Information Systems 57 (2016) 111–128

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

GeoSRS: A hybrid social recommender system for geolocated data

Joan Capdevila [a,1], Marta Arias [b,2], Argimiro Arratia [b,2,3]

[a] Barcelona Supercomputing Center, Barcelona Tech/Universitat Politècnica de Catalunya, Spain
[b] Computer Science Department, Barcelona Tech/Universitat Politècnica de Catalunya, Spain

A R T I C L E I N F O

Article history:
Received 23 November 2014
Received in revised form
5 October 2015
Accepted 13 October 2015
Available online 25 October 2015

Keywords:
Recommender systems
Text mining
Quadtree
Crawling
Social networks
Location-based social network

A B S T R A C T

We present *GeoSRS*, a hybrid recommender system for a popular location-based social network (LBSN), in which users are able to write short reviews on the places of interest they visit. Using state-of-the-art text mining techniques, our system recommends locations to users using as source the whole set of text reviews in addition to their geographical location. To evaluate our system, we have collected our own data sets by crawling the social network *Foursquare*. To do this efficiently, we propose the use of a parallel version of the *Quadtree* technique, which may be applicable to crawling/exploring other spatially distributed sources. Finally, we study the performance of *GeoSRS* on our collected data set and conclude that by combining sentiment analysis and text modeling, *GeoSRS* generates more accurate recommendations. The performance of the system improves as more reviews are available, which further motivates the use of large-scale crawling techniques such as the *Quadtree*.

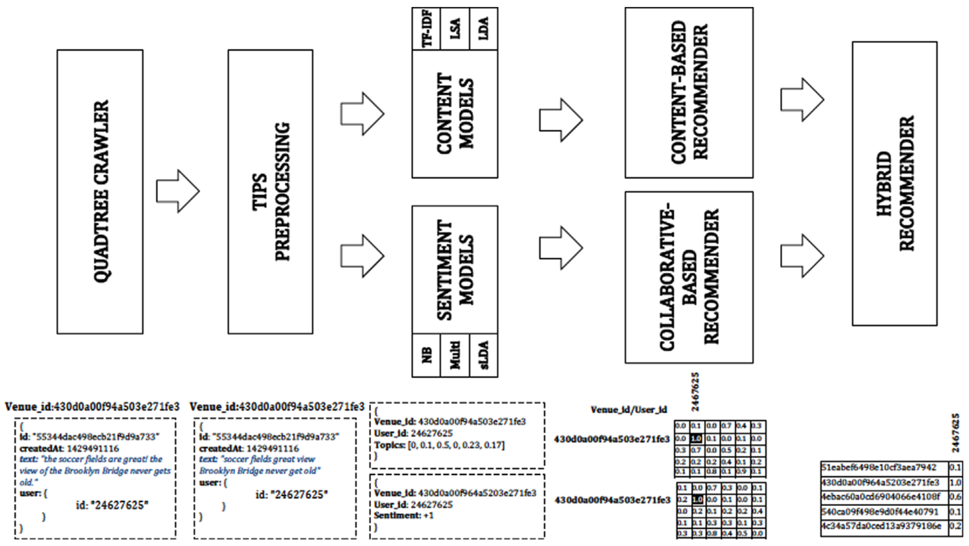
1. Introduction

The proliferation of mobile communication and GPS technologies has allowed users to add geographical identification metadata to various social media, such as photographs, text reviews or video, among many others. Location-based social networks (LBSNs) [54] integrate into a single network user relations (the “social” part) and geo-spatial information (the “location-based” part). By taking into account the physical location of users, LBSNs are bridging the gap between physical world and virtual communities such as *Foursquare*,[4] *Facebook*[5] or *Twitter*.[6]

The extensive use of these social networking sites has made them invaluable sources of information. However, the sheer volume of data flowing through these sites, even for a single user, has made it increasingly difficult for humans to track all this information. Therefore, most social networking sites implement some sort of Social Recommendation System (SRS) [16]: for example, Twitter suggests who to follow, Facebook filters and prioritizes posts in users' walls and Foursquare

E-mail addresses: jc@ac.upc.edu (J. Capdevila), marias@cs.upc.edu (M. Arias), argimiro@lsi.upc.edu (A. Arratia).
[1] Supported by Obra Social “la Caixa”.
[2] Supported by MICINN project TIN2011-27479-C04-03 (BASMATI), MINECO project TIN2014-57226-P (APCOM) and Gen. Cat. project SGR2014-890 (MACDA).
[3] Additional support by MEC project MTM2012-36917-C03-03 (SINGACOM).
[4] https://foursquare.com/
[5] https://facebook.com/
[6] https://twitter.com/

http://dx.doi.org/10.1016/j.is.2015.10.003
0306-4379/© 2015 Elsevier Ltd. All rights reserved.



```
Venue_id:430d0a00f94a503e271fe3
{
id: "55344dac498ecb21f9d9a733"
createdAt: 1429491116
text: "the soccer fields are great! the view of the Brooklyn Bridge never gets old."
user: {
        id: "24627625"
    }
}
```

```
Venue_id:430d0a00f94a503e271fe3
{
id: "55344dac498ecb21f9d9a733"
createdAt: 1429491116
text: "soccer fields great view Brooklyn Bridge never get old"
user: {
        id: "24627625"
    }
}
```

```
{
Venue_id: 430d0a00f94a503e271fe3
User_id: 24627625
Topics: [0, 0.1, 0.5, 0, 0.23, 0.17]
}
```

```
{
Venue_id: 430d0a00f964a5203e271fe3
User_id: 24627625
Sentiment: +1
}
```

Venue_id/User_id

2467625

430d0a00f94a503e271fe3

0.0	0.1	0.0	0.7	0.4	0.3
0.0	1.0	0.1	0.0	0.1	0.0
0.3	0.7	0.0	0.5	0.2	0.1
0.2	0.2	0.2	0.4	0.1	0.2
0.1	0.1	0.8	0.1	0.9	0.1

430d0a00f94a503e271fe3

0.1	0.0	0.7	0.3	0.0	0.1
0.2	1.0	0.0	0.1	0.0	0.1
0.0	0.2	0.1	0.2	0.2	0.4
0.1	0.1	0.3	0.3	0.1	0.3
0.3	0.3	0.8	0.4	0.5	0.0

	2467625
51eabef6498e10cf3aea7942	0.1
430d0a00f964a5203e271fe3	1.0
4ebac60a0cd6904066e4108f	0.6
540ca09f498e9d0f44e40791	0.1
4c34a57da0ced13a9379186e	0.2

Fig. 1. Fully integrated location-based Information System. It includes modules for crawling, preprocessing crawled data, modeling and recommendation. The bottom part of the figure shows examples of data chunks that each module produces.

recommends locations where to go. When the social content is geotagged, it becomes strictly necessary to consider user and item localization in the recommendation paradigm.

Location-based recommendation constitutes a unique application in LBSNs and it substantially differs from traditional recommender systems in the fact that the latter does not take into account the spatial properties of users and items [34]. Moreover, location-based recommendation on top of LBSNs might also benefit from the interaction between the three layers composing a LBSN, namely the user, the location and the content layer [47].

In this paper, we propose a fully integrated system for *information retrieval* of geolocated data and *end-to-end location-based recommender*, suitable for the popular social network *Foursquare*. The reader should note, however, that our methods are applicable to any other social network that contains geolocated time-referenced reviews and hence, in our presentation we abstract from the fact that we are using this particular site.

We believe that fully operative recommender systems on top of LBSNs require end-to-end designs, capable of performing data retrieval from social networks, cleaning the noisy and duplicated data, extracting relevant features and, not least, performing recommendation.

Our proposed location-based information system is summarized in Fig. 1. It retrieves the short reviews together with their geolocated venues and reviewers identification as its basis for recommendations. In Foursquare, users are able to *check-in* to places of interest (venues), write short reviews (tips) for the venues where they checked into, and share this information with users within their social network. For the task of crawling venues, users and tips, we had to use Foursquare's API, an interface which imposes restrictions to the amount of information one can query and the amount of requests one can make. Therefore, it was imperative to devise a crawling mechanism that would make optimal use of the queries available to us. To achieve this task, we have designed a parallel version of the *Quadtree* algorithm [42], which is very well suited for crawling venues that are spatially distributed, while at the same time gaining considerable throughput. We have found that crawling all venues from large urban areas such as Mexico D.F. or New York in reasonable time was possible by the proposed Quadtree algorithm. We consider the application of the parallel Quadtree algorithm to this problem an important contribution of our paper and we believe that problems that require sweeping spatial devices (sensors) could also benefit from it.

To make recommendations our system makes extensive use of user's reviews (tips). In order to extract meaningful information from these free-form reviews, *GeoSRS* relies upon many state-of-the-art techniques for text mining and sentiment analysis, which are evaluated in terms of recommendation accuracy and the ones that outperform are selected to be used in *GeoSRS*. Another relevant contribution from our paper is the increase of accuracy when mixing the review's sentiment and content into a simple but rather effective weighted hybrid recommender setup [9]. Sentiment refers to the global opinion that is reflected in the review (positive, negative or neutral) while content indicates the topics that the review addresses. This enforces the idea that pure review-based choices are not merely based on the opinion reflected on a short review (The service was too slow), but also on the content relevant to the user (This is a kinda working place rather than a coffee shop).

To evaluate our system, we have collected our own data set of restaurants and tips from the area of Manhattan in New York City. We have chosen Manhattan due to the high density of venues and the number of active users, to validate both the scalability of the Quadtree crawler and the effectiveness and coverage of the recommender system. Recommender system is evaluated in terms of retrieval accuracy (performance) measures rather than statistical accuracy measures since we do not intend to predict individual venue ratings but relative order among them. Moreover, we propose an evaluation method in which we divide historically the tips data set in training and testing. Test tips are taken as ground truth to comparatively assess the recommendation. Last but not least, the simple weighted hybrid recommender setup employed in *GeoSRS* is compared against other state-of-the-art configurations such as meta-level and cascade models.

To summarize, this paper proposes for a hybrid recommender system for a location-based social network which is uniquely built upon text reviews. Our main contributions are:

1. Using a parallel version of the *Quadtree* technique as the basic strategy for crawling spatially distributed data.
2. Using sentiment analysis on text reviews to generate the source for collaborative-filtering.
3. Using the aggregated reviews by user and venue to generate the profiling information for content-based recommendation.
4. Using a simple but powerful hybridization technique to improve recommendation performance.
5. Putting it all together into a working information system.
6. Evaluating and comparing *GeoSRS* against other state-of-the-art hybrid systems in terms of IR figures.

The rest of this paper is organized as follows. Section 2 presents an overview of related work in social recommendation on top of LBSNs, sentiment and content analysis systems and general hybrid recommendation techniques. Section 3 proposes a parallel efficient technique to retrieve spatially distributed data sources. Our system *GeoSRS* is described in Section 4. Section 5 brings together the geolocated reviews data set from Foursquare with the working *GeoSRS* system, to assess different possible set-ups using an offline evaluation methodology. Last but not least, Section 6 derives several conclusions from our work and includes directions for future work.

2. Related work

The recommender system proposed in this paper falls within the class of *location-based social recommender systems*, using *sentiment and content analysis of text* combined withcollaborative filtering techniques leading to a *hybrid recommender system*. In order to place our system in context of the known literature we briefly review each of these research areas relevant to our work.

Social recommender systems (*SRS*): SRS have arisen as an application of recommender systems to social networks, although they have been accepted lately as a separate discipline in itself [16]. Nonetheless, researchers have been proposing novel recommender set-ups based on social content for the last five to ten years [12,16,21,25].

Classical recommendation paradigms such as content-based recommenders have been enhanced by adding friendship information into the matrix factorization objective function [25], configuring the so-called Social Content-based Recommender (SoCo). Collaborative-based filtering has also been redefined by including social information to improve the user neighborhood identification and deal with data sparsity [15,29]. According to [15], social filtering outperforms collaborative-based filtering. Trust-based recommenders refine the notion of social filtering even further by defining several notions of propagation of trust or reputation through the network of users. Examples of this are *FeedbackTrust* [33] and *TrustWalker* [20] although many others exist [30,3,37]. *FeedbackTrust* improves user-based recommendations by enhancing user's similarities with trust-based similarities; *TrustWalker* makes item-based recommendations by combining the result of repeated random-walks over the network of connected users.

Others have proposed mechanisms to include interpersonal influences in traditional recommender systems arguing that the interpersonal influence plays a critical role in this scenario [18]. The term *social regularization* has been coined to refer to the use of a regularization based on social content. In the context of *group recommendation*, which consists in making recommendations to a group of individuals based on their interests, the work of Gartrell et al. [13] highlights the benefits of incorporating social structure into the recommender.

Several practical SRS can be found in the literature applied to distinct recommendation situations. For example, Diaby et al. [11] describe an online social network-based job recommender system for recruiters. Tu et al. [49] present an online dating recommending system that learns the user preferences from a Latent Dirichlet Allocation (LDA) model. Xia et al. [50] propose a system that takes into account social ties in the area of scientific articles to recommend scholarly papers to users.

Location-based social recommender systems: Recommender systems can be further improved by exploiting geolocated data to take into account the spatial dimension [43]; such systems are generally referred to as *location-based social recommender systems*. Examples in this line of research are Berjani and Strufe [4], Yang et al. [51], Ye et al. [52], and Zheng et al. [55]; for an introduction to LBSN recommender systems and updates on the state-of-the-art see the tutorial [54] or the recently published textbook by Symeonidis et al. [48]. Compared to Yang et al. [51], we are using the same data source, and due to similarity of contributions we hereby present our contributions with theirs. Yang et al. [51] proposed a location-based recommendation system that uses geolocated tips from Foursquare jointly with check-in information to improve the recommendation performance. Our hybrid recommender system *GeoSRS* differs from this work in various aspects. While Yang et al. [51] use only sentiment information from tips, we also extract the topics structure from them to model user and venue profiles. Additionally, authors based the recommendation scheme on a probabilistic factorization of the user-item matrix that considers social influence, whereas we found out that by using social influence on the collaborative-based branch of our hybrid system performed poorly. Moreover, we evaluate the system by assessing the relative order of near venues compared with the actual attendance rather than calculating the statistical accuracy of the estimated ratings as they do. In our opinion, our evaluation methodology is closer to the real recommendation behavior of a location-based system. Therefore we believe that our work complements their work on other important aspects of location-based social recommender systems that have not been considered before.

Sentiment and text analysis: In this work, we propose the use of some of the already mentioned and more of the state-of-the-art text mining techniques to build a purely review-based recommender system. Several text modeling techniques are assessed under our recommendation scheme, such as Latent Dirichlet Allocation (LDA) [6], Latent Semantic Analysis (LSA) [10], or TF–IDF [46,41]. Sentiment analysis is also included in these models by using trained traditional classifiers such as Naive Bayes [31], Multinomial Logistic Regression [7] or supervised LDA classifiers [5].

There are many other recommender systems that leverage the information found in free-form text. In fact, previous work show that considering text sources improves standard collaborative filtering techniques. As an example, Aciar et al. [1] build a recommender system for consumer products based on product reviews, or Jakob et al. [19] improve movie recommendations based on movie reviews. Reschke et al. [40] propose a recommendation dialog system built upon narrow questions from reviews, which slightly differs from the recommendation problem definition. In contrast to all these systems, our recommender system bases the whole recommendation on the reviews text data which is modeled in a flexible and general procedure rather than building a text ontology or extracting multiple aspects [45,35].

Social context has been used to improve results in problems other than recommendation. An example that is directly relevant to our work here is the problem of text analysis and in particular of *review quality detection* [28]. Most previous work in this field treat each review as a stand-alone text document, extracting features from the text and learning a function based on the features [53,23,26]. Naturally, taking into account additional information in the form of social relations between authors of reviews should help improve the predictions.

Hybrid social recommenders: A good deal of researchers have been combining multiple recommendation techniques to boost the performance of the so-called hybrid systems [8]. Standalone collaborative or content-based recommendation systems suffer from several shortcomings which can be overcome by coupling their individual ratings. In this work, we show that these deficiencies of individual recommenders happen as well in our problem scenario and highlight the benefits of hybridization. Among a broad range of hybridization techniques [9], linear weighted combination seems to be one of the most simple but rather effective mechanisms, and it is the one we implement here. For example, Mobasher et al. [32] mix two components, a collaborative and a content-based branch, with a linear weighting scheme to perform movie recommendation. For the sake of comparison, we further develop two other hybrid mechanisms named the *cascade* and the *meta-level* hybrid, which Burke [9] found out to work well when combining two components of differing strength. A cascade hybrid called *Entree* was developed by Burke [8], who combined a knowledge-based and collaborative recommender in a hierarchical manner based on the strength of their recommendation. Finally, a meta-level hybrid that uses content-based recommendation to identify the collaborative neighbors can be found in Pazzani [38].

3. Data retrieval process

3.1. Social networks as sources of open data

Social networks (Facebook, Twitter, Foursquare, LinkedIn, Instagram, etc.) act as consolidated data warehouses unifying distinct users' social activities into common data schemes, which are often mined by in-house analytic systems or accessible to application developer communities. These platforms implement data security and privacy policies, accepted beforehand by the user, which rule the accessibility to the users' social data.

We refer to social open data as the content (posts, tweets, tips, publications, photos, etc.) which is visible or public to anyone. In the last decade, open social data became essential in many research fields ranging from recommender systems to urban design and planning [39]. Among all types of social open data, geolocated data cover a broad range of media types that include geographical coordinates. This is the case of location-based social networks which link some media (tweets, tips, etc.) to the user location gathered through the smart-phone GPS system. It also considers modern urban cities containing thousands of physical sensors spread over large geographical areas, from which their sensed information (e.g. air-pollution, traffic, light-level) is linked to a geographical coordinate.

Geolocated data, generated either by users acting as social sensors or real physical sensors, can be consulted from third-party data providers (social networks, open data portals, etc.) by means of geospatial queries. Typically, these entities implement control mechanisms to avoid server traffic overload, which hamper the retrieval of the totality of the data at once. Hence, any retrieval process for geolocated data has to carefully take into account these traffic limitations.

3.2. Data retrieval from social networks

Most of social networks give access to their data via Representational State Transfer (REST) Application Programming Interfaces (API). This protocol enables an easy but effective interaction with the social network data contents. However, these web services typically limit the rate of requests per registered application and the response data volumes in order to avoid incoming traffic overload.

For example, the geospatial query to obtain all the venues or places of a given geographical area from the Foursquare social network is constrained to return 50 venues at most. This necessarily implies that this area has to be divided into smaller sub-areas with less than 50 venues each to effectively retrieve all venues. Moreover, the Foursquare platform also limits up to 5000 requests per hour and per registered application, forcing the retrieval process to prioritize optimal requests. Here, we propose and motivate the use of *Quadtree* structures and the Quadtree construction algorithm, to effectively and efficiently retrieve all geolocated content from a given region generated in location-based social networks exhibiting the above-mentioned constraints.

3.3. The Quadtree algorithm

A Quadtree is a data structure based on the principle of recursive decomposition of space. Quadtrees are used as hierarchical data representations in the domains of computer vision, image processing, pattern recognition and geographic information systems. The interest in this data structure stems from the fact that it is designed to focus resources on the areas where the information is of greatest density. For a comprehensive survey on quadtrees and related hierarchical data structures see Samet [42].

The general Quadtree construction scheme is presented in Algorithm 1, and it works as follows. The algorithm iteratively divides each region or cell into four subregions or subcells when the maximum capacity per cell, denoted N_{max}, is reached. Given two coordinates (the southern-west and northern-east limits) defining the geographical bounding box, the algorithm defines and queues a quadcell object which contains its geographical limits.

Algorithm 1. Quadtree algorithm.

```
quad ← Quadcell(NElim, SWlim);
queue ← List(quad);
while Length(queue) > 0 do
    q ← Pop(queue;          //  Obtains quad from the queue
    if CheckQuad(q) then
        aux ← SplitQuad(q);
        Extend(queue, aux);      //  Push quad into the queue
    end
end
```

For each quadcell in the queue, the algorithm queries the social network REST API, and compares the cardinality of the response against the maximum number of sensors per cell, N_{max}. This task is performed by the *CheckQuad* function described in Algorithm 2.

Algorithm 2. Check Quadcell function.

```
Function CheckQuad(quad: Quadcell) : boolean
    sensors ← APIrequest(quad.NElim, quad.SWlim);
    if Length(sensors) < $N_{max}$ then
        SaveSensors(sensors);
        return False;        //  Do not split quad
    else
        return True        //  Splitquad
    end
end
```

In case the response size exceeds N_{max}, the quadcell is divided into four subregions or children, whose geographical limits are computed from its parents bounds, and these are also added into the pending queue. This task is performed by the *SplitQuad* function described in Algorithm 3. Leaves or quadcells whose API query returned fewer sensors than the maximum allowed, are stored into disk and the quadcells removed from the queue.

Algorithm 3. Split Quadcell function.

```
Function SplitQuad(quad: Quadcell) : List
        //  Computes the NElim and SWlim for each child
    NElims, SElims, SWlims, NWlims ← ChildrenLim(quad.NElim, quad.SWlim);

        //  Creates Listof children
    children ← List(Quadcell(NElims.NE, NElims.SW),
    Quadcell(SElims.NE, SElims.SW),
    Quadcell(SWlims.NE, SWlims.SW),
    Quadcell(NWlims.NE, NWlims.SW);
    return children;
end
```

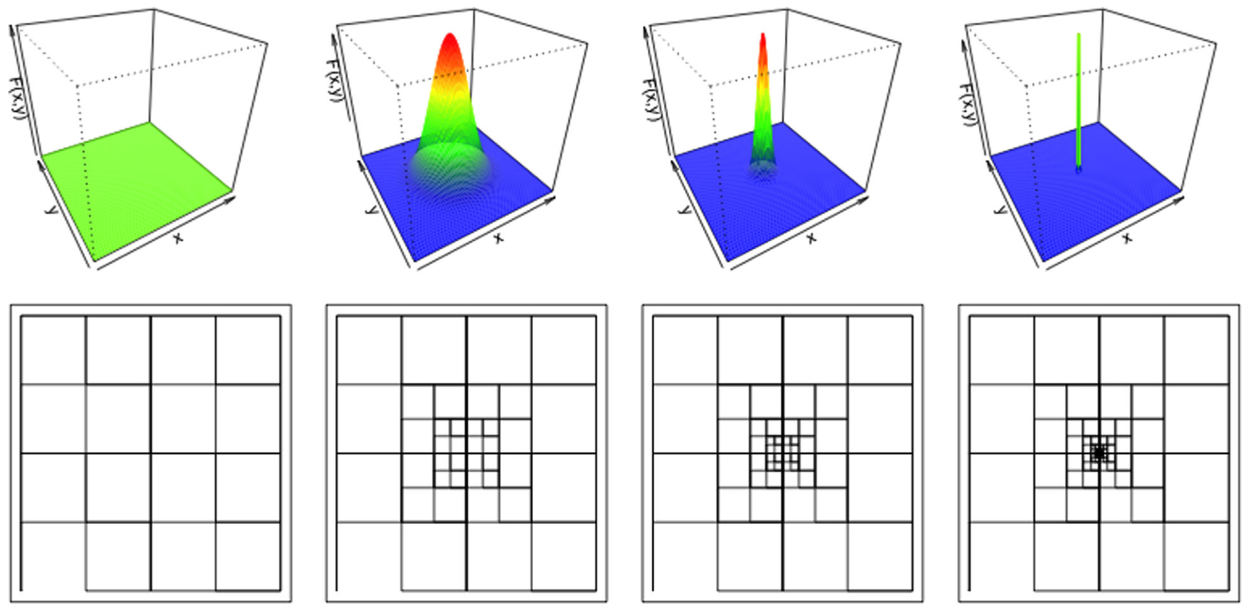


Fig. 2. At the top, spatial distribution of geolocated data uniform and normally distributed with standard deviation $\sigma = 100, 10, 0.25$. Corresponding quadtree structures at the bottom.

In order to interpret the results of the Quadtree algorithm for sensors spread over a bounded region, we simulate the Quadtree execution for different spatial distributions, all containing equal number of sensors (top of Fig. 2). In other words, all distributions integrate to N, the number of sensors in the region.

The resulting Quadtree structures (bottom of Fig. 2) clearly show the dependency between the peakiness of the distribution and the number of quadcells. Given that each quadcell implies at least one HTTP query to the REST API, peaky distributions of geospatial data result in less efficient quadtree executions compared to flatter and more uniform distributions.

The Quadtree algorithm overcomes the API limitations by first partitioning the geographical space into appropriate subregions and second by keeping the HTTP request rate bounded. We have not included the appropriate instructions for the control process over the request rate to keep the description in Algorithm 1 simple. However, the implementation of the Quadtree algorithm should keep control over the time between consecutive requests or the overall number of requests per hour, and sleep the program when necessary.

3.4. Scaling-up the Quadtree algorithm

Efficiency and scalability are key with today's volume of data available. Here, we propose a parallel version of the Quadtree algorithm. First, we state the main scalability drawbacks of the proposed algorithm and then revisit the algorithm to propose a new parallel Quadtree algorithm.

We assume that the maximum number of request per hour and per registered application is R_{max}. This limit is imposed by the social network administrator. When the number of sensors, N, is large R_{max} becomes the limiting factor of the algorithm performance.

Our approach uses parallelization of the algorithm into K sub-processes, each using a different registered application key and hence, enabling greater request rates (with K sub-processes the limit becomes KR_{max}). These subprocesses could either run at the same machine or into different machines with distinct public IP addresses, depending on social network directives.

One classical way to tackle this multiprocess problem is by means of the producer–consumer paradigm. The producer is in charge of querying the social network API, generating and queuing the quadcells into a processing queue. Then, the consumer takes each queued quadcell and stores it into the disk. By splitting the two most time consuming sub-processes (querying the API and storing to disk), we experience considerable gains into the parallelization of the Quadtree algorithm.

The producer–consumer scheme for the parallelization of Quadtree, described in Algorithms 4 and 5, works as follows. The producer threads (Algorithm 4) produce quadcells (*prodQuad*) by querying the Social Network API about the quadcells stored in the pending queue (*PendQueue*), checks whether the cardinality of the responses exceed the maximum number of sensors per cell (N_{max}) and splits the quadcells if they exceed. The split quadcells whose number of sensors exceeds N_{max} are also pushed into the pending queue for the next producer. Then, the producer stores the leaves or quadcells that have less sensors than N_{max} into the processing queue (*ProcQueue*), and notifies the condition variable (*Cond*) to release the underlying lock. This notification awakes the consumers threads (Algorithm 5) waiting in the condition variable, which consume quadcells (*consQuad*) from the processing queue. Consuming means pulling the quadcell from the queue and storing the sensors' values into disk. When the consumer thread completes its task, it waits in the condition variable for the next release triggered by the producers.

Algorithm 4. Producer Quadtree algorithm.

```
Function Producer (PendQueue: Queue, ProcQueue: Queue,Cond: Condition,API: APIhand) : void
  while countQuads(PendQueue) > 0 do
    with(Cond)
      q = prodQuad(PendQueue, API);
      putQuad(ProcQueue, Q);
      notify(Cond);
  end
  with Cond ::
    notifyAll(Cond)
end
```

Algorithm 5. Consumer Quadtree algorithm.

```
Function Consumer (PendQueue: Queue, ProcQueue: Queue, Cond: Condition) : void
  while countQuads(PendQueue) > 0 do
    with(Cond);
    Wait(Cond);
    if countQuads(ProcQueue) > 0 then
      consQuad(ProcQueue)
    end
  end
end
```

3.5. Case study: the Foursquare platform

The Foursquare platform provides a REST API to interact with its components as well as to access its open social data from registered applications. Querying the platform on the geolocated venues data requires satisfying the response size constraints (50 venues per request) and rate limitations (5000 request per hour and application). The parallel Quadtree algorithm enables an effective and efficient data retrieval process by parallelizing API requests into K_p subprocesses and storing to disk into K_c subprocesses. We found out that using $K_p=3$ producers and $K_c=10$ consumers exhibits a proper performance, although the optimization of these parameters is beyond the scope of this paper.

Fig. 3 shows the Quadtree structures for three large urban areas: Athens, Manhattan and Mexico City. As we could expect, the Quadtree grows deeper in the city downtown and business and commercial subareas, while being shallower in residential areas, parks and city surroundings. Table 1 presents some other features about the Quadtrees built in each of these urban areas.

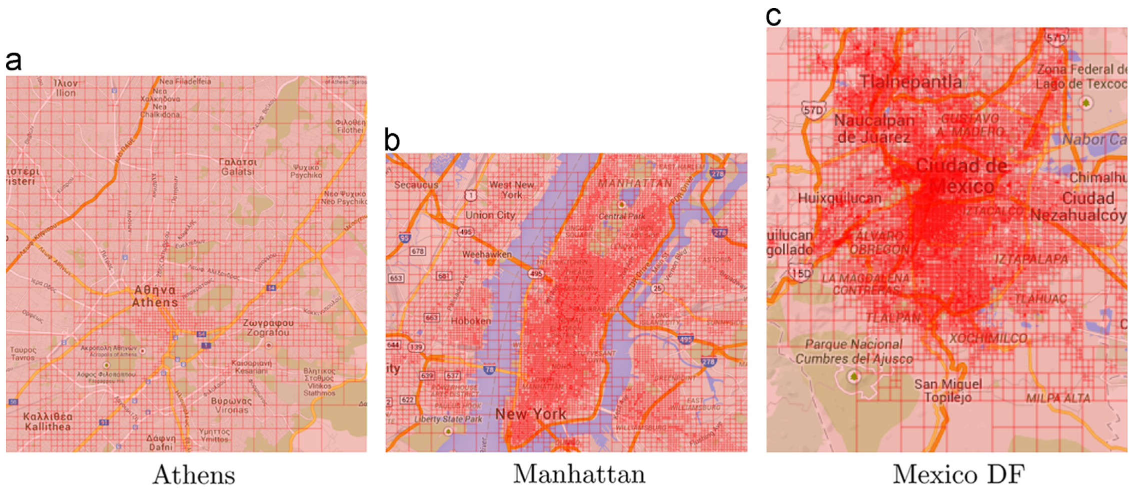
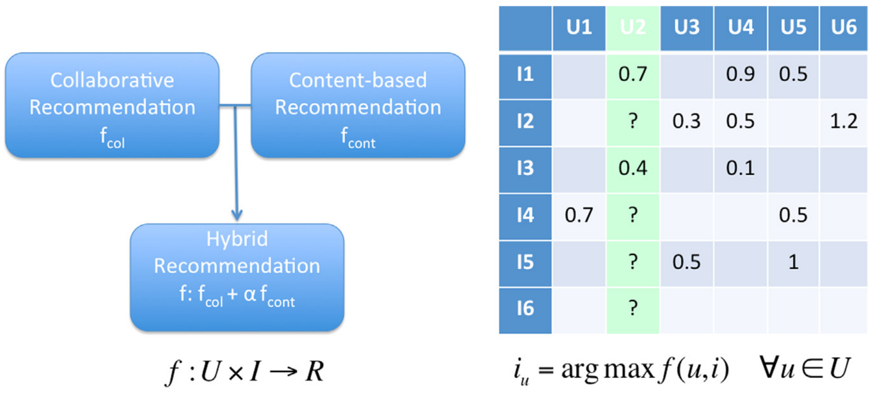


Fig. 3. Parallel Quadtree structures. (a) Athens, (b) Manhattan, and (c) Mexico DF.

Table 1
Urban Quadtrees with $K_p = 3$ producers and $K_c = 10$ consumers.

Features	*Athens*	*New York*	*Mexico D.F.*
NE lim	38.03, 23.79	40.80, −73.91	19.59, −98.94
SW lim	37.95, 23.69	40.70, −74.07	19.13, −99.36
Venues	48.215	297.924	511.096
Quadcells	2.997	18.682	32.270
Quadcell leaves	2.248	14.011	24.202
Quadtree time (s)	4.432	27.980	32.015


Collaborative Recommendation f_{col}
Content-based Recommendation f_{cont}
Hybrid Recommendation f: $f_{col} + \alpha f_{cont}$
$f: U \times I \rightarrow R$
$i_u = \arg\max f(u,i) \quad \forall u \in U$

	U1	U2	U3	U4	U5	U6
I1		0.7		0.9	0.5	
I2		?	0.3	0.5		1.2
I3		0.4		0.1		
I4	0.7	?			0.5	
I5		?	0.5		1	
I6		?				

Fig. 4. A graphical overview of the hybrid recommender system and its recommendation matrix.

4. *GeoSRS* system description

4.1. Overview

GeoSRS uses a rating-based scheme to perform item recommendation. This means that given a utility function $f: U \times I \rightarrow R$, a rate or measure of importance is estimated from the set of all users, U, and the set of all items, I. The recommendation problem can be seen as the following optimization problem:

$$i_u = \arg\max_{i \in I} f(u,i) \quad \forall u \in U \tag{1}$$

in which, for each user u, the goal is to determine the item, i, that maximizes the utility function among all available items. The item, i_u, that satisfies this maximization is recommended to the user u. However, the utility function is usually not defined for the whole space $U \times I$, and consequently, the recommender needs to estimate the missing ratings to perform an accurate recommendation, as shown in the recommendation matrix from Fig. 4.

GeoSRS is a hybrid social recommender system that combines a collaborative and a content-based subsystems to define the utility function which estimates the ratings for the unseen items. In particular, the utility function is defined from the user-item affinity as well as the community opinion by combining them into a hybrid system, given by the following linear combination:

$$f = f_{col} + \alpha f_{cont}$$

where α is a positive real, and while the content-based branch, f_{cont}, mainly uses the inferred features of the item and the profile of the user, the collaborative branch, f_{col}, takes advantage of the “wisdom of the crowd”.

4.2. Content-based branch

The content-based recommendation branch brings into the system the user preferences and item features with the purpose of making recommendations according to the user profile. Following the introduced nomenclature, the utility function for the content-based branch predicts the rate for both seen and unseen items. The rate, R, is predicted from the dot product of the users' profile vectors, w_u, and items' feature vectors, w_i. Thus, the utility function can be redefined as

$$f_{cont}: w_u \times w_i \rightarrow R \tag{2}$$

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

4.2.1. 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 [2]. Our proposed approach for the content-based branch 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

$$f_{cont} = \cos(w_u, w_i) = \frac{\overrightarrow{w_u}\ \overrightarrow{w_i}}{\| \overrightarrow{w_u} \| \| \overrightarrow{w_i} \|} \tag{3}$$

where w_u and w_i are respectively the profile vectors for the users and items defined from the reviews content models described below.

4.2.2. Review content models

Text reviews have been preprocessed using standard techniques, in particular we have removed stop-words and punctuation marks, stripped white space, and converted all text to lower case.

In this work we consider several state-of-the-art techniques for review modeling. The aim of modeling reviews is to build descriptive profiles for both users and items. To this end, we aggregate all reviews by a user, and consider the aggregated reviews as a single text document to be modeled, w_u. Analogously, we aggregate all reviews of a given item in order to build a document that once modeled will represent the item, w_i. In what follows, we list and briefly explain the modeling techniques we have used to model aggregated text reviews into K components for both profile vectors.

Term frequency–inverse document frequency: TF–IDF maps the contents of a review into a set of K keywords depending on the measurement of importance through the calculation of a numeric statistic [46,41]. The numeric statistic is calculated for each word from the review and then, the K more relevant words are kept.

Latent semantic analysis: LSA can be used to map the contents of a review into K latent concepts which turn out to be words that are close in meaning [10]. LSA is performed applying single value decomposition of a term-document matrix. This matrix contains the terms or words in the rows and the documents or reviews in the columns. The real number in the intersection of rows and columns indicates the occurrence of terms in the document. The K largest eigenvalues and their corresponding eigenvectors lead to a rank K approximation of the term-document matrix.

Latent Dirichlet Allocation: 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 [6]. LDA is 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 lead to different topics if the space of topics is large enough to differentiate among them.

4.3. Collaborative branch

The collaborative branch of the review-based recommender system aims to gather the opinion of the crowd from the posted reviews. Recommendation is done based on the opinion that the close neighborhood has about a specific item. Thus, we define the utility function f for the seen items $r_{u',i}$ to be the sentiment indicator that comes from the sentiment analysis of a text review that a user wrote about an item. For the unseen items, the heuristic proposed is an averaging over the rates of those similar users (neighbors) who have seen the item. Formally,

$$f_{col} = r_{u,i} = \frac{1}{N} \sum_{u' \in N_u} r_{u',i} \tag{4}$$

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 unseen items.

4.3.1. 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 [11]. 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 as the Pearson correlation coefficient or cosine-based similarity [2].

We propose 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 $\overrightarrow{u_x}$, $\overrightarrow{u_y}$. This is

$$sim(u_x, u_y) = \cos(u_x, u_y) = \frac{\overrightarrow{u_x}\overrightarrow{u_y}}{\| \overrightarrow{u_x} \| \| \overrightarrow{u_y} \|} \tag{5}$$

The user vectors are defined from sentiment ratings over all items. With this approach, pairs of users, who have both rated positively or negatively pairs of items, share similar opinions 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. Groh and Ehmig [15] propose to use social friendships or relationship to generate the neighborhood for a collaborative recommender. According to them, this approach could outperform the traditional collaborative filtering.

4.3.2. Review sentiment models

Text reviews have also been preprocessed using the aforementioned techniques before applying the following supervised sentiment models. These models have been trained with tagged reviews data and then used to predict the sentiment for the out-of-sample reviews. In fact, we have used three different data sets to build, test and validate the sentiment classifiers:

- *A training data set*: It is composed of 1500 text reviews. These reviews are tagged using AFINN [36], a list of English words rated for valence with an integer between minus five (negative) and plus five (positive). The rating is discretized in three values -1, 0 and 1 via a modified version of sign function, indicating the negative, neutral and positive reviews respectively.
- *A testing data set*: It is composed of 500 text reviews which also tagged using the AFINN approach.
- *A validation data set*: It is composed of 200 text reviews that we have manually tagged as positive, neutral or negative reviews.

Bernoulli Naive Bayes sentiment classifier: A Naive Bayes classifier considers a probabilistic model with naive independence assumptions between the features to predict the probability distribution of a sample over the set of classes. We aim to simply model the review sentiment by using a Naive Bayes classifier which considers a bag-of-words model for the review text. This means that we consider words as the model features without taking their position in the review into account and assuming independence among words of a given class (positive, negative or neutral). This can be achieved by using the Bernoulli Naive Bayes model, which defines features as independent binary inputs describing whether or not a word occurs in a given review (see [31] for a clarification on the different types of Naive Bayes classifiers).

Multinomial logistic regression sentiment classifier: 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 regression generalizes the logistic regression to a multi-class setting [22]. This classifier estimates the probabilities of the possible outcomes based on a set of features. It differs from the Naive Bayes classifier in that there is no need for statistical independence on the set of features or words used in the multinomial logit regression. 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.

Supervised Latent Dirichlet Allocation: The SLDA classifier for sentiment analysis extends the LDA topic model with response variables for each document [5]. The response variable for a sentiment analysis classifier corresponds to the sentiment class: positive, negative or neutral. Documents and their responses are jointly modeled to find the latent topics that best predict the responses for future unlabeled documents.

4.4. Hybrid set-up

Using a hybrid linear combination of the collaborative and content-based branches, we pretend to minimize the drawbacks of each individual approach. The hybrid model can be represented as follows, with $\alpha > 0$,

$$f : U \times I \rightarrow R \Rightarrow f : f_{col} + \alpha f_{cont} \tag{6}$$

On the one hand, collaborative approaches highly depend on the availability of a critical mass of users who have rated enough items to effectively predict 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 minimizes 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 this overspecialization. By combining the collaborative sentiment into the proposed system, unseen item features are also recommended enhancing the overall accuracy.

There also exist drawbacks which are common to both approaches, such as the cold start problem [24]. For example, the *new user problem* happens when a user has not reviewed any items or very few. In this situation, the recommender is not capable of performing a recommendation and this lowers the system coverage. 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 the system coverage.

These two drawbacks are mainly due to the lack of data and hence their solution necessarily undergoes through the process of effectively crawling more data from users and items.

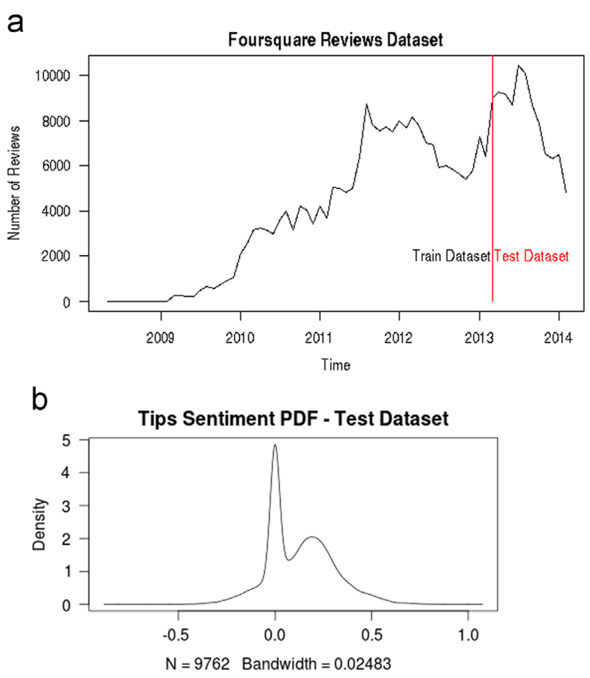


Fig. 5. Description of the Foursquare data set used in our evaluation. (a) Number of Foursquare restaurant reviews over time and (b) tips sentiment probability density function.

5. System evaluation

The evaluation methodology will measure how effectively *GeoSRS* recommends an unseen item to a given user. It will then make use of the social network to assess the performance accuracy of the system by comparing the actual user feedback about an item against the potential recommendation rate for that item. Hence, we assume that there exists some degree of causality between the fact of purchasing/experiencing an item and the subsequent action of reviewing the product/experience.

This approach also provides a simple but powerful mechanism to measure the recommendation coverage, or simply, the proportion of recommendations that the system is able to output. The inability to make a recommendation can happen due to the facts that either the user provides feedback for the first time – *new user problem* – or the item has not yet been reviewed – *new item problem*. For example, a user who checks into a Foursquare venue like a restaurant to have some food would probably provide its own feedback in the form of a review at the social network. The interpretation of this feedback (positive, negative or neutral) is used in our evaluation approach to be compared against the recommendation rate from *GeoSRS* at the reviewing time and within the neighborhood of that restaurant.

Next, we describe the Foursquare data set that is later used to assess *GeoSRS* in terms of recommendation coverage and performance accuracy.

5.1. *The Foursquare restaurant tips data set*

The Foursquare restaurant tips data set consists of 309.640 short reviews or tips from the Manhattan region. The whole data set is split into training (70% oldest tips) and test sets (30% newest tips).

In Fig. 5a, we show the number of reviews as a function of time, from the first review in 2008 until the last crawling execution in February 2014. Furthermore, this graphic shows a positive tendency in the number of reviews created since 2008, which indicates us that Foursquare has been growing since then, despite some seasonal patterns. As it has been said, the whole data set is split into two subsets, which are also indicated in the plot.

Fig. 5 b shows the distribution of sentiment polarity of tips in the test data set. While negative values in the pdf express 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. For evaluation purposes, we consider neutral feedback as a positive experience since most of them are statements such as "*Try the hot chocolate*!"; we make the assumption that the user implicitly rates the experience as positive.

We also note that the test data set is unbalanced in terms of positive and negative ratings, reflecting 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.

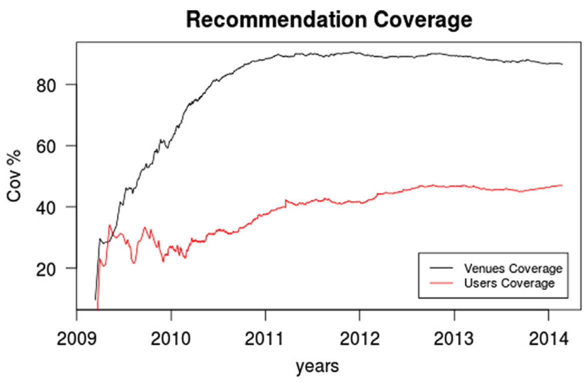


Fig. 6. Recommendation coverage for venues and users.

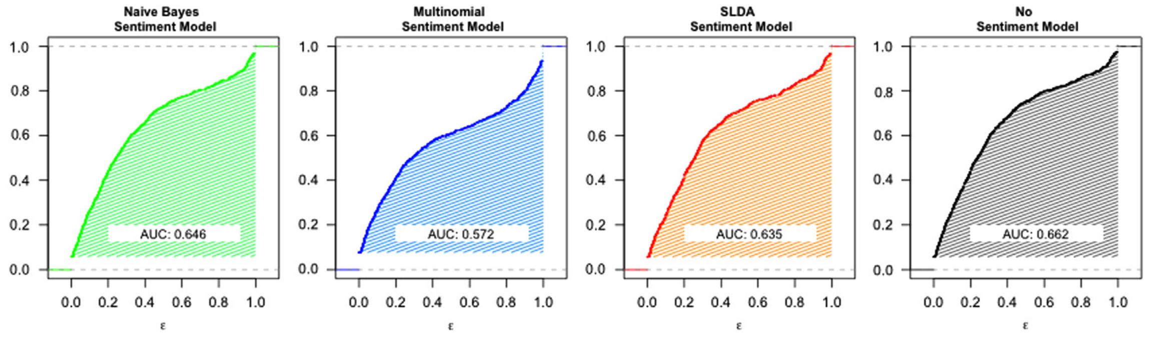


Fig. 7. AUCs from the recommendation error cdf for different sentiment models.

5.2. Recommendation coverage

Recommendation coverage measures the domain of items over which the system can perform recommendations [14]. 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 are effectively recommended. Here, we adopt the former definition since we gather the latter in the concept of performance accuracy defined later. Thus, formally, *Coverage* is defined as the percentage of available items (I) for which the recommender system can output a prediction (I_P) i.e. $Cov = \left(|I_P|/|I|\right) \cdot 100$.

In *GeoSRS* and within the proposed evaluation scheme, I_P only depends on the presence or absence of historical data (tips) in the training data set for an item and user from the test data set. Given that the content-based branch outputs a rate always that a given item and user have more than one tip, the overall recommendation coverage does not depend on the collaborative branch.

In Fig. 6, we plot the recommendation coverage as a function of time in order to show how the coverage softly increases when the number of tips starts to be large (cf. Fig. 5a) and the proportion between new and old items/users stabilizes.

From this picture, we can also see that while there is a good coverage value on restaurants venues (85% in 2014), coverage on users is quite low (45% in 2014). This low recommendation coverage on users is mainly due to two reasons. On the one hand, there are new users who signed up to Foursquare during the test data set time frame (2013–2014); on the other hand, some users might have tipped just once Manhattan restaurants, but they might have tipped restaurants out of Manhattan. This poor recommendation coverage on users could be mitigated by adapting the crawling system to also crawl tips from users, instead of just crawling tips from venues.

5.3. Performance accuracy

Performance accuracy measures the goodness of the rate prediction against the actual rate [44,17]. Typically, it is formulated as the number of successful cases over all the cases in the test data set:

$$accuracy = \frac{\text{good cases}}{\text{overall cases}}. \tag{7}$$

Table 2
Number of positive and negative predicted ratings.

Rating	*SLDA*	*Multinomial*	*Naive Bayes*	*No Sentiment*
positive ratings	191.809	142.758	205.585	212.947
negative ratings	21.138	70.189	7.362	0

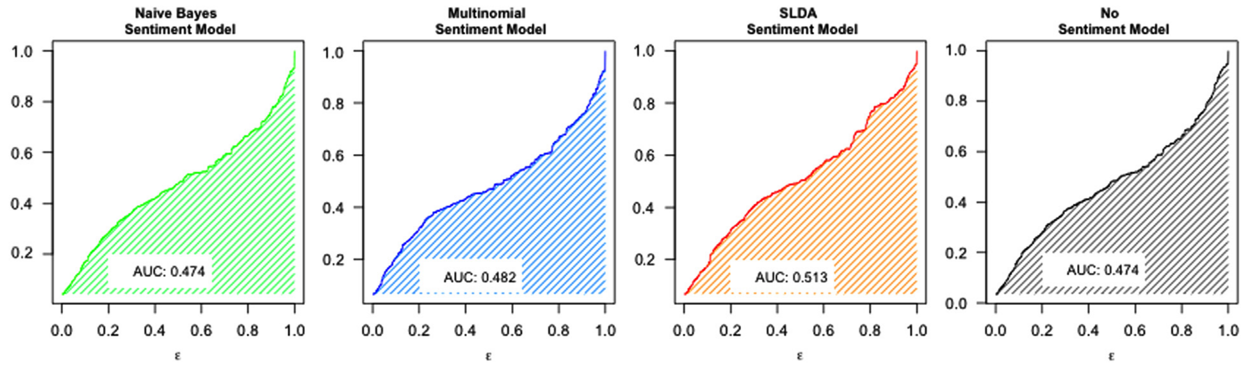


Fig. 8. AUCs from the recommendation error cdf for different sentiment models in a balanced data set.

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

$$accuracy = 1 - \frac{\text{negative cases}}{\text{overall cases}} = 1 - \overline{\epsilon} \tag{8}$$

where ϵ_k is the recommendation error rate for the kth experiment, defined as follows:

$$\epsilon_k = \begin{cases} \dfrac{N_{pos_k}}{N_k} & \text{Sentiment is positive} \\ 1 - \dfrac{N_{pos_k}}{N_k} & \text{Sentiment is negative} \end{cases} \tag{9}$$

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.

Notice that if a review is positive, then ϵ_k is low if the item is ranked at the top of the recommendation list, while if a review is negative, ϵ_k is low if the item is ranked at the bottom of the list.

5.3.1. *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, a.k.a. neighborhood. As introduced earlier, the N neighbors are identified through the k-nearest neighbor algorithm that uses the cosine similarity of the historical user experiences.

First, we plot the Area Under the Curves (AUC) of the recommendation error cumulative density functions (cdf) which let us interpret the goodness of the models with the available Foursquare data sets.

As Fig. 7 suggests that Naive Bayes outperforms the Multinomial and SLDA models. Even more relevant is the fact that using no sentiment performs better 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 with more positive than negative experiences benefits those models that are biased towards positive ratings. As shown in Table 2, Naive Bayes, and obviously the no-sentiment model, have bigger proportions of positive ratings in the training data sets (because the trained sentiment classifier was generated in this way) and they perform better in test data set which has greater number of positive experiences.

In order to mitigate the unbalancing effects described above, we propose to subsample the testing data set in two balanced data sets and then compute the performance accuracy. In order to estimate properly the cumulative density

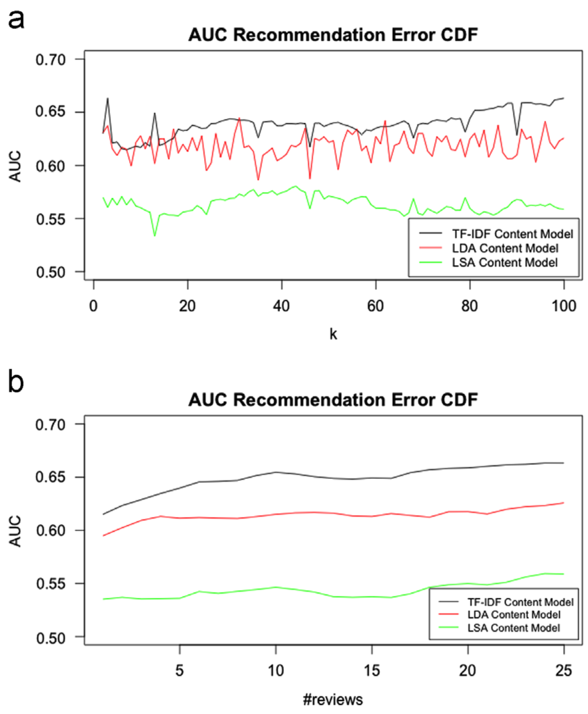


Fig. 9. Performance of the content-based branch. (a) AUC of the cdf as a function of the number of features and (b) AUC of the cdf as a function of the number of reviews.

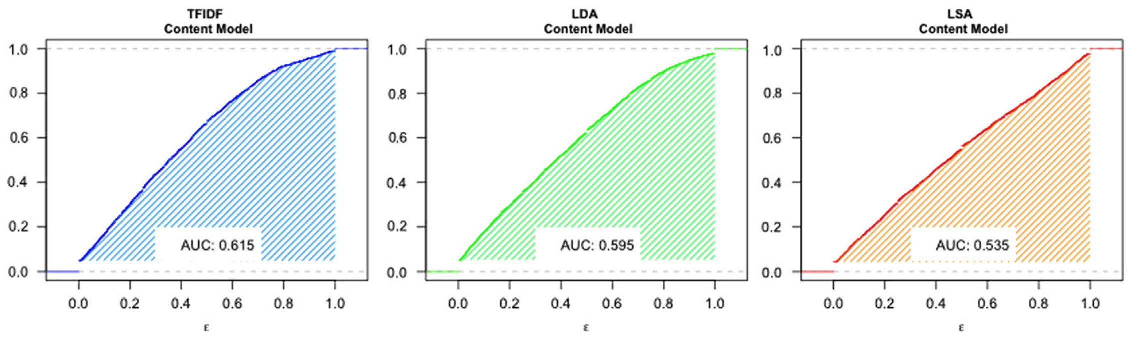


Fig. 10. AUC from the cdf of the recommendation error.

function and the AUC statistic, we repeat the sub-sampling process hundreds of times and average the results. The results of this assessments are shown in Fig. 8 for each of the sentiment models.

The results of this analysis show that the sentiment models behave 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.

5.3.2. *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 state-of-the-art text models to represent 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 AUC 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.

Fig. 9a 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 noisy text reviews, 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 use $k=100$ for all three models, which guarantees a fair tradeoff 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. 10.

One of the aspects that could affect the performance accuracy of the content-based models is the number of reviews or tips per venue and user. As shown in Fig. 9b, 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.

In general, it can be interpreted that not only the number of the reviews influence the performance accuracy and coverage of a review-based recommendation system, but also other features about the data such as the quality of the reviews, the heterogeneity of topics reviewed, the influence of the reviewer, amongst others.

As a closing remark for this section we point out that we believe our selection of metrics for capturing the quality of our rankings are adequate for the task and data at hand. The reader should note that we are not predicting ratings, and so more common measures such as MAE or RMSE are not appropriate in our context. Refer to [2,27] for more in depth discussion on the topic of classification accuracy metrics for ranking systems.

5.3.3. GeoSRS: our hybrid recommender system

Finally, we show results of *GeoSRS*, obtained by linearly combining both content-based and collaborative approaches, as discussed in Section 4.1. Our aim in this section is twofold. Firstly, we want to optimize the coefficient α from Eq. (6) that minimizes the recommendation error. In other words, we seek to find α values that maximize the AUC of the recommendation error cumulative density function. Hence, we simulate several recommendations for different α values and computed its associated AUC to find the one that maximizes it. Secondly, we aim to show that the hybrid approach consistently outperforms each individual component.

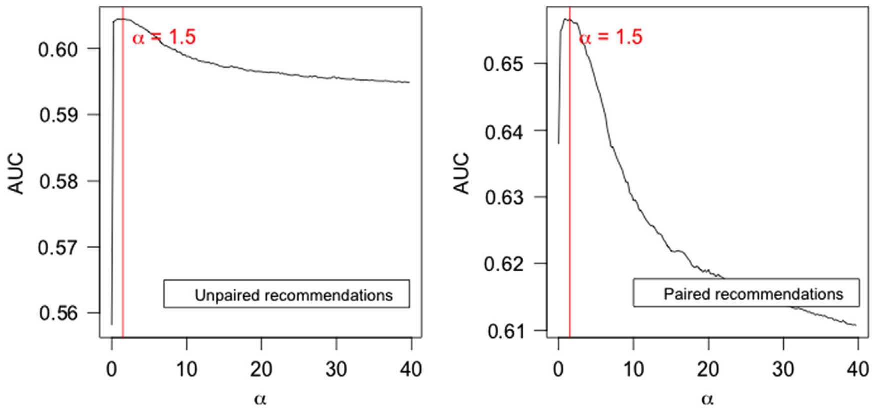


Fig. 11. Hybrid linear combiner optimization.

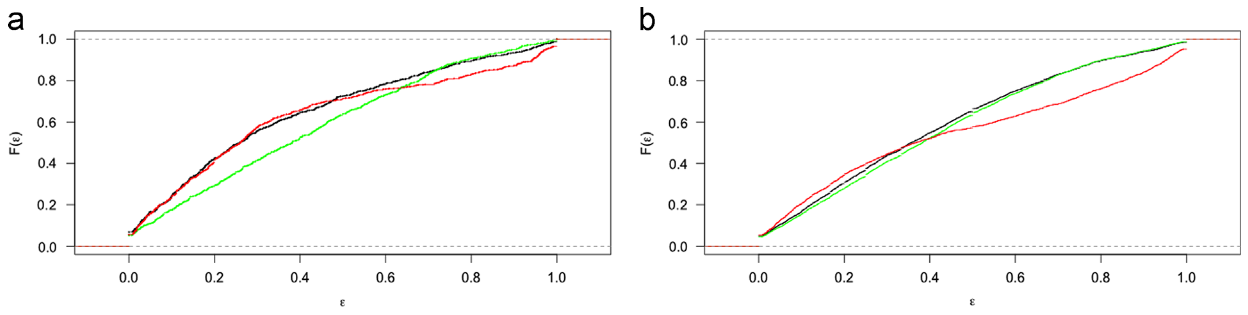


Fig. 12. Empirical cumulative density functions for content-based (green), collaborative (red) and hybrid (black) set-ups. (a) Paired recommendations and (b) unpaired recommendations. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Table 3
AUC and coverage for different set-ups of the review-based recommender.

Recommendation	Hybrid	Content-based	Collaborative	Coverage%
Paired rec.	0.6566	0.5984	0.6351	8.34
Unpaired rec.	0.6044	0.5922	0.5514	39.81

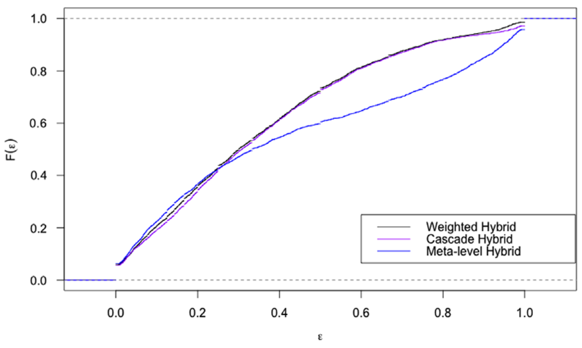


Fig. 13. Comparison against state-of-the-art hybridization techniques.

To globally assess the recommender, we consider again the evaluation model with the testing experiences classified as positive or negative. The SLDA sentiment model is chosen since it scored higher in the balanced evaluation test. Furthermore, TF–IDF is chosen as the content-based model to be included in this hybrid approach.

Fig. 11 plots two curves, both representing the AUC of *GeoSRS* for different values of α ranging from 0 to 20, see Eq. (6). Notice that the collaborative branch does not always output predictions due to lack of close neighbors to the given user. The left-most plot shows the performance of *GeoSRS* for all reviews in the test data set, whereas the right-most plot shows the performance of *GeoSRS* reviews that can be covered (make a recommendation) by both system branches ("paired" recommendations). As the graphic shows, an α equal to 1.5 maximizes AUCs from both scenarios. Note also that $\alpha = 0$ corresponds to the collaborative recommender, while $\alpha \to \infty$ corresponds to the content-based recommender.

Next, we plot the empirical cdf for both situations with $\alpha = 1.5$. Fig. 12a shows the performance accuracy for paired recommendations. That the hybrid system has greater AUC can also be clearly seen in Table 3.

Fig. 12 b 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. Particularly, 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 Fig. 12b, 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.

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 performance benefits from either when the collaborative branch can work out a rating or when it cannot. Nonetheless, the biggest contributions happen when the collaborative branch can recommend.

It is relevant to say that restricting the recommender just to those users with a collaborative prediction (paired recommendation) has a huge impact to the overall recommendation coverage since the collaborative branch has low coverage due to the high sparsity in commonly rated venues. Table 3 shows the relationship between the performance accuracy in terms of AUC for different system configurations and the recommendation coverage in each data set situation mentioned above.

For the sake of comparison, we implement two state-of-the-art hybridization techniques that have been shown to work specially well when employing two components of different strengths (for example, collaborative and content-based) named meta-level and cascade [9].

Regarding the meta-level setup, we build a collaborative through content approach similar to Pazzani [38], which uses the content-based recommendation matrix to identify the close neighborhood of each user under evaluation. The unseen rates are estimated by means of averaging their neighbor's sentiment rates, computed beforehand through the sentiment analysis model. On the contrary, the cascade configuration builds a strictly hierarchical hybrid which first applies the stronger branch, in our scenario this is the collaborative, and then it uses the content-based branch for those items which the collaborative could not decide well. A very similar system was proposed by Burke [8] for restaurant recommendation, but their hybrid system relied on a knowledge-based in place of a content-based recommender branch.

Fig. 13 shows the empirical density curves of the error for weighted, meta-level and cascade hybrids. As it is shown, the simple weighted hybrid system used in *GeoSRS* outperforms the other two enabling a simple but rather powerful hybridization technique.

6. Conclusions and future work

GeoSRS enables social network users to mitigate the information overload problem by digging into text reviews data and recommending personalized items according to their preferences based on their past reviews. *GeoSRS* supposes a novel attempt to purely integrate text reviews data into a working recommending system.

This paper assesses different *GeoSRS* set-ups which use several state-of-the-art text mining techniques. According to the proposed offline evaluation approach, the results justify combining text review content and sentiment into an hybrid recommending engine. This is particularly relevant in balanced and paired data sets. Concretely, our results show that for our data set the best configuration is given by TF–IDF for the content-based branch and SLDA for the collaborative branch with a coefficient of $\alpha = 1.5$. Moreover, the evaluation outcomes point out the importance of the data retrieval process and the quality of the data set, by showing that the larger the number of reviews, the greater the performance accuracy. Not least, we have also shown the benefits of using a simple hybridization technique like the weighted linear combination compared to employing more complex techniques such as cascade or meta-level.

Within this evaluation scheme and *GeoSRS*, we show that the goodness of recommendation coverage is tied to the fact of having historical reviews for all users and all items. We observe that the growth of the social network activity over the years directly improves this evaluation figure, but we also state that the more the retrieved data, the better the coverage. We speculate that the quality of our recommendations should improve as the number of reviews increases over time.

In order to directly impact the performance accuracy and recommendation coverage of the *GeoSRS*, this paper also motivates the use of *Quadtree* algorithm to efficiently retrieve geolocated data, such as reviews from Foursquare. A parallel version of *Quadtree* is detailed in this paper with the aim to effectively crawl data from large urban areas. As a result, *GeoSRS* together with the *Quadtree* crawling process becomes key components of a recommending system capable of working in large data sets of reviews for large urban areas.

Finally, we can conclude that *GeoSRS* results, as in many other recommender systems, highly depend on the availability and quality of the data. We have shown that by balancing data sets or making paired recommendations in the hybrid set-up, the outcome of the system could differ. Because of this, we deeply encourage *GeoSRS* to undergo through online evaluation tests, such as randomized experiments or A/B testing, in order to mitigate the skewness on the Foursquare test data sets and get more reliable results.

Acknowledgements

The authors gratefully acknowledge the extensive comments of the anonymous reviewers that lead to an improvement of the original manuscript.

References

[1] S. Aciar, D. Zhang, S. Simoff, J. Debenham, Informed recommender: basing recommendations on consumer product reviews, IEEE Intell. Syst. 22 (3) (2007) 39–47.

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

[3] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Kalai, V. Mirrokni, M. Tennenholtz, Trust-based recommendation systems: an axiomatic approach, In: Proceedings of the 17th International Conference on World Wide Web, ACM, Beijing, 2008, pp. 199–208.

[4] B. Berjani, T. Strufe, A recommendation system for spots in location-based online social networks, In: Proceedings of the Fourth Workshop on Social Network Systems, Salzburg, ACM, 2011, p. 4.

[5] D. Blei, J. McAuliffe, Supervised topic models, In: J. Platt, D. Koller, Y. Singer, S. Roweis (Eds.), Advances in Neural Information Processing Systems, vol. 20, MIT Press, Vancouver, 2008, pp. 121–128.

[6] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet allocation, J. Mach. Learn. Res. 3 (March) (2003) 993–1022.

[7] D. Böhning, Multinomial logistic regression algorithm, Ann. Inst. Stat. Math. 44 (1) (1992) 197–200.

[8] R. Burke, Hybrid recommender systems: survey and experiments, User Model. User-Adapt. Interact. 12 (4) (2002) 331–370.

[9] R. Burke, Hybrid web recommender systems, In: The Adaptive Web, Springer, 2007, pp. 377–408.

[10] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman, Indexing by latent semantic analysis, J. Am. Soc. Inf. Sci. 41 (6) (1990) 391–407.

[11] M. Diaby, E. Viennet, T. 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, Niagara Falls, 2013, pp. 821–828.

[12] K. Falahi, N. Mavridis, Y. Atif, Social networks and recommender systems: a world of current and future synergies, In: A. Abraham, A.-E. Hassanien (Eds.), Computational Social Networks, 2012, pp. 445–465.

[13] M. Gartrell, X. Xing, Q. Lv, A. Beach, R. Han, S. Mishra, K. Seada, Enhancing group recommendation by incorporating social relationship interactions, In: Proceedings of the 16th ACM International Conference on Supporting Group Work, GROUP '10, Sanibel Island, 2010, pp. 97–106.

[14] M. Ge, C. Delgado-Battenfeld, D. Jannach, Beyond accuracy: evaluating recommender systems by coverage and serendipity, In: Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys '10, Foster City, 2010, pp. 257–260.

[15] G. Groh, C. 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, Sanibel Island, 2007, pp. 127–136.
[16] I. Guy, D. Carmel, Social recommender systems, In: Proceedings of the 20th International Conference on Companion on World Wide Web, WWW '11, Hyderabad, 2011, pp. 283–284.
[17] F. Hernández del Olmo, E. Gaudioso, Evaluation of recommender systems: a new approach, Expert Syst. Appl. 35 (3) (2008) 790–804.
[18] J. Huang, X.-Q. Cheng, J. Guo, H.-W. Shen, K. Yang, Social recommendation with interpersonal influence, In: Proceedings of the 2010 Conference on ECAI 2010: The 19th European Conference on Artificial Intelligence, 2010, pp. 601–606.
[19] N. Jakob, S.H. Weber, M.C. Müller, I. Gurevych, Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations, In: Proceedings of the First International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion, TSA '09, Hong Kong, 2009, pp. 57–64.
[20] M. Jamali, M. Ester, Trustwalker: a random walk model for combining trust-based and item-based recommendation, In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, Paris, 2009, pp. 397–406.
[21] A. 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, Gallipoli, 2004, pp. 48–54.
[22] S.-B. Kim, H.-C. Rim, D. Yook, H.-S. Lim, Effective methods for improving Naive Bayes text classifiers, In: PRICAI 2002: Trends in Artificial Intelligence, Springer, Tokyo, 2002, pp. 414–423.
[23] S.-M. Kim, P. Pantel, T. Chklovski, M. Pennacchiotti, Automatically assessing review helpfulness, In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP '06, Sydney, 2006, pp. 423–430.
[24] B. Lika, K. Kolomvatsos, S. Hadjiefthymiades, Facing the cold start problem in recommender systems, Expert Syst. Appl. 41 (4 (Part 2)) (2014) 2065–2073.
[25] X. Liu, K. Aberer, SoCo: a social network aided context-aware recommender system, In: Proceedings of the 22nd International Conference on World Wide Web, WWW '13, Rio, 2013, pp. 781–802.
[26] Y. Liu, X. Huang, A. An, X. Yu, Modeling and predicting the helpfulness of online reviews, In: The Eighth IEEE International Conference on Data Mining, ICDM '08, Pisa, 2008, pp. 443–452.
[27] L. Lü, M. Medo, C.H. Yeung, Y.-C. Zhang, Z.-K. Zhang, T. Zhou, Recommender systems, Phys. Rep. 519 (1) (2012) 1–49.
[28] Y. Lu, P. Tsaparas, A. Ntoulas, L. Polanyi, Exploiting social context for review quality prediction, In: Proceedings of the 19th International Conference on World Wide Web, WWW '10, Hong kong, 2010, pp. 691–700.
[29] H. Ma, H. Yang, M.R. Lyu, I. King, SoRec: social recommendation using probabilistic matrix factorization, In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM '08, Napa Valley, 2008, pp. 931–940.
[30] P. Massa, P. Avesani, Trust-aware recommender systems, In: Proceedings of the 2007 ACM Conference on Recommender systems, ACM, Minnesota, 2007, pp. 17–24.
[31] A. McCallum, K. Nigam, et al., A comparison of event models for Naive Bayes text classification, In: AAAI-98 Workshop on Learning for Text Categorization, vol. 752, Madison, 1998, pp. 41–48.
[32] B. Mobasher, X. Jin, Y. Zhou, Semantically enhanced collaborative filtering on the web, In: Web Mining: From Web to Semantic Web, Springer, Cavtat-Dubrovnik, 2004, pp. 57–76.
[33] S. Moghaddam, M. Jamali, M. Ester, J. Habibi, Feedbacktrust: using feedback effects in trust-based recommendation systems, In: Proceedings of the Third ACM Conference on Recommender Systems, RecSys '09, New York, 2009, pp. 269–272.
[34] M.F. Mokbel, J. Bao, A. Eldawy, J.J. Levandoski, M. Sarwat, Personalization, socialization, and recommendations in location-based services 2.0, In: PersDB 2011 Workshop, Seattle, Washington, USA, 2011.
[35] C.-C. Musat, Y. Liang, B. Faltings, Recommendation using textual opinions, In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, AAAI Press, Bellevue, 2013, pp. 2684–2690.
[36] F.Å. Nielsen, A new ANEW: evaluation of a word list for sentiment analysis In microblogs, In: Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big Things Come in Small Packages, Heraklion, 2011, pp. 93–98.
[37] D. O'Doherty, S. Jouili, P. Van Roy, et al., Trust-based recommendation: an empirical analysis, In: The Sixth ACM Workshop on Social Network Mining and Analysis (SNA-KDD 2012), Beijing, 12 August, 2012.
[38] M.J. Pazzani, A framework for collaborative, content-based and demographic filtering, Artif. Intell. Rev. 13 (5–6) (1999) 393–408.
[39] B. Resch, A. Zipf, E. Beinat, P. Breuss-Schneeweis, M. Boher, Towards the live city: paving the way to real-time urbanism, Int. J. Adv. Intell. Syst. 5 (3–4) (2012) 470–482.
[40] K. Reschke, A. Vogel, D. Jurafsky, Generating recommendation dialogs by extracting information from user reviews, In: ACL (2), 2013, pp. 499–504.
[41] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Inf. Process. Manag. 24 (5) (1988) 513–523.
[42] H. Samet, The quadtree and related hierarchical data structures, ACM Comput. Survey 16 (June (2)) (1984) 187–260.
[43] M. Sarwt, J.J. Levandoski, A. Eldawy, M.F. Mokbel, Lars*: an efficient and scalable location-aware recommender system, IEEE Trans. Knowl. Data Eng. (2013) 99.
[44] G. Shani, A. Gunawardana, Evaluating recommendation systems, In: Recommender Systems Handbook, 2011, pp. 257–297.
[45] B. Snyder, R. Barzilay, Multiple aspect ranking using the good grief algorithm, In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL), Rochester, 2007, pp. 300–307.
[46] K. Sparck Jones, A statistical interpretation of term specificity and its application in retrieval, J. Doc. 28 (1972) 11–21.
[47] P. Symeonidis, D. Ntempos, Y. Manolopoulos, Location-based social networks, In: Recommender Systems for Location-based Social Networks, Springer, 2014, pp. 35–48.
[48] P. Symeonidis, D. Ntempos, Y. Manolopoulos, Recommender Systems for Location-based Social Networks, Springer, 2014.
[49] K. Tu, B. Ribeiro, D. Jensen, D. Towsley, B. Liu, H. Jiang, X. Wang, Online dating recommendations: matching markets and learning preferences, In: Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion, Seoul, 2014, pp. 787–792.
[50] F. Xia, N.Y. Asabere, H. Liu, N. Deonauth, F. Li, Folksonomy based socially-aware recommendation of scholarly papers for conference participants, In: Proceedings of the 23rd International Conference on World Wide Web Companion, Seoul, 2014, pp. 781–786.
[51] D. Yang, D. Zhang, Z. Yu, Z. Wang, A sentiment-enhanced personalized location recommendation system, In: Proceedings of the 24th ACM Conference on Hypertext and Social Media, ACM, Paris, 2013, pp. 119–128.
[52] M. Ye, P. Yin, W.-C. Lee, Location recommendation for location-based social networks, In: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, San Jose, 2010, pp. 458–461.
[53] Z. Zhang, B. Varadarajan, Utility scoring of product reviews, In: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, CIKM '06, Arlington, 2006, pp. 51–57.
[54] Y. Zheng, Tutorial on location-based social networks, In: Proceedings of the 21st International Conference on World Wide Web (WWW), ACM, Lyon, 2012.
[55] Y. Zheng, L. Zhang, Z. Ma, X. Xie, W.-Y. Ma, Recommending friends and locations based on individual location history, ACM Trans. Web 5 (1) (2011) 5:1–5:44.