Social Review-based Recommender Systems from Theory to Practice

Master Thesis Project

Master of Information and Communication Technologies

Escola Tècnica Superior de Telecomunicacions de Barcelona (ETSETB)

Author:
Joan Capdevila Pujol

Advisors:
Dr. Argimiro Arratia
Dra. Marta Arias

June 26, 2014

Inferring knowns from unknowns.

Acknowledgements

I would like to devote some lines to the people who has encouraged me during the realization of the Master Thesis.

First of all, I would like to thank Prof. Argimiro Arratia and Prof. Marta Arias for their always valuable insights, guidelines and councils. Not least, I am also very grateful to them for their flexibility in letting me carry out this thesis during my spare time with all the logistic consequences that this has supposed.

Secondly, I deeply thank Emma, my partner, for encouraging me from the very first beginning in focusing the Thesis on data analytics. This probably was the best approach to learn about machine learning concepts and achieve the professional turning point that I was looking for, moving from descriptive to predictive analytics. Moreover, she payed for my less free time and mood swings when things did not work out as expected. I sincerely appreciate her love.

Many colleagues have helped me a lot to glimpse my professional path since I landed on Barcelona 9 years ago and they have contributed to create a big piece of me. Many thanks to all of them.

I also greatly appreciate the help of several classmates and professors with who we worked side by side during the lecture semester in the MINT program. I would like to especially thank Prof. Josep Sol Pareta, for encouraging me to enroll this program and for all his support as the Master coordinator. This brings to an end the hard work done in the whole Master program.

Last but not least, I would like to dedicate these last words to my family and friends for their understanding, support and love. I owe it all to them.

Abstract

Social Recommender Systems were born with the goal to mitigate the current information overload caused by the birth of Social Networks among other causes. They have enabled Internet actors (e.g. users, web browsers, sensors, actuators, etc.) to make more informed decisions based on the information that is been shown to them, up to the point that some actors even blindly trust the recommendation generated by these systems.

Within this scenario, this thesis proposes a novel Hybrid Social Recommender System purely based on the text reviews typed by users. The proposed engine treats the review content and sentiment separately and finally, combines both into a single recommendation. Very little scientific research has been published on mining text reviews with the aim of performing item recommendation. Moreover, among all Hybrid Recommendation Systems in the literature, none use the above-mentioned review features into a collaborative and content-based recommender.

With the purpose in mind of assessing the platform effectiveness, we present a methodology that goes from the process of extracting the data directly from a Social Network, cleaning and pre-processing the text data, building the predictive model with different state-of-the art machine learning techniques, up to the point of evaluating the system in terms of several key metrics.

The data extraction process gains our attention due to the challenges imposed by most social platforms in obtaining all the geo-positioned data generated in a bounded region. To overcome the platform limitations, we introduce the use of the Quadtree algorithm with the goal of crawling all the geo-positioned reviews [1]. The algorithm is enhanced with a module that copes with the time dynamics and captures the time-stamped data as well. Moreover, we study the effectiveness of the Quadtree partition method to crawl any type of spatial data, which tends to be softly distributed in the area.

This thesis draws several conclusions from the available data about the use of several state-of-the art text mining techniques and the effectiveness of the proposed recommender setup. Nonetheless, future work needs to design and propose novel evaluation methodologies that uncouple the system evaluation from the data.

Contents

Chapter 1

Introduction

1.1 Overview

This thesis paves the way to designing and building systems that mitigate the information overload in a social and interconnected world. By making use of the state-of-the-art text mining techniques we propose a novel Hybrid Recommender System, that tackles the information overload problem in user reviews feedback from social online communities.

In this thesis, we start surveying the state-of-the-art on Social Recommender Systems to understand the challenges and open points in this field in order to enable operational Social Recommender Systems. As part of the preliminary work, we have also done research and propose techniques to crawl geo-positioned and time-stamped social data. We then bring these techniques into practice by proposing and implementing a crawler system which retrieves reviews from a Social Network called Foursquare[1]. Text reviews are mostly ignored in Social Recommender Systems, but they are one of the most consulted web content before any purchase or decision. Hence, we propose a new Social Recommender System purely based on text reviews data. The proposed system combines content from the text reviews as well as the user opinion into a Hybrid Recommender set-up. In order to evaluate the Recommender System, we also present an evaluation methodology which assesses the system in terms of Prediction Coverage and Accuracy. We finally use the Foursquare data set to examine different text mining techniques with the aim of identifying which one is the best in terms of the evaluation methodologies. Interesting relationships arise from the coverage, the prediction accuracy and the quality of the data set.

1.2 Introduction to Social Recommender Systems (SRS)

The Internet, also known as the network of networks, has been interconnecting billions of people, devices and organizations for several decades. Lately, the Internet actors have started

[1]https://www.foursquare.com

establishing social contact through the network, constituting what are known as Social Networks. A Social Network is a social structure made up of the interaction between actors. In fact, the concept of Social Network was coined back into the nineteenth century and, clearly, it was not referring at all to the concept used today on the Internet.

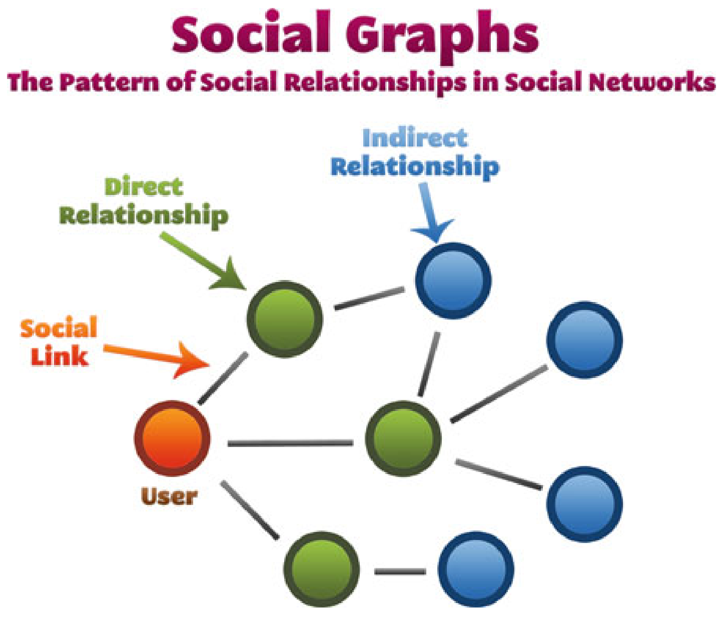


Figure 1.1: Social graph: the pattern of social relationships in social networks [2]

The birth of Social Networks on the Internet, also know as virtual or online communities, together with the increase of players are some of the main reasons that can be related with the phenomena of Big Data. Big Data refers to the tsunami of data that flooded the Internet, causing new challenges to technically process, handle and store this data.

This tsunami of data has not only caused new technological challenges, but it has also requested methodologies and systems capable of interpreting the data and turning it into valuable knowledge and actionable insights. Actors on the Internet are completely drown and they cannot assume the cost of making optimal decisions from the large amount of public information. For example, a traveler who wishes to spend his/her holidays in New York and looks for hotels positively rated and matching his/her preferences might require excessive time to make the right choice.

Because of this, we envision Social Recommender Systems (SRS), and more generally Recommender Systems (RS), that may or may not use Big Data technology, as helpers that can make decisions on behalf of others.

Social Recommender Systems (SRS) have been described by I. Guy et al. [3] as systems that aim to alleviate information overload over social media users by presenting the most attractive and relevant content. SRSs also aim at increasing adoption, engagement, and participation of new and existing users of social media sites.

Social Recommender Systems encompass a broader range of recommendable items than classical Recommender Systems due to their presence into the Internet. Some of the recommendable items fall into the following categories: web content (blogs, news, wikis,...), tags (trending topics, tagged images,...), multimedia (music, images, videos,...), people (friends, colleagues, ...), businesses (jobs, advertisements, stock shares, companies,...), places (hotels, restaurants, ...), items for sale (e-commerce, phone apps marketplaces,...) and groups of interests.

Social Recommender Systems also embrace a larger number of data sources. While classical RS have mainly been based on either a set of manually scored categories or a scored derived from the user neighbourhood popularity, SRS can have access into a vast set of data sources (page logs, likes/dislikes, friendships, reviews, posts, followers, amongst others) and comes up with novel recommendation methodologies that goes beyond the established frontiers.

All these sets of data sources provide the SRS distinct types of data, such as structured or unstructured data sets; different types of processing, like streaming/online, quasi-online or batch. Hence, the study of Social Recommender Systems as a whole also involves other disciplines such as information retrieval, data processing and cleaning.

The target audience has also become much greater and more generic due to the amount of actors present in today's Internet. While individual user recommendation has been already faced by traditional RS, emerging audiences organized in groups, communities, business or even things, now requires new paradigms, more complex than the simply sum of its members [4].

The classical challenges from traditional Recommender Systems, such as the cold start, the sparsity or the overspecialization, also occur within the Social Recommender System. Depending on which applications, these challenges can even be taken to the extremes. For example, an e-commerce site using a long tail business model to sell their products such Amazon or Netflix. In these type of business, the number of items is huge since they make significant profit out of selling small volumes of a lot of hard-to-find items.

In a nutshell, Social Recommender Systems mainly differs from the classical Recommender Systems in the fact that the Social Networking sites and virtual communities provide a new way of performing recommendations by means of the user profiles and relationships among users [2].

1.3 The Foursquare Network

Foursquare is a social networking site launched in 2009 specially thought for smartphones and other mobile devices. Users register or check-in in places, known as venues in Fourquare terminology, by means of the Foursquare application installed in their devices. Moreover, users can also review their experiences at the venues through short reviews or tips.

The Foursquare application contains its own Social Recommender System that recommends as per several features available in the system. Although the recommendation engine is not publicly available, Foursquare engineers stated in the past that they observed the system was skewed to restaurants with larger capacity and longer opening hours [2]. Several upgrades have been done to the Foursquare recommendation engine, but there is still a lot of work to do in this field.

Foursquare has an open Application Programming Interface (API) that allows developers to create applications (a.k.a apps) on top of the Foursquare platform. This not only allows applications to interact with the Foursquare Network, but it also gives access to most of the data available in the social site.

Due to the features and the open nature of the Foursquare Network, we propose this Social Network for the case study of this thesis.

1.4 Motivation for this Work

The motivation for this work is listed next:

- First, we aim to design and develop systems which are capable of handling the nowadays information overload in the Internet. Social Recommender systems combine traditional Recommender Systems with social networking content to cope with Big Data phenomena and provide valuable insights.
- Second, despite of security and limitations directives defined in Social Networks; extracting data from a Social Network is feasible and compliant. This allow us access to real data to build, test and validate the proposed platforms.
- Third, new forms of user content such as user profiles and their relationships with other actors, appear due to the birth of Social Networks. Both types of information bring new challenges into this field.
- Fourth, new form of item content such as reviews, pictures, tags, amongst others; also are result of new online communities. This content requires pre-processing the data to extract key features before feeding a Recommender System.
- Fifth, Social Networks enable us to measure the user feedback after experiencing any purchase or choice. This enables Social Recommender Systems to compare their recommendations against the actual or real experience of a given user.

[2]http://engineering.foursquare.com/2011/03/22/building-a-recommendation-engine-foursquare-style/

1.5 Related Work

Social Recommender Systems have arisen as a combination of Social Networks and Recommender Systems communities. Hence, there is a lot of related work already done to each individual community. In the last two decades, many Recommender Systems have been designed and proposed mainly for e-commerce web sites. Two of the most popular applications are Amazon.com [5] and Netflix [6]. In what follows, we will only refer to the research done in the crossroad of both fields.

The term Social Recommender System (SRS) has been lately accepted as a slightly different concept from the classical Recommender Systems [3]. However, researchers have been proposing novel recommender set-ups based on social content for the last 5 to 10 years [2, 3, 4, 7].

Classical recommendation paradigms such as Content-based recommenders have been enhanced by adding friendship information into the matrix factorization objective function [7], configuring the so-called Social Content-based Recommender(SoCo). Collaborative-based filtering has also been tailored to include social information to improve the user neighbourhood detection and deal with the data sparsity in [8, 9]. According to the authors in [8], social filtering outperforms collaborative-based filtering. Trust-based recommenders, a variation of the collaborative-based recommenders, have also been re-shaped to add user feedback from Social Networks creating a new recommender known as FeedbackTrust [10].

Others have proposed mechanisms to include interpersonal influences in traditional recommender arguing that the interpersonal influence plays a critical role in this scenario [11]. Some others have coined the term Social Regularization to refer to the use of a regularization based on social content. Location-aware Recommender Systems that exploits geo-positioned data have also been proposed to take into account the spatial dimension [12].

A lot of research has also been done in group recommendation since the birth of Social Networks. In [13], the authors highlight the benefits of incorporating social interests in group Recommender Systems.

Furthermore, several practical Social Recommender Systems can be found nowadays in the literature. For example, [14] describes an on-line social network-based job Recommender System for recruiters.

Social context is not only used for Recommender System, but it has also been lately proposed for Review Quality Detection in [15]. Most of the previous work in this field treated each review as a stand-alone text document, extracting features from the text and learning a function based on the features [16, 17, 18]. Other reviews models are proposed in this thesis, especially using novel text mining techniques such as Latent Dirichlet Allocation (LDA) [19] or other classical techniques like Latent Semantic Analysis (LSA) [20] or TF-IDF [21]. Sentiment analysis is also included in these models by using trained traditional classifiers as the simple Naives Bayes [22], Multinomial [23] or supervised LDA classifiers [24]

Last, we have also created our own data sets to validate the proposed models. To do that, we have surveyed Information Retrieval techniques [25] and Social Networks crawling systems such as Twitter Echo [26], a paradigm to overcome some limitations in the Twitter API.

1.6 About This Document

We have already introduced the Social Recommender Systems and presented the motivations for this work.

The rest of the document is organized in two parts. The first part establishes a methodology to design a Social Recommender System from theory to practice. The second part is a case study of the proposed Social Recommender System focused on Restaurants data set crawled from Foursquare.

The methodology part is explained in four chapters and each chapter presents a separate data process. In the second chapter of the thesis, we present the data crawling techniques to retrieve data from Social Networks. We also propose techniques to crawl geo-positioned and time-stamped data such *reviews* or *checkins* (sign up to a place) through a smartphone. In the third chapter, we introduce a model for the text reviews and we survey different state-of-the-art text mining techniques that can be used in conjunction with this model. In the fifth chapter, a hybrid Recommender System that exploits the previous reviews model is presented. Chapter six closes this methodology part with the presentation of a method for evaluating the system and some relationships among the performance figures.

The application part consist of two chapters which bring into practice the above-mentioned methodology. First, we present the Hybrid Recommender System purely based on the restaurant reviews data from Foursquare. Secondly, we evaluate the system following the proposed methodology and we reach several interpretations about the results.

Finally, we present the conclusions of this work and present several open issues for future work.

Part I

Methodology

Chapter 2

Geo-positioned and Time-stamped Data Retrieval

2.1 Data Retrieval Overview

Data retrieval is the first and the most relevant process in any data analysis application and hence, also in designing and implementing Social Recommender Systems (SRSs). Being able to retrieve accurately and consistently all required data enhances the prediction accuracy and coverage of any Recommender System. Consequently, we believe that any methodology to design and implement SRS needs to include techniques to crawl data effectively and efficiently.

Next, we present a paradigm, Web APIs, to access data from social networking sites. This paradigm is implemented in most of the existing Social Networks [25]. However, challenges come when handling different types of data and the limitations of specific APIs. Because of this, we devote this chapter to tackle some of these challenges with geo-positioned and time-stamped data in Social Web APIs, such as the Foursquare Platform.

2.2 Web Application Programming Interfaces (APIs)

An Application Programming Interface (API) is a set of functions used by a program to request services from another software system.

Within the Web, we might use the term Web API to refer either to a set of methods used to interact upon a web-server, called a server-side Web API; or upon a web-browser, called a client-side Web API. In what follows, we will make use of the term Web API to refer specifically to server-side APIs.

A server-side Web API, or simply a Web API; is a set of requests and responses messages exposed via the web typically by means of an HTTP-based web server. These messages are usually encoded in an internet media type such as XML or JSON.

One type of HTTP-based APIs are the so-called Representational State Transfer (REST) Web APIs. REST or RESTful Web Services have been introduced to replace other distributed Web Services architectures such SOAP (Simple Object Access Protocol) due to the lightness and simplicity of the formers.

REST-based Web APIs obey the following aspects:

- Base Uniform Resource Identifier (URI) which identifies the name of the web resource.
- Internet Media Type for the data which usually is JSON.
- Standard HTTP Methods (e.g. GET, PUT, POST)
- Hypertext links to reference state
- Hypertext links to reference related resources.

2.2.1 Social Network APIs

Social Networks APIs are the subset of APIs used to access content from a Social Network or to interact with its resources. Most Social Networks contain nowadays APIs which enable third-party developers to create web or mobile applications on top of it. These APIs provide a set of HTTP methods for accessing a web resource or URI such as a user, a venue, or a tip in Foursquare; or a tweet, a timeline or a message in Twitter [1]. Given a resource, one can often drill down into a particular aspect of it (e.g. friends of a user, retweets of the tweet, ...), which will be also identified by a different URI. A resource might also have some actions associated with it (accept a friendship of a user, create a new tweet, ...). These actions are linked to a URI too.

The most used HTTP methods for accessing URIs in the current Social Networks are:

- GET: Requests a representation of the specified resource or some aspects of it. Requests using GET only retrieve data.
- POST: Requests that might change the state of the web resource identified by the URI. The data posted might be, as examples, a new tweet, a tip, a comment or a message.

As an example of a POST request, the following URL is the URI to add a tip (a.k.a review) into Foursquare.

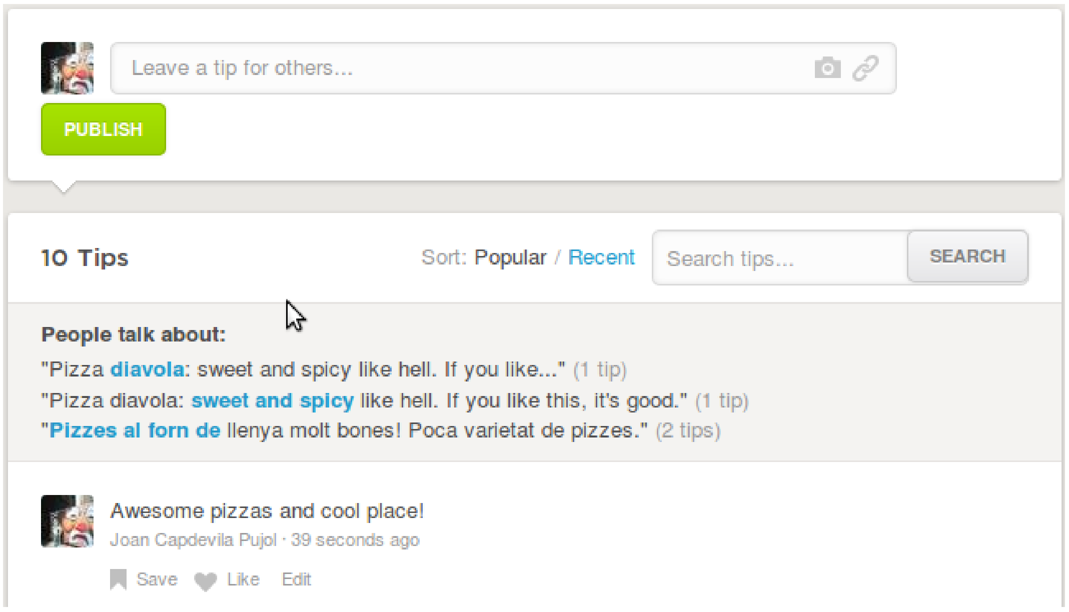
```
https://api.foursquare.com/v2/tips/add
```

[1] `https://www.twitter.com`

The POST request passes a set of key-value pairs. The *venuesId* identifies the place in Foursquare, which can be a restaurant, a shop, outdoors place, amongst others. The *text* is the tip content which is published to the venue's time-line. Last, the *oauth_token* and *v* are the authorization token value and the version respectively.

venueId: 4beff0bed4e4d13abcad15a7
text: Awesome pizzas and cool place!
oauth_token: XEPQYW33HGKXXY2HT4TATRAHN0YCQOGSCI5SWC5WYLDC4Y2A
v: 20130707

The result of executing the POST method is a change of the state on this resource, which is reflected in the Foursquare Social Network, Fig. 2.1. The POST request also returns a JSON which contains the tip.



People talk about:
"Pizza diavola: sweet and spicy like hell. If you like..." (1 tip)
"Pizza diavola: sweet and spicy like hell. If you like this, it's good." (1 tip)
"Pizzes al forn de llenya molt bones! Poca varietat de pizzes." (2 tips)

Awesome pizzas and cool place!
Joan Capdevila Pujol · 39 seconds ago

Figure 2.1: Update of the venue's time-line after the POST request.

2.2.2 Data Retrieval with APIs

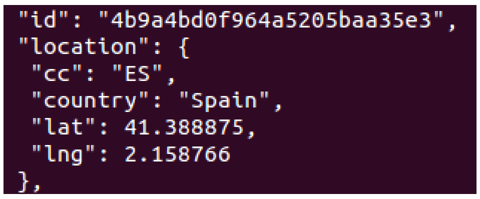
Retrieving data from Social Networks is feasible by means of the above-mentioned APIs. Nonetheless, Social Networks typically limit the rate of requests per application. Moreover, their policy also specifies several directives to not store data; but instead, cache it.

Techniques to overcome the request rate and response size limitations as well as methodologies to comply with the specified guidelines need to be taken into consideration when creating an application that works with social data owned by a different content delivery.

Data is usually retrieved using GET requests. The following URL is the URI to retrieve venues (a.k.a places) from Foursquare, via a GET method. In GET methods, parameters can be specified within the URL itself. For example, *near* indicates a pre-defined place where Venues are located.

```
https://api.foursquare.com/v2/venues/search?near=Barcelona
&oauth_token=XEPQYW33HGKXXY2HT4TATRAHN0YCQOGSCI5SWC5WYLDC4Y2A
&v=20130707
```

The result of the GET method provides a JSON structure with several Venues, as in Fig. 2.2. If the number of Venues is less than the API limitation (in this case 50 Venues), then all Venues are displayed. Otherwise, the request only return 50 results like in our example.



```
"id": "4b9a4bd0f964a5205baa35e3",
"location": {
 "cc": "ES",
 "country": "Spain",
 "lat": 41.388875,
 "lng": 2.158766
},
```

Figure 2.2: JSON structure of a Venue resulted from the GET request.

2.3 Quadtree: A Geo-positioned Data Retrieval Technique

In what follows, we describe a technique to retrieve geo-positioned data via an API which exhibits rate limitations in their requests as well as constrained responses.

Geo-positioned or geo-tagged data consist in adding geographical coordinates to various media such as pictures, videos, posts, comments, tweets, among others. This could be the case of a Social Network which links some media (tweets, tips, etc.) to the user position gathered through the smartphone GPS system. It could also be the situation for a big set of sensors spread over a huge geographical area, from which we wish to get their sensed information (e.g. air-pollution, traffic, etc.). Next, we will use the term sensor or item to refer indistinctly to geo-positioned data sources spread over an area. In either case, we must consider that the API will have some rate limitation mechanisms in order to not overload the web servers with request-response messages.

These limitations might take effect depending on the goal of the request messages. If applications need to query the API sporadically and the goal of the query is some detailed and concrete information, then we might not have to deal with these limitations. On the other hand, if queries are persistent and broad, then the limitations need to be taken into consideration.

A classical situation in which the query is persistent and wide at the same time is when querying information such the following: *What is the content for all traffic sensors in Europe?*. In case it exists an API to answer this question, the API would certainly limit the size of the response message and it would probably reply only with the top t sensors data.

Note that if we could query this information for each sensor in Europe, then we would be able to query individually each sensor in the following way: *What is the content for sensor with ID X?*. In this case, the API probably would also limit the rate of request per hour, r, in order to prevent answering the wide question by means of several narrower queries, which would cause the web server to overload.

The following technique aims to solve efficiently the problem of querying an API with wide geo-positioned questions while ensuring data integrity.

In what follows, we consider that the number of items, N, with geo-spatial information for which the query asks for, is much greater than the response limitation, denoted N_{max}, ($N \gg N_{max}$). Hence, the solution to this problem necessarily goes through the process of dividing the broad question into several narrower sub-queries.

The splitting process has to take into consideration that querying individually each item might not be possible nor efficient. Some APIs do not offer the possibility of querying each item individually and if they do, this solution possibly would not be the most efficient in terms of time. The overall time to query all items one-by-one could be quite large due to the fact that APIs impose a maximum number of requests per hour, R_{max}.

Our proposal takes into account these restrictions and we will show the conditions under which the overall time using the proposed crawling technique, T_{QUAD}, is less than the time querying individually all items or sensors, T_{IND}.

$$T_{QUAD} < T_{IND} \tag{2.1}$$

A Quadtree is a data structure which divides each region into four spaces when the maximum capacity per cell, N_{max}, is reached [1]. They are mostly used to partition two-dimensional spaces where nodes correspond to subspaces of the same space. We propose the Quadtree algorithm as the mechanism to partition a geographical area into sub-areas which the crawling system can query while still getting all items' contents. Moreover, we aim to know the conditions for which the inequality in (2.1) is satisfied.

Given two geographical coordinates (the south-west and north-east coordinates) defining a bounding box such the one shown in Fig. 2.3, we would like to gather all content from each sensor in the area defined by these two coordinates.

We start querying the full geographical area indicating the south-west and the north-east coordinates. If the query returns the maximum number of results, N_{max}, the process splits the area into four sub-areas with identical sizes, which are queried as well. If the results per sub-area are still greater than the maximum number of results, the splitting process continues following the same logic. Otherwise, that area or sub-area whose query returns a number of results less than the maximum stops the splitting process and it assigns the number of crawled items to that cell.

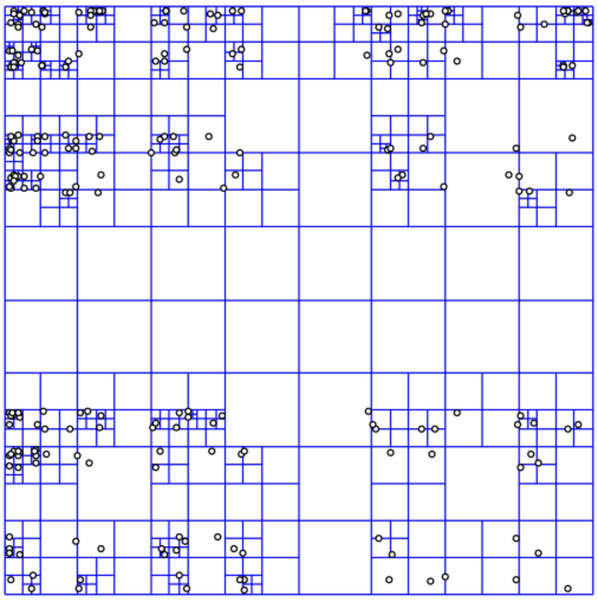


Figure 2.3: A region quadtree with point data. [27]

The maximum capacity per cell, N_{max}, is constrained by the API. However, the Quadtree technique does not guarantee that each cell will have exactly N_{max} results. Typically, there will be a number between 0 and N_{max} results. The average number of results or capacity per cell is called N_{cell}.

The time to query individually each sensor can be upper bounded by means of the number of sensor in the area N, which also corresponds to the overall number of requests, and the maximum number of requests per hour ,R_{max}, as,

$$T_{IND} < \frac{N}{R_{max}} \tag{2.2}$$

And the time using Quadtree can also be expressed and bounded by the overall number of Quadtree requests, N_{QUAD} , and the maximum number of request per hour R_{max} as,

$$T_{QUAD} < \frac{N_{QUAD}}{R_{max}} \tag{2.3}$$

Most of the APIs limit the maximum number of request per hour, R_{max}, to some numbers which are easily reachable. Hence, this upper-bound is reached in most of the APIs.

In what follows, we assume that sensors are spread uniformly over an area with the aim to find a closed analytic solution. Nonetheless, we know that this hypothesis only responds to an ideal situation, difficult to find in real applications. Because of this, in the next section we experimentally assess the same problem with other distributions.

Assuming that the sensors are spread uniformly through the area, the overall number of Quadtree request can be expressed as follows,

$$N_{QUAD} = \sum_{k=0}^{1/2log_2(N/N_{cell})} 4^k \tag{2.4}$$

where the summation adds up the four requests in each level of the Quadtree, which goes from level 0 to level $1/2log_2(N/N_{cell})$.

Hence, the average capacity per cell, N_{cell}, has to satisfy the following inequation which combines (2.2), (2.3) and (2.4) into (2.1).

$$\sum_{k=0}^{1/2log_2(N/N_{cell})} 4^k < N \tag{2.5}$$

The summation can be expressed in the form of a geometric series and the sum it becomes,

$$\sum_{k=0}^{1/2log_2(N/N_{cell})} 4^k = \frac{1 - 4^{1/2log_2(N/N_{cell})+1}}{1-4} \tag{2.6}$$

Hence, the average capacity per cell has to obey the following inequality

$$N_{cell} > \frac{4N}{3N+1} \tag{2.7}$$

Considering the initial assumption that $N \gg N_{max}$ and obviously $N_{max} \gg 1$, we can assume that in the limit, the minimum value for the average cell capacity parameter is independent of N:

$$N_{cell} > 1.33 \tag{2.8}$$

This result concludes that the Quadtree paradigm will always take less time than querying individually each sensor if the average cell capacity parameter is greater than 1.33 under the assumption that $N \gg N_{max}$ and that geo-positioned data sources are spread uniformly. In fact, the average cell capacity will always satisfy equation (2.8), since N_{max} tends to be much greater than 1.33 and hence, N_{cell} will also be.

However, the hypothesis that sensors are uniformly geo-distributed cannot be assumed in several applications in which the density of sensors does not follow a uniform distribution. With the aim to generalize this result, we next assess the efficiency of the Quadtree technique in scenarios which sensors are spread over a region following normal distributions.

Last but not least, the Quadtree structure needs maintenance in order to keep the lowest cells, also knows as leaves, up-to-date. This means that if the number of sensors per leave decreases below a threshold, then it might be necessary to prune the tree, or remove those leaves. Moreover, if the lowest cells exceed the maximum cell capacity at some point, the tree is forced to grow. The overhead caused by the Quadtree maintenance has not been considered in the present assessment, nor in the next section.

2.3.1 Quadtree Assessment for Normal Spatial Distributions

In this section, we aim to empirically assess the efficiency of the proposed Quadtree crawler for different density distributions other than the uniform. Our goal is to generalize the result obtained in equation (2.8) to demonstrate that the efficiency of the Quadtree technique will ultimately depend on the peakiness of the distribution.

To achieve this, we compute the Quadtree structure for several normally distributed sensor densities. All these densities will contain the same number of nodes, N, hence all density curves will integrate to N. In what follows, we set N equal to 1 in order to remove a parameter that only supposes a scaling factor. Consequently, the two-dimensional normal distribution for the geo-positioned sensors can be written as,

$$F(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\left(\frac{(x-x_o)^2}{2\sigma_x^2}+\frac{(y-y_o)^2}{2\sigma_y^2}\right)} \tag{2.9}$$

where we force the standard deviation in each dimension to be equal ($\sigma_x = \sigma_y = \sigma$) and we also consider that the distribution is centered at the origin $(x_o, y_o) = (0, 0)$.

Fig. 2.4 shows a uniform and three normal distributions with parameter σ equal to 100, 10 and 0.25 from left to right.

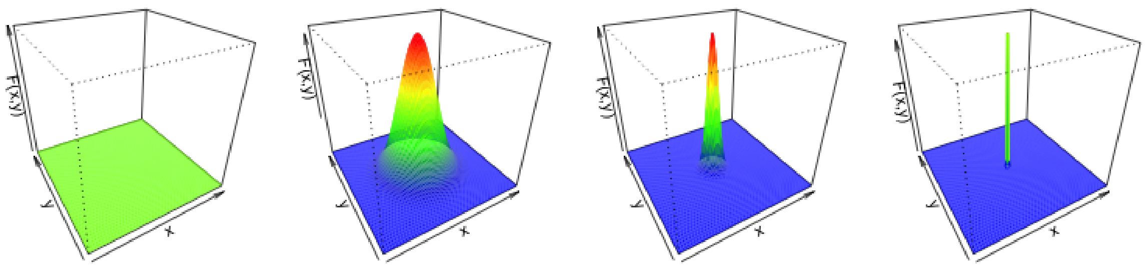


Figure 2.4: Uniform and Normal Distributions with $\sigma = 100, 10, 0.25$ respectively.

In each of these distributions, we apply the Quadtree algorithm to recursively find the quadcells that contains less than $N_{\max}$ sensors. $N_{\max}$ is defined here to be a 10 % of the overall number of nodes, N, hence $N_{\max}$ is equal to 0.1.

The output of the Quadtree algorithm can be found at Fig. 2.5 in the form of the data structures generated.

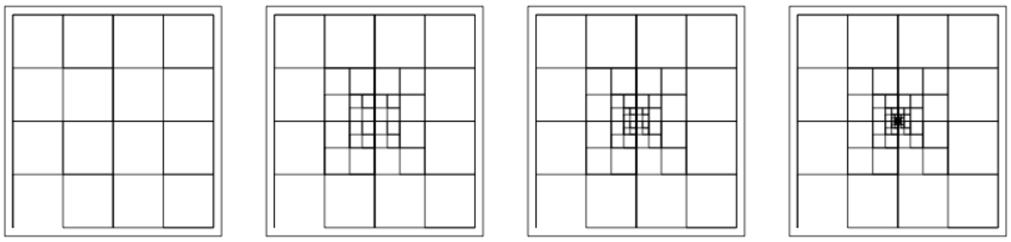


Figure 2.5: Quadtree structures for the Uniform Distribution and the Normal Distributions with $\sigma = 100, 10, 0.25$ from left to right.

Observing these results, there is a clear evidence that the lower the standard deviation is, the more quadcells need to be generated by the Quadtree to capture the underlying spatial distribution. Consequently, the more peaky the distribution is, the more overhead the algorithm creates to crawl all sensors.

In the next plot, Fig. 2.6, we represent the number of quadcells, that is equivalent to the number of queries that the Quadtree technique performs to the API, as a function of the standard deviation. We also add to the plot the number of quadcells due to the uniform distribution.

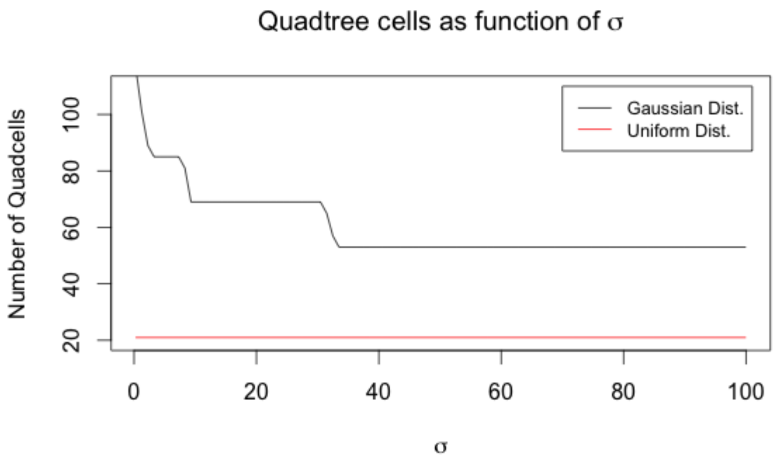


Figure 2.6: Quadtree cells as a function of the standard deviation

Finally, we compute the Quadtree structure of spatial data distributed according to a Mixture model of two Normal Distributions, see Fig. 2.7. From these results, the number of quadcells of a given distribution modeled as a mixture distribution could be intuitively upper-bounded by the number of quadcells for each Normal distribution.

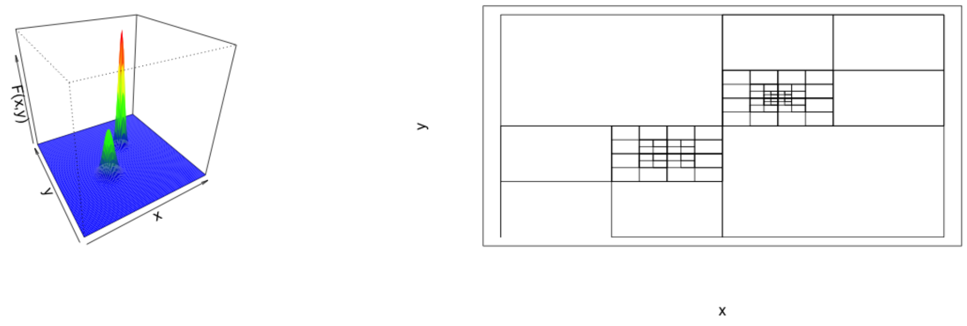


Figure 2.7: Mixture of two Normal Distributions (left) and the corresponding Quadtree structure (rigtht).

These results shows that the Quadtree performance depends on the underlying spatial distribution of data. Moreover, it points out that future research in Quadtree for spatial data crawling has to consider the spatial distribution estimation to enhance its effectiveness.

2.4 Delta: A Time-stamped Data Retrieval Technique

In many social-based applications one needs to load historical time-stamped data from a Social Network. Data in resources such as time-lines, posts, publications is usually ordered and stored by date. Retrieving past information from these resources requires having either a mechanism to query the past or an application connected to a real-time API from that time on.

A real-time API enables applications to subscribe to the updates of a resource. When changes occurs, an HTTP POST request is sent to a callback URL belonging to the application. This allows the application avoiding to continuously querying the API. Real-time APIs are not always available in a Social Network platform and they sometimes requires some type of authorization to retrieve data from specific objects. For example, Foursquare offers a real-time API for venues which is only available to the owners of the venues.

Because of this, it is necessary a mechanism which enables the retrieval of all historical time-stamped data from a specific resource and taking into account the API limitations.

When querying a resource, the API restricts the query response by limiting the result to few records in order to avoid the web server overload. Hence, it is not possible to retrieve the full history of a resource at once.

The so-called delta technique, used in many commercial ETL tools [28], consists in re-building the full history of a resource from the oldest to the newest with several conditioned requests without overlapping on the selection criteria.

In order to guarantee that conditioned requests do not overlap, it is important to maintain a delta time-stamp field which indicates the oldest data retrieved by the application. By comparing the time-stamp of the newest record in the query with the delta time-stamp, we are able to identify if new data has been generated and if so, extract it. In the case that both time-stamps match, the application does not update the database, since there is no new data.

Note that if the time-stamp in the record matches a field that signals about the last changes, the delta mechanism would be able to identify changes in existing data. However, sometimes the record time-stamp only reflects the time in which the record was created.

2.5 A Case Study: The Foursquare platform

The Foursquare Platform is composed by two APIs: the Core API and the Real-time API.

The **Core API** provides a set of methods, also known as endpoints, to access the Foursquare resources such as venues, users, tips. Moreover, the API allows the application to consult aspects of a resource or trigger actions which are associated to them.

The **Real-time API** permits the application to receive real-time updates on the checkins of an authenticated user or real-time content from venues managed by one of the authorized users in the application.

In this case study, we aim to use the Foursquare Platform in order to crawl all geo-positioned and time-stamped data like tips from a large geographical area like Manhattan. Tips in Foursquare are the user reviews in plain text about a specific venue. Tips also contain a time-stamp, that is the date at which the user creates the tip, and they also are geographically located with the coordinates of the venue.

The real-time API would probably help in keeping an up-to-date data set of tips. However, this API requires the venue manager authorization for each venue, which is not feasible in Manhattan. On the other hand, the Core API provides fewer limitations on querying venues.

In what follows, we will describe the methods to connect with the Foursquare Platform; then, we will introduce the endpoints used to retrieve tips while pointing out the rate limitations that they exhibit. Finally, we will propose a Foursquare crawler to retrieve the whole history of tips geo-positioned in a dense area such as Manhattan.

2.5.1 Connection

To connect to the Foursquare API platform, there are some steps that have to be done before querying the platform.

1. Register the application. The application developer needs to register it to obtain the Foursquare API credentials. The credentials consist of a client ID as well as a client secret.

2. Obtain an access token. The access tokens allows the application to query the Foursquare platform on behalf of a user. There are different mechanisms to request the Access token depending whether your application is a mobile or desktop app.

3. Query the Foursquare platform.

   - Authenticate access. With endpoints that require this type of access, it is needed to add the access token at the end of the GET or POST request, *oauth_token = ACCESS_TOKEN*
   - Userless access. With endpoints that do not require user authorization, the GET or POST request only needs to specify the client ID and secret instead of the access token, *client_id=CLIENT_ID & client_secret=CLIENT_SECRET*.

2.5.2 Core API Endpoints

Tips can be queried in three different ways in the Foursquare Platform, as standalone resources, as aspects of venues and as aspects of users.

- Tips standalone.

  An application can directly query tips from the Foursquare platform via an endpoint that allows to search nearby tips. The endpoint consists of an HTTP GET method which does not require authenticated access. Moreover, the endpoint has a set of required and optional parameters that allows the application narrow the query. The response is a JSON structure with 500 tips at most.

  For example, we can see below a HTTP GET method to search for tips near the latitude and longitude coordinates encoded in the parameter *ll*. Moreover, the userless request does not require appending the access token, but the Foursquare credentials in terms of client_id and client_secret. Last, it is also necessary to end the request with parameter v which encodes today's date.

```
https://api.foursquare.com/v2/tips/search?ll=40.7,-74
&client_id=JDRIE3Z2IE2AIDWMMLOMEDE0SPRGBZCPTFSXUV2AESCTGU4R
&client_secret=BAN1B1PBWHHYW2KTQGP31ED5LCZRTIVMT1S0YSNUVWGD3ZCN
&v=20140214
```

- Tips, as aspects of Venues.

  Tips can be accessed as aspects of venues from the Foursquare platform via an endpoint that provides a list of tips for a specific venue. The endpoint consist of a HTTP GET request which does not require to act as user. It also has a list of parameters that allows to specify the type of response. The response of the request is also a JSON structure.

Table 2.1 shows the list of parameters that can be appended to the GET request. Note that the maximum number of tips returned by the request will be limited to 500. Hence, if a Venue has more than 500 tips, it will be necessary to use the page through results parameter in order to crawl all tips. In other words, parameter *offset* is used to select the next results set composed of 500 tips at most.

Table 2.1: Tips from a Venue Endpoint

VENUE_ID	XXX123YYYY	required. The venue you want tips for.
sort	recent	One of friends, recent, or popular.
limit	100	Number of results to return, up to 500.
offset	100	Used to page through results.

- Tips, as aspects of Users.

  Tips can also be accessed as aspects of users via an endpoint that returns the tips done by a user. However, this endpoint consists of a HTTP GET request that requires user authentication. The access token has to be appended in to the URL. The list of parameters and the HTTP request are shown next.

  Table 2.2 also display the list of parameters available in the Tips from a User endpoint. To contrast with the previous endpoint, this includes a parameter to look for tips nearby a latitude and longitude coordinates from the user.

Table 2.2: Tips from a User Endpoint

USER_ID	XXX123YYYY	Identity of the user to get tips from. Pass self to get tips of the acting user.
sort	recent	One of recent, nearby, or popular. Nearby requires geolat and geolong to be provided.
ll	33.7,44.2	Latitude and longitude of the user's location.
limit	100	Number of results to return, up to 500.
offset	100	Used to page through results.

Note that, the HTTP GET requests appends the token access value into the url, since this endpoint requires acting as a user.

```
https://api.foursquare.com/v2/users/self/tips?sort=recent
&oauth_token=WMGWPDKI0BBHKLTLCHGWW34YQYDNIS4M3F2HMAA00GMCQHZF
&v=20140213
```

2.5.3 Rate Limits

Foursquare also controls the API usage by establishing limits on the number of requests per hour. The limits distinguish between authenticated request and userless requests. It also grants some privileges to userless venue endpoints.

- Userless venue endpoint can be queried up to 5.000 times per hour.
- Other userless endpoints are limited to 500 request per hour.
- Authenticated endpoints can be queried 500 times per hour and per authenticated user.

2.5.4 The Foursquare Crawler

The proposed Foursquare Crawler aims to extract geo-positioned and time-stamped tips from venues in a dense geographical area such as Manhattan.

To tackle this case study, we first need to extract all venues in the delimited region. In order to achieve this, we propose to use the Foursquare venues/search endpoint in conjunction with the Quadtree data retrieval technique presented in previous section.

The Foursquare venues/search endpoint provides up to 50 venues objects in the JSON results structure. The most relevant parameters from this endpoint are the location related parameters. Among them, *sw* and *ne* parameters allow to specify the south-west and north-east corners of the rectangular region. It is also possible to specify a circular region through *ll* and *near* parameters, but we choose a rectangular shape because of its ease of use in combination with Quadtree. Last, *intent* parameter also need to be set to the value 'browse' in order to ensure that the query will return venues in the specified area without taking any other consideration.

Using the venues/search endpoint in conjunction with Quadtree means that in each area and sub-area of the Quadtree, the system will query the Foursquare platform through this endpoint. Consequently, the maximum cell capacity will be 50 venues, what means that the average cell capacity should be in between $1.33 < N_{cell} < 50$. The system implementing this solution has to also take into account the rate limitations of 5.000 requests/hour.

Once the Quadtree ends collecting all venues, the next step is to gather all tips from each venue. This can be achieved through the endpoint which provides the Tips from a Venue, which was presented previously. As it was shown there, given a venue id, the endpoint returns up to 500 tips in a JSON structure which belongs to that venue. In case the venue has more tips, the system will need to page through the results using the parameter. Note that this end-point can also query 5.000 times per hour the API.

Venues and tips are dynamic information, meaning that new venues and/or tips might be created and the system should be able to incorporate them into the already cached or stored

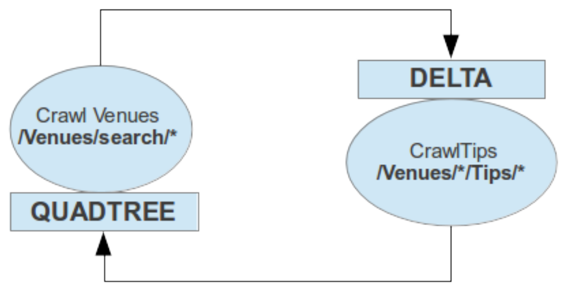


Figure 2.8: Foursquare Crawler System to retrieve geo-positioned tips.

data. Even though, it might also be technically possible that venues and/or tips are deleted from the Foursquare Platform, the tips objects, so far, do not have a time-stamp indicating the change or deletion of a tip; hence the only way to identify them would be crawling all them again. Hence, our implementation would not take into account deleted or changed tips, but it does consider if venues are deleted from the Foursquare platform by pruning the Quadtree when necessary.

To keep the system up-to-date, we propose to iterate in a close-loop between the two afore-mentioned stages: collecting all venues via Quadtree and crawling the tips from venues. Even though, the iteration between the two stages could also happen concurrently, in what follows, we consider each process to be executed after the other finishes.

Fig. 2.8 represents the two states in which the Foursquare crawler might be. Either crawling venues via the execution or maintenance of the Quadtree or crawling tips from venues by means of the delta process.

As stated before, the Quadtree can be easily maintained by pruning and/or creating the leaves of the tree, avoiding the complete re-execution of the Quadtree process. Hence, our proposal is to execute the Quadtree maintenance after the initial Quadtree set-up, instead of re-creating the full Quadtree in every step.

It happens similarly with the Venue's tips. Retrieving the full set of tips from each venue is possible but not desirable. We can do it more efficient by using the Delta technique and the parameter from the endpoint 'Tips from a Venue' called *sort* which allows ordering the result by time of creation.

By querying each venue with this parameter and then checking whether new tips have been created after the last delta, the system would not waste several useless requests.

Foursquare Crawler Results

We have executed the proposed Foursquare Crawler for the Manhattan area delimited limited by the bounding box with latitude and longitude coordinates south-west (40.70,-74.07) and north-east (40.80,-73.91) for all venues with a category belonging to restaurants.

The results of the Quadtree execution were 29.177 venues or restaurants extracted in 1.982 queries which took less than an hour. Intuitively, it is already clear that 1.892 queries taking up to 1 hour is a better result than 29.177 individual queries which would take more than 5 hours due to the Foursquare limitation of 5.000 requests/hour . By knowing that the Quadtree has 1450 nodes leaves, it is easy to compute the average capacity per cell, $\mathrm{N}_{\mathrm{cell}}$, which clearly satisfies the condition in equation (2.8) which states that $N_{\mathrm{cell}} = 19.6 \gg 1.33$.

Chapter 3

User Text Reviews Model

3.1 Model Overview

In order to recommend items to a user, we need to uncover user's preferences, *profiling*, as well as the item's features. For example, in order to recommend a buffet Japanese restaurant which serves high quality expensive food, it is essential to identify the restaurant main features (e.g. Japanese, High quality food, Expensive, Buffet) and check whether it might appeal to a given user according to his/her preferences.

The importance of user's feedback about a specific item is two-fold. On the one hand, it provides information about the item's features. When averaging user's feedback, the item's features can also be inferred from them. On the other hand, the user's feedback helps to determine what the user thinks about different features constituting the so-called user profile.

User's feedback usually comes in the form of numeric rating. E-commerce portals, such amazon.com, offer the possibility to assign stars or ratings to each item. Without a doubt, this type of user feedback indicates the acceptance/rejection of the overall products' features, but it says nothing about the individual features. Other web portals, such tripadvisor.com, allows users to assign ratings to a set of hotel's features. Undoubtedly, this is even more insightful for other users surfing the portal. However, this still limits the features that one can assess.

User text reviews or feedback provide full freedom to users to express their feelings and opinion about a specific item. Most of today's web portals already offer the chance to write text reviews. However, there are few Recommender Systems in the literature that are based on the user text reviews. This is due to the difficulties to mathematically model and interpret the text reviews in such a way that can provide a better recommendation accuracy.

Our proposed review model assumes that a good review will be composed by features or topics indicating the characteristics of an item and it will also contains the sentiment or opinion about these characteristics. Consequently, our text review model will be composed of:

- **Review topics or features**, which will be represented by a set of k keywords, objects or proportions of them.
- **Review sentiment or opinion**, which will be expressed by means of a number that ranges from an extreme value indicating negative sentiment to another extreme indicating positive sentiment.

Note that the model assumes the fact that a good review is provided. However, most of the times reviews are very noisy, not normalized and often they have poor quality. Because of this, it is extremely necessary to first clean text reviews data.

3.2 Cleaning Reviews Data

Handling reviews text data tends to be a complex pre-processing task due to the unstructured nature of text and the different writing styles, languages and argots. Moreover, the usefulness of a review also plays a critical role when trying to gain some knowledge from reviews.

In our methodology, we identify three critical sub-processes when cleaning the social reviews data which are key to achieve a high quality data set. It is important to highlight that we might be losing some relevant information for the application in mind when cleaning text data. Hence, is it highly recommendable to identify which information needs to be kept and do not always implement all data cleaning sub-processes listed next.

3.2.1 Language Identification

Detecting or identifying the language or even the argot of a review usually needs to be done beforehand, since most of the language processing techniques are language dependent.

Language identification from written and oral source material has been relatively well studied over several decades. Automated systems are widely used and they perform with quite high accuracies (99% in 3-gram models) in long and noiseless texts. However, social media sites brought a new challenge: short and noisy text. Classical language identification methods perform poorly (90-95%) in this type of data sets [29].

3.2.2 Text Normalization

Text normalization refers to the process of converting reviews text data into a canonical form, which can be then interpreted by Natural Language Processing (NLP) techniques.

Text normalization has been widely used in text-to-speech applications or in normalizing numbers, dates or acronyms. However, the appearance of SMS, chat rooms, and, lately, social media sites has motivated its use to normalize non-canonical text content generated in this sites.

Expressions such *See u 2night @ rest A* needs to be somehow translated into a standard sentence interpretable by the NLP systems.

Authors in [30] propose a method which identifies and normalizes lexical variants based on morphophonemic similarity. The proposed system also takes into account the context of the word.

3.2.3 Other Normalization Techniques

Simpler standardization or normalization techniques are often used in Natural Language Processing (NLP) with suitable performance for the applications in mind. Most of them are direct mappings to lower case versions of the text, text without punctuation or text without white spaces.

- **Convert to lower case.** As it indicates the naming, text is all converted to lower case.
- **Strip white spaces.** Consecutive white spaces are usually meaningless and hence they are removed.
- **Stemming.** It consist in reducing derived words to the base or root form of the word.
- **Removing stop words.** It consist in filtering out a list of stop words which is not unique for all systems. It is mostly used in search machines which look for keywords.
- **Removing punctuation.** Punctuation is often removed in some information retrieval applications.
- **Removing numbers.** Some NLP application might also obviate numbers.

In our methodology, we aim to use at least these normalization techniques, which will provide a good recommendation accuracy at a lower computational cost. Nonetheless, we deeply motivate future work in this field to pay more attention on the performance gains of using more complex normalization techniques.

3.3 Modelling the Reviews Sentiment

Text sentiment analysis has been deeply studied in the past decade and several commercial applications of sentiment analysis have already been embedded into operational systems. However, the rise of social networks has fueled the study of these systems too.

To model the sentiment of the reviews, we consider the Bag-of-words model for the text corpora. This model assumes text documents consist of set (Bag) of words without taking into account the word's position, neither the sentence's grammar. However, this simple model does account for the word multiplicity and it can be jointly used with context-aware NLP techniques to synthesize the dependencies among words.

The reviews model consist in a sentiment category assigned to each review done by a user about an item. Categories can be either positive, negative or neutral depending on the polarity of the text.

A review can have several sentences and each sentence, different sentiments; the model averages all sentiments in a review and outputs only one sentiment per review. Similarly, each review or sentence could have several aspects with different sentiments. The proposed model considers the overall sentiment of the review.

3.3.1 Sentiment Classifiers

The basic task in Sentiment Analysis is polarity classification of a review given a set of features (words, topics, synsets, etc.).

The features for the proposed reviews model will be words or topics, although we will ultimately get the topics from the context in a bag of words.

With the aim of identifying the statistical model that outperforms the sentiment classification for reviews, we will next present several classifiers which can be used with the reviews model for sentiment.

Naives Bayes Classifier

The Naives Bayes Classifier for Sentiment Analysis is a simple probabilistic bayesian paradigm which assumes independence among the features of a review [22]. Features in the proposed reviews model are the bag-of-words composing the review.

The probability model for the Naives Bayes Sentiment Classifier is derived from a conditional model that can be re-written in terms of the Bayes Theorem as follows:

$$p(C|W_1, W_2, ...W_N) = \frac{p(C)p(W_1, W_2, ...W_N|C)}{p(W_1, W_2, ...W_N)} \tag{3.1}$$

where C represents the sentiment category which can be either positive, negative or neutral. $W_1, W_2, \ldots W_N$ are the N features or words from the review model.

Since the denominator or evidence does not depend on the category, the probability model only considers the numerator which can also be expressed in terms of the join probability and its equivalence using the chain rule.

$$\begin{aligned} p(C|W_1, W_2, ...W_N) &\propto p(C, W_1, W_2, ...W_N) \\ &\propto p(C)p(W_1|C)p(W_2|C, W_1)...p(W_N|C, W_1, W_2, ...W_{N-1}) \end{aligned} \tag{3.2}$$

Given that Naive Bayes assumes independence among words, this implies that the probability of a feature given a class and other features simply equals to the probability of the feature given a class. Hence,

$$p(C|W_1, W_2, ...W_N) \propto p(C) \prod_{n=1}^{n=N} p(W_n|C) \tag{3.3}$$

The parameters of the Naive Bayes probability models are usually estimated by calculating the frequencies of each class from a training data set, named the Maximum Likelihood Estimation (MLE)

Finally, the classification task is performed by picking the most probable hypothesis or maximum a posteriori (MAP) that arises from the probability model. In other words, the classifier chooses the category C that maximizes the conditional probability.

$$\underset{c}{\operatorname{argmax}}\, p(C) \prod_{n=1}^{n=N} p(W_n|C) \tag{3.4}$$

Multinomial Logistic Classifier

The Multinomial Logistic regression generalizes the logistic regression to the multi-class situation [23]. Since we aim to estimate three possible outcomes for each review, positive, negative and neutral reviews, a multi-class classifier is needed.

The Multinomial Logistic Classifier estimates the probabilities of the possible outcomes based on a set of features. To differentiate with the Naive Bayes classifier, there is no need for statistical independence on the set of features or words used in multinomial logit regression.

One strong assumption that the Multinomial Classifier needs to satisfy is the so-called Independence of Irrelevant Alternatives. This means that if a review is considered more positive than negative in a binary or two-class classification problem, adding the neutral class, multi-class problem, should not change the sentiment of the review.

The mathematical model for multinomial logit regression relies on the independence of irrelevant alternatives in the fact of modeling the multinomial as a set of independent binary logistic regressions. Hence, the K-multinomial classification of a review modeled by a bag-of-words, W, consist in estimating the probability for each class of belonging to that class and hence, not to the others.

$$
\begin{aligned}
p(C=1) &= \frac{e^{\beta_1 W}}{1+\sum_{k=0}^{K-1} e^{\beta_1 W}} \\
p(C=2) &= \frac{e^{\beta_2 W}}{1+\sum_{k=0}^{K-1} e^{\beta_2 W}} \\
&\dots \\
p(C=K) &= \frac{e^{\beta_K W}}{1+\sum_{k=0}^{K-1} e^{\beta_K W}}
\end{aligned}
\tag{3.5}
$$

Where β_k are the regression coefficients associated to class k and W is the set of words associated with the review.

A drawback, compared with Naive Bayes, is the fact that estimating the regression coefficients from a Multinomial classifier is much complex and generally requires an iterative process.

sLDA Classifier

The Supervised Latent Dirichlet Allocation (sLDA) classifier for Sentiment Analysis is based on a topic model called LDA which incorporates a response variable with each document [24]. The response model for a Sentiment Analysis classifier corresponds to the sentiment class: positive, negative or neutral.

The documents and the responses are jointly modeled to find the latent topics that will best predict the responses for future unlabeled documents. In Fig. 3.1, a sLDA model is represented graphically. As it can be seen, the sentiment response results from the topics assignments in each document.

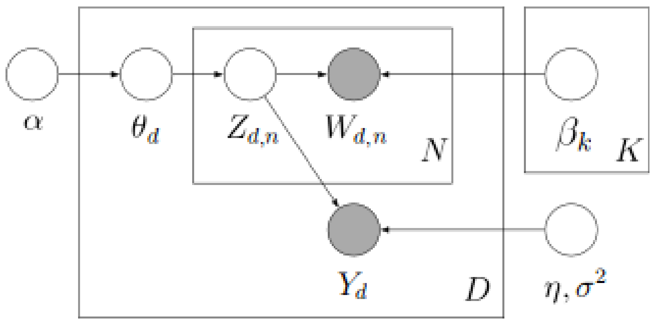


Figure 3.1: A graphical model representation of Supervised Latent Dirichlet Allocation [24].

In fact, the sentiment response comes from a normal linear model whose covariates are the frequencies of topics per document, $Z_{d,n}$, and the regression coefficients, η, are estimated in an iterative process.

3.4 Modelling the Review Contents

The review contents represent the description or summary of the key points expressed in a review organized in such a structured way that enables to computationally handle reviews with this representation. Extracting the contents from text reviews requires using Natural Language Processing (NLP) techniques on the text corpora.

Our aim is to achieve a representation of the review contents by means of a vector of K values. Preferably, K should be some value large enough to guarantee a good review model and as short as possible to make the computation fast. This trade-off needs to be satisfied in all NLP techniques that will be presented afterwards.

3.4.1 Content Models

There is a variety of content models available in the current literature. In what follows, we will introduce three approaches to be jointly used with the review contents model. All three techniques map the contents of a review into different terms in the vector of length K.

TF-IDF Keyword Model

The TF-IDF (Term Frequency - Inverse Document Frequency) can be used to map the contents of a review into a set of K keywords depending on the measurement of importance through the calculation of a numeric statistic [21].

The numeric statistic is calculated for each word from the review. The text review can be pre-processed using some of the techniques mentioned earlier. Nonetheless, the statistic will give few importance to stop-words, punctuation, etc. Therefore, some of the pre-processing techniques might not need to be applied, since the TF-IDF will filter out them.

The metric is composed of two sub-metrics, named TF and IDF. The former, $TF_{i,j}$, represents the normalized frequency, $f_{i,j}$, of term k_i in a document or review r_j which can be written as:

$$TF_{i,j} = \frac{f_{i,j}}{max_z f_{z,j}} \tag{3.6}$$

where the maximum is computed over the frequencies $f_{z,j}$ of all keywords k_z that appear in the review r_j. The latter sub-metric, IDF_i, is included to negatively weight those words that appear in many reviews and hence, they are not useful to distinguish between reviews (e.g. stop words). The Inverse Document Frequency is defined as,

$$IDF_i = log\frac{N}{n_i} \tag{3.7}$$

where N is the total number of reviews and n_i is the number of reviews that keyword k_i appears.

The final TF-IDF statistic for keyword k_i in the review r_j is defined by combining 3.6 and 3.7 into,

$$w_{i,j} = TF_{i,j} IDF_i \tag{3.8}$$

The content for a review r_j is finally defined as,

$$Content(r_j) = (w_{1,j}, w_{2,j}, ...w_{K,j}) \tag{3.9}$$

where the number of keywords, K, will be set to satisfy the trade-off between the computational cost and the recommendation accuracy.

LSA Concept Model

The LSA (Latent Semantic Analysis) Model can be used to map the contents of a review into concepts which turns out to be words that are close in meaning [20].

The LSA is generated from computing the single value decomposition (s.v.d.), X, of a term-document matrix which contains the terms or words, t_i, in the rows and the documents or reviews, r_j, in the columns. The real number in the interjection of rows and columns indicates the occurrence of terms in the document.

$$\mathrm{X} = \begin{array}{c c} & r_j \\ t_i & \begin{pmatrix} k_{1,1} & k_{1,2} & \cdots & k_{1,j} & \cdots & k_{1,r} \\ k_{2,1} & k_{2,2} & \cdots & k_{2,j} & \cdots & k_{2,r} \\ \vdots & \vdots & \ddots & \ddots & \cdots & \vdots \\ k_{i,1} & k_{i,2} & \cdots & k_{i,j} & \cdots & k_{i,r} \\ \vdots & \vdots & \ddots & \ddots & \cdots & \vdots \\ k_{w,1} & k_{w,2} & \cdots & k_{w,j} & \cdots & k_{w,r} \end{pmatrix} \end{array}$$

Assuming that exist a single value decomposition of X, this can be expressed in terms of its left and right eigenvectors and its eigenvalues matrix,

$$X = U\Sigma V \tag{3.10}$$

When picking the K largest eigenvalues from matrix Σ, and their corresponding left and right eigenvectors, one gets the rank K approximation of X with minimum error according to the Frobenius norm.

This approximation allow us to work in a lower dimensional space in which a given document or review, r_j, can now be approximated to a K-length vector, $\bar{r_j}$, in the lower dimension. Even though the new dimensions do not intuitively relate to the higher dimensions, some properties such as the similarity among documents or reviews are kept.

LDA Topic Model

LDA (Latent Dirichlet Allocation), [19], is often used to model reviews contents into topics that occurs in a review. Intuitively, a review can talk about the *service* of a *Japanese* restaurant, and another review refers to the *food* of a *local* restaurant. Both reviews will lead to different topics if the space of topics, K, is large enough to differentiate among them. LDA can be used to model other collection of discrete data rather than text corpora.

LDA is a generative probabilistic model in which documents or reviews are represented as random mixtures over latent topics, and each topic is characterized by a distribution over words.

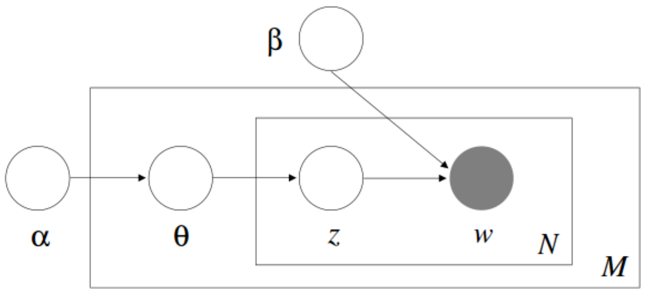


Figure 3.2: A graphical model representation of Latent Dirichlet Allocation [19].

Fig. 3.2 is a graphical representation of the Latent Dirichlet Allocation in plate notation. The boxes in the figure are plates. The outer box represents the plate of M documents. The inner box, the plate of N words and the topics assigned to these words. α is the Dirichlet prior parameter on the per-document topic distributions. β is the Dirichlet prior parameter on the per-topic word distribution. θ_i is the topic distribution for document i. $z_{i,j}$ is the topic for the jth word in document i. Finally, $\omega_{i,j}$ is the specific word or observable variable.

To solve this probabilistic model, we need to compute the posterior distribution of the hidden variables given a document:

$$p(\theta, z|w, \alpha, \beta) = \frac{p(\theta, z, w|\alpha, \beta)}{p(w|\alpha, \beta)} \tag{3.11}$$

However, the computation of the posterior distribution is intractable for exact inference and several approximate inference algorithms have been proposed in the literature such Laplace approximation, variational approximation, and Markov Chain Monte Carlo (MCMC).

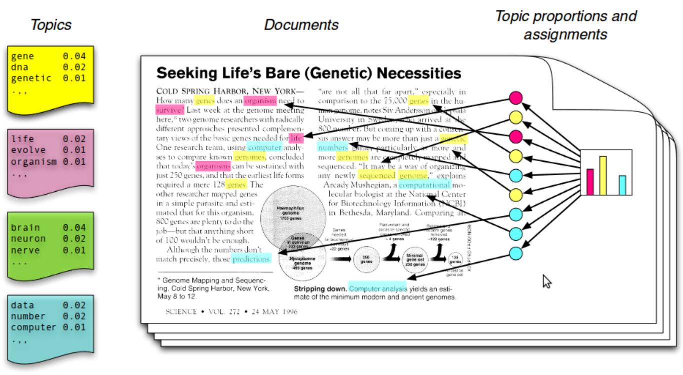


Figure 3.3: The interpretation of the LDA results [31].

As we can see in Fig. 3.3, topics (in different colors) are assigned to words in a document by means of $z_{i,j}$. Each document has a distribution over topics, θ_i, as it is shown in the right hand bar plot. At the same time, it also exists a distribution of words on a topic as it is shown in the left side of the figure, which is generated by the Dirichlet process with prior β.

Chapter 4

Review-based Hybrid Recommender System

4.1 System Overview

In this chapter, we present the proposed Hybrid Recommender System that uses reviews from Social Networks to make more accurate recommendations purely based on this type of user feedback.

The proposed Recommender System uses the reviews models introduced in Chapter 3 to estimate rates for the unseen items of a given user. A rating-based Recommender System can be understood as follows. One defines a function $f : U \times I \to R$, called the utility function from the set of all users, U, and the set of all items that can be recommended, I. This function provides a rate or measure of importance, R, associated to each pair of user and item. The problem of recommendation can be expressed as,

$$i_u = \operatorname*{argmax}_{i \in I} f(u, i) \forall u \in U \tag{4.1}$$

meaning that for each user u we need to get the item that maximizes the utility among all available items. However, the utility function is usually not defined for the whole space $U \times I$, and hence the recommender needs to estimate the missing ratings to perform an accurate recommendation.

There are mainly two approaches to estimate the missing ratings:

- Using heuristics that define the utility function and then validate empirically its performance.
- Estimating the utility function that optimizes a performance criteria, such the mean square error.

Once the missing ratings are found, the recommendation can be done over the whole space as specified in equation (4.1).

The proposed Recommender System will use heuristic methods to define the utility function from the user preferences and item features which are, in fact, inferred from the reviews.

Depending how the recommendation is made, the recommender engines are usually classified as:

- *Content-based recommendations.* It uses the item's features and the user profile to make the recommendation. *"Show me more of the Facebook groups that I like".*
- *Collaborative recommendations.* It uses the preferences of the community or the wisdom of the crowd. *"Show me more of the Facebook groups that are popular among my friends".*
- *Hybrid approaches.* It somehow combines both types of recommendation. *"Show me more of the Facebook groups that I like and that they are popular among my friends".*

The proposed system will combine Content-based and Collaborative recommendation into an hybrid final recommendation. The Hybrid approaches is capable of minimizing the two main drawbacks of Content-based and Collaborative methods: the overspecialization and the sparsity respectively. Both approaches as well as their drawbacks will be treated in depth in the following chapters.

4.2 Collaborative branch

The collaborative branch of the review-based Recommender Systems aims to gather the opinion of the crowd from the posted reviews. Recommendation will be done based on the opinion that the close neighborhood has on a specific item. The individual opinion of your neighbors can be extracted from the review sentiment model introduced in the previous chapter.

Following the nomenclature introduced in (4.1), the *utility function*, f, of the collaborative branch for the rated items will directly be the *Reviews Sentiment Model* which extracts the opinion from user reviews done about the items. For the unseen items, the heuristic proposed is an averaging over the rates of those neighbor users who have seen the item as shown next,

$$r_{u,i} = \frac{1}{N} \sum_{u' \in N_u} r_{u',i} \tag{4.2}$$

where N_u represent the N users in the neighborhood of u. Determining N and the neighborhood is key to achieve a good rate estimate for the unseen items.

4.2.1 Neighborhood Identification

The neighborhood identification plays a key role to achieve an accurate prediction of the recommender rate.

Recommender Systems have traditionally used the similarity among users from the matrix of user-item ratings [14]. Similarity usually takes into account pairs of ratings that both users have rated. Similarity measurement has been usually calculated by means of a heuristic mechanism such the Pearson correlation coefficient or cosine-based similarity.

We proposed to use the cosine-based similarity for the review-based recommender collaborative branch, as the heuristic measurement of similarity of two pairs of user vectors $\overrightarrow{u_x}$, $\overrightarrow{u_y}$. The user vectors are defined from sentiment ratings over all items.

$$sim(u_x, u_y) = cos(u_x, u_y) = \frac{\overrightarrow{u_x}\overrightarrow{u_y}}{||\overrightarrow{u_x}||||\overrightarrow{u_y}||} \tag{4.3}$$

With this approach, pairs of users who have both rated positively or negatively pairs of items, they will share similar opinion for the unseen items. The cosine-based similarity becomes a useful metric to identify them.

Lately, Social Recommender Systems have introduced novel mechanism to identify the neighbor for collaborative filtering. Authors in [8] propose to use social friendships or relationship to generate the neighborhood for a collaborative recommender. According to them, this approach outperforms the traditional collaborative filtering. We will examine this mechanism in our case study with Foursquare data.

4.3 Content-based branch

The content-based recommendation branch will bring into the system the user preferences with the purpose of making recommendations according to the user profile. Hence, user and item profiles, defined from the reviews content models introduced in Chapter 3, will be compared to find the most similar matches.

Following the introduced nomenclature, the *utility function* for the content-based branch will predict the rate for both seen and unseen items. The rate, R, will be predicted from the users profile vectors, w_u, and items features vectors, w_i. As a result, the *utility function* can be redefined as $f_{col} : w_u \times w_i \to R$.

Due to the fact that items and users spaces are large, mechanism to reduce dimensionality or filtering items/users are often used.

4.3.1 Similarity Measurement

Similarity measurement among pairs of users and items can be either calculated using heuristic models or statistical models learned from the underlying data. The former was already introduced for the neighborhood identification in Section 4.2.1. The latter precises having previously tagged some items as relevant or irrelevant in order to learn a model which can predict unseen items.

Our proposed approach for the content-based branch also uses a cosine-based heuristic model in order to compute the similarity rate between user profiles and items features. Hence, the *utility function* formulation can be expressed as:

$$f_{cont} : w_u \times w_i \to R \implies f_{cont} = cos(w_u, w_i) = \frac{\overrightarrow{w_u}\overrightarrow{w_i}}{||\overrightarrow{w_u}||||\overrightarrow{w_i}||} \tag{4.4}$$

where w_u and w_i are the profile vectors for the users and items defined from the reviews content models previously introduced. The K vector values might represent keywords in TF-IDF, concepts in LSA or topics in LDA.

4.4 Hybrid Linear Combiner

Using a hybrid linear combiner of the collaborative and content-based branches, we pretend to minimize the drawbacks of each individual approach.

$$f : U \times I \to R \implies f : f_{col} + \alpha f_{cont} \tag{4.5}$$

On the one hand, collaborative approaches highly depend on the availability of a critical mass of users who has rated enough items to effectively predict the the unseen items. One way to overcome *the sparsity of the user-item rating matrix* is to jointly use profiling data to identify the close neighborhood.

Our approach minimize the sparsity problem by combining the rates from the content-based branch when the collaborative rates are not accurate enough.

On the other hand, content-based approaches basically fail to recommend items whose features have not been rated yet by the user. In other words, the content-based recommender is *overspecialized* to the features from the seen items since user profiling arises from the items already seen by the user.

Item diversity is added into our system to overcome the overspecialization problem. By combining the collaborative sentiment into the proposed system, unseen features from unseen item will also be recommended enhancing the overall accuracy.

There also exist drawbacks which are common to both approaches. For example, the *new user problem* will happen when a user has not reviewed any items or very few. In this situation, the recommender will no be capable of performing a recommendation and this will lower the system coverage. If few reviews exist, it happens that the recommender makes unreliable recommendations. The *new item problem* is also a drawback difficult to minimize in both set-ups. It occurs when an item is new into the system and it has not yet been reviewed by any user. Since no user feedback exist for these items, the system is not capable of performing an accurate recommendation, also impacting to the system coverage.

Chapter 5

Evaluation Approach

5.1 Evaluation Overview

In this chapter, we present the approach used to evaluate the effectiveness of the Review-based Recommender System. The proposed methodology also relies on the user feedback materialized in short reviews via Social Networks.

The approach measures how effective would be the Recommender System on proposing the item that the user has just reviewed. The approach assumes that causality exists between the fact of purchasing an item and the posterior action of reviewing it. Moreover, the approach provides a simple but powerful mechanism to measure the system coverage, or simply, the proportion of accurate recommendations that the system outputs.

As explained before, Social Networks provide a channel to gather user feedback after having experienced an item. Therefore, Social Networks also enable us to compare the proposed Recommender System against the feedback provided by the user, u, about the specific item, i. The evaluation approach takes all available review data from this user and this item until the last feedback. With all this data, the aforementioned models are built and the recommendation rate is generated as explained in Chapter 4.

The recommendation rate, $f(u, i)$, is then compared with the user feedback F for this item. However, as a difference with other evaluation methodologies, the comparison between the recommendation rate and the actual opinion of the user about this item is not done absolutely such as in MAE (Mean Absolute Error), but relative to an ordered list.

The recommendation rates provided by the utility function are ordered downwards and the relative position of the item is then compared with the actual feedback F. The actual user feedback is separately extracted from the review, by means of the Sentiment model early introduced. The relative position and the actual user feedback are then compared absolutely.

Recommendation systems are usually assessed through the Receiver Operating Characteristic (ROC) curves. Here, we will also introduce this framework based in the previous approach in order to evaluate the proposed Review-based Recommender System.

Not least, the user might be reviewing an item for the first time, or even, it might happen that the item receives the first review. None of these cases, the proposed review-based system cannot accurately make a recommendation due to the lack of history of data. This phenomenon is also captured in this approach by measuring the percentage of recommendation that can be done among the overall.

The methodology is based in two metrics or indicators, named Coverage and Performance accuracy.

5.2 Measurement Indicators

Next, we present and define the indicators used to assess the Recommender System.

5.2.1 Coverage

Coverage is a measure of the domain of items over which the system can make recommendations [32]. Typically, the term coverage has been associated with (1) the percentage of the items for which the system is able to generate a recommendation and (2) the percentage of the available items which effectively are recommended. Here, we adopt the former definition since the latter will be coped with Accuracy and ROC presented later.

Coverage is usually defined as the percentage of available items, I, for which the Recommender System can output a prediction, I_P.

$$Coverage = \frac{\mid I_P \mid}{\mid I \mid} \tag{5.1}$$

Although the coverage value often depends on the recommender engine as well as the available data, we prove next that the coverage only depends on the available reviews for the proposed reviews-based Recommender System.

The proposed recommender engine combines a collaborative and a content-based branch. While the collaborative branch is based on the average of rates of the K-nearest users, the content-based directly outputs a rate based on the profile similarity of user and item. Because of this, the content-based branch will always output some rate, as long as some information exists for that user and item. On the other hand, the collaborative-branch output will have always less or the same coverage than the content-based branch, since it also depends on the items that are commonly reviewed by the users. If no user is similar to the base user (a user who has reviewed items common with the base user), this branch is not able to output a recommendation value. Nonetheless, the overall coverage will always be the content-based coverage, since the combiner ensures a rate as long as either the content-based or the collaborative is able to recommend.

Therefore, the system coverage is constrained to the existence of a user and item profile modeled using the content models introduced in Chapter 3. As described, a profile only exists in case that there are reviews related to the item and from the user. Otherwise, the item or user are considered to be new into the system, respectively.

Since all existing items (items with one or more reviews) can be predicted and recommended, the item coverage can be reformulated in terms of the new, I_N, and existing items, I_E as follows,

$$Coverage = \frac{\mid I_P \mid}{\mid I \mid} = \frac{\mid I_E \mid}{\mid I \mid} = \frac{\mid I_E \mid}{\mid I_N + I_E \mid} = \frac{\mid 1 \mid}{\mid 1 + \frac{I_N}{I_E} \mid} = \frac{\mid 1 \mid}{\mid 1 + \alpha \mid} \tag{5.2}$$

where α is the proportion between new and existing items. Whenever this proportion tends to zero, meaning that the system has reached a stable number of items, the coverage tends to one. User coverage will also tend to one when the proportion between new and existing users tends to zero.

Given that Social Networks are continuously growing due to the fact that new actors enrol into them, assuming that the proportion will tend to zero, might not be realistic at all. However, formulation in (5.2) also points out that the system coverage can be enhanced by adding new information into the Recommender System. By enhancing the contents model with other existing information rather than reviews (e.g. friendship relationships, check-ins, amongst others); the system coverage can be greatly impacted [33].

5.2.2 Performance Accuracy

Accuracy measures the quality of nearness to the true value achieved by a Recommender System. Typically, it is formulated as the number of successful cases over all the cases,

$$accuracy = \frac{\#\text{ good cases}}{\#\text{ overall cases}} \tag{5.3}$$

As a difference with most of the evaluation methodologies in Recommender Systems, our approach assumes that a successful recommendation is the one that leads the user to a positive experience. The positiveness of the experience is measured through the user feedback in the review, while the recommendation is generated by the engine.

Since the overall number of cases can be decomposed by positive and negative cases, the accuracy can be also expressed in terms of the average error rate, $\overline{\epsilon}$, as,

$$accuracy = 1 - \frac{\#\text{ negative cases}}{\#\text{ overall cases}} = 1 - \overline{\epsilon} \tag{5.4}$$

where ϵ is the recommendation error rate for each experiment, k, and it is defined as follows,

$$\epsilon_k = \begin{cases} \frac{N_{pos_k}}{N_k} & \text{Sentiment is } \textbf{positive} \\ 1 - \frac{N_{pos_k}}{N_k} & \text{Sentiment is } \textbf{negative} \end{cases} \tag{5.5}$$

where N_k is the number of items recommended in the k experiment and N_{pos_k}, the position in a ordered list that occupies the item that the user has experienced. Moreover, the positiveness of the experience is determined based on the sentiment models introduced in Chapter 3.

5.2.3 Receiver Operating Characteristic (ROC) Curves

Recommender Engines are often assessed via the Receiver Operating Characteristic (ROC) curves. The ROC curves provides a powerful mechanism to compare different Recommender set-ups taking into account the *precision* (True Positive Rate) and the *fallout* (False Positive Rate).

These metrics can be understood by means of a confusion matrix, which links the recommendable items and the success of the experience. Note that the construction of the confusion matrix requires discretizing the predicted recommendation list in two: recommended and non-recommended items. Moreover, the actual values also have to be split into two according to the positiveness of the experience.

	Success	
	Positive Sent.	**Negative Sent.**
Recom.	True Positive	False Positive
Non-Recom.	False Negative	True Negative

Table 5.1: Confusion Matrix for a Recommender System.

The True Positive Rate (TPR) or *precision* is defined in this matrix as,

$$TPR = \frac{TP}{TP + FP} \tag{5.6}$$

And, the False Positive Rate (FPR) or *fallout* is defined as,

$$FPR = \frac{FP}{FP + TN} \tag{5.7}$$

In general, if the probability distributions for precision and fallout are known, the ROC curve can also be generated by plotting the Cumulative Distribution Function of the precision probability in the y-axis versus the CDF of the fallout probability in x-axis.

Both probabilities as well as their respective cumulative functions can be experimentally calculated from the samples in the data. *Precision* is equivalent to the *accuracy* defined in Eq. (5.4) for the successful experiences. While the *fallout* is the reverse of the *accuracy* for the unsuccessful experiences.

Precision and Fallout expression can be formulated as follows:

$$\begin{aligned} Precision &= \tfrac{N_{pos_k}}{N_k} \quad \text{when Pos. Sentiment} \\ Fallout &= \tfrac{N_{pos_k}}{N_k} \quad \text{when Neg. Sentiment} \end{aligned} \tag{5.8}$$

where N_k is the number of items recommended in the k experiment and N_{pos_k}, the position in a ordered list that occupies the item that the user has experienced.

ROC Optimization can be graphically represented as pushing the ROC curves to the upper-left corner, in which the recall is maximum and the fallout is minimum. Fig. 5.1 shows this interpretation.

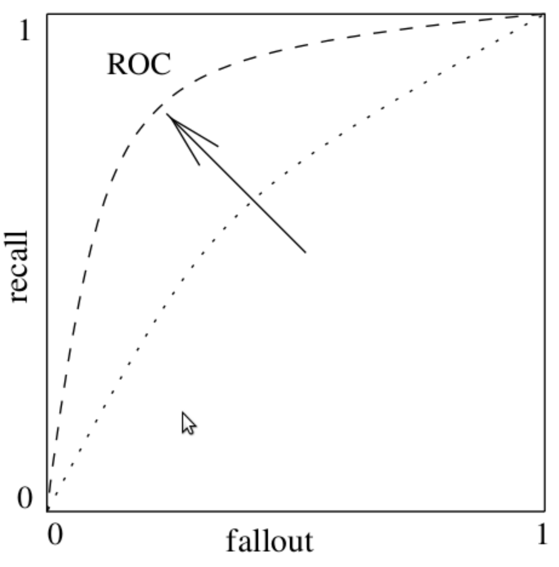


Figure 5.1: ROC Optimization curves [34]

Finally, the Area Under the Curve (AUC) statistic which is the integral of the ROC curve and it represents the probability that a classifier will rank a randomly chosen positive sample higher than a randomly chosen negative one.

Part II

Application

Chapter 6

A Foursquare Tip-based Restaurant Recommender System

6.1 Overview

In the application part of the thesis, we bring the methodological concepts into practice. We build a Social Recommender System based on the restaurant reviews crawled from Foursquare using the Foursquare Crawler described in Section 2.5.4. Reviews are also known as tips in the Foursquare platform and they are the spontaneous user feedback from his/her experience in a venue. There are different types of venues, but the recommender engine will only be trained and assessed on restaurant data, although it could be used with other types of venues too.

Reviews or tips data is cleaned and modeled according to the proposed paradigms in Chapter 3. Based on these models, the introduced Hybrid Recommender System is built on top.

In the following chapters, the Foursquare Recommender System will be evaluated in terms of the metrics defined in Chapter 5.

6.2 The Foursquare Restaurant Tips Data set

The *Foursquare Restaurant Tips Data Set* consist of 309.640 short reviews from the Manhattan region. The whole data set is split into a train and a test data sets for learning and evaluation purposes. The train data set contains the 70% of the oldest tips, while the test data set, the 30% newest tips.

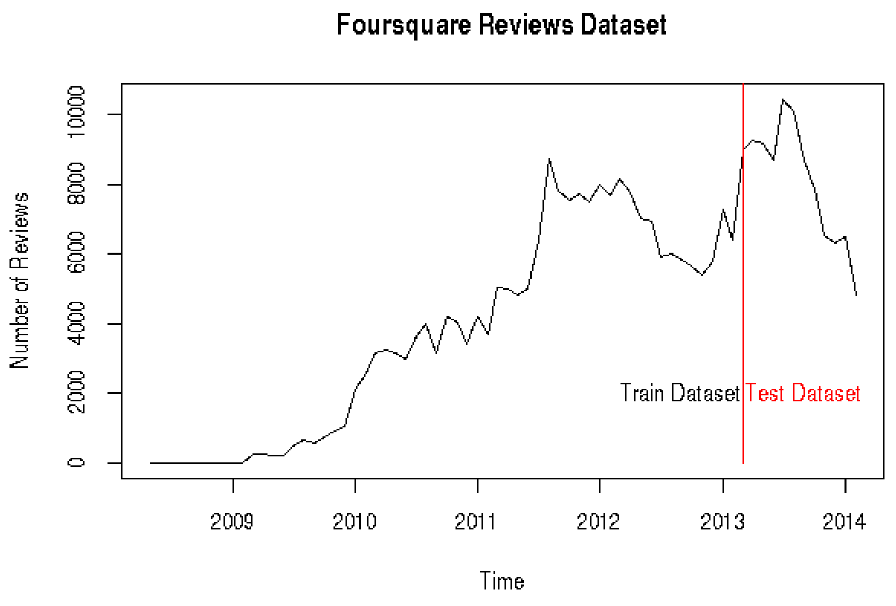


Figure 6.1: The number of Foursquare Restaurant Reviews as a function of time.

In Fig. 6.1, we can see the number of reviews as a function of time, from the first review in 2008 till the last crawling execution in February 2014. Moreover, this graphic shows a positive tendency in the number of reviews created since 2008, what indicates us that Foursquare platform has been growing since then, despite some seasonal patterns. As it has been said, the whole data set is split in two subsets, which are also indicated in the plot.

Data set	Num. Reviews	Num. Restaurants	Num. Users	Avg. Words/Rev.	Avg. Likes/Rev.	% Positive Words/Rev.	% Negative Words/Rev.
Train	216.748	12.664	76.448	15,6	1,2	1,05%	0,075%
Test	92.892	10.341	41.234	15,8	1,6	0,95%	0,068%
Full	309.640	14.782	104.040	15,7	1,3	1,02%	0,073%

Table 6.1: The Foursquare train and test data set features.

In Table 6.1, we describe the two datasets in terms of the text review features. Columns in the right side of the table, represents the average number of words per review, likes per review and percentage of positive and negative words per review. These features show that both data sets are balanced in terms of these review features.

6.3 Tip Models

6.3.1 Tip Normalization

Foursquare tip or review data is quite noisy due to the spontaneity of the tipping situation as well as the writer freedom associated with text. Tips are usually written directly from the venue itself through smartphones or other wearable device which can also entail spelling mistakes. Because of this, the first step in modeling tips consist in normalizing text data.

In the proposed Recommender System, we suggest to apply the following techniques which has been described in Chapter 3.

- Filter out non-English tips
- Remove English stop words
- Convert all text into lower case
- Strip white spaces
- Remove punctuation

Note that by pre-processing text data, we normalize tips achieving less dimensionality, but we might also lose some essential features that can be relevant when modeling reviews.

In Appendix A, we list the tools used to pre-process text data and generate normalized tips.

6.3.2 Tip Content Models

We apply to Foursquare tips data sets the three review content models introduced in Chapter 3: TF-IDF, LSA and LDA. The behaviour of each model will be reviewed in the next chapter in terms of recommendation accuracy and model complexity.

All three models work with text documents. In our recommender, we can have three types of documents, that constitute three types of corpus, all sharing the same vocabulary.

- Each venue (or Restaurant), a document composed of all tips for that venue.
- Each user, a document composed of all tips for that user.
- Each tip, a document.

While the two first represent the content profiling of venues and users respectively, the third document is used to model the contents of a tip.

TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF provides an statistic to reflect how relevant a word is to a document in the restaurant reviews Foursquare corpus introduced earlier.

The statistic computation can be programmed using statistical software as it is suggested in Appendix A.

The result of TF-IDF is shown in a *wordcloud*, Fig. 6.2, which reflects the relevance of words in a document from the corpus by means of different font sizes.

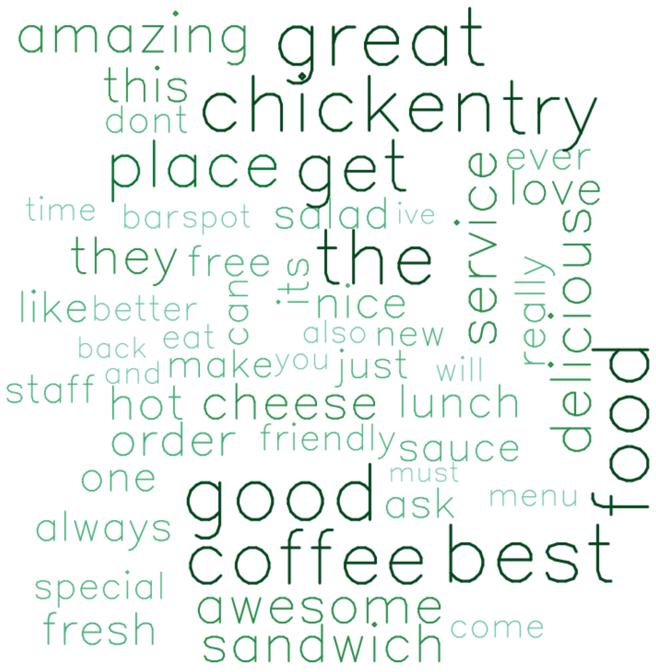


Figure 6.2: Foursquare Restaurant Tips Wordcloud

The vocabulary relevancy resulted from applying TF-IDF is then reduced to the K most relevant words or keywords. The combination of these keywords will model the contents of each document (venue, user or tip). The effects of choosing different K values are shown in the following Chapter 7 at section 7.2.2.

Latent Semantic Analysis (LSA)

LSA produces a set of concepts related to the documents and their terms, assuming that words close in meaning will occur in similar documents. The singular value decomposition (SVD) generates concepts from the Term-Document matrix.

LSA can be also computed using the statistical software shown in Appendix A. The resulted K concepts are not intuitive, but they can be seen as linear combinations of the underlying words. Different values of K will lead to different recommendation performances which will be shown later.

Latent Dirichlet Allocation (LDA)

LDA generates a mixture of topics that describe documents (venues, users or tips) composed by words attributable to at least one of the document's topics. Hence, each topic can be seen as a probability distribution over the word vocabulary.

There are software implementations of LDA, which assist computing the topics from a text corpus. In Appendix A, we list one statistical software package.

Table 6.2 shows 10 topics generated from applying LDA to the Corpus of restaurant Foursquare tips. It can be intuitively seen in the words of each topic, that different types of venues could be described by the combination of these topics. Moreover, different user tastes can be understood by means of topics too.

Topic 5	Topic 10	Topic 12	Topic 21	Topic 28	Topic 32	Topic 37	Topic 40	Topic 48	Topic 49
the	chicken	pizza	pork	burger	sushi	hour	food	cheese	beer
pasta	salad	slice	dumplings	fries	great	happy	mexican	eggs	great
italian	falafel	best	soup	best	lunch	night	margaritas	the	wings
amazing	sandwich	good	chicken	burgers	roll	great	burrito	chicken	bar
great	sauce	slices	fried	cheese	roll	drinks	get	bacon	good
delicious	get	chicken	rice	the	fresh	every	good	get	beers
try	the	crust	get	food	best	music	tacos	mac	selection
food	hummus	the	chinese	sweet	sake	live	place	sandwich	happy
salad	good	cheese	noodles	potato	restaurant	late	the	steak	hour
wine	lamb	great	beef	veggie	japanese	free	margarita	fried	the

Table 6.2: LDA Topics from a generative process of k=50

In Fig. 6.3, a restaurant called *Home Restaurant*, which it is an american cuisine restaurant, is represented by the LDA topics. As it can be seen, Topic 49 has most of the proportional weight.

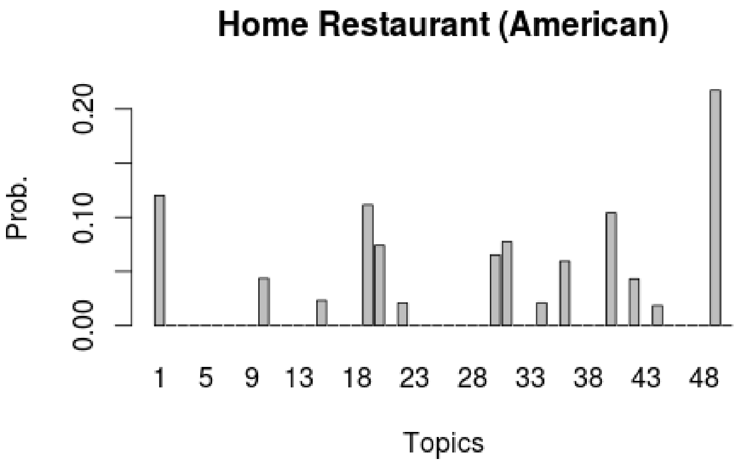


Figure 6.3: Venue content model from LDA topics.

6.3.3 Tip Sentiment Models

In this section, we apply Sentiment Analysis to the Foursquare tips data set. In particular, we extract the sentiment for each tip from the data set using the three introduced sentiment models: Naive Bayes, Multinomial and sLDA. Here, we always assume that one tip is composed of one document.

All three models are cases of supervised learning in which we train a model based on some tagged data. Then, the model is tested against a testing data set and validate it against a validation data set. In what follows, we describe how we build these three data sets from the training Foursquare set introduced earlier in section 6.2. Note that the testing Foursquare set is not used to assess the sentiment model, but it will be used when assessing the recommender.

- Training data set. It is composed of 1400 tips from the Foursquare training data set. These tips are tagged using AFINN [35], a list of english words rated for valence with an integer between minus five (negative) and plus five (positive). The rating is discretized in two values -1 and 1 via the sign function, indicating the most negative and most positive tips respectively.
- Testing data set. It is composed of 600 tips from the Foursquare training data set. Tips are also tagged using the AFINN approach.
- Validation data set. It is composed of 200 tips from the Foursquare training data set. In order to validate the model, we have manually tagged tips as positive or negative.

Naive Bayes

A Naive Bayes model is built from the Training data set using the sixth most relevant features or words based on a chi-squared test.

The feature selection as well as the model are built using the statistical tools described in Appendix A.

	Training Data set	Testing Data set	Validation Data set
Error Rate (%) with Feature Selection	30%	33,5%	35,4%
Error Rate (%) without Feature Selection	53%	49,6%	60,2%

Table 6.3: Bayes Sentiment Classifier

Table 7.2 shows the misclassification error rate in each of the data sets using a Naive Bayes classifier with the sixth most relevant features and without any feature selection. It clearly shows the benefits of using a feature selector in this type of model.

Multinomial

The Multinomial Logistic Regression is also built from the tagged data set by training a multinomial model as shown in Appendix A. Here, we do not perform feature selection and all vocabulary words are being used.

	Training Data set	Testing Data set	Validation Data set
Error Rate (%) with Feature Selection	29,14%	33%	35,4%
Error Rate (%) without Feature Selection	0,28%	3,16%	28,57%

Table 6.4: Multinomial Sentiment Classifier

Table 6.4 shows the misclassification error rate in each of the data sets using the multinomial classifier. As we can see, using feature selection does not guarantee a lower misclassification error.

Supervised Latent Dirichlet Allocation (sLDA)

The supervised LDA classifier builds the model on the training data set generating the topics associated to the output variable. The Appendix A shows a statistical package to built this classifier in R.

	Training Data set	Testing Data set	Validation Data set
Error Rate (%) with K=2	7,49 %	13,78 %	29,81 %
Error Rate (%) with K=50	9,51 %	13,78%	32,3%
Error Rate (%) with K=100	16,7 %	19,15%	29,19%

Table 6.5: sLDA Sentiment Classifier

Table 6.5 shows the misclassification error rate in each of the data sets using the supervised LDA classifier. The misclassification error rate is shown for different numbers of topics (K). Since the classifier differentiate between two possibles classes, choosing a K=2 leads to acceptable error rates.

Table 6.6 list the most common words for each topic in a Supervised Latent Dirichlet Allocation model with two topics. It can be seen that *Topic 1* contains positive-related words, while *Topic 2*, negative-related words.

Topic 1	get	go	best	great	good	order	amazing	s	like
Topic 2	service	food	bad	i	worst	the	terrible	ever	place

Table 6.6: Supervised LDA Topics for K=2

6.4 The Foursquare Hybrid Tip-based Recommender

The Foursquare Hybrid Tip-based Recommender system is composed of a collaborative filtering branch on the tips sentiment and a content-based branch which measures the similarity between user preferences and item features. This recommendation approach, which was proposed in Chapter 4, is used in this section to recommend Foursquare restaurants to users.

Both branch are combined by using a linear combiner with parameter α which allows different possible recommender configurations as shown in formula 6.1. The different set-ups are presented and evaluated in the next chapter.

$$f : U \times I \rightarrow R \implies f : f_{col} + \alpha f_{cont} \tag{6.1}$$

Chapter 7

Results and System Performance

7.1 Coverage Accuracy

Recommendation Coverage Accuracy was introduced in Chapter 5 and particularized for the proposed Recommender System. There, we saw that Coverage only depended on the existence of previous tips or reviews for the evaluated user/venue. We also proved that the proportion between new and old items/users directly impacts the item/user coverage respectively.

In Fig. 7.1, we plot the Coverage Accuracy as a function of time in order to show how the coverage softly increases when the number of tips starts to be large (see Fig 6.1) and the proportion between new and old items/users stabilizes (see Eq. (5.2)).

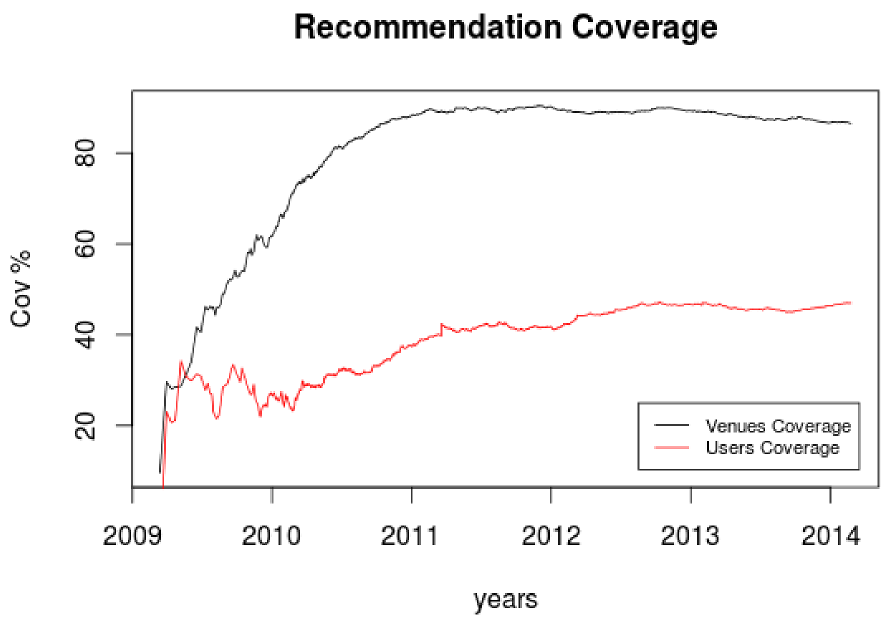


Figure 7.1: Recommendation Coverage Accuracy for venues and users.

From this picture, we can also see that while there is a good Coverage value on venues or restaurants (85% in 2014), Coverage on users is still low (45% in 2014). This is mainly due to two reasons. On the one side, there are users who still sign up in Foursquare in 2014, on the other, there are some users who have tipped Manhattan restaurants in 2014 for the first time. The same reason apply to Manhattan restaurants, but restaurant are not as dynamic as users and most of them has been receiving tips since 2009, when Foursquare was launched.

As suggested in Section 5.2.1, these numbers could be improved by adding into the recommender engine other information rather than tips. Tips and reviews are types of user feedback content which require some kind of typing effort. Other types of Foursquare contents such check-ins, likes and dislikes, would potentially increase the coverage numbers. However, since our aim was to evaluate a pure review-based engine, we did not pursue this direction.

7.2 Performance Accuracy and ROC Curves

Performance accuracy was defined in section 5.2.2 as the proportion of good cases against the overall cases. The accuracy was related with the recommendation error rate and the later was defined by the relative position that the evaluated restaurant occupies in the ordered recommendation list.

The position in this list is then compared with the positiveness of the experience. *Positive* experiences, expressed through positive sentiment tips in the test data set, require top positions in the ordered list in order to achieve minimum recommendation error. While *negative* experiences, need positions at the bottom of the recommendation list to achieve the same goal.

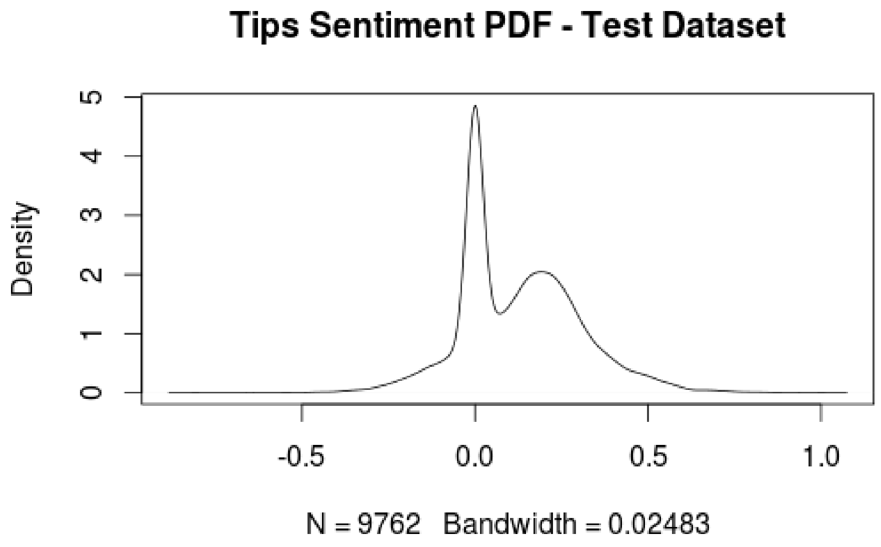


Figure 7.2: Tips Sentiment Probability Density Function

Fig. 7.2 plots the sentiment Probability Density Function (PDF) of tips in the test Foursquare data set. While negative values expresses negative sentiment, positive sentiment is contained in the interval (0,1]. On the other hand, neutral user feedback is represented by a sentiment value of zero. In our approach, we consider neutral feedback as a positive experience since most of them are tipping advice such as *"Try the hot chocolate!"*, in which the user implicitly rates the experience as positive.

Note also that the data set is quite unbalanced in terms of positive and negative ratings, although it reflects the reality of Foursquare tips data from early 2013 till 2014. This might not be extremely relevant when assessing the performance accuracy of this data set, but it is relevant when comparing different recommender set-ups and the generalization capabilities. In order to assess the Recommender Systems independently of the data set, we propose to use the ROC curves introduced in earlier chapters. ROC curves can be seen as the proposed evaluation model, which balances the numbers of experience opinions. In fact, we show that maximizing the AUC of the ROC curves is equivalent to maximizing the AUC of the error Cumulative Density Function (CDF).

7.2.1 Collaborative-based Model Results

The proposed collaborative models gather the opinion of the crowd by averaging the feedback from users whose taste is similar to the recommended user. As introduced in section 4.2, the N neighbors are identified through the k-nearest algorithm that uses the cosine similarity of the historical user experiences.

Three sentiment models were introduced in Chapter 3 to characterize the opinion hided in the user reviews. We have already seen in Section 6.3.3, the accuracy of these sentiment analysis classifiers when comparing the predicted sentiment against the sentiment in the tagged data set. Next, we assess Naive Bayes, Multinomial and SLDA models in terms of recommendation accuracy and compare their performance figures against the collaborative model without sentiment. The closer the AUC is to 1, the better it performs the model.

First, we plot the Area Under the Curves (AUC) of the Recommendation Error Cumulative Density Functions which let us interpret the goodness of the models with the available Foursquare data sets.

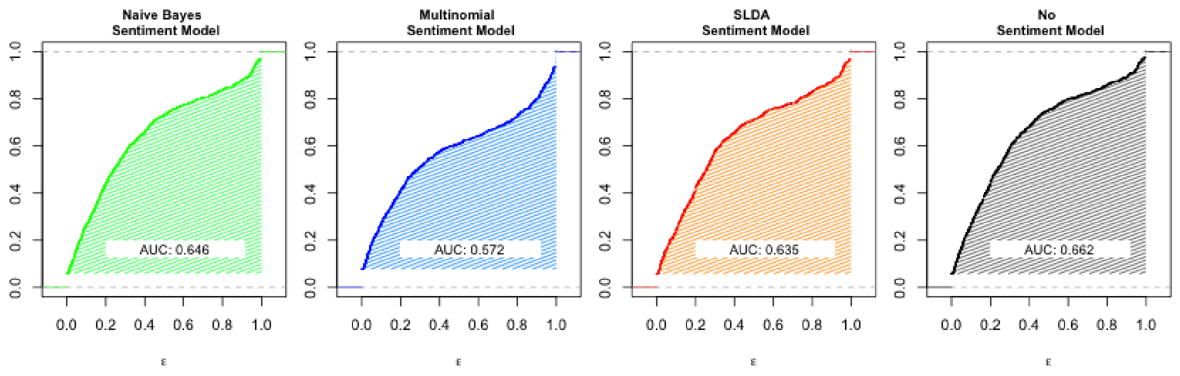


Figure 7.3: AUCs from the Recommendation Error Cumulative Density Functions for different Sentiment models.

As Fig. 7.3 suggests Naives Bayes outperforms the Multinomial and SLDA models since its AUC is greater than the other two. Even more relevant is the fact that using no sentiment looks more accurate than using a sentiment model. The collaborative model without sentiment considers all tips as positive instead of using the sentiment model which extracts the opinion from the reviewer.

However, the fact of using an unbalanced data set, which has more positive experiences than negative, benefits those models that the predicted rates has more positive ratings. As shown in the table below, Naive Bayes, and obviously the no-sentiment model, have bigger proportions of positive ratings in the training data sets and they perform better in test data set which has greater number of positive experiences.

	SLDA	Multinomial	Naive Bayes	Without Sentiment
# positivie ratings	191.809	142.758	205.585	212.947
# negative ratings	21.138	70.189	7.362	0

Table 7.1: Number of positive and negative predicted ratings

In order to mitigate the unbalancing effects described above, we propose to sub-sample the testing data set in two balanced data sets and then compute the performance accuracy. In order to estimate properly the Cumulative Density Function and the AUC statistic, we repeat the sub-sampling process hundreds of times and average the results. The results of this assessments is shown in the figure below for each of the sentiment models.

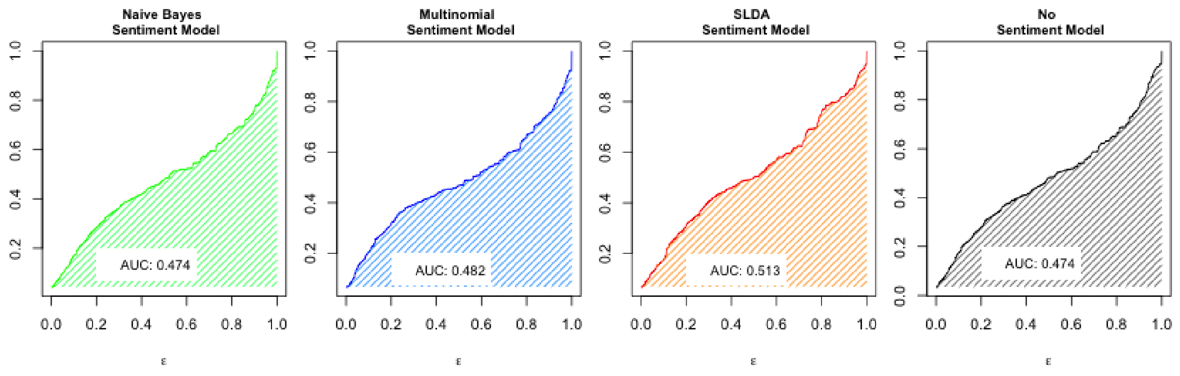


Figure 7.4: AUCs from the Recommendation Error Cumulative Density Functions for different Sentiment models in a balanced data set.

The results of this analysis show that the sentiment models behaves better than the collaborative model without sentiment, except for Naive Bayes which behaves similarly. Furthermore, we realize that the SLDA models, which takes into account the word polysemy, performs also better than others when used in a collaborative recommender. Note that, SLDA classifier also outperforms other models when tested as a standalone classifier, see Sec. 6.3.3.

Another mechanism to evaluate the Recommender System without having to deal with unbalanced data sets is by means of the Receiver Operating Curves, Sec. 5.2.3. These curves relate the True Positive and the False Positive rates of a recommender. As the names suggest, these values are rates and hence normalized to the overall number of cases.

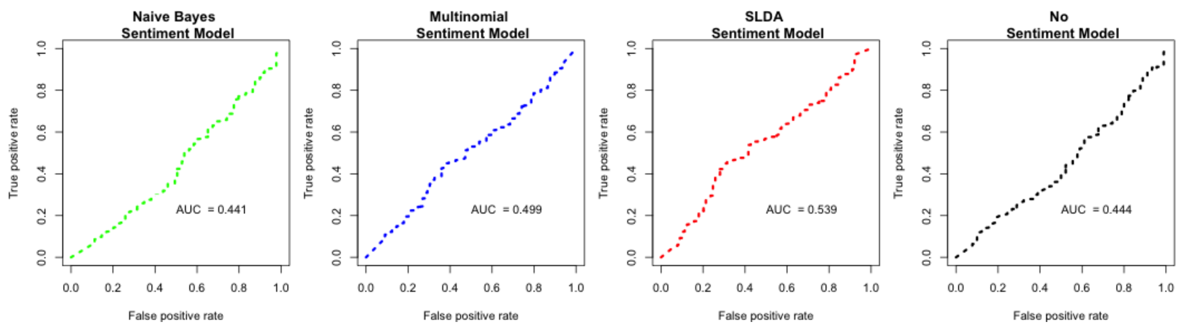


Figure 7.5: AUCs from the Receiver Operating Curves (ROCs) for different Sentiment models

The ROC curves and the associated AUC also point out the aforementioned conclusion that the SLDA sentiment model makes more accurate recommendations in a collaborative framework than other simpler sentiment models. It also supports the use of a sentiment model instead of simply performing collaborative filtering on the past activity of the similar users.

7.2.2 Content-based Model Results

Content-based models cope with the user tastes, which are represented by features extracted from the tips data. We have presented earlier three possibilities to model the user tastes into data features. Note that the content model gathers the a priori knowledge about a restaurant. In other words, it takes into account the similarity between a user and a venue based on their preferences and traits, respectively. Consequently, we assess the content model against the fact that user has gone or not to the recommended venue, more than evaluating the experience on it, since we understand that the goal of the content-based model is to match similar profiles more than taking into account the global opinion about a restaurant.

In what follows, we compare the performance accuracy of these three models by examining the Area Under the Curve of the recommendation error CDF. In this section, the recommendation error is defined as if all tips in the test data set were positive experiences.

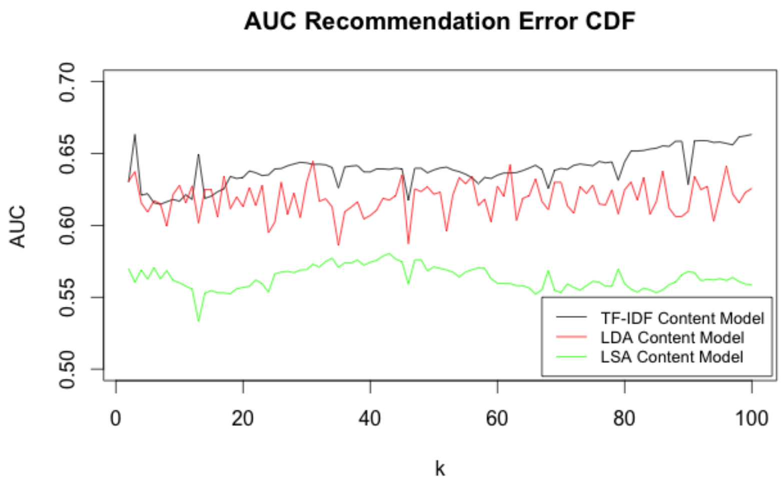


Figure 7.6: AUC of the CDF as a function of the number of features

Fig. 7.6 shows the Performance Accuracy as a function of the feature space size. Note that, in TF-IDF, features are keywords; in LSA, semantic concepts; and in LDA, document topics. Although the complexity that entails LSA and LDA models, the simplicity of TF-IDF content model tends to generate more accurate recommendations for feature space size ranging from 2 to 100 features since it outputs a greater AUC and hence less recommendation error.

Despite the noisiness of these curves, clearly related to the noisiness of text data, there is a tendency that states that the greater the number of features, the more AUC and the better performance accuracy is. However, the feature space size directly impacts the computation time and hence the velocity to make a recommendation. From now on, we consider a k of 100 for all three models, which guarantees a fair trade off between performance accuracy and computation cost. The different CDF and their AUC for each of the content models with k=100 features can be seen in Fig. 7.7.

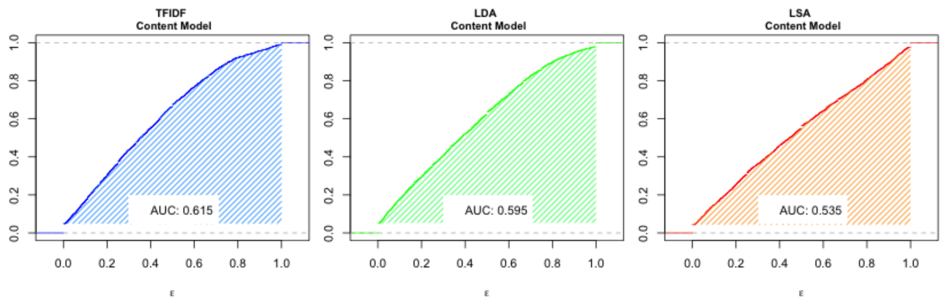


Figure 7.7: AUC from the Cumulative Density Function of the Recommendation Error

One of the aspects that could drive less or more performance accuracy of the contents models is the number of reviews or tips per venue and user. Clearly, the more a user has tipped, the better the content model can cope with his/her tastes and the most effective recommendation can be generated by the engine. Similarly, the same happens with venues when defining the venue profile through the content models. Since the proposed recommendation system uses user tastes and venue profiles to measure the similarity among them, we expect that the better defined these tastes and profiles are, the greater performance accuracy will be achieved.

With the aim of performing this analysis, we first represent ten percentiles of the number of reviews in the train data set per venue and user in the test data set in table 7.2. Here, we see again that more than 30% of users was new and they did not have any tip in the training data set. The same occurs with the number of venues who has not received any tip, more than a 10%.

	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
# tips per venue	0	5	13	21	31	43	59	79	111	183
# tips per user	0	0	0	0	0	1	3	7	21	455

Table 7.2: Deciles on the number of reviews per venue and user

Intuitively, by restricting the recommendations to only those users that have tipped or venues that have been tipped more than a x number of times, the recommendation system will suffer a reduction of coverage because of the scarcity of tips. Nonetheless, the following analysis is only proposed to show the goodness of the system when having abundance of data.

As shown in Fig 7.8, the performance accuracy increases with the number of reviews per venue and user. The graphic plots the AUC for observations of users and venues in the test data set with more than a given number of reviews. Consequently, it says that the more knowledge we have about the venue and the user, the better we can estimate its recommendation. Note also that TF-IDF outperforms the LSA and LDA for any number of tips.

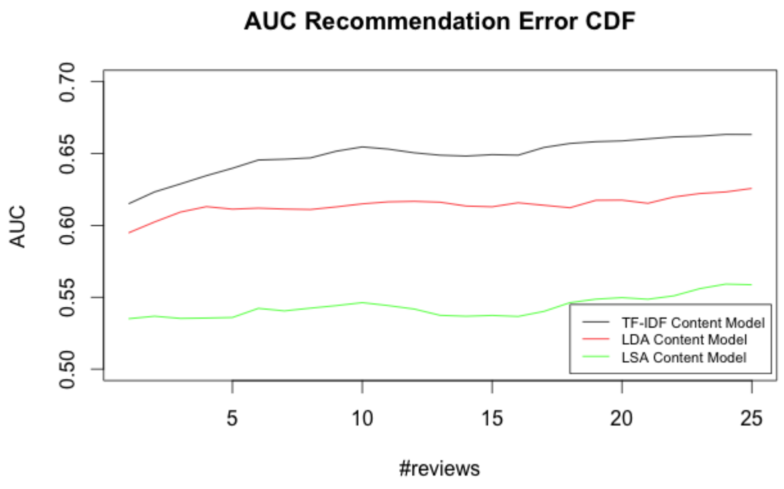


Figure 7.8: Performance accuracy in terms of AUC as function of number of reviews

In general, it can be interpreted that not only the number of the reviews will influence the performance accuracy and coverage of a review-based recommendation system, but other features about the data such as the quality of the reviews, the heterogeneity of topics reviewed, the influence of the reviewer, amongst others. Future work in this area should focus on these items.

7.2.3 Hybrid Linear Combiner Results

The final proposed recommendation approach consist in combining the collaborative and the content-based predicted rates into a linear mixer constituting an Hybrid Recommender System. In this way, we are able to bring together the recommendation based on the similarity between users and items, as well as the neighborhood opinion expressed through the reviews.

Our aim in this section is two fold. Firstly, we pretend to optimize the α parameter defined in equation 4.5 to minimize the recommendation error. Secondly, we show that by using linear combination, the performance accuracy and coverage of the recommender engine is always better than using simply a content-based or collaborative branches.

To globally assess the recommender, we consider again the evaluation model with the testing experiences classified as positive or negative. However, the performance accuracy will be calculated on the real unbalanced data set with the sentiment model (SLDA) that scored the highest in the balanced analysis. Furthermore, TF-IDF is chosen as the content-based model to be included in this hybrid approach.

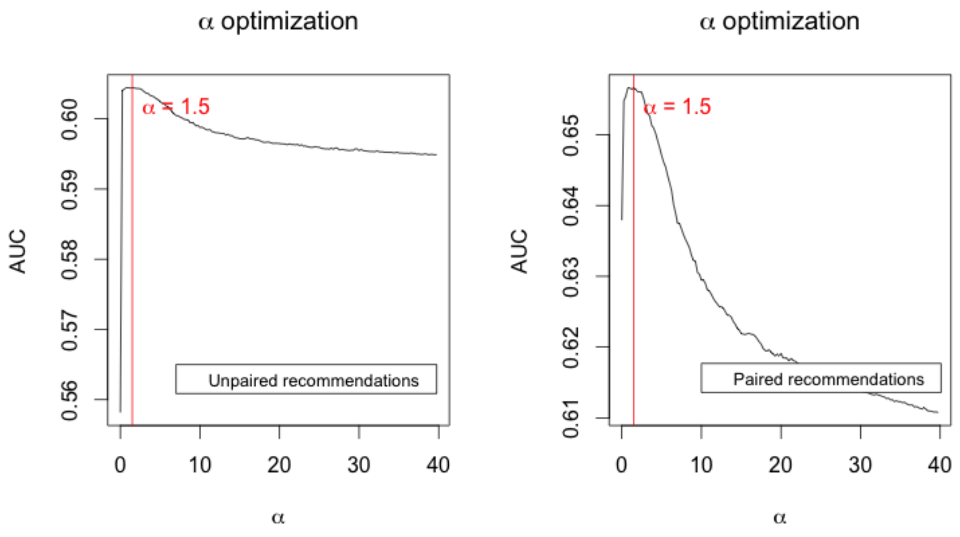


Figure 7.9: Hybrid Linear Combiner Optimization

Fig. 7.9 plots two curves, both representing the Area Under the Curve of an Hybrid Recommender for different values of α ranging from 0 to 20, see Eq. (6.1). The difference between them is that the figure on the right is the AUC calculated for a test data set with pairs of users and venues which the system can provide a predicted rating for both the collaborative and the content-based branches. Whereas the graphic on the left is the AUC for the Foursquare test data set, in which there are pairs of users and venues which the collaborative filtering cannot estimate the output rating, since there are not enough neighbors close to the recommended users to average their ratings.

As the graphic shows, an α equal to 1.5 maximizes AUCs from both scenarios. Note also that $\alpha = 0$ is simply the collaborative recommender, while $\alpha \to \infty$ is simply the content-based recommender.

Next, we plot the empirical cumulative density functions for both situations and the α parameter set to the optimum value, 1.5. Fig. 7.10 shows the performance accuracy for the data set with paired recommendations meaning that all recommendations estimated by a linear combination of the collaborative and the content-based rate. Clearly, the result of applying a linear combiner results in a CDF which has greater AUC as it is noted in table 7.3

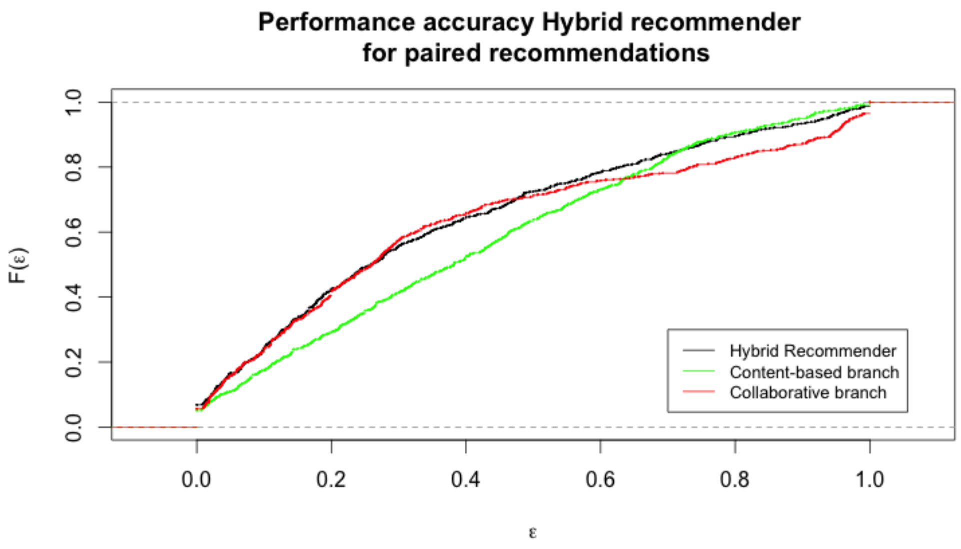


Figure 7.10: Empirical cumulative density functions for paired recommendations.

Fig 7.11 plots the empirical cumulative density function for the real Foursquare data set dominated by unpaired recommendations, what this means is that recommendations do not necessarily have a collaborative prediction for each content-based. In fact, we observe that in our situation for every 5 content-based recommendations, there is only one collaborative recommendation. This has a huge impact into the overall system evaluation because unpaired recommendations weights more than paired recommendations and the content-based branch seems to perform more accurately. As shown in the plot below, the Hybrid curve is now closer to the content-based curve than to the collaborative which is much lower because it cannot output an appropriate recommendation for those users without neighborhood.

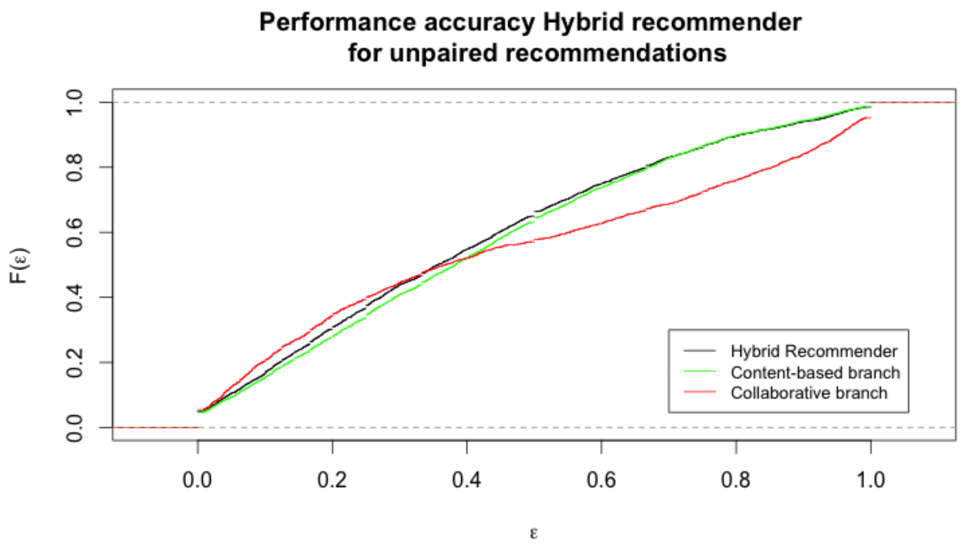


Figure 7.11: Empirical cumulative density functions for unpaired recommendations.

By exposing the results split into two separate data sets (paired and unpaired recommendations), the fact of using a linear model always guarantees greater performance. The Hybrid Recommender basically benefits the overall performance when the collaborative branch can work out a rating and when it cannot. Nonetheless, the biggest contributions happens when the collaborative branch can recommend.

It is relevant to say that restricting the recommender just to those users with a collaborative prediction has a huge impact to the overall recommendation coverage. The next table, 7.3 shows the relationship between the performance accuracy in terms of AUC for differents system configurations and the recommendation coverage in each data set situation mentioned above.

	Hybrid	Content-based	Collaborative	Coverage
# Paired recommendations	0.6566	0.5984	0.6351	8.34%
# Unpaired recommendations	0.6044	0.5922	0.5514	39.81%

Table 7.3: AUC and coverage for different set-ups of the review-based recommender

7.2.4 Other Recommendation Accuracy

In this section, we examine other possible recommendation set-ups that arises from redefining the proposed models.

Most Popular

Recommendation based on the most popular venues in the Foursquare data set consist in recommending the most popular venue independently to the user tastes or venue profile. The concept most popular in our review-based recommendation means recommending the most reviewed venue or the better reviewed venue.

In fact, we propose to evaluate both recommendation paradigms: *the most reviewed venue* and *the better reviewed venue.* The former will simply consist in counting the number of tips and always recommend the most tipped venue in the nearby area. The latter will sum up the sentiment ratings of each close venue and recommended the best rated venue.

Note that this type of recommendation is equivalent to saying that the neighborhood of our collaborative branch are now all users that have previously been in the restaurants nearby. Moreover, *the most reviewed venue* paradigm will be equivalent to this extremist collaborative model without sentiment, while *the better reviewed venue*, will incorporate the sentiment models.

Not least, this type of recommendation is capable of offering greater coverage accuracy since it does not depend on a neighborhood neither the past user activity. In fact, as long as the item, venue in our scenario, has been tipped in the past, it can be recommended.

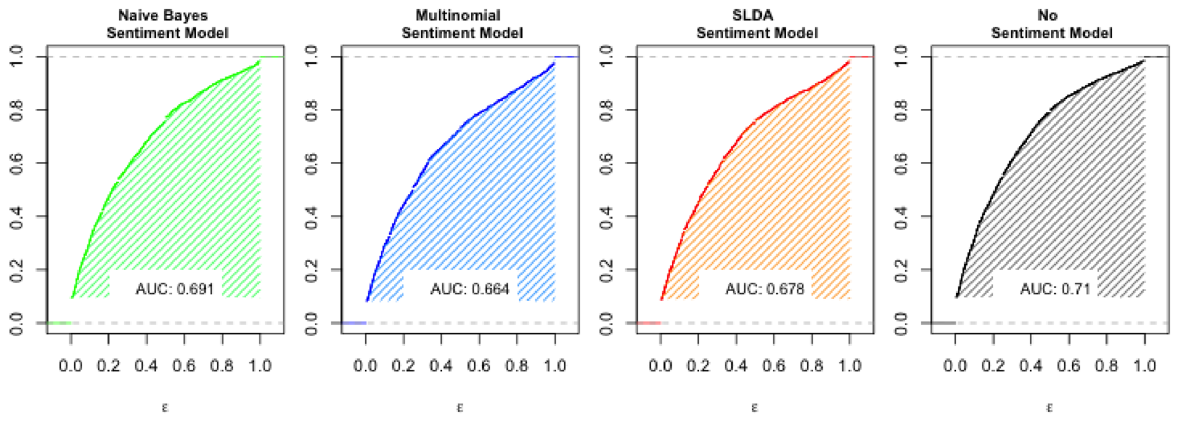


Figure 7.12: Most popular recommendation accuracy for different sentiment models.

Fig. 7.12 corresponds to the recommendation accuracy measured through the area under the curve for the recommendation error density functions when using different Sentiment models, *the better reviewed venue*, or without Sentiment, *the most reviewed venue.* Clearly, the most popular recommender without sentiment outperforms the others in a unbalanced data set as shown in the above picture.

As we have also studied previously, the performance accuracy value in a balanced data set shows again that using sentiment improves the accuracy of the recommender, see Fig. 7.13. Hence, it can be concluded that it is better to recommend *the better reviewed venue* rather than *the most reviewed venue*. Furthermore, the SLDA model outperforms the other proposed sentiment models in this recommendation approach too.

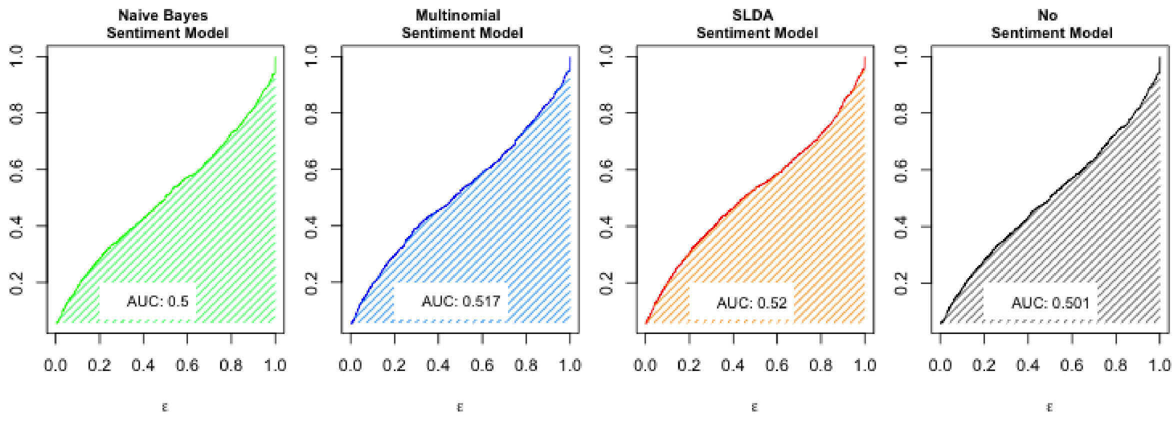


Figure 7.13: Most popular recommendation accuracy for different sentiment models in a balanced data set.

Finally, Fig. 7.14 supports the above idea by plotting the corresponding ROC regions and their AUC values.

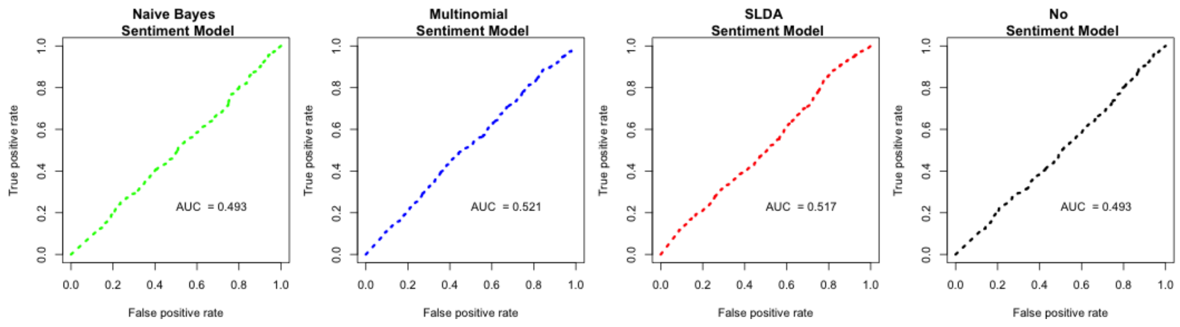


Figure 7.14: Most popular Receiver Operating Curves (ROCs) for different Sentiment models

These results show that the Hybrid recommendation paradigm scores similarly to the most popular. As a result, one could state that there is no benefit at all in performing personalized recommendations and it is fair enough by simply recommending the better reviewed venue. However,we realize that this question cannot be fully answered with the available data, since this data is conditioned to the situation that motivated the tip and hence, the fact of selecting one or another restaurant. Moreover, this data belongs to the Foursquare service who has their particular recommendation engine, which could be based on different goals rather than making a better user experience. As mentioned in Section 1.3, Foursquare engineers found out that the system was skewed to restaurants with larger capacity and longer opening, and hence the data might also be.

In order to evaluated whether a personalized recommendation is better than a popular recommendation, we motivate future work to tackle this problem through online testing in a controlled experiment such as randomized experiments or A/B tests [36].

Friendship popularity

Another resource often used in Social Recommender Systems is the friend relationship that a particular user has in the Social Network.

To conclude this section, we propose to assess a recommender that outputs the better reviewed venue among those reviewed by his/her friends. The result, in terms of AUC, is shown in Fig. 7.15 and there is not a clear evidence that this model could outperform better than the proposed Hybrid recommender, neither the most popular.

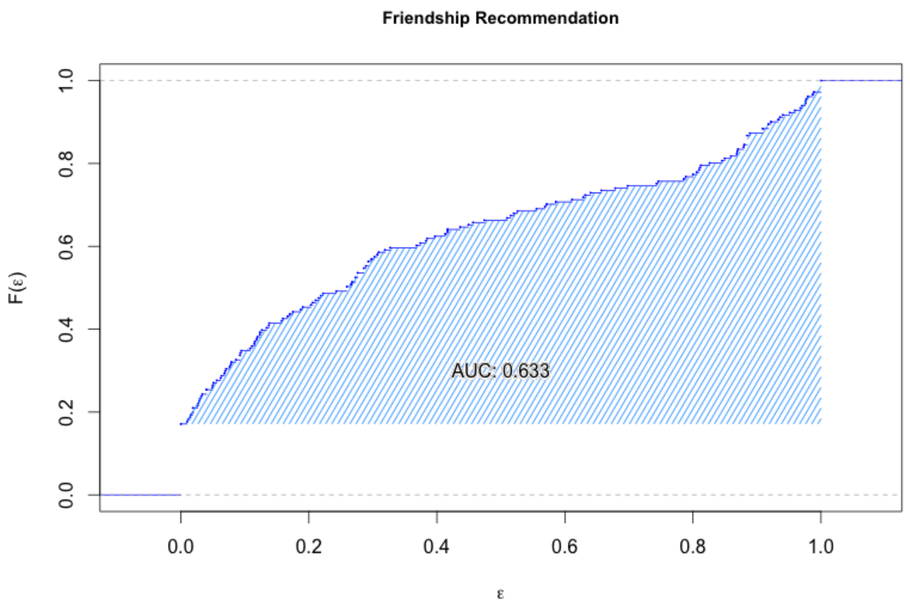


Figure 7.15: Friendship Popularity with SLDA Sentiment model.

One of the main drawbacks of this last engine is the poor recommendation coverage, 5 %, since it strongly relies on the friendships in the Social Network. Mechanisms to increase this coverage could make use of relationships in similar Social Networks, combine friendship and collaborative filtering or other neighbor identification techniques that guarantees greater coverage.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

We motivate the use of **Social Recommender Systems to overcome the information overload tsunami** caused by the increase of actors, links and networking sites that boosted the data volumes on the Internet. These engines are especially good at improving decision outcomes in any data-informed decision-making process.

The design, development and assessment of any SRs entails to follow a **methodology** which we split and present in five different stages: **data retrieving, data cleaning, data modeling, recommendation rates prediction and system evaluation**.

Efficient mechanisms to acquire data from third-party Service Providers are essential in building any data-based algorithm. In this line, we propose a **data retrieval technique that makes use of Quadtree algorithm** to crawl geo-positioned and time-stamped data. We show how Quadtree can help crawling geo-positioned data with different spatial distributions.

User's feedback in the form of text reviews are a powerful source of information to understand users tastes and preferences, while determining and rating item's features. We motivate **the use of text reviews in Social Recommender Systems** and we propose **a novel hybrid recommender purely based on models of text reviews** which takes advantage of the potential of user's feedback.

Text reviews are an unstructured type of noisy data and require intensive pre-processing tasks in order to extract the right information from them. We enumerate and describe state-of-the-art text processing techniques with the final purpose of embedding text reviews into a RS.

With the aim of performing recommendation, **we propose to model reviews in terms of content and sentiment**. This modeling captures the most relevant features from the user's feedback.

Due to the lack of a common evaluation paradigm for Recommender Systems, we detect the necessity to propose a methodology to assess these systems. We present **an evaluation approach that uses the available data from Social Networks** to find out how well the proposed Recommender System works in different setups.

The evaluation results justify **the proposal of combining review content and sentiment** into an hybrid recommendation set-up. Moreover, the test outcomes point out to **the importance of the data retrieval process and the quality of the data set**, by showing that the greatest performance accuracy happens when the system has lot of information. Variations of the classical collaborative-based recommenders, which include user relationships in Social Networks into the neighborhood identification algorithm, does not seem to clearly improve the results of our approach.

Finally, we make clear one of the weak points of the proposed evaluation methodology: **the bias introduced in the data by the recommendation engine of the social networking site**. Future work needs to decouple the evaluation methodology from the available data, enabling to test and compare different recommenders setups.

8.2 Future Work

Although there are many Social Recommender Systems already implemented in several social sites, the community still lacks of a common methodology to assess the performance accuracy of Recommender Systems. Therefore, Recommender Systems are nowadays tested through randomized experiments, such as A/B testing, which achieve more robust reliability and validity on the performance assessment, but it unavoidably requires to put these systems online and perform the tests with real users.

Not only we encourage to test the proposed Social review-based Recommender System through a randomized experiment and obtain even more robust accuracy, but we also stimulate the research community to figure out evaluation methodologies for recommender engines which enable to easily compare them among different setups.

Throughout this thesis, we have also left behind several open issues that we consider that have to be tackled in the future work on this field.

First, the community should investigate new data retrieval mechanisms or variations of Quadtree that exploit the spatial distribution of the geo-positioned data, to reduce times to obtain complete operational data sets. Moreover, real-time applications should also be taken into consideration since the achieved latency times of the current methods cannot be tolerated. Parallelism and other Big Data solutions could also enhance the basic Quadtree algorithm.

Secondly, we need to pay close attention to pre-processing text reviews due to the high levels of noise present is social text data. Novel strategies to normalize text, or to classify between good and bad reviews could notably impact the performance of any review-based Recommender System.

Next, future review models could take into account the sentiment for each feature hided in the text data instead of having an overall sentiment for the whole review. Although, Foursquare reviews, or tips, are short and they usually represent one concept or feature per tip, longer reviews could miss to capture different sentiments associated to each feature.

To conclude, we also deeply encourage to make the best of social artifacts to improve the coverage and accuracy of recommenders. For example, the new user problem could be extensible tackled through user profile matching, which enables to identify and match users in different Social Networks. Similarly, the new user item could also be overcome through the available data in other web platforms (e.g. for venues in Foursquare, yelp.com or tripadvisor.com could enrich the features content). Last but not least, the sparsity of the recommendation matrix in collaborative-based recommenders can be fixed thanks to the social links established in the Social Network.

List of Figures

Appendices

Appendix A

The Tip Models Tools

Tips pre-processing

To pre-process and normalize tips data, we have used the following functions from the *tm* packaged of *R*:

```
library(tm)
# Omitted code
cor <- tm_map(cor, removeWords, stopwords("english"))
cor <- tm_map(cor, tolower)
cor <- tm_map(cor, stripWhitespace)
cor <- tm_map(cor, removeNumbers)
cor <- tm_map(cor, removePunctuation)
```

Tips content models

TF-IDF

```
library(tm)
# Omitted code
tdm   <- TermDocumentMatrix(cor)
tfidf <- weightTfIdf(tdm)
```

LSA

```
library(lsa)
library(tm)
# Omitted code
dtm <- DocumentTermMatrix(cor)
lsa <- lsa(dtm,k=100)
```

LDA

```
library(lda)
# Omitted code
corpus <- lexicalize(cor)
to.keep <- corpus$vocab
result <- lda.collapsed.gibbs.sampler(documents=corpus, K=50,
vocab=to.keep, num.iterations=50, alpha=0.1, eta=0.1)
```

Tips Sentiment models

Naive Bayes

```
library(FSelector)
library(e1071)
# Omitted code
weights<-chi.squared(annTrain~.,data=docsTrain)
subset <- cutoff.k(weights,6 )
f <- as.simple.formula(subset, "annTrain")
model <- naiveBayes(f, data = docsTrain)
```

Bibliography

[1] Hanan Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260, June 1984.

[2] KannaAl Falahi, Nikolaos Mavridis, and Yacine Atif. Social networks and recommender systems: A world of current and future synergies. In Ajith Abraham and Aboul-Ella Hassanien, editors, *Computational Social Networks*, pages 445–465. Springer London, 2012.

[3] Ido Guy and David Carmel. Social recommender systems. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 283–284, New York, NY, USA, 2011. ACM.

[4] Anthony Jameson. More than the sum of its members: Challenges for group recommender systems. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, pages 48–54, New York, NY, USA, 2004. ACM.

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

[6] James Bennett, Stan Lanning, and Netflix Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[7] Xin Liu and Karl Aberer. Soco: A social network aided context-aware recommender system. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 781–802, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[8] Georg Groh and Christian Ehmig. Recommendations in taste related domains: Collaborative filtering vs. social filtering. In *Proceedings of the 2007 International ACM Conference on Supporting Group Work*, GROUP '07, pages 127–136, New York, NY, USA, 2007. ACM.

[9] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. Sorec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM Conference on*

Information and Knowledge Management, CIKM '08, pages 931–940, New York, NY, USA, 2008. ACM.

[10] Samaneh Moghaddam, Mohsen Jamali, Martin Ester, and Jafar Habibi. Feedbacktrust: Using feedback effects in trust-based recommendation systems. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 269–272, New York, NY, USA, 2009. ACM.

[11] Junming Huang, Xue-Qi Cheng, Jiafeng Guo, Hua-Wei Shen, and Kun Yang. Social recommendation with interpersonal influence. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 601–606, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[12] M. Sarwat, J. Levandoski, A. Eldawy, and M. Mokbel. Lars*: An efficient and scalable location-aware recommender system, 2013.

[13] Mike Gartrell, Xinyu Xing, Qin Lv, Aaron Beach, Richard Han, Shivakant Mishra, and Karim Seada. Enhancing group recommendation by incorporating social relationship interactions. In *Proceedings of the 16th ACM International Conference on Supporting Group Work*, GROUP '10, pages 97–106, New York, NY, USA, 2010. ACM.

[14] Mamadou Diaby, Emmanuel Viennet, and Tristan Launay. Toward the next generation of recruitment tools: An online social network-based job recommender system. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 821–828, New York, NY, USA, 2013. ACM.

[15] Yue Lu, Panayiotis Tsaparas, Alexandros Ntoulas, and Livia Polanyi. Exploiting social context for review quality prediction. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 691–700, New York, NY, USA, 2010. ACM.

[16] Zhu Zhang and Balaji Varadarajan. Utility scoring of product reviews. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 51–57, New York, NY, USA, 2006. ACM.

[17] Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 423–430, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[18] Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. Modeling and predicting the helpfulness of online reviews. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 443–452, Dec 2008.

[19] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[20] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990.

[21] Juan Ramos. Using tf-idf to determine word relevance in document queries.

[22] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification, 1998.

[23] Sang bum Kim, Hae-Chang Rim, Dongsuk Yook, and Heui seok Lim. Effective methods for improving naive bayes text classifiers, 2002.

[24] David Blei and Jon McAuliffe. Supervised topic models. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 121–128. MIT Press, Cambridge, MA, 2008.

[25] Matthew A. Russell. *Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites.* O'Reilly Media, 1 edition, 2011.

[26] Matko Boanjak, Eduardo Oliveira, Jos Martins, Eduarda Mendes Rodrigues, and Lus Sarmento. Twitterecho: a distributed focused crawler to support open research with twitter data. In Alain Mille, Fabien L. Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors, *WWW (Companion Volume)*, pages 1233–1240. ACM, 2012.

[27] Wikipedia. Quadtree — Wikipedia, the free encyclopedia, 2014. [Online; accessed 10-January-2014].

[28] Adam Aspin. Delta data management. In *SQL Server 2012 Data Integration Recipes*, pages 619–680. Apress, 2012.

[29] Timothy Baldwin and Marco Lui. Language identification: The long and the short of the matter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 229–237, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[30] Bo Han, Paul Cook, and Timothy Baldwin. Lexical normalization for social media text. *ACM Trans. Intell. Syst. Technol.*, 4(1):5:1–5:27, February 2013.

[31] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012.

[32] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 257–260, New York, NY, USA, 2010. ACM.

[33] Aline Bessa, Alberto H. F. Laender, Adriano Veloso, and Nivio Ziviani. Alleviating the sparsity problem in recommender systems by exploring underlying user communities. In Juliana Freire and Dan Suciu, editors, *AMW*, volume 866 of *CEUR Workshop Proceedings*, pages 35–47. CEUR-WS.org, 2012.

[34] Félix Hernández del Olmo and Elena Gaudioso. Evaluation of recommender systems: A new approach. *Expert Syst. Appl.*, 35(3):790–804, October 2008.

[35] F. Å. Nielsen. Afinn, mar 2011.

[36] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and RandalM. Henne. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*, 18(1):140–181, 2009.