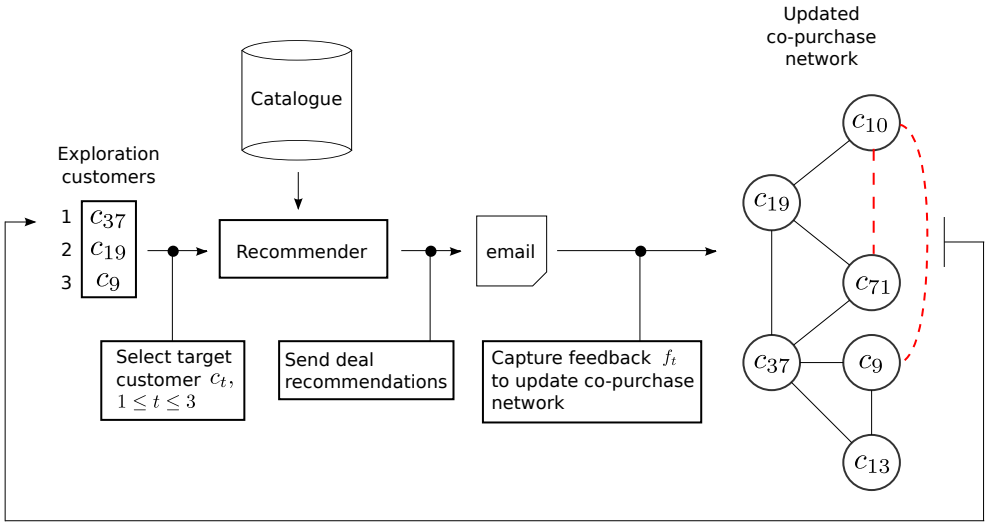


Figure 5.3: Recommendation process for customers in the exploration set. The entire process is repeated for each target customer c_t, where $1 \leq t \leq 3$.

updated co-purchase network) are placed ahead of all other recommended deals for this customer.

In Figure 5.4 we illustrate the exploitation phase of our approach. Here we are also considering the Full Explore strategy to select customers to exploitation set. The customers selected for exploitation are c_{13}, c_{71}, and c_{10}. For each timestep t, where $4 \leq t \leq 6$, we select the target customer c_t to send recommendations. Different from the process during exploration, here we used the update co-purchase network to re-rank the recommendations built by the recommender algorithm.

Exploration and exploitation are sequentially performed, i.e., every testing day starts with the exploration phase, which is then followed by the exploitation phase. In the end of each testing day, all purchases are used to update the historic data H, and the explore-then-exploit process is repeated. In the next section, we assess the recommendation effectiveness of our approach on a sample of real DDS usage data.

5.2 Experimental Evaluation

In this section we assess the effectiveness of our proposed algorithms. First, we present a detailed analysis of the dataset used in our experiments. This analysis is followed by experiments showing the differences in effectiveness between the proposed sorting criteria and splitting strategies. Finally, we show the results obtained by the proposed explore-then-exploit algorithms according to different parameter configurations.

Our evaluation follows the Interleaved Test-Then-Train methodology [Bifet *et al.*, 2010]. We first evaluate all recommendations a customer receives on day t considering

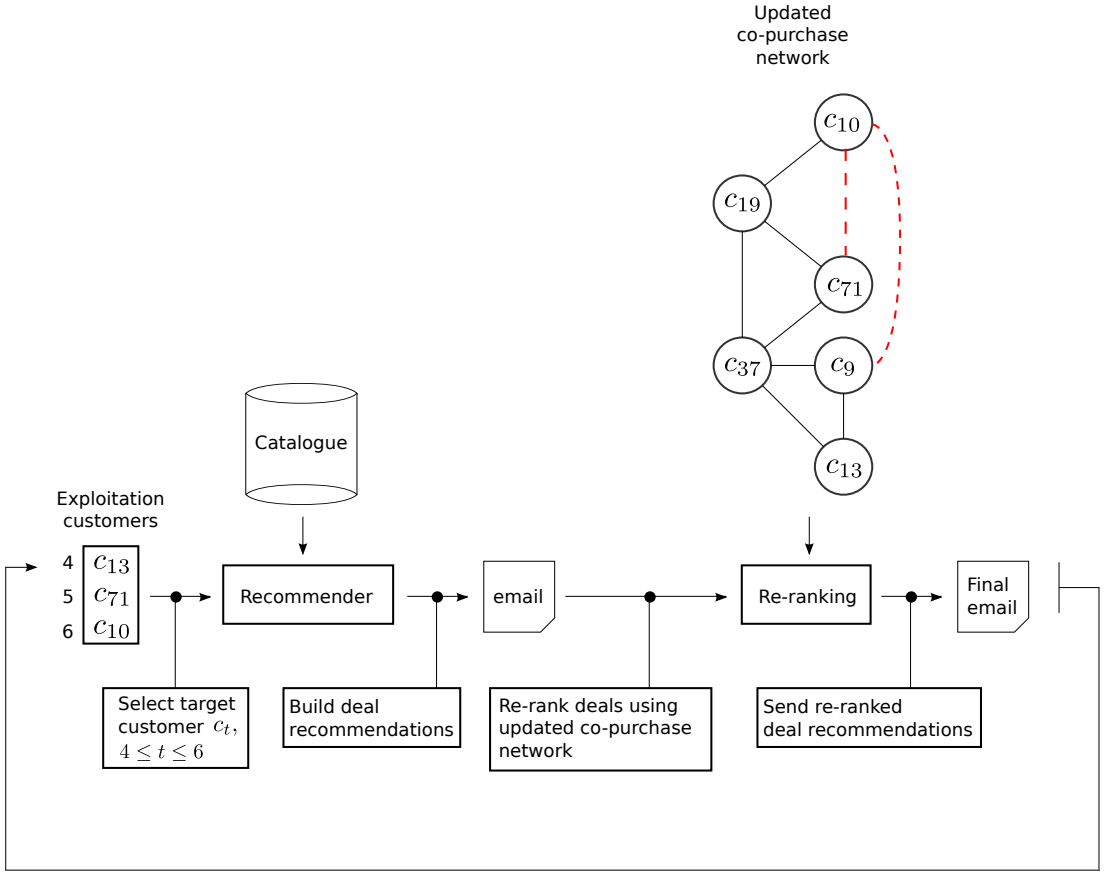


Figure 5.4: Recommendation process for customers in the exploitation set. The entire process is repeated for each target customer c_t, where $4 \leq t \leq 6$.

as training data all history from first day until day $t-1$. When considering testing day $t+1$, we follow the same methodology, i.e., we evaluate all recommendations a customer receives on day $t+1$ and we consider as training data all history from first day until day t. This process follows for all testing days. The results to be reported correspond to an average over all testing days. Statistical significance was verified using a t-test ($p < 0.05$) and the best results, including statistical ties, are shown in bold.

An effective ranked list of deals should place relevant deals in the top positions, since these are the positions more likely to be clicked by the customers [Feng *et al.*, 2007]. Therefore, we discuss the recommendation performance of our proposed algorithms in terms of standard evaluation metrics, namely precision, MAP and MRR.

5.2.1 Dataset

The experiments were performed using a real-world dataset obtained from Peixe Urbano. We used a sample of the 2-months version with 455 deals. The first 31 days were used to build the initial co-purchase network, and the last 30 days were used to test our algorithms.

In Figure 5.5, we show the evolution of the co-purchase network, in terms of the number of nodes and edges composing the network over time. On the left side we present the number of nodes (i.e. customers) added, and on the right side, we present the number of edges added (i.e. purchases). We present the evolution of graph properties as a function of testing days. From the plot, we can see that the number of new nodes added ranges from 16,976 to 31,280 and the number of new edges increases from 6,681,262 in first testing day to 16,648,608 in the last testing day. In the DDS scenario, we have both new customers and new deals available on a daily-basis, and both the number of nodes and the number of edges are always increasing throughout testing days. We can also see that the increment in the number of new nodes and new edges follows a similar growth rate.

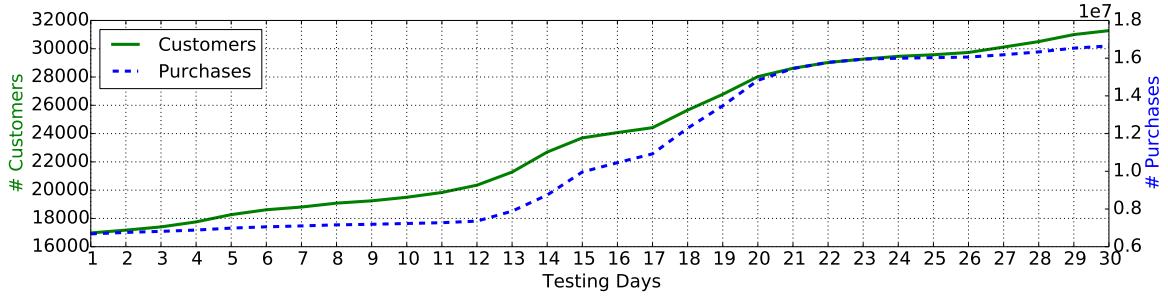


Figure 5.5: Co-purchase network evolution: nodes and edges added during the testing days.

5.2.2 Evaluation Results

The proposed explore-then-exploit algorithms for daily-deals recommendation are evaluated considering four dimensions: (i) the fraction ϵ of users to be explored, (ii) the recommendation algorithm used during the process (see Section 3.2), (iii) the five different criteria used to sort customers, and (iv) the two strategies used to separate customers into exploration and exploitation.

5.2.2.1 Framework Validation

In this section, we attempt to validate the recommendation framework proposed in Section 5.1. In particular, we aim to answer the following research questions:

Q1 How do existing recommendation algorithms perform in a DDS scenario?

Q2 Can we improve the performance of existing recommendation algorithms using our explore-then-exploit approach?

In order to answer question Q1, in Figure 5.6, we show the performance of our explore-the-exploit framework on top three representative recommendation algorithms from the literature, namely, MP, WRMF, and CLiMF. Performances is given in terms of MAP (Figure 5.6a), MRR (Figure 5.6b), and Precision (Figure 5.6c) as we vary the ϵ parameter. Recall that the larger the value of ϵ the more customers are used for exploration. To assess the effectiveness of our approach at improving existing recommendation algorithms, we executed 100 runs for each ϵ value, and in each run we randomly split users into exploration and exploitation. Therefore, we have 100 results for each ϵ value, from which we calculate the average. In Figure 5.6, our explore-the-exploit framework is applied with the Betweenness sorting criterion (see Equation (5.1)) and the k-Way Merge splitting strategy, proposed in Section 5.1.2. Results for other sorting criteria and splitting strategies are discussed later in this section.

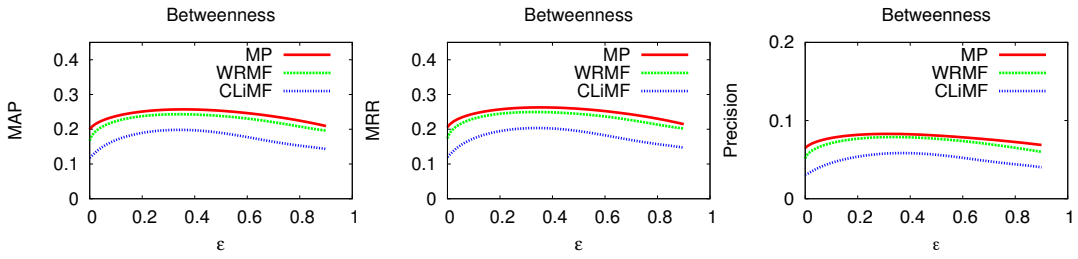


Figure 5.6: (Color online) MAP, MRR and Precision numbers for Most Popular, WRMF, and CLiMF.

By contrasting the three recommendation algorithms when ϵ = 0 in Figures 5.6 (a) (b) (c), we observe that MP is consistently the most effective, following by WRMF and then by CLiMF. This result is strictly different from the performance of these algorithms as reported in the literature. Recalling question Q1, we observe that the considered algorithms generally underperform in a DDS recommendation scenario. Indeed, WRMF is regarded as the state-of-the-art in recommendation scenarios with implicit feedback. This may happen because (i) data is extremely sparse compromising collaborative filtering algorithms, and (ii) the daily-deals recommendation scenario is more dynamic than typical recommendation scenarios in the sense that deals are volatile and new deals are constantly appearing.

Regarding question Q2, we observe that all three recommendation algorithms are improved by our proposed framework as ϵ increases, with a peak around $\epsilon = 0.3$. This result is consistent for different instantiations of our framework. In particular, Figure 5.7 provides a breakdown visualization of the performance in terms of MAP of our framework for different sorting criteria (besides Betweenness, we consider Degree, Eigenvector, Clustering Coefficient, and PageRank as sorting criteria, as introduced in Section 5.1.1) in each row and the three considered recommendation algorithms in each column.[1] In addition, besides the overall performance of our framework (i.e. the performance computed for all customers shown in blue in Figure 5.6), we also show its performance on the set of exploration customers (red in the figure) and on the set of exploitation customers (green in the figure).

From Figure 5.7, we first observe that the performance our framework on the exploited customers is naturally superior than its performance on the exploration users, in which case the underlying baseline recommendation algorithm is applied without any feedback. Indeed, as observed from the red curves the exploration performance is generally independent of the ϵ parameter. The exploitation performance on the other hand increases as the amount of feedback (i.e. the value of epsilon) increases, as depicted by the green curves. This behavior is consistent for all considered sorting criteria and baseline recommendation algorithms. Note, however, that simply exploiting more feedback does not necessarily guarantee the best overall performance as shown in the blue curves. Indeed, we observe that a value of $\epsilon \approx 0.3$ is generally the best trade-off between exploration and exploitation for all the considered instantiations of our framework, in which case the observed overall gain on top of the three recommendation baselines is maximized. Overall, these results answer question Q2 by further attesting the effectiveness of our explore-then-exploit framework for DDS recommendation.

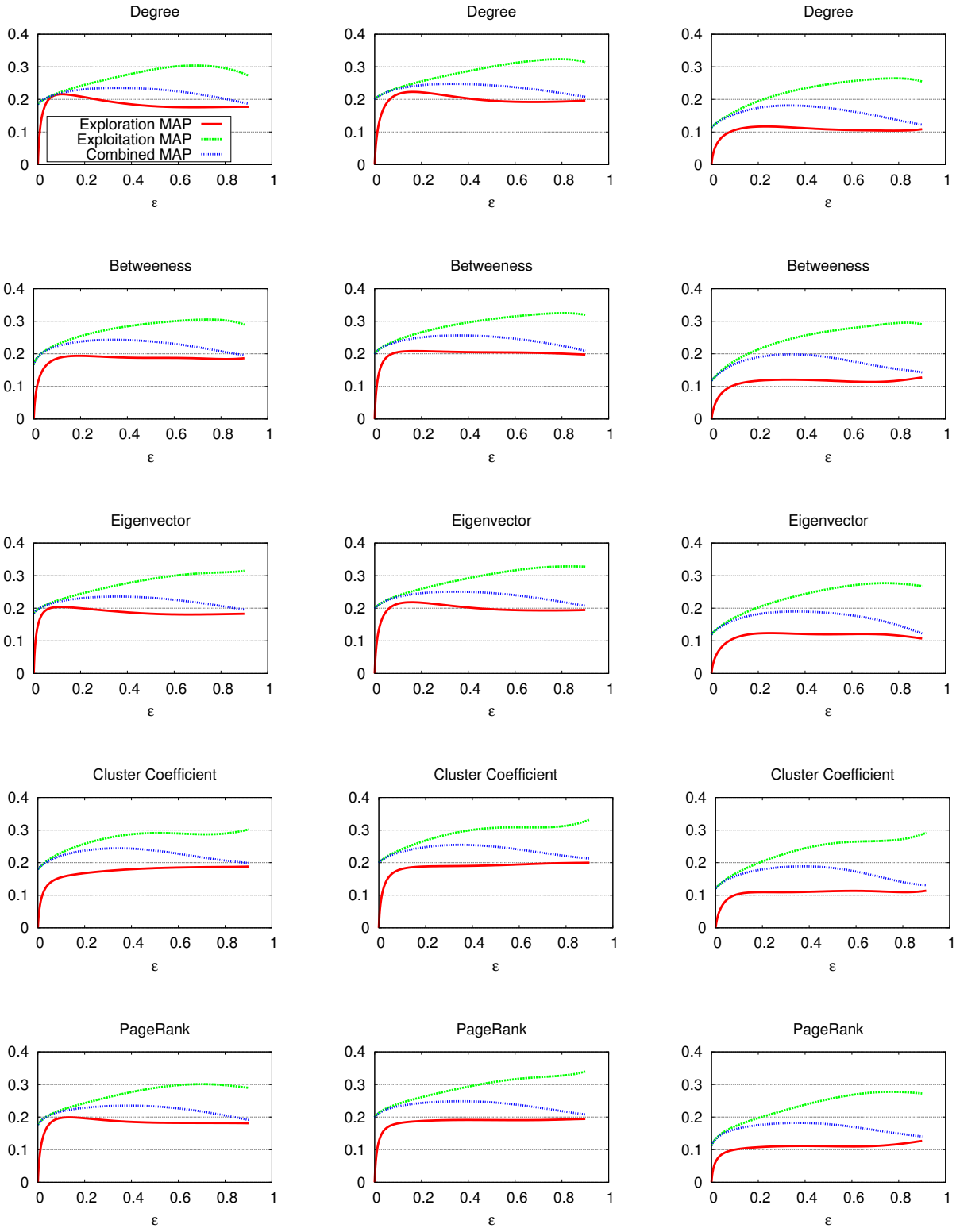
5.2.2.2 Analytical Results

The previous section demonstrated the effectiveness of our proposed framework at improving the performance of three representative recommendation algorithms from the literature in a DDS scenario. In this section, we further investigate the reasons behind the effective performance of our framework. In particular, we aim to answer the following research questions:

Q3 What is the impact of different criteria for sorting customers?

Q4 What is the impact of our proposed k-Way Merge splitting strategy?

[1] Results in terms of MRR and Precision show similar trends and are omitted for brevity. A complete analysis in terms of these three evaluation metrics is provided in Section 5.2.2.2



(a) WRMF (b) MostPopular (c) CLiMF

Figure 5.7: (Color online) MAP numbers obtained considering exploration, exploitation, and their combined values.

To answer these three research questions, Tables 5.1 and 5.2 provide a full breakdown analysis of our framework in terms of MAP and MRR, respectively. In each table, we show the performance of our framework on top of the MP recommendation algorithm, which was the best performing baseline in Section 5.2.2.1, for different choices of sorting criteria, splitting strategy, and the ϵ parameter. As sorting criteria, besides of aforementioned Betweennes, Degree, Eigenvector, Clustering Coefficient and PageRank, introduced in Section 5.1.1, we further consider a Random sorting criterion as a baseline. To assess the effectiveness of our proposed KWM splitting strategy, we compared to FE strategy proposed by Lacerda *et al.* [2013]. Regarding the ϵ values, recall that $\epsilon = 0$ corresponds to the performance obtained by the MP baseline, without using any feedback information. For each performance number, we use the symbols $\triangle$ and $\circ$ to indicate significant improvement and statistical ties, respectively, according to a t-test with $p < 0.05$. We also use symbols $\uparrow$, $\downarrow$ and $\bullet$ to indicate that the corresponding result is an improvement, decline, or remained the same considering the previous ϵ value. Finally, at the bottom of each table we provide summary figures for average gain of our framework compared to the MP baseline, as well as the average and the highest gains of KWM compared to the FE splitting strategy.

In Section 5.2.2.1, we showed that our proposed framework consistently improves on top of the different recommendation baselines for various choices of sorting criteria. To address question Q3, we further investigate the impact of each of these criteria on the performance of our framework. From Tables 5.1 and 5.2, we first note that the use of centrality metrics as sorting criteria is of utmost importance. Indeed, using the Random baseline as a sorting criterion results in negligible and no significant improvements compared the MP baseline with no feedback ($\epsilon = 0$). In contrast, all of the proposed sorting criteria based on centrality metrics significantly improve on top of MP for both MAP and MRR (Tables 5.1 and 5.2, respectively). In terms of MAP Tables 5.1, the Betweenness sorting criterion is the most effective for both FE (MAP = 0.264) and KWM (MAP = 0.267) splitting strategies. In terms of MRR 5.2, Betweenness performs the best with FE (MRR = 0.273), whereas Clustering Coefficient performs the best with KWM (MRR = 0.272). For both MAP and MRR, the best performance is consistently obtained with $\epsilon = 0.30$. Recalling question Q3, these results show that the customers sorting criterion has a significant impact on the performance of our explore-then-exploit recommendation framework, with Betweenness given the overall best performance. In particular, for the range of ϵ values, Betweenness gives a 19% improvement in terms of MAP on top of MP when using the FE splitting strategy, and 21.3% when using the KWM strategy. In terms of MRR, the average gains for Betweenness are 8.8% and 20.8% with FE and KWM, respectively. In terms of highest gains, KWM splitting strategy givens improvements in terms of MAP that range from 28.26% to 34.00%, and in terms of MRR the gain range from 27.79% to 32.79%.

Table 5.1: MAP numbers, considering the Most Popular algorithm. Symbols △ and ∘ indicate significant improvement and statistical tie, respectively, with 95% confidence. Symbols ↑, ↓ and • indicate that the corresponding result is an improvement, decline, or remained the same considering the previous ϵ value.

	Random	Degree		Betweenness		Eigenvector		Cluster Coefficient		PageRank	
ϵ	–	FE	KWM	FE	KWM	FE	KWM	FE	KWM	FE	KWM
0.00	0.199 •	0.199 •	0.199 •	0.199 •	0.199 •	0.199 •	0.199 •	0.199 •	0.199 •	0.199 •	0.199 •
0.01	∘ 0.201 •	∘ 0.201 •	△ 0.211 ↑	△ 0.214 ↑	△ 0.213 ↑	∘ 0.203 •	△ 0.208 ↑	△ 0.218 ↑	△ 0.209 ↑	∘ 0.203 ↑	△ 0.212 ↑
0.05	∘ 0.201 •	∘ 0.202 •	△ 0.225 ↑	△ 0.224 ↑	△ 0.235 ↑	∘ 0.204 •	△ 0.234 ↑	△ 0.242 ↑	△ 0.229 ↑	∘ 0.204 ↑	△ 0.232 ↑
0.10	∘ 0.202 •	∘ 0.206 •	△ 0.241 ↑	△ 0.240 ↑	△ 0.245 ↑	∘ 0.206 •	△ 0.245 ↑	△ 0.249 ↑	△ 0.241 ↑	△ 0.213 ↑	△ 0.242 ↑
0.20	∘ 0.201 •	△ 0.213 ↑	△ **0.258** ↑	△ 0.258 ↑	△ 0.259 ↑	△ 0.216 ↑	△ 0.251 ↑	△ 0.252 ↑	△ 0.255 ↑	△ 0.222 ↑	△ 0.247 ↑
0.30	∘ 0.200 •	△ 0.224 ↑	△ 0.250 ↓	△ 0.264 ↑	△ **0.267** ↑	△ 0.223 ↑	△ **0.258** ↑	△ 0.248 ↓	△ **0.265** ↑	△ 0.237 ↑	△ 0.251 ↑
0.40	∘ 0.201 •	△ 0.231 ↑	△ 0.249 ↓	△ 0.255 ↓	△ 0.259 ↓	△ 0.229 ↑	△ 0.254 ↓	△ 0.248 •	△ 0.263 ↓	△ 0.237 •	△ **0.256** ↑
0.50	∘ 0.202 •	△ 0.228 ↓	△ 0.247 ↓	△ 0.252 ↓	△ 0.259 •	△ 0.229 •	△ 0.250 ↓	△ 0.243 ↓	△ 0.254 ↓	△ 0.240 ↑	△ 0.251 ↓
0.60	∘ 0.201 •	△ 0.227 ↓	△ 0.241 ↓	△ 0.244 ↓	△ 0.248 ↓	△ 0.226 ↓	△ 0.245 ↓	△ 0.240 ↓	△ 0.240 ↓	△ 0.235 ↓	△ 0.242 ↓
0.70	∘ 0.201 •	△ 0.224 ↓	△ 0.230 ↓	△ 0.233 ↓	△ 0.240 ↓	△ 0.220 ↓	△ 0.232 ↓	△ 0.231 ↓	△ 0.229 ↓	△ 0.232 ↓	△ 0.234 ↓
0.80	∘ 0.202 •	△ 0.213 ↓	△ 0.220 ↓	△ 0.221 ↓	△ 0.226 ↓	△ 0.211 ↓	△ 0.220 ↓	△ 0.222 ↓	△ 0.219 ↓	△ 0.221 ↓	△ 0.224 ↓
0.90	∘ 0.201 •	∘ 0.204 •	∘ 0.208 •	∘ 0.207 ↓	△ 0.209 ↓	∘ 0.202 •	∘ 0.207 •	△ 0.212 ↓	△ 0.213 ↓	∘ 0.205 •	∘ 0.208 •
Avg. Gain over MP		△ 8.2%	△ 17.60%	△ 19.07%	△ **21.23%**	△ 7.9%	△ 18.7%	△ 18.7%	△ 19.3%	△ 11.3%	△ 18.0%
Highest Gain over MP		△ 15.59%	△ 29.36%	△ 32.51%	△ 34.00%	△ 14.94%	△ 29.54%	△ 26.31%	△ 32.95%	△ 18.74%	△ 28.26%
Avg. Gain over FE		–	△ 8.7%	–	∘ 1.8%	–	△ **10.0**%	–	∘ 3.4%	–	△ 6.0%
Highest Gain over FE		–	△ **21.0**%	–	△ 4.7%	–	△ 19.2%	–	△ 7.0%	–	△ 13.9%

Table 5.2: MRR numbers, considering the Most Popular algorithm. Symbols △ and ∘ indicate significant improvement and statistical tie, respectively, with 95% confidence. Symbols ↑, ↓ and • indicate that the corresponding result is an improvement, decline, or remained the same considering the previous ϵ value.

	Random	Degree		Betweenness		Eigenvector		Clustering Coefficient		PageRank	
ϵ	–	FE	KWM	FE	KWM	FE	KWM	FE	KWM	FE	KWM
0.00	0.205 •	0.205 •	0.205 •	0.205 •	0.205 •	0.205 •	0.205 •	0.205 •	0.205 •	0.205 •	0.205 •
0.01	∘ 0.207 •	∘ 0.206 •	△ 0.218 ↑	△ 0.220 ↑	△ 0.220 ↑	∘ 0.209 •	△ 0.214 ↑	△ 0.225 ↑	△ 0.215 ↑	∘ 0.208 •	△ 0.219 ↑
0.05	∘ 0.206 •	∘ 0.207 •	△ 0.232 ↑	△ 0.230 ↑	△ 0.242 ↑	∘ 0.209 •	△ 0.242 ↑	△ 0.250 ↑	△ 0.236 ↑	∘ 0.209 •	△ 0.240 ↑
0.10	∘ 0.207 •	∘ 0.212 •	△ 0.249 ↑	△ 0.246 ↑	△ 0.252 ↑	∘ 0.212 •	△ 0.252 ↑	△ 0.256 ↑	△ 0.249 ↑	△ 0.218 ↑	△ 0.250 ↑
0.20	∘ 0.207 •	△ 0.219 ↑	△ **0.265** ↑	△ 0.264 ↑	△ 0.265 ↑	△ 0.221 ↑	△ 0.259 ↑	△ 0.258 ↑	△ 0.261 ↑	△ 0.228 ↑	△ 0.253 ↑
0.30	∘ 0.207 •	△ 0.231 ↑	△ 0.255 ↓	△ **0.273** ↑	△ 0.270 ↑	△ 0.228 ↑	△ **0.265** ↑	△ 0.254 ↓	△ **0.272** ↑	△ 0.243 ↑	△ 0.258 ↑
0.40	∘ 0.207 •	△ 0.236 ↑	△ 0.254 ↓	△ 0.260 ↓	△ 0.265 ↓	△ 0.234 ↑	△ 0.260 ↓	△ 0.253 ↓	△ 0.270 ↓	△ 0.238 ↓	△ **0.262** ↑
0.50	∘ 0.208 •	△ 0.234 ↓	△ 0.252 ↓	△ 0.258 ↓	△ 0.265 •	△ 0.234 •	△ 0.256 ↓	△ 0.248 ↓	△ 0.260 ↓	0.246 ↑	△ 0.257 ↓
0.60	∘ 0.208 •	△ 0.233 ↓	△ 0.246 ↓	△ 0.250 ↓	△ 0.253 ↓	△ 0.232 ↓	△ 0.250 ↓	△ 0.246 ↓	△ 0.246 ↓	△ 0.241 ↓	△ 0.247 ↓
0.70	∘ 0.207 •	△ 0.230 ↓	△ 0.235 ↓	△ 0.238 ↓	△ 0.245 ↓	△ 0.225 ↓	△ 0.238 ↓	△ 0.237 ↓	△ 0.235 ↓	△ 0.238 ↓	△ 0.235 ↓
0.80	∘ 0.207 •	△ 0.218 ↓	△ 0.225 ↓	△ 0.226 ↓	△ 0.232 ↓	△ 0.216 ↓	△ 0.225 ↓	△ 0.227 ↓	△ 0.224 ↓	△ 0.227 ↓	△ 0.222 ↓
0.90	∘ 0.207 •	∘ 0.210 •	∘ 0.213 •	∘ 0.212 •	△ 0.214 ↓	∘ 0.208 •	∘ 0.213 •	△ 0.217 ↓	△ 0.218 ↓	∘ 0.210 •	∘ 0.213 •
Avg. Gain over MP		△ 8.0%	△ 17.1%	△ 18.8%	△ **20.8%**	△ 7.7%	△ 18.7%	△ 18.4%	△ 19.1%	△ 11.1%	△ 17.8%
Highest Gain over MP		△ 15.26%	△ 29.07%	△ 33.12%	△ 31.82%	△ 14.43%	△ 29.36%	△ 26.02%	△ 32.79%	△ 19.78%	△ 27.79%
Avg. Gain over FE		–	△ 8.5%	–	∘ 1.7%	–	△ **10.11**%	–	∘ 0.5 %	–	△ 6.0%
Highest Gain over FE		–	△ **20.1**%	–	△ 5.3%	–	△ 19.2%	–	△ 7.2%	–	△ 14.8%

5.2.2.3 The Impact of Sorting Criteria

A natural question for further improving the performance of our proposed framework is whether we could combine multiple sorting criteria as arms within our MAB approach. In the following we analyze the impact of different sorting criteria. We used the five aforementioned centrality metrics sort customers. This first experiment shows how similar are the rankings generated by different criteria. This is important because MAB algorithms assume that the arms are independent from one another. We use a rank correlation metric, namely Kendall Tau, to measure the extent to which the order of the observations (or customers) in a ranking A differs from the order of the observations in B, as defined in Section 4.3.1. In Table 5.3, we show all the values of τ are very close to 0, showing that the rankings generated are not correlated. As previously discussed, this result attests the utility of the several considered centrality rankings as arms within our MAB approach.

Table 5.3: Kendall Tau considering rankings Generated by different Centrality Measures

	PageRank	Betweenness	Clustering Coefficient	EigenVector	Degree
PageRank	1.00000	0.00365	-0.00128	0.00017	-0.00053
Betweenness	0.00365	1.00000	-0.00069	0.00042	0.00048
Clustering Coefficient	-0.00128	-0.00069	1.00000	-0.00204	-0.00148
EigenVector	0.00017	0.00042	-0.00204	1.00000	-0.00011
Degree	-0.00053	0.00048	-0.00148	-0.00011	1.00000

5.2.2.4 The Impact of Splitting Strategies

To address question Q4, we contrast the performance of our framework using KWM splitting strategy introduced in Section 5.1.2 to its performance using the FE strategy proposed by Lacerda *et al.* [2013] across the five considered sorting criteria. From Tables 5.1 and 5.2, we observe that the KWM strategy consistenltly improves compared to FE for almost all sorting criteria and values of ϵ. The average gains in terms of MAP range from 1.8% to 10.0% and are significant in most cases (the only exception are when using the Betweenness and Clustering Coefficient sorting criteria, in which cases the average gains are not significant). In turn, the highest MAP improvements range from 4.5% (Betweenness) to 21% (Degree). In terms of MRR, the average gains range from 1.7% to 10.11%, once again significant in most cases, except for Betweenness and Clustering Coefficient. Similarly to the MAP results, the highest gains in terms of MRR range from 5.3% (Betweenness) to 20.1% (Degree). Recalling question Q4, these

results attest the effectiveness of our proposed KWM splitting strategy and its positive impact on the overall performance of our recommendation framework.

To understand the improved benefits of the KWM splitting strategy, we compare the ranking of customers when using different k values and the impact of this rankings generated co-purchase network. Remember that the k parameter refers to the size of each window in the latter strategy. In particular, the fixed-size window splitting strategy was proposed based on the hypothesis that, in highly connected graphs, central users are usually connected to a large number of common users, and hence receiving feedback only from central users may not be the most effective approach. In other words, we hypothesize that the more dispersed the explored customers the more their feedback will influence the remaining customers.

In Figure 5.8, we show evidence that supports this hypothesis. For each graph, we measured the density of the subgraph used during the exploration phase, i.e., built from users $\{u_1, u_2, \ldots, u_{b-1}\}$. The density of a graph G is given by the fraction of the number of edges in G and the maximal possible number of edges in G. Since our graph is undirected, the density is computed as:

$$Density(G) = \frac{2 \times |E|}{V \times (V - 1)} \tag{5.5}$$

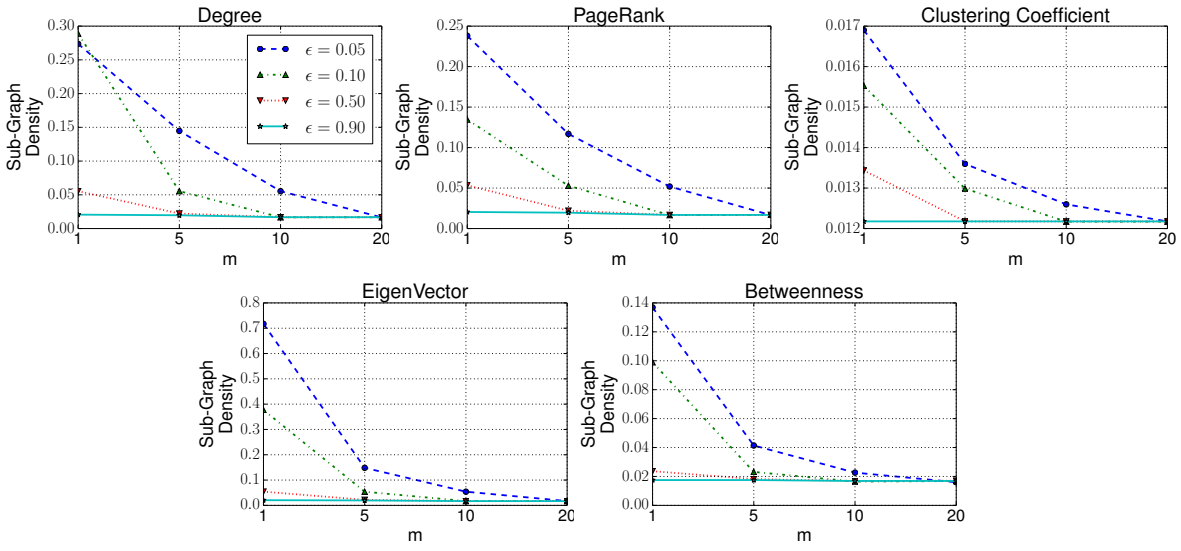


Figure 5.8: Density of the subgraphs versus window size k. We consider the graph generated during the exploration phase by different ordering metrics.

The denser the sub-graph, the higher the probability of two central users having a large number of common neighbors. In the first graph, for different values of ϵ (which have direct impact on the size of the subgraph used in the exploration phase), we show

how the density of the graph decreases as we reorder users by interposing different fixed-size windows. The larger the size of the window, the less dense the graph becomes, and hence the probability of user feedback affecting different users increases. This result demonstrates the potential benefit of our k-Way Merge strategy for splitting users into exploration and exploitation groups. In the next section, we thoroughly assess the effectiveness of our explore-the-exploit approach in an online recommendation scenario.

5.3 Time Performance

In this section we discuss the time performance of the proposed approach for deal recommendation, and analyze the four steps of the process presented in Section 5.1. We start by sorting customers off-line according to their purchase history data as illustrated in Figure 5.1 on Page 63. This step is divided into three phases: building the co-purchase graph, calculating the centrality metrics and sorting users according to the metrics. The cost to build the graph is $O(c + n)$, where c is the number of customers and n is the number of deals. The centrality metrics were calculated using the implementations available in the SNAP library.[2] The most expensive metric is Betweenness, which is $O(c^2 \log c + ce)$, where e is the number of edges of the co-purchase network. The phase of sorting customers is $O(c \log c)$. The overall complexity of the sorting step is $O(c^2 \log c + ce)$. The sorting step is performed off-line.

The second step of our approach is the splitting strategies to separate customers into exploration and exploitation sets, illustrated in Figure 5.2 on Page 66. Our Full Explore strategy is $O(1)$ because it does not re-rank the previously computed ranking of customers. The k-Way Merge strategy is $O(ck \log k)$ [Knuth, 1973], where k is the number of chunks. The splitting step is performed off-line.

The third step of our approach is the recommendation process for customers in the exploration set, illustrated in Figure 5.3 on Page 68. The recommender is built off-line using the implementation available in the MyMediaLite package.[3] The most expensive recommendation algorithm used in our investigation is WRMF, which is $O(c + N)$, where N is the number of non-zero observations in the customer/deal matrix [Hu *et al.*, 2008]. The online part of this step refers to sending emails for each customer and updating the co-purchase network. The running time to send emails is $O(c)$. The running time to update the co-purchase network is $O(p)$, where p is the number of co-purchases that happened during exploration.

The last step of our process is the recommendation process for customers in the exploitation set, illustrated in Figure 5.4 on Page 69. We used the recommender trained during exploration to compute recommendations. Hence, the additional step here is

[2] http://www.snap.stanford.edu/snap/
[3] http://www.mymedialite.net

the re-ranking of recommendations using the co-purchase network. This is a sorting step that is $O(r \log r)$, where r is the number of deals recommended to customers. The overall running time of the off-line and online methods are $O(c^2 \log c + ce)$ and $O(c)$, respectively.

5.4 Final Remarks

In this chapter, we focused on the important problem of suggesting relevant and appealing products and services to potential customers, considering the particularly challenging scenario of daily-deals recommendation. Specifically, we consider the task of sending personalized email messages featuring potentially relevant deals to users. In this case, we can impose a restriction on the order that users receive their messages, that is, some users receive their messages before others.

We propose explore-then-exploit recommendation algorithms that are devised to: (i) gather feedback from users that receive their messages first (i.e., during the exploration phase), and (ii) use the gathered feedback in order to send personalized messages to the remaining users (i.e., during the exploitation phase). An important advantage of our method is that it is agnostic in terms of the recommendation algorithm used, i.e., we are able to use any recommendation algorithm proposed in the literature.

There is a trade-off between exploration and exploitation, in the sense that the more customers are explored, more feedback is gathered but less feedback is indeed used. To deal with this trade-off, we propose sorting customers according to their centrality in an evolving co-purchase graph, considering that more central customers are those that share the same taste with more customers. According to different centrality metrics, we split customers into exploration and exploitation by taking into account how central they are in the current co-purchase network, so that we achieve a proper balance between the amount of feedback acquired during exploration and the amount of feedback indeed used during exploitation.

We use a sample of the 2-month real data obtained from Peixe Urbano, the largest daily-deals site in Brazil, and show that: (i) the recommendation performance of existing algorithms is no better than the naive strategy of suggesting non-personalized deals for all customers (i.e., all customers receive the same email featuring the same deals), and (ii) the proposed exploration-exploitation algorithms are effective and well-suited to the DDSs recommendation scenario, providing improvements ranging from 7.9% to 34%.

Chapter 6

Conclusions and Future Work

Daily-Deals Sites (DDSs) provide online discount coupons for diverse products and services, and operate as a mediator between local merchants and customers. This thesis focused on two problems related to the Daily-Deals business: how to *maximize revenue* and *increase customer satisfaction*. We proposed three different methods to deal with the challenges associated with these problems, namely deal size prediction, email prioritization and deals recommendation.

The problem of deal size prediction, i.e., predicting the number of coupons a deal is expected to sell, is crucial to DDS administrators. With this information, administrators can simulate different sets of catalogs, choosing the one that maximizes profitability. The drawback of current methods in the literature is that they ignore complex interactions between deals, an issue solved by the method proposed here.

A variety of channels are used to suggest deals to customers in DDSs, but the primary approach is yet to send email featuring the deals of the day [Freed and Berg, 2012]. As customers are overloaded with non-personalized daily messages, with time they start considering these messages as spam. In order to avoid that, the second problem solved in this thesis is prioritizing emails, i.e.. choosing which customers should receive emails first (or at all) according to their probability of clicking the emails.

Finally, our last method tackles the problem of the email content, i.e., the content of the recommendations sent. Although there is a variety of recommendation algorithms available, the daily-deals scenario has characteristics that make the problem particularly challenging, including extremely sparse, noisy and very dynamic data. We solve this problem by proposing a method that benefits from customer feedback to improve recommendations.

The remainder of this section shows the main contributions and conclusions of this thesis, and discusses various directions of future research.

6.1 Summary of Contributions

This section summarizes the main contributions of this thesis.

The exploitation of Markets to tackle deal interaction: When simulating deals catalogs, DDS administrators have difficulties in dealing with complex interactions among deals, as explained in Chapter 3. This is also true for automatic methods for deal size prediction. The two main types of interactions are: (i) deals that compete for customer preference (possibly decreasing their sizes), or (ii) deals that complement other deals (possibly leveraging their sizes). We dealt with this problem by using the concept of Markets, which is defined as a set of deals that attract the interest of customers with similar preferences. As showed in Section 3.2, we proposed a content-based method to identify markets by exploring deal structure, and showed that their use minimizes deal prediction error.

A new method for Deal Size Prediction: Still in Chapter 3, we incorporated the concept of markets when automatically predicting deal size. In Section 3.3, we generated different predictors for different markets, and then presented an expectation-maximization algorithm that models market interplay and intra-market competition by minimizing the error in prediction. The results showed that the method outperforms current state-of-the-art methods that do not consider deals interactions.

Criteria to sort customers according to their probability of email click: In Chapter 4, we introduced several criteria that can be used to sort customers based on aggregated statistics and past history according to their intention to click a link and buy a product. The best criterion at a given time step is not known in advance and may vary with time. In isolation, the time since the last purchase showed to have a high predictive power, but it was combined with other criteria in a multi-armed bandit strategy to prioritize emails online.

A new method for Prioritizing Emails: Strategies to daily-deals recommendation usually rely on email marketing, and the perceived value of these emails is seriously compromised given the large number of non-personalized emails sent to customers. Given different criteria to sort customers according to their intention to buy, in Section 4.2 we proposed an alternative strategy that sends emails featuring the daily-deals only to customers that are more likely to click the email using different multi-armed bandit algorithms to decide the order and the number of emails that should be sent.

Results showed that we can significantly reduce the number of emails sent while keeping the same number of clicks.

The exploitation of Customer Feedback for DDS Recommendation: In Chapter 5, we tested whether customer feedback on current deals at the catalog can improve the recommendation in DDS scenarios. We did that by modeling the problem using a co-purchase graph, and concluded feedback is indeed important.

A new method for DDS Recommendation: Having customer feedback, we tackled the problem of suggesting relevant and appealing products and services to potential DDS customers via email in Section 5.1. We propose exploration-exploitation recommendation algorithms devised to: (i) gather feedback from customers that receive their messages first (i.e., during the exploration phase), and (ii) use the gathered feedback in order to send personalized messages to the remaining customers (i.e., during the exploitation phase). There is a trade-off between exploration and exploitation, in the sense that the more customers are explored, more feedback is gathered but less feedback is indeed used. The method showed that the recommendations performed are better than using stand-alone state-of-the-art recommendation algorithms.

Use of Peixe Urbano real-world dataset: Throughout Chapters 3, 4 and 5, the proposed methods were tested in a real-world dataset provided by Peixe Urbano. In some cases, finding baselines for our methods was not trivial, since most problems in DDS are not studied due to lack of real-world data.

6.2 Summary of Conclusions

In this section, we summarize the main conclusions drawn from the evaluation of our three methods proposed in Chapters 3, 4 and 5. The results corroborate with our initial goal, which was to maximize DDSs revenue and improve customer satisfaction.

Deals Interaction and Markets: A characterization of deals showed that the terms used to describe them can discriminate deals into markets. These markets are used to create separate models for deal size prediction for each market. These different models can then be combined to improve deal size predictions while taking deals interactions into account. Experiments on three real datasets showed the proposed method obtained gains in root mean squared error that range from 8.18% to 17.67% over previously proposed methods.

Prioritizing Emails: We model the problem of prioritizing email as a sequential decision problem, and employed two well-known multi-armed bandit algorithms as a way to map customers to criteria, in order to maximize the click rate by selecting the best criterion to apply at each time step. The best algorithm deals efficiently with the trade-off between maximizing the cumulative reward and minimizing the number of emails sent to customers. For Peixe Urbano, the proposed method enables a 10% reduction in terms of the number of emails sent without affecting reward. If a small reduction in terms of reward is allowed (i.e., 10%), then we may avoid sending about 30-40% of the emails.

Email Recommendation in DDSs: When tackling the problem of suggesting relevant and appealing products and services to potential DDS customers via email, we also used multi-armed bandit algorithms. First, we modeled the problem as a co-purchase graph, and worked with information provided by the graph in two phases. First, we sorted customers according to their centrality in an evolving co-purchase graph, considering that more central customers are those that share the same taste with more customers. According to different centrality metrics, we divide customers into exploration and exploitation by taking into account how central they are in the current co-purchase network, so that we achieve a proper balance between the amount of feedback acquired during exploration and the amount of feedback indeed used during exploitation. Experiments in the Peixe Urbano dataset showed that: (i) the recommendation performance of existing algorithms is no better than the naive strategy of suggesting non-personalized deals for all customers (i.e., all customers receive the same message featuring the same deals), and (ii) the proposed exploration-exploitation algorithms are very effective and well-suited to the daily-deals recommendation scenario, providing improvements in precision ranging from 7.9% to 34.0%.

6.3 Directions for Future Research

This section discusses possible directions for future research, directly inspired by or stemming from the results of this thesis. These directions cover topics related to deal size prediction, email prioritization and daily-deals recommendation.

Deal Features In Chapter 3, we used only textual features to represent deals and LDA to identify deal markets. However, markets can be characterized by more than the vocabulary used to describe them. For instance, in Tables 3.3 and 3.4, we observe that the average price of deals is also very different among markets. Hence, it might

be interesting to describe deals through textual and non-textual features. One idea to model this problem is, for each market, to build a representative vector that contains, for example, the average value of all non-textual features. In a second step, we can combine the textual vector of each deal with the representative vector of the market it belongs to. In this case, we can build a single predictor, since markets are already represented in the deal vectors. A possible advantage of this approach is that we need to learn only one SVR for all deals.

Deal Structure In Chapter 3, we exploit the structure of deals, i.e., title, merchant's name, description, and highlight to describe them. We used a previously proposed heuristic to assign weights to terms present in each structure. However, this heuristic was not proposed to assess term weighting for collections of deals. We could investigate other methods to assign weights to terms or to select only terms useful to identify deal markets. A machine learning algorithm could be used for this purpose. For instance, in Lacerda *et al.* [2006], the authors proposed a Genetic Programming framework to explore different weight schemes of advertising structure in the computational advertising context. Another possible line of investigation is to learn a model to classify the terms according to their importance in describing deals, given the market identification task.

Multiple Market Assignment In Chapter 3, we have used a Latent Dirichlet Allocation (LDA) strategy to identify markets in our framework of deal size prediction. We assign each deal to the most representative market. However, a deal may be present in more than one market. For instance, a deal may belongs to markets "restaurants" and "Chinese food", or "beauty" and "hair cut". Hence, the idea is to study a more flexible assignment of deals to markets allowing a deal to be present in more than one market. We could use simple strategies to cluster deals and, for each deal, compute its proximity to the centroid of clusters.

Learning to Rank Criteria In Chapters 4 and 5 we sort customers using a single criterion. In Chapter 4, the sorting criteria were related to the domain of DDS, such as last purchase date and total number of purchases. In Chapter 5, our criteria were based on centrality metrics, such as pagerank and betweenness. In both cases, we assigned a weight given by a single criteria to each user. However, we could use more than one criterion and apply different learning to rank algorithms to combine the weights output by each criterion. The idea is that different orders provided by the learning to rank algorithms would be the arms, and the MAB algorithm would learn their best order.

Creating Deals Catalogs In Chapter 3 we argument that the deal size prediction problem may be used to select a set of deals that are the most profitable, but we did not propose a systematic way to create this catalog. Both deal selection (considers one day) and deal scheduling (considers interactions among deals in different days) can be performed. The current solutions can be directly used for deal selection, but it can also be adapted to perform deal scheduling in multiple days.

Contextual Multi-Armed Bandit Algorithm Chapter 4 and 5 used multi-armed bandit algorithms for sequential decision problems, namely prioritizing emails and the search for feedback. However, both algorithms did not take into account any context about the task, as done by the recommendation algorithms described in Section 2.2.2. In our context, we could consider the current catalog as context for the recommender. For the prioritizing email task, the idea would be to sort user according to purchase history **and** the current catalog. For getting user feedback, we would also look at the most central users that would give feedback for the current catalog.

Combining Purchase Probability and Feedback The methods presented in Chapters 4 and 5 could be integrated in a way that we would reduce the number of emails sent while obtaining as much feedback as possible, improving the recommendation in the emails. For that, we would have to investigate which criterion could simultaneously return users with high purchase probability and chance of providing feedback. A naive solution for this problem would be to use linear combinations of the criteria used in both methods.

Bibliography

Adomavicius, G. and Kwon, Y. (2009). Toward more diverse recommendations: Item re-ranking methods for recommender systems. In *Proceedings of the 19th Workshop on Information Technology and Systems*, pages 89--97.

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 Transactions on Knowledge and Data Engineering*, 17(6):734--749.

Agarwal, D. and Chen, B.-C. (2010). flda: matrix factorization through latent dirichlet allocation. In *Proceedings of the 3rd ACM International Conference on Web Search and Web Data Mining*, pages 91–100.

Aizenberg, N., Koren, Y., and Somekh, O. (2012). Build your own music recommender by modeling internet radio streams. In *Proceedings of the 21st International Conference on World Wide Web*, pages 1--10.

Albert, R. and Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47--97.

Ariely, D. (2008). *Predictably Irrational: The Hidden Forces That Shape Our Decisions*. HarperCollins, 1 edition.

Audibert, J.-Y., Munos, R., and Szepesvári, C. (2009). Exploration–exploitation trade-off using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876--1902.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multi-armed bandit problem. *Machine learning*, 47(2-3):235--256.

Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (2011). *Modern Information Retrieval: The Concepts and Technology Behind Search*. Addison Wesley, second edition.

Barragáns-Martínez, A. B., Costa-Montenegro, E., Burguillo, J. C., Rey-López, M., Mikic-Fonte, F. A., and Peleteiro, A. (2010). A hybrid content-based and item-based collaborative filtering approach to recommend tv programs enhanced with singular value decomposition. *Information Sciences*, 180(22):4290--4311.

Basak, D., Pal, S., and Patranabis, D. C. (2007). Support vector regression. *Neural Information Processing-Letters and Reviews*, 11(10):203--224.

Belch, G. E., Belch, M. A., Kerr, G. F., and Powell, I. (2008). *Advertising and promotion: An integrated marketing communications perspective*. Mcgraw-Hill.

Bennett, J. and Lanning, S. (2007). The Netflix prize. In *Proceedings of KDD cup and workshop*, pages 3--6.

Berry, D. A. and Fristedt, B. (1995). Bandit problems. *IEEE Transactions on Automatic Control*, 29(20):199--209.

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). MOA: massive online analysis. *Journal of Machine Learning Research*, 11(1):1601--1604.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(1):993--1022.

Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., and Hwang, D.-U. (2006). Complex networks: Structure and dynamics. *Physics reports*, 424(4):175--308.

Bonacich, P. (1987). Power and centrality: A family of measures. *The American Journal of Sociology*, 92(5):1170--1182.

Borgatti, S. (2005). A graph-theoretic perspective on centrality. *Social Networks*, 28(4):466--484.

Bouneffouf, D., Bouzeghoub, A., and Gançarski, A. L. (2012). A contextual-bandit algorithm for mobile context-aware recommender system. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, pages 324--331.

Boyd-Graber, J. L., Blei, D. M., and Zhu, X. (2010). Topic models for word sense disambiguation and token-based idiom detection. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1138--1147.

Bubeck, S. and Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Machine Learning*, 5(1):1--122.

Burke, R. (2002a). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331--370.

Burke, R. (2002b). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331--370.

Byers, J., Mitzenmacher, M., and Zervas, G. (2012a). Daily deals: prediction, social diffusion, and reputational ramifications. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, pages 543--552.

Byers, J. W., Mitzenmacher, M., Potamias, M., and Zervas, G. (2011). A month in the life of Groupon. *arXiv preprint arXiv:1105.0903*.

Byers, J. W., Mitzenmacher, M., and Zervas, G. (2012b). The daily deals marketplace: Empirical observations and managerial implications. *SIGecom Exchanges*, 11(2):29--31.

Byers, J. W., Mitzenmacher, M., and Zervas, G. (2012c). The Groupon effect on Yelp ratings: A root cause analysis. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 248--265.

Cesa-Bianchi, N. and Fischer, P. (1998). Finite-time regret bounds for the multiarmed bandit problem. In *Proceedings of the 15th International Conference on Machine Learning*, pages 100--108.

Chakrabarti, D., Kumar, R., Radlinski, F., and Upfal, E. (2008). Mortal multi-armed bandits. In *Advances in Neural Information Processing Systems*, pages 273--280.

Choi, S.-M., Ko, S.-K., and Han, Y.-S. (2012). A movie recommendation algorithm based on genre correlations. *Expert Systems with Applications*, 39(9):8079--8085.

Chu, W. and Park, S.-T. (2009). Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of the 18th International Conference on World Wide Web*, pages 691--700.

Collard, C., Pustay, M., Roquilly, C., and Zardkoohi, A. (2001). Competitive cross-couponing: A comparison of French and US perspectives. *Journal of Public Policy and Marketing*, 20(1):64--72.

Davidsson, C. and Moritz, S. (2011). Utilizing implicit feedback and context to recommend mobile applications from first use. In *Proceedings of the 2011 Workshop on Context-awareness in Retrieval and Recommendation*, pages 19--22.

De Francisci Morales, G., Gionis, A., and Lucchese, C. (2012). From chatter to headlines: harnessing the real-time web for personalized news recommendation. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, pages 153--162.

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

Dholakia, U. M. (2010). How effective are Groupon promotions for business. `http://www.ruf.rice.edu/~dholakia`.

Dholakia, U. M. (2012). How businesses fare with daily deals as they gain experience: A multi-time period study of daily deal performance. *Available at SSRN 2091655*.

Donnelly, K. (2012). Coupons of the 21st century: The golden age of the daily deal industry. *The Elon Journal of Undergraduate Research in Communications*, 3(2):85.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A., and Vapnik, V. (1997). Support vector regression machines. *Advances in Neural Information Processing Systems*, pages 155--161.

Ebrahimi, S., Villegas, N. M., Müller, H. A., and Thomo, A. (2012). Smarterdeals: a context-aware deal recommendation system based on the smartercontext engine. In *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research*, pages 116--130.

Evans, D. S. (2008). The economics of the online advertising industry. *Review of network economics*, 7(3):1--33.

Farahat, A., Ahmed, N., and Dholakia, U. M. (2012). Does a daily deal promotion signal a distressed business? an empirical investigation of small business survival. *CoRR*, abs/1211.1694.

Feng, J., Bhargava, H., and Pennock, D. (2007). Implementing paid placement in Web search engines: computational evaluation of alternative mechanisms. *INFORMS Journal on Computing*, 19(1):37–49.

Fernandes, D., de Moura, E. S., Ribeiro-Neto, B., da Silva, A. S., and Gonçalves, M. A. (2007). Computing block importance for searching on web sites. In *Proceedings of the 16th ACM on Conference on Information and Knowledge Management*, pages 165--174.

Figueiredo, F., Belém, F., Pinto, H., Almeida, J., Gonçalves, M., Fernandes, D., Moura, E., and Cristo, M. (2009). Evidence of quality of textual features on the web 2.0. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 909--918.

Freed, L. and Berg, R. (2012). Daily deal websites and emails bring in new and existing customers for retailers. `http://www.foreseeresults.com/research-white-papers/_downloads/daily-deal-commentary-2012-foresee.pdf`. [Online; accessed 01-November-2013].

Freeman, L. (1979). Centrality in social networks: conceptual clarification. *Social Networks*, 1(3):215--239.

Goel, A., Guha, S., and Munagala, K. (2006). Asking the right questions: Model-driven optimization using probes. In *Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 203--212.

Groupon, I. (2012). FORM 8-K – May 2012. `http://investor.groupon.com/secfiling.cfm?filingid=1104659-12-33493&cik=1490281`.

Groupon, I. (2013). Groupon Statement Presentation – 3Q13 Earnings. `http://files.shareholder.com/downloads/AMDA-E2NTR/2803965490x0x704836/3f006dcf-649a-4fa1-adfd-140540d6b720/3Q13_GRPN_Earnings_Slides.pdf`.

Groupon, Inc (2012). FORM 10-Q – May 2012. `http://investor.groupon.com/secfiling.cfm?filingID=1490281-13-33`.

Guha, S. and Munagala, K. (2007). Multi-armed bandits with limited exploration. In *Proceedings of the Annual Symposium on Theory of Computing*, pages 1--19.

Herlocker, J. and Konstan, J. (2002). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*, 5(4):287--310.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 263--272.

Jun, T. (2004). A survey on the bandit problem with switching costs. *De Economist*, 152(4):513--541.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4(1):237--285.

Kelly, D. and Teevan, J. (2003). Implicit feedback for inferring user preference: a bibliography. *ACM SIGIR Forum*, 37(2):18--28.

Knuth, D. E. (1973). *Sorting and searching*, volume 3. Addison-Wesley Professional.

Krasnova, H., Veltri, N. F., Spengler, K., and Günther, O. (2013). "Deal of the day" platforms: What drives consumer loyalty? *Business and Information Systems Engineering*, 5(3):165--177.

Krestel, R., Fankhauser, P., and Nejdl, W. (2009). Latent dirichlet allocation for tag recommendation. In *Proceedings of the 3rd ACM Conference on Recommender systems*, pages 61--68.

Kuleshov, V. and Precup, D. (2010). Algorithms for the multi-armed bandit problem. *Journal of Machine Learning*. Submitted.

Kumar, V. and Rajan, B. (2012). Social coupons as a marketing strategy: A multifaceted perspective. *Journal of the Academy of Marketing Science*, 40(1):120--136.

Lacerda, A., Cristo, M., Gonçalves, M., Fan, W., Ziviani, N., and Ribeiro-Neto, B. (2006). Learning to advertise. In *Proceedings of the 29th Annual International ACM Conference on Research and Development in Information Retrieval*, pages 549--556.

Lacerda, A., Veloso, A., and Ziviani, N. (2013). Exploratory and interactive daily deals recommendation. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 439--442.

Lai, T. L. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4--22.

Lappas, T. and Terzi, E. (2012). Daily-deal selection for revenue maximization. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 565--574.

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010a). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661--670.

Li, W., Wang, X., Zhang, R., Cui, Y., Mao, J., and Jin, R. (2010b). Exploitation and exploration in a performance based contextual advertising system. In *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining*, pages 27--36.

Li, X. and Wu, L. (2013). Measuring effects of observational learning and social-network word-of-mouth (wom) on the sales of daily-deal vouchers. In *Proceedings of the 46th Hawaii International Conference on System Sciences*, pages 2908--2917.

Madani, O., Lizotte, D. J., and Greiner, R. (2004). Active model selection. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 357--365.

Mahajan, D. K., Rastogi, R., Tiwari, C., and Mitra, A. (2012). Logucb: an explore-exploit algorithm for comments recommendation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 6--15.

Mitchell, T. M. (1997). *Machine learning. 1997.*

Oard, D. W., Kim, J., *et al.* (1998). Implicit feedback for recommender systems. In *Proceedings of the AAAI Workshop on Recommender Systems*, pages 81--83.

Orengo, V. M. and Huyck, C. R. (2001). In *Proceedings of the 8th String Processing and Information Retrieval Symposium*, pages 186--193.

Park, J. Y. and Chung, C.-W. (2012). When daily deal services meet Twitter: understanding Twitter as a daily deal marketing platform. In *Proceedings of the 3rd Annual ACM Web Science Conference*, pages 233--242.

Pinto, D., Benedí, J.-M., and Rosso, P. (2007). Clustering narrow-domain short texts by using the Kullback-Leibler distance. In *Proceedings of the 8th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 611--622.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130--137.

Radlinski, F., Kleinberg, R., and Joachims, T. (2008). Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th International Conference on Machine Learning*, pages 784--791.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 452--461.

Resnick, P. and Varian, H. (1997). Recommender systems. *Communications of the ACM*, 40(3):56--58.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527--535.

Salkind, N. J. (2010). *Encyclopedia of Research Design*, volume 1. SAGE Publications.

Schafer, J. B., Konstan, J., and Riedi, J. (1999). Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 158--166.

Shevade, S. K., Keerthi, S. S., Bhattacharyya, C., and Murthy, K. R. K. (2000). Improvements to the SMO algorithm for svm regression. *IEEE Transactions on Neural Networks*, 11(5):1188--1193.

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 6th ACM Conference on Recommender Systems*, pages 139--146.

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

Sontag, D. and Roy, D. (2011). Complexity of inference in latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 1008--1016.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. Cambridge University Press.

Thompson, W. R. (1963). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3):285--294.

Tran-Thanh, L., Chapman, A., Munoz De Cote Flores Luna, J. E., Rogers, A., and Jennings, N. R. (2010). Epsilon-first policies for budget-limited multi-armed bandits. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1211–1219.

Vermorel, J. and Mohri, M. (2005). Multi-armed bandit algorithms and empirical evaluation. In *Proceedings of the 16th European Conference on Machine Learning*, pages 437--448.

Wansink, B. and Park, S. B. (2000). Methods and measures that profile heavy users. *Journal of Advertising Research*, 40(4):61--72.

Weng, J., Lim, E.-P., Jiang, J., and He, Q. (2010). Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM International Conference on Web Search and Data Mining*, pages 261--270.

Ye, M., Sandholm, T., Wang, C., Aperjis, C., and Huberman, B. A. (2012). Collective attention and the dynamics of group deals. In *Proceedings of the 21st International Conference Companion on World Wide Web*, pages 1205--1212.

Yelp, I. (2012). Yelp's Academic Dataset. `http://www.yelp.com/academic_dataset`.

Yoo, S., Yang, Y., Lin, F., and Moon, I.-C. (2009). Mining social networks for personalized email prioritization. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 967--976.

Zhao, X., Zhang, W., and Wang, J. (2013). Interactive collaborative filtering. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 1411--1420.