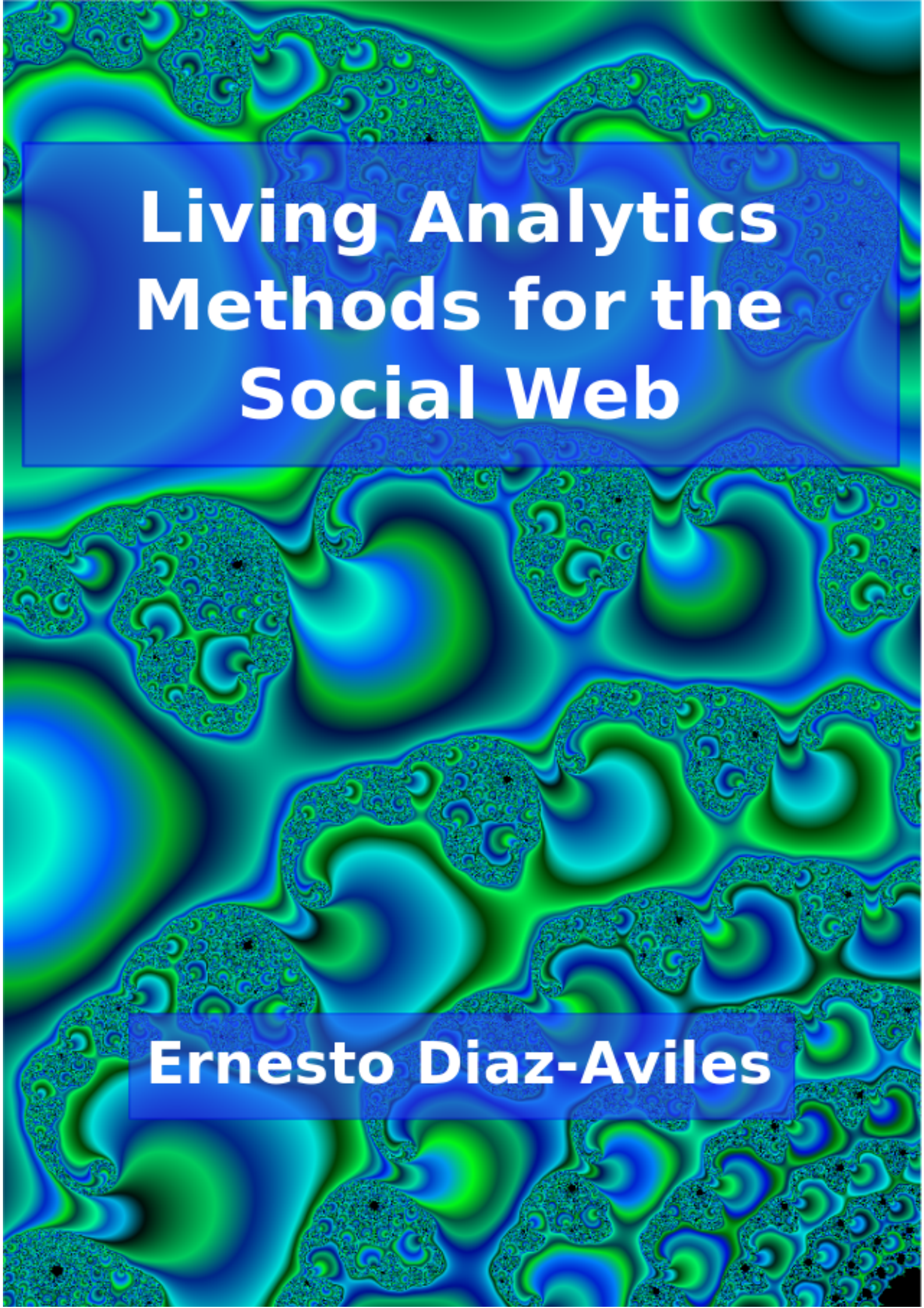

Living Analytics Methods for the Social Web
Ernesto Diaz-Aviles

Living Analytics Methods for the Social Web

Ernesto Diaz-Aviles
L3S Research Center
Gottfried Wilhelm Leibniz Universität Hannover
Appelstraße 9a
30167 Hannover
Germany
E-mail: diaz@L3S.de

ISBN 978-1-4921-9424-8
ISBN 1492194247

ACM Categories: H.3.3; I.5.1; K.4.
ACM General Terms: Algorithms; Performance; Experimentation; Human Factors.

published by vedax ⊙

About the cover: fractal generated using *Endlos*, which is available at http://endlos.sourceforge.net/.

2013-08-08

0 1 1 2 3 5 8 13

Living Analytics Methods for the Social Web

by

Ernesto Diaz-Aviles

Para Carolina. "La mamá".

Preface

The collective effervescence of social media production has been enjoying a great deal of success in recent years. The hundred of millions of users who are actively participating in the *Social Web* are exposed to ever-growing amounts of sites, relationships, and information.

This work contributes state-of-the-art methods and techniques to the emerging field of *Living Analytics*, whose main goal is to capture people interactions in real-time and to analyze these data in order to relieve information overload. We introduce intelligent filtering approaches that exploit social interactions, multidimensional relationships, metadata, and other data becoming ubiquitous in the social web, in order to discover and recommend the most relevant and attractive information that meets users' individual needs.

In particular, the contributions of this work fall into mainly two categories: (i) *Recommender Systems:* We present novel algorithms that advance the state-of-the-art in Online Collaborative Filtering. Moreover, we propose an approach based on Swarm Intelligence to directly optimize ranking functions for item recommendations. New approaches to address the cold-start problem in social recommender systems are also part of our contributions. In addition, we also offer a personalized ranking algorithm for Epidemic Intelligence.
(ii) *Collective Intelligence:* Our contributions in the field of computational social science are twofold. First, we explore how social media streams can be exploited for Epidemic Intelligence and show its potential for early warning detection and outbreak analysis and control. Second, we show how the real-time nature of social media streams can be leveraged to take the pulse of political emotions.

In total, the methods and studies included in this work constitute an *analytics toolbox* to help understand and analyze the social web.

Keywords: Machine Learning; Collaborative Filtering; Social Media.

Contents

1 Introduction

1.1 Living Analytics and the Social Web

The Web of people is highly dynamic and the life experiences between our on-line and "real-world" interactions are increasingly interconnected. The massive amounts of information from people's daily living interactions require new and innovative analytic models to observe, understand and predict the dynamics of the actors and components of the Web.

The objective of such *Living Analytics* models is to help people live better, for example, by assisting them with the overwhelming amount of choices they face as they consume goods, services, social media and leisure time. *Living Analytics* aims to capture people interactions in real-time to analyze and process this information in order to produce valuable output that is fed back to the Web of people for the benefit of its members.

The highly dynamic and huge volume of potentially relevant items from an immense number of sources (e.g., blogs, social networks), and information seeking therein make it harder for existing systems and applications to obtain and process a complete and accurate impression of the situation, in a timely manner.

Consider a system envisioned to support its users to conduct reliable assessments of dynamics topics on the Web, such as: views on political developments, economic events and crises, as well as pandemics or natural catastrophes. Moreover, Social Media technologies have given rise to citizen journalism, so the public at large may actively contribute to generating new content in these areas. The goal of such system is to go beyond individual resources to an aggregated overview, by automatically collecting the relevant sources, extracting the required data,

aggregating the results, and finally enabling in-depth investigation with tools designed to support visual analysis. The need for overviews of high volume and user-generated content is crucial for many stakeholders, for example, journalists, opinion analysts, social scientists, public safety institutions, product designers and marketers, and well as the general public.

On-line applications that tackle the information deluge problem exist, at least for selected topics. For example, detecting the global trends based on the volume of Twitter[1] messages (*tweets*) or generating visualizations from publicly available statistics. While these tools are very useful, the respective services tend to be limited to a very small fraction of interesting information. In addition, users are presented with "one-size-fits-all" solutions that do not necessarily reflect their individual interests.

Having this scenario in mind, several interesting research questions arise in the scope of *Living Analytics*. For example:

- **Understanding and Predicting Behavior in Real-Time Context.** What theoretical, methodological, and empirical extensions are needed to observe and analyze the behavior of users and groups in the network in near or real-time, as it is occurring? and how is it possible to follow the evolution of network behavior over extended periods of time?

- **Collective Intelligence**. What content do people create and share? How can the collective behavior of people be harnessed to solve complex tasks? How does collective intelligence evolve over time? How can the trends observed in social media be employed to improve discovery performance?

- **Real-Time Modeling and Experimentation**. How can the ability to update predictive models, as well as, execute and interpret experiments in these real-time networked settings of users, and their group interactions, be realized and improved?

[1] **Twitter:** twitter.com

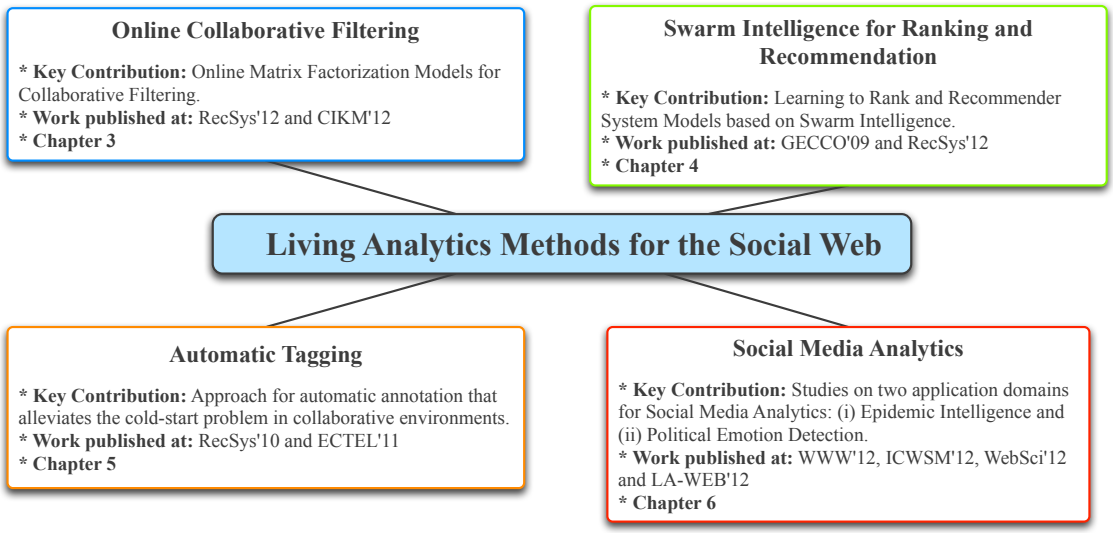


Figure 1.1: **Main Components of the Work. The figure shows this work's areas of research, the main contributions, and the international conferences where the material has been already published.**

In this work, we tackle some of the above-mentioned issues and integrate our techniques and results into one coherent framework. We present the outline of the book and our contributions in the next section. The main components of the work are depicted in Figure 1.1.

1.2 Outline and Contributions

Living Analytics field of research is distinctive in that it combines the key technologies of statistical machine learning, large scale data mining, and computational tools for the analysis of dynamic social networks with analytics focused on user behavior and social media [LARC, 2012].

In this work, our main goal is to provide a set of tools to capture people's interactions from a highly dynamic stream of data, automatically annotate resources on the Web, and to understand and predict user actions and preferences. Therefore, our contributions refer to various disciplines, e.g., information filtering and retrieval, recommender systems, and computational social science. Note that all these building bricks are *contributions in their own right*, being important not only for the overall *Living Analytics* field, but also for numerous other applications, e.g., matrix factorization in collaborative filtering systems [Koren

et al., 2009], social stream mining [Zubiaga, 2012], social tagging systems [Jäschke et al., 2008], epidemic intelligence [Fisichella et al., 2011], computational social science [Giles, 2012] and sentiment analysis [Liu, 2012].

Online Collaborative Filtering (Chapter 3)

Collaborative Filtering (CF) has shown to be an effective approach to recommender systems. The essence of CF lies in analyzing past user and item interactions to generate personalized recommendations based on the preferences of other users with similar behavior. One of CF's most successful techniques is low dimensional linear factor models, that assume user preferences can be modeled by only a small number of *latent* factors [Koren et al., 2009]. Although latent factor models are able to generate high quality recommendations, coping with fast changing trends in the presence of large scale data is a challenge, since retraining such models is costly. Our proposed approach features two important contributions:

- **Online Personalized Ranking based on Matrix Factorization.** We introduce a novel framework for online collaborative filtering based on a pairwise ranking approach for matrix factorization in the presence of streaming data. We propose novel online learning algorithms and show experimentally that our approaches achieve state-of-the-art performance when recommending a short list of interesting and relevant topics to users from a continuous high volume stream data, and under the constraints of bounded space and time. In total, we provide an approach for integrating large-scale collaborative filtering with the real-time nature of social media streams.
- **Selective Model Updates for Collaborative Filtering.** For unpersonalized learning to rank, many studies have been made in the field of information retrieval [Liu, 2011]. This work presents an innovative personalized ranking perspec-

tive to matrix factorization for social media streams, which has not been reported before in the literature.The novelty of our approach lies in a selective sampling strategy to update the model based on personalized small buffers. Our empirical study used real-world data, from the micro-blog service Twitter, as a test bed and showed that models updated using the selective sampling approach proposed here, significantly outperform online methods that use random samples of the data.

Swarm Intelligence for Ranking and Recommendation (Chapter 4)

We propose an approach to learn ranking functions that directly optimize non-smooth Information Retrieval (IR) metrics using Swarm Intelligence for the ranking and recommendation tasks. In particular, our approaches are based on Particle Swarm Optimization (PSO), a global non-linear optimization algorithm based on swarm intelligence, which is inspired by the social behavior of biological organisms.

Our work addresses the item recommendation task in the context of recommender systems. While learning to rank algorithms use hand-picked features to represent items, we learn such features based on user-item interactions and apply PSO, which does not require, nor approximate, gradients of the cost function, as in the case of the models explored in Chapter 3. This key characteristic and its resilience to local minima, allow us to directly optimize non-smooth IR measures, such as MAP (Mean Average Precision) [Baeza-Yates and Ribeiro-Neto, 2011], a desirable property to tackle the ranking and top-N recommendation problems.

Automatic Tagging (Chapter 5)

Social Tagging has proven to be an intuitive and flexible Web 2.0 mechanism to enhance the users' online experience [Jäschke et al., 2008]. Tags are capable of facilitating search, easing navigation

(e.g., tag clouds), and improving personalization in collaborative tag recommendations and across disparate media types. When a resource does not have any associated tags or users, a collaborative tagging recommender lacks valuable information needed to provide useful recommendations, an issue known as *the cold start problem*. We present an approach to addressing the dynamics associated with online environments, where novel items appear rapidly, e.g., news articles. We use probabilistic topic models, in specific, Latent Dirichlet Allocation, and show the ability of our method to enrich sparse and limited textual information by means of exploiting the resource redundancy and latent topic overlap between similar resources found in an auxiliary domain. The approach developed in this chapter nicely complements the limitations of the models for CF discussed in Chapter 3 and Chapter 4 when facing cold start problems.

Social Web Analytics (Chapter 6)

We explore and demonstrate the potential of monitor social media streams (e.g., Twitter) for early warning of disease outbreaks. Furthermore, for outbreak analysis and control, many studies have been made for systems that return documents in response to a query. Little effort has been devoted to exploiting learning to rank in a personalized setting, specially in the domain of epidemic intelligence. We present an innovative personalized ranking approach that offers decision makers the most relevant and attractive tweets for risk assessment, by exploiting latent topics and social hash-tagging behavior in Twitter. For Computational Social Science, we offer an empirical study that shows how the real-time nature of social media streams, in particular, Twitter, can be leveraged to take the pulse of political emotions in emerging regions of the world, namely: Latin America. We performed a sentiment analysis of tweets and brief blog posts over a period of six months. This work presents, not only the extracted emotions and polarity, but also goes a step forward and quantifies which combination of emotions explains better the public's opinion.

1.3 Published Work

Large portions of contributions made in this work have been published in international conferences. Results of **Chapter 3** have been published in

- Diaz-Aviles, E., Drumond, L., Gantner, Z., Schmidt-Thieme, L., and Nejdl, W. (2012a). What Is Happening Right Now ... That Interests Me? Online Topic Discovery And Recommendation In Twitter. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12.

and

- Diaz-Aviles, E., Drumond, L., Schmidt-Thieme, L., and Nejdl, W. (2012b). Real-time Top-N Recommendation In Social Streams. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 59–66, New York, NY, USA. ACM.

Chapter 4 relates to

- Diaz-Aviles, E., Nejdl, W., and Schmidt-Thieme, L. (2009). Swarming To Rank For Information Retrieval. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 9–16, New York, NY, USA. ACM.

and

- Diaz-Aviles, E., Georgescu, M., and Nejdl, W. (2012c). Swarming To Rank For Recommender Systems. In *Proceedings of the sixth ACM Conference on Recommender Systems*, RecSys '12, pages 229–232, New York, NY, USA. ACM.

Results of **Chapter 5** have been published in

- Diaz-Aviles, E., Georgescu, M., Stewart, A., and Nejdl, W. (2010). Lda For On-the-fly Auto Tagging. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 309–312, New York, NY, USA. ACM.

- Diaz-Aviles, E., Fisichella, M., Kawase, R., Nejdl, W., and Stewart, A. (2011a). Unsupervised Auto-tagging For Learning Object Enrichment. In *Proceedings of the 6th European Conference on Technology Enhanced Learning: Towards Ubiquitous Learning*, EC-TEL'11, pages 83–96, Berlin, Heidelberg. Springer-Verlag (**Best Paper Award**).

Finally, **Chapter 6** is documented in

- Diaz-Aviles, E., Stewart, A., Velasco, E., Denecke, K., and Nejdl, W. (2012f). Towards Personalized Learning To Rank For Epidemic Intelligence Based On Social Media Streams. In *Proceedings of the 21st International Conference Companion on World Wide Web*, WWW '12 Companion, pages 495–496, New York, NY, USA. ACM

- Diaz-Aviles, E., Stewart, A., Velasco, E., Denecke, K., and Nejdl, W. (2012e). Epidemic Intelligence For The Crowd By The Crowd. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012*.

- Diaz-Aviles, E. and Stewart, A. (2012). Tracking Twitter For Epidemic Intelligence. Case Study: Ehec/hus Outbreak In Germany 2011. In *Proceedings of the 4th ACM International Conference on Web Science*, WebSci '12.

- Diaz-Aviles, E., Orellana-Rodriguez, C., and Nejdl, W. (2012d). Taking The Pulse Of Political Emotions In Latin America Based On Social Web Streams. In *LA-WEB '12: Proceedings of the 2012 Latin American Web Conference*. IEEE Computer Society.

Additional published works towards the completion of this book include:

- Diaz-Aviles, E. and Kawase, R. (2012). Exploiting Twitter As A Social Channel For Human Computation. In *Proceedings of the First International Workshop on Crowdsourcing Web Search, Collocated with WWW'12. Lyon, France, April 17, 2012.*, CrowdSearch'12.

- Stewart, A., Diaz-Aviles, E., and Nanopoulos, A. (2011a). Self-supervised Detection Of Disease Reporting Events In Outbreak Reports. In *IEEE International Conference on Information Reuse and Integration (IRI'11). August 2011, Las Vegas, Nevada, USA*, pages 416–421.

- Diaz-Aviles, E., Siehndel, P., and Naini, K. D. (2011b). Exploiting Social #-tagging Behavior In Twitter For Information Filtering And Recommendation (microblog Track). In *Proceedings of The Twentieth Text REtrieval Conference, TREC 2011, Gaithersburg, Maryland, USA, November 15-18, 2011*, TREC

- Lage, R., Diaz-Aviles, E., Stewart, A., Dolog, P. (2011): Personalized Event-based Surveillance and Alerting Support for the Assessment of Risk. *International Meeting on Emerging Diseases and Surveillance (IMED'11), Vienna, Austria, February 4-7, 2011.*

- Denecke, K., Diaz-Aviles, E., Dolog, P., Eckmanns, T., Fisichella, M., Gomez-Lage, R., Linge, J., Smrz, P., Stewart, A. (2011). The Medical Ecosystem [M-Eco] Project: Personalized Event-based Surveillance. *International Meeting on Emerging Diseases. (IMED'11), Vienna, Austria. February 4-7, 2011.*

- Stewart, A., Diaz-Aviles, E., Nejdl, W., Marinho, L. B., Nanopoulos, A., and Schmidt-Thieme, L. (2009). Cross-tagging For Personalized Open Social Networking. In *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*, HT '09, pages 271–278, New York, NY, USA. ACM.

- Abel, F., Diaz-Aviles, E., Henze, N., Krause, D., and Siehndel, P. (2010). Analyzing The Blogosphere For Predicting The Success Of Music And Movie Products. *International Conference on Advances in Social Network Analysis and Mining (ASONAM'10)*, pages 276–280.

- Stewart, A., Diaz-Aviles, E., and Nejdl, W. (2008). Mining User Profiles To Support Structure And Explanation In Open Social Networking. *International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW 2008). Collocated with the 3rd International Conference on Semantic and Digital Media Technologies (SAMT 2008). Koblenz, Germany, Dec. 03 2008.*

2 Background and Preliminaries

In this chapter, we present background material describing key concepts that will provide the context and theoretical framework required for the rest of the book. In addition, we describe the datasets used in the experimental evaluations conducted in this work.

2.1 On Recommender Systems

The information overload problem facing today's Web users has made the decision-making task really challenging, and in many cases complex to the user, who has to face an overwhelming set of options that outstrips his capability to survey them and reach a decision. Automated recommender systems make product suggestions that are tailored to the human user's individual needs and represent powerful means to combat information saturation [Resnick and Varian, 1997].

One of the most successful technologies for recommender systems is *Collaborative Filtering* (CF) [Goldberg et al., 1992]. CF analyzes interdependencies among items as well as relationships and interactions between users to identify new user-item associations. It is domain independent and has the advantage to perform well in scenarios where there is not much content associated with items, or where the content is too difficult to collect or analyse. Besides, it can provide *serendipitous* recommendations, for example, suggesting items that are not particularly similar to a user's previous items.

Two main tasks of CF are the rating prediction (i.e., regression) and the item prediction task. In the rating prediction task, the recommender system aims to predict the user's rating for a new item, based on the past rating history. On the other hand, the task of item predic-

tion is to predict a user-specific ranking for a set of items, this is also referred to as a *Top-N* recommendation task.

Such recommender systems rely on different types on input. Users may explicitly express their interest in products by providing, for example, star ratings for a book in Amazon.com, or by selecting thumbs-up/down buttons in YouTube.com to indicate their preferences after watching a video. This *explicit feedback* is of high quality and most convenient, however is also scarce and not always available. A more abundant source of information for recommender systems is *implicit feedback*, which indirectly reflects users opinions and preferences. Some instances of implicit feedback include search and browsing history, eye movement and repeat buying. For example, a user that listens many songs by *U2* probably likes that band.

In the past few years, much research was devoted to the Netflix contest [Bell and Koren, 2007; Koren, 2008, 2009], many works were published addressing the rating prediction task focused on processing explicit feedback. *Top-N* recommendation is a harder problem than rating prediction, since the gathered information represents only positive feedback, e.g., by observing user behavior, we can infer which items he likes. However, it is hard to reliably infer which items he does not like. Thus, methods that optimize error metrics, such as RMSE or MAE, cannot directly be applied [Cremonesi et al., 2010].

Proposed approaches that address the *Top-N* task do exist [Hu et al., 2008; Rendle et al., 2009b; Cremonesi et al., 2010] but they do not consider the highly dynamic and time-evolving nature of social streams and concentrate on user-item interactions that have been aggregated over time.

We focus in this work on the *Top-N* recommendation task in domains where implicit feedback is derived from the evolution of recurring events, in particular, in the presence of social media data streams.

One of CF's most successful techniques are low dimensional linear factor models, that assume user preferences can be modeled by only a small number of *latent* factors [Koren et al., 2009], which we will discuss later in this section.

First we introduce some notation that will be useful in our setting. Let $U = \{u_1, \ldots, u_n\}$ and $I = \{i_1, \ldots, i_m\}$ be the sets of all users and all items, respectively. We reserve special indexing letters to distinguish users from items: for users u, v, and for items i, j. Suppose we have interactions between these two entities, and for some user $u \in U$ and item $i \in I$, we observe a *relational score* $x_{ui} \in \mathbb{X}$, where $\mathbb{X}$ is the set of discrete values associated to the relational score. For example, x_{ui} can be represented by an integer denoting: (i) an explicit rating scale, e.g., from one to five *stars*: $\mathbb{X} = \{1, \ldots, 5\}$ or a binary one, e.g., *thumbs-up/thumbs-down*: $\mathbb{X} = \{1, -1\}$; or (ii) implicit feedback captured from the user-item interactions, e.g., number of times that a user has listened to a particular song, e.g., $\mathbb{X} \subset \mathbb{N}$.

Thus, each instance of the data is a tuple (u, i, x_{ui}). For example in the movie recommendation case, the tuple might correspond to an explicit "rating" given by user u to movie i or, in the case of hashtag/topic recommendation, to a "weight" that is implicitly derived from user u's interaction patterns. Typical CF models organize these tuples into a sparse matrix $\mathbf{X}$ of size $|U| \times |I|$, using (u, i) as index and x_{ui} as entry value. The task of the recommender system is to estimate the score for the missing entries. We consider the relational scores as ordinal. Thus, we assume a total order between the possible score values. We distinguish predicted scores from the known ones, by using $\hat{x}_{ui}$, where $\hat{x}_{ui} \in \mathbb{X}$.

Finally, we denote by S the set of all observed user-item interactions, such that $S \subseteq U \times I \times \mathbb{X}$, with

$$(u, i, x_{ui}) \in S :\Leftrightarrow \text{the interaction between user } \textit{'u'} \text{ and item } \textit{'i'} \text{ is observed.}$$

For convenience, we also define for each user the set of all items with an observed score, denoted by B_u^+:

$$B_u^+ := \{i \in I \mid (u, i, x_{ui}) \in S\} .$$

As mentioned above, low dimensional linear factor modeling are popular collaborative filtering approaches [Koren et al., 2009]. These models consider that only a small number of *latent* factors can influence the preferences. Their prediction is a real number, $\hat{x}_{ui}$, per user item pair (u, i). Some of the most successful realizations of latent factor models are based on matrix factorization (MF). In its basic form, matrix factorization estimates a matrix $\hat{\mathbf{X}} : U \times I$ by the product of two low-rank matrices $\mathbf{W} : |U| \times k$ and $\mathbf{H} : |I| \times k$:

$$\hat{\mathbf{X}} := \mathbf{W}\mathbf{H}^{\intercal} \tag{2.1}$$

where k is a parameter corresponding to the rank of the approximation. Each row, w_u in W and h_i in H can be considered as a feature vector describing a user, u, and an item, i, correspondingly. Thus the final prediction is the linear combination of the factors:

$$\hat{x}_{ui} = \langle \mathbf{w}_u, \mathbf{h}_i \rangle = \sum_{f=1}^{k} w_{uf} \cdot h_{if}$$

Singular value decomposition (SVD) provides the best approximation of $\hat{X}$ to X with respect to the least squares optimization problem, but is prone to overfitting. Therefore, other matrix factorization techniques have been proposed, including regularized least square matrix factorization, maximum margin matrix factorization and non-negative matrix factorization [Koren et al., 2009].

2.1.1 Stochastic Gradient Descent for Matrix Factorization

Stochastic Gradient Descent (SGD) is an iterative stochastic optimization algorithm, that has performance advantages for large-scale problems, where the number of data points and the problem dimensionality are both very large [Bottou, 2010].

SGD finds the value $\theta^* \in \mathbb{R}^k (k >= 1)$ that minimizes a loss function $L(\theta)$ by estimating its gradient $\nabla L(\theta_t)$ at each iteration, based on a

single randomly picked example θ_t:

$$\theta_{t+1} = \theta_t - \eta_t \nabla L(\theta_t) \tag{2.2}$$

where t denotes the iteration number, η_t is the gain at iteration t, also known as *learning rate*, and $-\eta_t \nabla L(\theta_t)$ is a noisy approximation of the direction of steepest descent.

SGD convergence has been studied extensively in stochastic approximation theory, which establishes almost sure convergence under mild conditions [Bottou, 2010].

Since SGD does not need to remember which examples were seen during the previous iterations, it can process examples within stream data, on the fly.

SGD can be applied to learn a MF model by setting $\theta = (\mathbf{W}, \mathbf{H})$ and defining an application-dependent loss function L, that measures how well the learning process performs on each example. The general formulation including $L2$ regularization terms is:

$$\underset{\theta=(\mathbf{W},\mathbf{H})}{\operatorname{argmin}} L(\mathbf{X}, \mathbf{W}, \mathbf{H}) + \frac{\lambda_W}{2}||\mathbf{W}||_2^2 + \frac{\lambda_H}{2}||\mathbf{H}||_2^2 \ .$$

That is, our objective is to find $\mathbf{W}$ and $\mathbf{H}$ that lead to the smallest loss and has low model complexity, represented by the Frobenius norm of the low-rank matrices. We assumed that the aggregated loss $L(\mathbf{X}, \mathbf{W}, \mathbf{H})$ can be decomposed, as follows:

$$L(\mathbf{X}, \mathbf{W}, \mathbf{H}) = \frac{1}{|S|} \sum_{(u,i,x_{ui}) \in S} \ell(x_{ui}, \langle \mathbf{w}_u, \mathbf{h}_i \rangle) \ ,$$

where $\ell(x_{ui}, \hat{x}_{ui})$ is a loss function on a single example, defined on the observed relational score x_{ui} and the predicted one $\hat{x}_{ui}$.

Figure 2.1 shows a general alternating procedure applying SGD for MF. Note that at each iteration, the algorithm draws one single instance $(u, i, x_{ui}) \in S$ uniformly at random, and performs a local approximation of the gradient at this point.

SGD for MF

Input:

Training data S; Regularization parameters λ_W and λ_H; Learning rate η_0; Learning rate schedule α; Number of iterations T

Output: $\theta = (\mathbf{W}, \mathbf{H})$

1: initialize $\mathbf{W}_0$ and $\mathbf{H}_0$
2: **for** $t = 1$ **to** T **do**
3: $\quad (u, i, x_{ui}) \leftarrow \text{randomExample}(S)$
4: $\quad \mathbf{w}_u \leftarrow \mathbf{w}_u - \eta \frac{\partial}{\partial \mathbf{w}_u} \ell(x_{ui}, \langle \mathbf{w}_u, \mathbf{h}_i \rangle) - \eta \lambda_W \mathbf{w}_u$
5: $\quad \mathbf{h}_i \leftarrow \mathbf{h}_i - \eta \frac{\partial}{\partial \mathbf{h}_i} \ell(x_{ui}, \langle \mathbf{w}_u, \mathbf{h}_i \rangle) - \eta \lambda_H \mathbf{h}_i$
6: $\quad \eta = \alpha \cdot \eta$
7: **end for**
8: **return** $\theta_T = (\mathbf{W}_T, \mathbf{H}_T)$

Figure 2.1: **General stochastic gradient procedure to learn matrix factorization models with regularization parameter λ_θ, learning rate η, and learning rate schedule α. At each iteration, the algorithm draws one single instance $(u, i, x_{ui}) \in S$ uniformly at random, and performs a local approximation of the gradient at this point. Only the latent factors of user u and item i are updated.**

We do not need to update the entire matrices $\mathbf{W}$ and $\mathbf{H}$, but only the latent factors of user u and item i, i.e., $\mathbf{w}_u$ and $\mathbf{h}_i$.

Even though the *squared loss* has been successfully used for MF in the context of rating prediction (e.g., [Koren, 2008; Gemulla et al., 2011]) and item prediction [Hu et al., 2008], we are interested in a *ranking* approach to MF, and therefore require an *ordinal loss* to guide the factorization process.

In particular, we are interested in a *pairwise* approach, similar to the one used by RankSVM [Joachims, 2002], a popular ranking method in the field of learning to rank [Liu, 2011].

2.2 Learning to Rank for Information Retrieval

Although the most popular Information Retrieval (IR) model for document retrieval is still the vector space model [Baeza-Yates and Ribeiro-Neto, 2011], in recent years supervised learning-based methods have

been proposed to automatically learn an effective ranking model based on training data and then applied to unseen test data (e.g., [Joachims, 2002; Freund et al., 2003; Cao et al., 2007]). This task is referred to as "Learning To Rank for IR" in the field.

Learning to rank for Information Retrieval is a problem formalized as described next. In learning (training), a collection of queries and their corresponding retrieved documents are given. Furthermore, the labels (i.e., relevance judgements) of the document with respect to the queries are also provided. The relevance judgements, provided by human annotators, can represent ranks (e.g., categories in a total order). The objective of learning is to construct a ranking model, e.g., a ranking function, that achieves the best result on test data in the sense of optimization of a performance measure (e.g., error rate, classification accuracy, Mean Average Precision, etc.) [Baeza-Yates and Ribeiro-Neto, 2011].

In retrieval (test phase), given a query, the learned ranking function is applied, returning a ranked list of documents in descending order of their relevance scores. Suppose that $Q = \{q_1, \cdots, q_{|Q|}\}$ is the set of queries, and $D = \{d_1, \cdots, d_{|D|}\}$ the set of documents, the training set is created as a set of query-document pairs, $(q_i, d_j) \in Q \times D$, upon which a relevance judgement (e.g., a label) indicating the relationship between q_i and d_j is assigned by an annotator. Suppose that $Y = \{y_1, \cdots, y_{|Y|}\}$ is the set of labels and $y_{ij} \in Y$ denotes the label of query-document pair (q_i, d_j). A feature vector $\phi(q_i, d_j)$ is created from each query-document pair $(q_i, d_j), i = 1, 2, \cdots, |Q|; j = 1, 2, \cdots, |D|$. The training set is denoted as $T = \{(q_i, d_j), \phi(q_i, d_j), y_{ij}\}$. The ranking model is a real valued function of features:

$$f(q, d) = \vec{w} \cdot \phi(q, d) \tag{2.3}$$

where $\vec{w}$ denotes a weight vector corresponding to the ranking model to be learned from the data. In ranking for query q_i the model associates a score to each of the documents d_j as their degree of relevance with respect to query q_i using $f(q_i, d_j)$ and sorts the documents based on their scores. Table 2.1 gives a summary of notations described above.

Notations	Explanations
$Q := \{q_1, \cdots, q_{\|Q\|}\}$	Set of queries
$q_i \in Q$	Query
$D := \{d_1, \cdots, d_{\|D\|}\}$	Set of documents
$d_j \in D$	Document
$Y := \{y_1, \cdots, y_{\|Y\|}\}$	Set of relevance judgements
$y_{ij} \in Y$	Relevance judgement of query-document pair $(q_i, d_j) \in Q \times D$
$\phi(q_i, d_j)$	Feature vector w.r.t. (q_i, d_j)
$\phi_k(q_i, d_j)$	k^{th} dimension of $\phi(q_i, d_j)$
$T := \{((q_i, d_j), \phi(q_i, d_j), y_{ij}) \mid q_i \in Q \text{ and } d_j \in D \text{ and } y_{ij} \in Y\}$	Training set
$1 \leq i \leq \|Q\|$	
$1 \leq j \leq \|D\|$	

Table 2.1: **Summary of notations.**

2.3 Particle Swarm Optimization

The particle swarm optimization (PSO) algorithm is a population-based probabilistic optimization algorithm inspired by the social behavior of biological organisms, specifically the ability of groups of species of animals to work as a whole in locating desirable positions in a given area, e.g., bird flocking or fish schooling [Eberhart and Kennedy, 1995; Kennedy and Eberhart, 1995; Poli et al., 2007]. In a PSO algorithm, a population of agents called *particles* move through the solution space of an optimization problem, updating their velocity according to the information collected by the group called *swarm*. PSO algorithms are global non-linear optimization algorithms and do not require nor approximate gradients of the cost function.

In every iteration, each particle is attracted to the best solution that it has found individually, and toward the best solution that any particle in their *neighborhood* has found. In PSO, a neighborhood is defined for each individual particle as the subset of particles which it is able to communicate with. Neighborhoods are usually defined by graph topologies, e.g. $G = \{V_g, E_g\}$, where each vertex V_g corresponds to a particle in the swarm and the edges in E_g establish the neighbor relation between a pair of particles.

The algorithm updates the entire swarm at each time step by updating the velocity and position of each particle i in every dimension k as follows

$$v_{i,k}^{t+1} = \chi \cdot (v_{i,k}^{t} + c_1 \cdot \epsilon_1 \cdot (p_{i,k}^{t} - x_{i,k}^{t}) + c_2 \cdot \epsilon_2 \cdot (\ell_{i,k}^{t} - x_{i,k}^{t})) \tag{2.4}$$

and

$$x_{i,k}^{t+1} = x_{i,k}^{t} + v_{i,k}^{t+1} \tag{2.5}$$

where $v_{i,k}^{t}$ and $x_{i,k}^{t}$ are the particle's velocity and position at time step t respectively, $p_{i,k}^{t}$ is the particle's best position so far, $\ell_{i,k}^{t}$ is the best position found by the particle's neighbors; and ϵ_1 and ϵ_2 are uniformaly distributed random numbers between 0 and 1 generated at every update for each individual dimension k. The term $c_1 \cdot \epsilon_1 \cdot (p_{i,k}^{t} - x_{i,k}^{t})$ is associated with cognition since it takes into account the particle's own experience, and the term $c_2 \cdot \epsilon_2 \cdot (\ell_{i,k}^{t} - x_{i,k}^{t})$ is associated with social interaction between the particles. In view of this similarity, parameters c_1 and c_2 are known as *Cognitive Acceleration* and *Social Acceleration*, respectively. The *Constriction Coefficient* χ is used to avoid an "explosion" of the particle's velocity, and it is defined as follows [Poli et al., 2007; Bratton and Kennedy, 2007] :

$$\chi(\kappa, \varphi) = \frac{2\kappa}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \tag{2.6}$$

where $\kappa \in [0, 1]$, and $\varphi = c_1 + c_2 \geq 4$.

An in-depth introduction to PSO is given in [Poli et al., 2007; Bratton and Kennedy, 2007].

2.4 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [Blei et al., 2003] is a generative probabilistic model for collections of discrete data such as text corpora. The basic idea is that documents are represented as random mixtures

over latent topics, where each topic is characterized by a distribution over terms. More formally, assume that a text collection consists of a set of documents D. Furthermore, consider the set of topics Z, the distribution $P(z \mid d)$ over topics $z \in Z$ in a particular document $d \in D$ and the probability distribution $P(t \mid z)$ over terms $t \in T$ given topic $z \in Z$, where T is the set of terms. Each term $t_i \in T$ in a document (where the index refers to the ith term token) is generated by first sampling a topic from the topic distribution, then choosing a term from the topic-term distribution. We write $P(z_i = j)$ as the probability that the jth topic was sampled for the ith term token and $P(t_i \mid z_i = j)$ as the probability of term t_i under topic j. The model specifies the following distribution over terms within a document:

$$P(t_i \mid d) = \sum_{j=1}^{Z} P(t_i \mid z_i = j) P(z_i = j \mid d) \tag{2.7}$$

where $|Z|$ is the number of topics. $P(t \mid z = j)$ and $P(z \mid d)$ indicate which terms are important for which topic and which topics are important for a particular document, respectively.

In LDA the goal is to estimate the distribution topic-term $P(t \mid z)$ and the document-topic distribution $P(z \mid d)$, these distributions are sampled from Dirichlet distributions.

There are several methods developed for making inference in LDA such as variational expectation maximization [Blei et al., 2003], expectation propagation [Department et al., 2002], and Gibbs sampling [Griffiths and Steyvers, 2004].

The Gibbs Sampling algorithm, for example, considers each term token in the text collection in turn, and estimates the probability of assigning the current term token to each topic, conditioned on the topic assignments to all other term tokens. From this conditional distribution, a topic is sampled and stored as the new topic assignment for this term token.

This conditional distribution can be written as $P(z_i = j \mid t_i, d_i, z_{-i})$, and calculated by [Griffiths and Steyvers, 2004]:

$$P(z_i = j \mid t_i, d_i, z_{-i}) \propto \frac{C^{TZ}_{t_i j} + \beta}{\sum_t C^{TZ}_{tj} + |T|\beta} \frac{C^{DZ}_{d_i j} + \alpha}{\sum_z + |Z|\alpha} \tag{2.8}$$

where C^{ZT} and C^{DZ} are matrices of counts with dimensions $|T| \times |Z|$ and $|D| \times |Z|$ respectively; C^{ZT}_{tj} contains the number of times term t is assigned to topic j, not including the current instance i and C^{DZ}_{dj} contains the number of times topic j is assigned to some term token in document d, not including the current instance i. $z_i = j$ represents the topic assignment of token i to topic j, z_{-i} represents all topic-term and document-topic assignments except the current assignment z_i for term t_i, and α and β are the (symmetric) hyper-parameters for the Dirichlet priors. Based on the count matrices the posterior probabilities in Equation 2.7 can be estimated as follows:

$$P(t_i \mid z_i = j) = \frac{C^{TZ}_{t_i j} + \beta}{\sum_t C^{TZ}_{tj} + |T|\beta} \text{ and} \tag{2.9}$$

$$P(z_i = j \mid d) = \frac{C^{DZ}_{d_i j} + \alpha}{\sum_z + |Z|\alpha} \,. \tag{2.10}$$

LDA is an intensively studied model and its performance compares favorably to other known text information retrieval techniques, in addition to the large number of applications in this field, LDA has also been applied to several other problem scenarios, including entity resolution [Bhattacharya and Getoor, 2006], image processing [Kim et al., 2005; Li and Perona, 2005], fraud detection [Xing and Girolami, 2007], and many more.

2.5 Datasets

In this section, we provide a description of the data collection used in our empirical studies.

- ***476 million Twitter tweets*** dataset[1] [Yang and Leskovec, 2011] includes over 476 million Twitter posts from 20 million users, covering a 7 month period from June 1, 2009 to December 31, 2009. The number of hashtags present in the dataset is 49,293,684. It is estimated that this is about 20-30% of all public tweets published on Twitter during this particular time frame. We used this dataset in the empirical evaluations documented in Chapter 3.

- **Learning to Rank Dataset: LETOR**. This benchmark dataset is used in the experiments conducted in Chapter 4.

  - ***LETOR 2.0*** includes two datasets [Liu et al., 2007]: TD2003 and TD2004. The datasets are part of the topic distillation task of TREC 2003 and TREC 2004. TD2003 has 50 queries and TD2004 has 75 queries. The document collection is based on a January, 2002 crawl of the ".gov" domain. For each query, there are about 1,000 associated documents (webpages). Each query-document pair is given a binary judgement: *relevant* or *not relevant*.

    The features of LETOR TD2003 and TD2004 datasets include low-level content features such as term frequency (tf), inverse document frequency (idf), document length (dl) and their combinations (e.g., tf*idf) [Baeza-Yates and Ribeiro-Neto, 2011], as well as high-level content features as BM25 [Robertson, 1997] and LMIR [Zhai and Lafferty, 2004].

[1] `http://snap.stanford.edu/data`

Feature type	Feature name	References	Number of Features
Low-level	tf	[Baeza-Yates and Ribeiro-Neto, 2011]	4
Content	idf	[Baeza-Yates and Ribeiro-Neto, 2011]	4
	dl	[Baeza-Yates and Ribeiro-Neto, 2011]	4
	tfidf	[Baeza-Yates and Ribeiro-Neto, 2011]	4
High-level	BM25	[Robertson, 1997]	4
Content	LMIR	[Zhai and Lafferty, 2004]	9
Hyperlink	PageRank	[Page et al., 1999]	1
	Topical PageRank	[Nie et al., 2006]	1
	HITS	[Kleinberg, 1999]	2
	Topical HITS	[Nie et al., 2006]	2
	HostRank	[Xue et al., 2005]	1
Hybrid	Hyperlink-based relevance propagation	[Shakery and Zhai, 2003]	6
	Sitemap-based relevance propagation	[Qin et al., 2005]	2
			total: 44

Table 2.2: **Features extracted for TD2003 and TD2004.**

Hyperlink features are also part of the datasets: PageRank [Page et al., 1999], HITS [Kleinberg, 1999] and their variations (HostRank [Xue et al., 2005], topical PageRank and topics HITS [Nie et al., 2006]).

Hybrid features are also included and they correspond to those features holding both, content and hyperlink information, including "hyperlink-based relevance propagation" [Shakery and Zhai, 2003] and "sitemap-based relevance propagation" [Qin et al., 2005]. In total, there are 44 features extracted for each query-document pair. Table 2.2 lists all the features.

- ***LETOR 4.0*** uses the Gov2 web page collection (∼25M pages) and two query sets from Million Query track of TREC 2007 and TREC 2008, denoted as MQ2007 and MQ2008 for short. There are about 1700 queries in MQ2007 with labeled documents and about 800 queries in MQ2008 with labeled documents. Each query-document pair is given a binary judgment: *relevant* or *not relevant*.

The features of LETOR MQ2007 and MQ2008 datasets include low-level content features such as term frequency (tf), inverse document frequency (idf), document length (dl) and their combinations (e.g., tf*idf) [Baeza-Yates and Ribeiro-Neto, 2011], as well as high-level content features as BM25 [Robertson, 1997] and LMIR [Zhai and Lafferty, 2004]. Hyperlink features are also part of the datasets: PageRank [Page et al., 1999], HITS [Kleinberg, 1999].

Hybrid features are also included and they correspond to those features holding both, content and hyperlink information, including "hyperlink-based relevance propagation" [Shakery and Zhai, 2003] and "sitemap-based relevance propagation" [Qin et al., 2005]. In total, there are 46 features extracted for each query-document pair. Table 2.3 lists all the features.

Feature type	**Feature name**	**Category**	**Number of Features**
Low-level	tf	Q-D	5
Content	idf	D	5
	tf*idf	Q-D	5
	dl	D	5
High-level	BM25	Q-D	5
Content	LMIR	Q-D	15
Hyperlink	PageRank	D	1
Hybrid	Hyperlink-based	Q-D	5
			total: 46

Table 2.3: **Features extracted for MQ2007 and MQ2008. *Q* and *D* denote query and document features, respectively. Note: linear ranking functions cannot make use of the class-Q features, since these features are the same for all the documents under a query.**

- **Last.fm Dataset – 1K users**. This publicly available dataset obtained from *Last.fm*, a major social media Internet radio station, represents the whole listening habits, until May, 2009, for nearly 1,000 users [Celma, 2010]. We used this data collection in the experimental evaluation reported in Chapter 4.

- **BibSonomy Dataset ECML/PKDD 2009**. An empirical evaluation documented in Chapter 5 is based on the BibSonomy dataset from [Eisterlehner et al., 2009]. This dataset is almost a complete dump of BibSonomy, i.e., all users, resources (publication references and bookmarks) and tags publicly available until December 31^{st}, 2008. The dataset includes 3,617 users; 93,756 different tags; 378,378 resources; 1,401,104 tag assignments, and 1,401,104 posts.

- **OpenScout Dataset**. Additional experiments reported also in Chapter 5 make use of a dataset sampled from the OpenScout project collection [Niemann et al., 2010]. The project gathers metadata information from learning resources located at different learning content repositories. For our evaluation, we selected learning objects whose language is English and have at least five keywords added by their author. In total, 563 learning objects, 1692 unique keywords and 3150 keywords assignment were considered for the experiments.

- **Twitter M-Eco Dataset for EHEC/HUS Outbreak in Germany**. In the context of the European project M-Eco[2], we monitor over 500 diseases and symptoms on Twitter. During May and June 2011, we incrementally collected 7,710,231 tweets related to medical conditions, where 456,226 of them were related to the EHEC outbreak in Germany, and were produced by 54,381 distinct users. We used this data collection in the experiments conducted in Chapter 6.

- **Tweets and Blog posts for Emotion Detection**. We perform one of the studies reported in Chapter 6 on a collection of 165,484 documents, from them, 155,280 are 140-character Twitter messages or *tweets*, and 10,204 are snippets of weblog posts. The total number of documents was produced by 55,013 distinct users during the six-month period between 1st of October, 2011 and 1st of April, 2012. We chose this period of time because it allowed us to discuss and contrast our findings against an independent opinion poll published in April 2012. We discuss the details of the data collection process also in Chapter 6.

2.6 Evaluation Measures

In this section we define three evaluation measures widely used in information retrieval, namely, precision at position n ($P@n$), mean average precision (MAP), and normalized discount cumulative gain (NDCG). We use these measures in different experimental evaluations across the book. In some of the experiments we also used other metrics, which are defined in the corresponding chapters. The definitions are as follows.

[2]**Medical Ecosystem Personalized Event-Based Surveillance**
M-Eco: `meco-project.eu`

Precision at Position n ($P@n$)

Precision at n [Baeza-Yates and Ribeiro-Neto, 2011] measures the relevance of the top n documents in the ranking list with respect to a given query:

$$P@n = \frac{\#\text{ of relevant docs in top } n \text{ results}}{n} \tag{2.11}$$

Mean Average Precision (MAP)

The average precision (AP) [Baeza-Yates and Ribeiro-Neto, 2011] of a given query is calculated as Eq. (2.12), and corresponds to the average of $P@n$ values for all relevant documents:

$$AP = \frac{\sum_{n=1}^{N} (P@n * rel(n))}{\#\text{ of relevant docs for this query}} \tag{2.12}$$

where N is the number of retrieved documents, and $rel(n)$ is a binary function that evaluates to 1 if the n^{th} document is *relevant*, and 0 otherwise. Finally, MAP [Baeza-Yates and Ribeiro-Neto, 2011] is obtained by averaging the AP values over the set of queries.

Normalized Discount Cumulative Gain (NDCG)

For a single query, the NDCG [Järvelin and Kekäläinen, 2002] value of its ranking list at position n is computed by Eq. (2.13):

$$NDCG@n = Z_n \sum_{j=1}^{n} \frac{2^{r(j)} - 1}{\log(1 + j)} \tag{2.13}$$

where $r(j)$ is the rating of the the j-th document in the ranking list, and the normalization constant Z_n is chosen so that the perfect list gets NDCG score of 1.

For the LETOR datasets, we define two ratings $\{1, 0\}$ corresponding to "*relevant*" and "*not relevant*" in order to compute NDCG scores.

3 Online Collaborative Filtering

Users engaged in the Social Web increasingly rely upon continuous streams of Twitter messages (*tweets*) for real-time access to information and fresh knowledge about current affairs. However, given the deluge of tweets, it is a challenge for individuals to find relevant and appropriately ranked information. We propose to address this knowledge management problem by going beyond the general perspective of information finding in Twitter, that asks: "What is happening right now?", towards an individual user perspective, and ask: "What is interesting to *me* right now within the social media stream?". In this chapter, we consider collaborative filtering as an online ranking problem and present: (i) `RMFO`, a method that creates, in real-time, user-specific rankings for a set of tweets, based on individual preferences that are inferred from the user's past system interactions and (ii) `RMFX`, a novel approach that follows a selective sampling strategy to perform online model updates based on *active learning principles*, that closely simulates the task of identifying relevant items from a pool of mostly uninteresting ones. In particular, we focus on recommending personalized *hashtag*-topics to users. Our methods are based on matrix factorization, stochastic gradient descent, and reservoir sampling, three key components that make them particularly suitable for large scale applications. Experiments on the *476 million Twitter tweets* dataset show that our online approaches outperform recommendations based on Twitter's *global trend*, and they are also able to deliver a recommendation quality at the level of state-of-the-art matrix factorization techniques for Collaborative Filtering, such as Weighted Regularized Matrix Factorization (WRMF), much faster and more space efficient, demonstrating the efficacy of our approaches.

3.1 Introduction

Given a continuous stream of incoming tweets, arriving at a high rate, we are interested in the task of filtering and recommending topics to users that meet their particular information needs. We seek to go beyond the general perspective of "What is happening right now?" [Lin et al., 2011], a one-size-fits-all view, that only reflects the current *global* trend, which does not take into account individual preferences. Instead, we take an individual user perspective, and ask: "What is interesting to *me* right now?" In particular, we use hashtags as surrogates for topics, and learn, online, a personalized ranking model based on low-rank matrix factorization for collaborative prediction.

In this chapter, we focus on a fundamental subtask of CF, namely *item prediction*. The goal of item prediction is to produce a user-specific ranking for a set of items. In the absence of high quality *explicit feedback* (e.g., ratings), CF can infer user preferences about items using *implicit feedback*.

For example, in Twitter, if user *Alice* has been tagging her tweets lately with the hashtag *#Olympics2012*; and, so far, she has never used the hashtag *#fashion*, we can exploit this information, and use it as a good indicator for her up-to-date preferences. We can infer that currently, *Alice* is more interested in *Olympic Games* than, for instance, in *fashion*. Thus the task can be cast as that of recommending hashtags to users. This kind of data is generated continuously in the form of a stream of tweets as shown in Figure 3.1.

In this chapter, we present our online matrix factorization approaches: `RMFO` and `RMFX` for addressing these research challenges. We view matrix factorization in the light of a ranking problem, and use stochastic gradient descent (SGD) to optimize the SVM loss, or hinge-loss, by sampling the twitter stream. The low-rank matrix factorization model is therefore learned online on *pairwise* comparisons of sampled tweets containing hashtags. We recommend interesting topics to users, based on the latest state of the model, which evolves over time.

Our methods were extensively evaluated on a large-scale stream of tweets containing millions of tweets, hashtags and user interactions [Yang

Figure 3.1: **Illustration of how Pairwise Preferences can be inferred based on observed real-time interactions, for example for Alice we can say that she prefers both hashtags #Olympics2012 and #London, over #fashion and #cikm, which she has never used. In turn, based on the Alice's hashtag frequencies, we can also assume that she prefers #Olympics2012 over #London.**

and Leskovec, 2011]. To the best of our knowledge this work is the first empirical study demonstrating the viability of online collaborative filtering for Twitter.

To summarize, the main contributions of this work are:

1. We introduce a novel framework for online collaborative filtering based on a pairwise ranking approach for matrix factorization, in the presence of streaming data.

2. We propose `RMFO`, an online learning algorithm that is based on stochastic gradient descent. We explore different variations of the algorithm and show that it achieves state-of-the-art performance when recommending a short list of interesting and relevant topics to users from a continuous high volume stream of tweets, under the constraints of bounded space and time.

3. We propose `RMFX`, which represents an innovative personalized ranking perspective to matrix factorization for social media streams. The novelty of our approach lies in a selective sampling strategy to update the model based on personalized small buffers.

4. Personalized and unpersonalized offline learning to rank have been previously studied in the literature. This chapter proposes an innovative perspective to the problem, directed to social media streams and based on online learning and matrix factorization techniques.

5. Finally, this chapter provides an example of integrating large-scale collaborative filtering with the real-time nature of Twitter.

3.2 Online CF for Social Streams

In the presence of a continuous stream of incoming tweets, arriving at a high rate, our objective is to process the incoming data in bounded space and time, and recommend a short list of interesting topics that meet users' individual taste.

The high rate makes it harder to: (i) capture the information transmitted; (ii) compute sophisticated models on large pieces of the input; and (iii) store the amount of input data, which we consider significantly larger than the memory available to the algorithm.

This problem setting fits a streaming model of computation by Muthukrishnan [Muthukrishnan, 2005], which establishes that by imposing a space restriction on algorithms that process streaming data, we may not be able to store all the data we see. The impact is that the data generated in real-time carries high-dimensional information which is difficult to extract and process.

Consider, for example, an epidemiologist monitoring Twitter using CF techniques to enhance his capabilities for epidemic detection and control. Social media data has to be processed in real-time, additionally, the surveillance models must be updated online, otherwise the timeliness required for such a critical system will be heavily impacted. Any time lag in modeling the data could render its outcome obsolete and useless. In the other word, an online CF algorithm should quickly learn the best Top-N recommendations based on real-time user interactions and prevent repeatedly suggesting highly relevant, but old information.

In our scenario, we assume that topics of interest are captured by the hash-tagging behavior in Twitter. Hashtags are words or phrases prefixed with the symbol #, e.g., *#eurovision*, a form of metadata tag used to mark keywords or topics in a tweet. Hashtags were created by Twitter users as a way to categorize messages, the practice is now a Twitter standard. Any user can categorize or follow topics with the hashtags service.

For example, a tweet about health might be tagged with hashtag *#health*:

> An hour of Tai Chi a day can help people with Parkinson's disease to walk - http://t.co/LsO5ao5X #taichi #martialarts #**health**

Hashtags evolve over time, reflecting the dynamics of user preferences in the social stream. Our approach seeks to incorporate these dynamics to produce a short list of interesting recommendations based on a matrix factorization model for CF, which is learned online. In this section, we formally define the problem of matrix factorization for collaborative filtering in presence of streaming data.

3.2.1 Problem Definition

We focus on learning a matrix factorization model for collaborative filtering in presence of streaming data. To this end, we will follow a pairwise approach to minimize an ordinal loss. Our formalization extends the work of Sculley [Sculley, 2010] for unpersonalized learning to rank, to an online collaborative filtering setting.

Remember from Chapter 2 that we denote by S the set of all observed user-item interactions, such that $S \subseteq U \times I \times \mathbb{X}$, with

$$(u, i, x_{ui}) \in S :\Leftrightarrow \text{the interaction between user } \textit{`u'} \text{ and item } \textit{`i'} \text{ is observed.}$$

We also define for each user the set of all items with an observed score, denoted by B_u^+:

$$B_u^+ := \{i \in I \mid (u, i, x_{ui}) \in S\} .$$

With slight abuse of notation, we also use S to represent the input stream $s_1, s_2, \ldots$ that arrives sequentially, instance by instance. Let $p_t = ((u, i), (u, j))_t$ denote a pair of training instances sampled at time t, where $(u, i, x_{ui}) \in S$ has been observed in the stream and $(u, j, x_{uj}) \notin S$ has not.

Formally, we define the set P as the set of tuples $p = ((u, i), (u, j))$ selected from the data stream S, as follows: $P := \{((u, i), (u, j)) \mid i \in B_u^+ \ \wedge \ j \notin B_u^+\}$.

We require pairs that create a *contrast* in the preferences for a given user u over items i and j. Since we are dealing with implicit, positive only feedback data (i.e. the user never explicitly states a negative preference for an item) we follow the rationale from Rendle et al. [Rendle et al., 2009b] and assume that user u prefers item i over item j. We will restrict the study to a binary set of preferences $x_{ui} = \{+1, -1\}$, e.g., *observed* and *not-observed*, represented numerically with $+1$ and -1, respectively. For example, if a user u in Twitter posts a message containing hashtag i, then we consider it as a positive feedback and assign a score $x_{ui} = +1$. More formally, $x_{ui} = +1 \iff i \in B_u^+$. In future work we plan to explore how repeated feedback can be exploited to establish a total order for items in B_u^+.

With P defined, we find $\theta = (\mathbf{W}, \mathbf{H})$ that minimizes the pairwise objective function:

$$\underset{\theta=(\mathbf{W},\mathbf{H})}{\operatorname{argmin}} L(P, \mathbf{W}, \mathbf{H}) + \frac{\lambda_W}{2}||\mathbf{W}||_2^2 + \frac{\lambda_H}{2}||\mathbf{H}||_2^2 \ . \tag{3.1}$$

In this chapter, we explore the use of the SVM loss, or *hinge-loss*, used by RankSVM for the learning to rank task [Joachims, 2002]. Given the predicted scores $\hat{x}_{ui}$ and $\hat{x}_{uj}$, the ranking task is reduced to a pairwise classification task by checking whether the model is able to correctly

rank a pair $p \in P$ or not. Thus, $L(P, \mathbf{W}, \mathbf{H})$ is defined as follows:

$$L(P, \mathbf{W}, \mathbf{H}) = \frac{1}{|P|} \sum_{p \in P} \hbar(y_{uij} \cdot \langle \mathbf{w}_u, \mathbf{h}_i - \mathbf{h}_j \rangle) \quad , \tag{3.2}$$

where $\hbar(z) = max(0, 1 - z)$ is the hinge-loss; $y_{uij} = sign(x_{ui} - x_{uj})$ is the $sign(z)$ function, which returns $+1$ if $z > 0$, i.e., $x_{ui} > x_{uj}$, and -1 if $z < 0$. The prediction function $\langle \mathbf{w}_u, \mathbf{h}_i - \mathbf{h}_j \rangle = \langle \mathbf{w}_u, \mathbf{h}_i \rangle - \langle \mathbf{w}_u, \mathbf{h}_j \rangle$ corresponds to the difference of predictor values $\hat{x}_{ui} - \hat{x}_{uj}$.

Please note that in this special case of binary rank values of *observed* and *not-observed*, the optimization problem defined by Eq. (3.2) is equivalent to the problem of optimizing area under the ROC curve (AUC) for binary-class data [Sculley, 2010]. Other convex loss functions can also be applied, e.g., *squared* or *logistic* loss [Rendle et al., 2009b; Sculley, 2010], as well as any prediction function besides the dot product $\langle \cdot, \cdot \rangle$ [Rendle and Schmidt-Thieme, 2008].

To conclude this section, we compute the gradient of the pairwise loss at instance $p_t \in P$ with non-zero loss, and model parameters $\theta_t = (\mathbf{w}_u, \mathbf{h}_i, \mathbf{h}_j)$, as follows:

$$-\nabla \hbar(p_t, \theta_t) = \begin{cases} y_{uij} \cdot (\mathbf{h}_i - \mathbf{h}_j) & \text{if } \theta_t = \mathbf{w}_u, \\ y_{uij} \cdot \mathbf{w}_u & \text{if } \theta_t = \mathbf{h}_i, \\ y_{uij} \cdot (-\mathbf{w}_u) & \text{if } \theta_t = \mathbf{h}_j, \\ 0 & \text{otherwise.} \end{cases}$$

3.3 RMFO: Online Matrix Factorization for CF

In this section, we introduce our approach for personalized ranking through online matrix factorization in the presence of stream data: `RMFO`. Furthermore, we present three variations of the online algorithm namely: *Single Pass*, *User Buffer*, and *Reservoir Sampling*. The overall steps of our approach are shown in Figure 3.2.

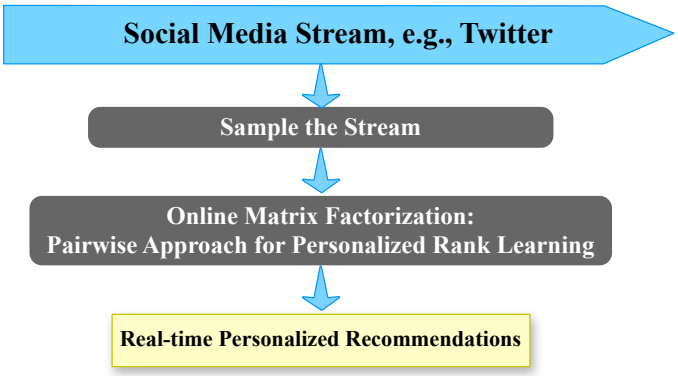


Figure 3.2: **Main steps of our approach for personalized ranking through online matrix factorization – `RMFO`.**

Our goal is to develop an algorithm to efficiently optimize the objective function (3.1). Based on the stochastic gradient descent concepts [Bottou, 2010], we present the framework of our algorithm in Figure 3.3. The main components of this framework are: (i) a sampling procedure done on the streaming data, and (ii) a model update based on the sample.

Framework for Online CF

Input: Stream representative sample at time t: S_t; Regularization parameters λ_W, λ_{H^+}, and λ_{H^-}; Learning rate η_0; Learning rate schedule α; Number of iterations T_S and T_θ; Parameter c to control how often to perform the model updates

Output: $\theta = (\mathbf{W}, \mathbf{H})$

initialize $\mathbf{W}_0$ and $\mathbf{H}_0$
initialize sample stream $S' \leftarrow \emptyset$
counter $\leftarrow 0$
for $t = 1$ **to** T_S **do**
    $S' \leftarrow$ **updateSample**(S_t)
    counter $\leftarrow$ counter $+ 1$
    if $c =$ counter **then**
        $\theta \leftarrow$ **updateModel**$(S', \lambda_W, \lambda_{H^+}, \lambda_{H^-}, \eta_0, \alpha, T_\theta)$
        counter $\leftarrow 0$
    end if
end for
return $\theta_T = (\mathbf{W}_T, \mathbf{H}_T)$

Figure 3.3: **Framework for Online CF.**

The model update procedure performed by RMFO is shown in Figure 3.4, which includes three regularization constants: λ_W, λ_{H^+}, and λ_{H^-}, one for the user factors, the other two for the positive and negative item factors updates. Moreover, we include a learning rate η and a learning rate *schedule* α that adjusts the step size of the updates at each iteration.

RMFO Model Update based on SGD for MF

Input: Stream representative sub-sample at time t: S'; Regularization parameters λ_W, λ_{H^+}, and λ_{H^-}; Learning rate η_0; Learning rate schedule α; Number of iterations T_θ

Output: $\theta = (\mathbf{W}, \mathbf{H})$

procedure updateModel($S', \lambda_W, \lambda_{H^+}, \lambda_{H^-}, \eta_0, \alpha, T_\theta$)
  $\eta \leftarrow \eta_0$
  for $t = 1$ **to** T_θ **do**
    $((u, i), (u, j)) \leftarrow \text{randomPair}(S') \in P$
    $y_{uij} \leftarrow sign(x_{ui} - x_{uj})$
    $\mathbf{w}_u \leftarrow \mathbf{w}_u + \eta\, y_{uij}\, (\mathbf{h}_i - \mathbf{h}_j) - \eta\, \lambda_W\, \mathbf{w}_u$
    $\mathbf{h}_i \leftarrow \mathbf{h}_i + \eta\, y_{uij}\, \mathbf{w}_u - \eta\, \lambda_{H^+}\, \mathbf{h}_i$
    $\mathbf{h}_j \leftarrow \mathbf{h}_j + \eta\, y_{uij}\, (-\mathbf{w}_u) - \eta\, \lambda_{H^-}\, \mathbf{h}_j$
    $\eta \leftarrow \alpha \cdot \eta$
  end for
  return $\theta = (\mathbf{W}_{T_\theta}, \mathbf{H}_{T_\theta})$
end procedure

Figure 3.4: RMFO Model Update.

In the rest of the section we explore three variations of our online algorithm based on how the sampling is performed.

3.3.1 Sampling Techniques for Twitter Stream

In this work, we explore the following three variations of our approach based on different stream sampling techniques:

(1) Single Pass (RMFO-SP) takes a single pair from the stream and performs an update of the model at every iteration. This approach does not "remember" previously seen instances. That is, we sample a

pair $p_t \in P$ at iteration t, and execute procedure *updateModel*(p_t, λ_W, λ_{H^+}, λ_{H^-}, η_0, α, $T_\theta = 1$) (Figure 3.4).

(2) User Buffer (`RMFO-UB`) retains the most recent b instances per user in the system. In this way, we retain certain amount of history so that the algorithm will run in constant space. For each user, we restrict the maximum number of her items to be kept and denote it by b.

More precisely, after receiving the training instance $(u, i, x_{ui})_t$ at time t, the user buffer $|B_u^+|$ for u, is updated as follows:

if $|B_u^+| < b$ **then** $B_u^+ \cup \{i\}$
else
    Delete the oldest instance from B_u^+
    $B_u^+ \cup \{i\}$
end if

We update the model selecting pairs, $p_t \in P$, from the candidate pairs implied by the collection of all user buffers B, which is defined by the function $B := u \to B_u^+$.

(3) Reservoir Sampling (`RMFO-RSV`) involves retaining a fixed size of observed instances in a *reservoir*. The reservoir should capture an accurate "sketch" of history under the constraint of fixed space. The technique of random sampling with a reservoir [Vitter, 1985] is widely used in data streaming, and recently has been also proposed for online AUC maximization in the context of binary classification [Zhao et al., 2011]. We represent the reservoir as a list $R := [s_1, s_2 \ldots, s_{|R|}]$ that "remembers" $|R|$ random instances from stream S. Instances can occur more than once in the reservoir, reflecting the distribution of the observed data. We note that this approach also bounds the space available for the algorithm, but in contrast to the user buffer technique, we do not restrict the space per user, but instead randomly choose $|R|$ samples from the stream and update the model using this history.

3.3.2 Computational Complexity

The computational complexity of the `RMFO` top-N recommendation algorithm depends on the amount of time required to build the model

(i.e., $\mathbf{W}$ and $\mathbf{H}$), and the amount required to compute the recommendation using this model.

During the model building phase, at every iteration t, we need to update the factors for user u and items i and j, following the gradient steps, the upper bound on the complexity of this step is $O(k)$, where k is the number of factors of the approximation. In general, the complexity of *processing* the whole stream is given by $O(|S|)$, in practice, on the other hand, the model achieves a good performace with a fraction of observed interactions, i.e., $O(T_S\ T_\theta)$, with $(T_S\ T_\theta) << |S|$, where T_S corresponds to the *sampling rate*, equivalent to the number of *epochs* or iterations over the observed data at a given time, and T_θ to the number of model updates per iteration. In total, the building phase requires $O(T_S\ T_\theta\ k)$.

The space required by `RMFO-SP` is $O(1)$, since we need to store a single instance to perform the model update. For `RMFO-UB`, the space required is $O(|U|\ b)$, that is, the number of users times the items to be kept per user. In the case of `RMFO-RSV`, the space required is $O(|R|)$, where $|R|$ is the size of the reservoir. It is clear that all models require to store the model parameters, which takes $O(|\mathbf{W}| + |\mathbf{H}|)$.

Finally, the time required to compute the top-N recommendations for a user u is $O(|I|\ k^2 + |I|\ \log(|I|))$ in the worse case, that is, we need to compute the personalized score for each item by multiplying the user factors $\mathbf{w}_u$ by the item factor matrix $\mathbf{H}$, this step takes $O(|I|\ k^2)$, then sort the items based on the computed scores, which takes $O(|I|\ \log(|I|))$, and finally select the top-N items that the user has not consumed in the past as the recommendation list. In practice, a subset of the items I' (with $|I'| << |I|$) can be used to compute the recommendations, for instance a set of topics, books or movies in a specific genre or category depending on the user's context. Our experimental evaluation reflects this idea.

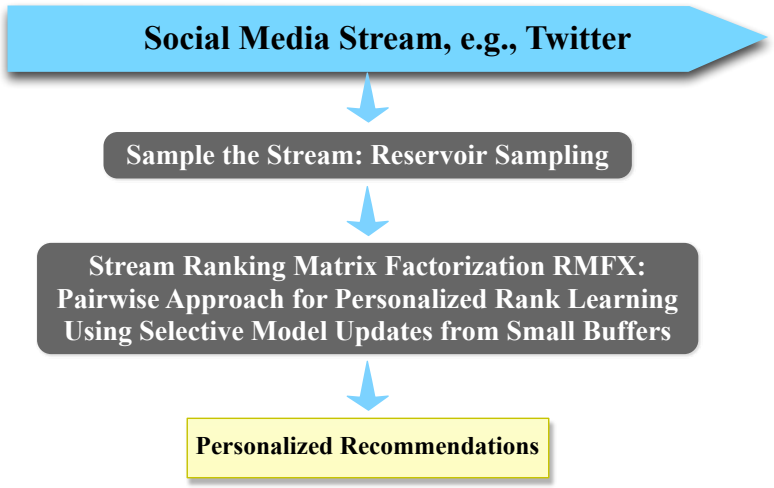


Figure 3.5: **Main steps of `RMFX` for online CF with selective model updates.**

3.4 RMFX: Online CF with Selective Model Updates

In this section, we introduce our approach `RMFX`, an innovative personalized ranking perspective to matrix factorization for social media streams. `RMFX` leverages the ideas developed in the previous section for `RMFO`, and extend them to include a selective sampling strategy, which updates the model based on personalized small buffers following *active learning* principles. We present our model in steps discussing the rationale behind them, such steps are illustrated in Figure 3.5.

The random sampling with a reservoir allows us to retain a fixed size of observed instances, bounding the space available for the algorithm to a set of $|R|$ randomly chosen samples from the stream and update the model using this history. Although simply updates of the model based on the reservoir may yield better results than single online updates, it is still far from the accuracy achieved by the offline cases. On top of that, in the reservoir we store only user and item pairs observed in the stream, and the question of how to sample the pairs needed for creating the contrasts P still remains.

In order to address this drawback, we need to exploit as much information as possible from the sampled tweets in the reservoir. In particular we propose to perform model updates and retraining on the

most informative examples present in the reservoir, then, the question is how to select such examples from this sketch of the stream. This scenario is similar to the one of active learning, where the system asks the user to evaluate a minimum set of items which will contribute the most to learning his/her preferences (e.g., [Karimi et al., 2011]).

Consider the case of binary classification using Support Vector Machines (SVM). SVM attempt to find a hyperplane that divides the two classes with the largest margin. From the theoretical foundations of SVM we know that only the support vectors have an effect on the solution. The support vectors are the points that lie closest to the hyperplane, therefore the *most informative* training points, and the goal of training is to discover them [Vapnik, 1995].

Usually, the training set is chosen to be a random sampling of instances, for example the tweets in our reservoir. However, in many cases principled criteria can be used to sample the training data with the goal to reduce its need for large quantities of labeled data.

Our scenario of dyadic data, i.e., user-item interactions, differs from the one of SVM in two fundamental ways: (i) since we are learning personalized rankings, there are as many hyperplanes as users, unlike an SVM, (ii) we are not just learning a hyperplane per user, but simultaneously also the item feature vectors, in contrast to SVM where the values of the features vectors, defining the training points, are known and given in advance.

Moreover, remember that we are concerned with learning personalized rankings from pairwise comparisons, hence the most informative instances are the ones that have opposite labels but are close to each other in the ranking induced by the user's hyperplane, intuitively they are more difficult to order than the ones away from each other in the ranking [Yu, 2005]. Figure 3.6 illustrates how user u's feature vector $\mathbf{w}_u$ induces a particular (personalized) ranking at a given iteration in a two dimensional example[1]. $\mathbf{w}_u$ determines the ordering of four item points. For any user *weight* vector $\mathbf{w}_u$, the items are ordered by the

[1]Observe that Figure 3.6 can be regarded as a *personalized* adaptation of Figure 2 in [Joachims, 2002].

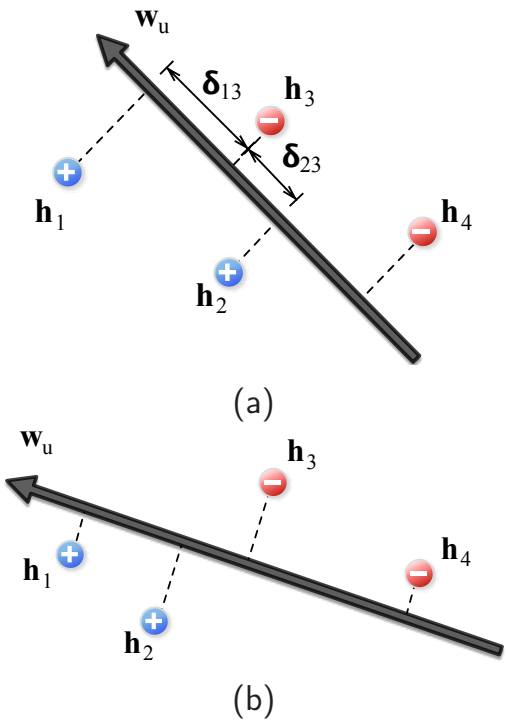


Figure 3.6: **Example of how a user *weight* vector $\mathbf{w}_u$ ranks four item points. (a) The vector $\mathbf{w}_u$ ranks the points as ($\mathbf{h}_1$, $\mathbf{h}_3$, $\mathbf{h}_2$, $\mathbf{h}_4$), erroneously ranking $\mathbf{h}_3$ higher than $\mathbf{h}_2$. (b) The model updates vector $\mathbf{w}_u$ (user features) and the item features iteratively based on pairwise differences and learns the correct ordering ($\mathbf{h}_1$, $\mathbf{h}_2$, $\mathbf{h}_3$, $\mathbf{h}_4$). In this example, the pair ($\mathbf{h}_2$, $\mathbf{h}_3$) with δ_{23} is considered more informative than ($\mathbf{h}_1$, $\mathbf{h}_2$) with δ_{13}, since $|\delta_{23}| < |\delta_{13}|$, i.e., the smaller the δ, the more informative the instances are for training the model.**

projection onto $\mathbf{w}_u$, or equivalently, by their signed distance to a hyperplane with normal vector $\mathbf{w}_u$. The items in the figure are ordered ($\mathbf{h}_1$, $\mathbf{h}_3$, $\mathbf{h}_2$, $\mathbf{h}_4$). We denote as δ the distance between two projections of data points with different labels on the induced ranking. The smaller the δ, the more informative the instances are for training the model. More formally, for a given user vector $\mathbf{w}_u$, and two items points $\mathbf{h}_i$ and $\mathbf{h}_j$, δ_{ij} can be computed as follows:

$$\delta_{ij} = \left| \frac{\langle \mathbf{w}_u, \mathbf{h}_i - \mathbf{h}_j \rangle}{\|\mathbf{w}_u\|} \right| .$$

Finally, to answer the question of how to select such examples from the reservoir, we will use an active learning inspired approach. In classical active learning [Tong and Koller, 2002], the search for the most informative instance is performed over the entire dataset, which involves the recomputation of each training example's distance to the new hyperplane. This process is prohibitively expensive for large datasets or unbounded data streams. Therefore, we propose a selection method based on the "59 trick" [Smola and Schölkopf, 2000; Ertekin et al., 2007], that establishes that randomly sampling only 59 instances, regardless the training set size, is enough to guarantee with 95% probability, that one of them is among the top 5% closest instances to the hyperplane. This approach also simulates the real world scenario of given a pool of items, ranking the positive ones higher than the negatives, modeled into the recommender system evaluation protocol proposed in [Cremonesi et al., 2010].

At each iteration, we select at random a user-item (u, i) interaction from the reservoir, which represents a positive feedback observation. Next, we construct a small buffer for user u by sampling 59 negative items j's, creating the required *contrast* in the preferences for user u over items i and j's. The user buffer contains exactly 59 pairs of the form $p_b = ((u, i), (u, j_b))$, $b = 1 \ldots 59$. Then, we compute the values δ_{uijb} between the projections on $\mathbf{w}_u$ of each instance in the pair p_b. Finally, we sample a pair p^* with probability proportional to its *informativeness*, which is given by $1/\delta_{uijb}$, and use p^* to perform the matrix factorization model updates. This procedure is shown in Figure 3.7, which includes three regularization constants: λ_W, λ_{H^+}, and λ_{H^-}, one for the user factors, the other two for the positive and negative item factors updates. Moreover, we include a learning rate, and a learning rate *schedule* α that adjusts the step size of the updates at each iteration.

RMFX Model Update based on SGD for MF using active learning with small buffers

Input:

Reservoir representing a sample of the stream at time t: R; Regularization parameters λ_W, λ_{H^+}, and λ_{H^-}; Learning rate η_0; Learning rate schedule α; Number of iterations T_θ.

Output: $\theta = (\mathbf{W}, \mathbf{H})$

procedure updateModel($R, \lambda_W, \lambda_{H^+}, \lambda_{H^-}, \eta_0, \alpha, T_\theta$)
  for $t = 1$ **to** T_θ **do**
    Select a user-item pair (u, i) from R uniformly at random
    Construct a small buffer for user u by sampling 59 negative items j's from R ("59 trick" [Smola and Schölkopf, 2000; Ertekin et al., 2007])
    Compute the distances δ_{uijb} for each pair $p_b = ((u, i), (u, j_b)) \in P$, $b = 1 \dots 59$ in the small buffer
    Sample a pair $p^* = ((u, i), (u, j))$ from the buffer with probability proportional to its *informativeness:* $1/\delta_{uijb}$
  // Perform the model updates as follows:
    $y_{uij} \leftarrow sign(x_{ui} - x_{uj})$
    $\mathbf{w}_u \leftarrow \mathbf{w}_u + \eta\ y_{uij}\ (\mathbf{h}_i - \mathbf{h}_j) - \eta\ \lambda_W\ \mathbf{w}_u$
    $\mathbf{h}_i \leftarrow \mathbf{h}_i + \eta\ y_{uij}\ \mathbf{w}_u - \eta\ \lambda_{H^+}\ \mathbf{h}_i$
    $\mathbf{h}_j \leftarrow \mathbf{h}_j + \eta\ y_{uij}\ (-\mathbf{w}_u) - \eta\ \lambda_{H^-}\ \mathbf{h}_j$
    $\eta = \alpha \cdot \eta$
  end for
  return $\theta = (\mathbf{W}_{T_\theta}, \mathbf{H}_{T_\theta})$
end procedure

Figure 3.7: **Matrix factorization model update based on SGD using personalized active learning with small buffers. The procedure minimizes the SVM loss, or hinge-loss, following a pairwise learning to rank approach for dyadic data.**

3.4.1 Computational Complexity

From Section 3.3.2, we know that the complexity of `RMFO-RSV` is $O(T_S\ T_\theta\ k)$, where T_S, as defined in Section 3.3.2, corresponds to the *sampling rate* or number of *epochs* over the observed data at a given time, and T_θ to the number of model updates per epoch. Note that in our setting, we set T_θ to be proportional to the size of the reservoir $|R|$.

In the case of RMFX Top-N recommendation algorithm, for each iteration, we need to perform an additional computation corresponding to the selective sampling step, which requires to compute the score of b pairs in the buffer (in our setting $b = 59$). For the multiplication of the scores we need to multiply the user vector, $\mathbf{w}_u$, by the item vectors $\mathbf{h}_j$ in the small buffer of size b. The time complexity is given then by $O(T_S\ T_\theta\ (b\ k + k)) = O(T_S\ T_\theta\ b\ k)$.

The space required by RMFX is $O(|R|)$, that is, proportional to the size of the reservoir.

Even though the time per iteration required by RMFX is b times higher that the one of RMFO-RSV, RMFX is expected to converge faster, since the model updates are performed on the most informative samples, requiring less number of iterations for a good recommendation performance, i.e., $(T_S\ T_\theta)_{\texttt{RMFX}} < (T_S\ T_\theta)_{\texttt{RMFO-RSV}}$, as shown in the experimental evaluation discussed in the following section.

3.5 Experimental Study

In this section, we demonstrate our approach by analyzing real-world data consisting of millions of tweets. The dataset used in our experiments corresponds to the *476 million Twitter tweets* as described in Section 2.5. For our evaluation, we computed a 5-core of the dataset, i.e., every user has used at least 5 different hashtags, and every hashtag has been used at least by 5 different users. The statistics of the 5-core are shown in Table 3.1.

Tweets	Users	Hashtags (items)
35,350,508	413,987	37,297

Table 3.1: ***476 million Twitter tweets*** **Dataset Statistics (5-core).**

3.5.1 Evaluation Methodology

Evaluation of a recommender in the presence of stream data requires a time sensitive split. Hence, we split the dataset S into two sets: a training set S_{train} and a testing set S_{test}. Consider we make the split at time t_{split}, then we put into S_{train} the individual training examples (tweets) with timestamps less than t_{split}. Into S_{test}, we put the user rankings with timestamps greater than t_{split}. The recommenders are trained on S_{train} and then their performance is measured on S_{test}. Note that given the dynamics in Twitter, there might be users in S_{train} not present in S_{test}.

To evaluate the recommenders we used a variant of the *all-but-1* protocol, also known as the leave-one-out holdout method. In particular, we follow a similar schema as the one described in [Cremonesi et al., 2010].

Our goal is to evaluate the system performance when it suggests *Top-N* topics to a user. For example, recommending the user a few specific hashtags which are supposed to be the most attractive to him. That is, to find the relative position of these interesting items within the total order of items ranked for a specific user.

- To this end, for each user $u \in |U_{test}|$ we aggregate his rankings in the test set S_{test} by accumulating the item frequencies across those rankings in order to produce a single total ranking. The items are again sorted in descending order of their accumulated frequencies.

  We take one item i at random from the top-10 of the aggregated ranking and hide it. The goal of a recommender system is to help users to discover new items of interest, therefore we impose the additional restriction that the hidden item has to be *novel* for the user, and therefore we remove from the training set all occurrences of the pair (u, i). In total, we have $|U_{test}| = 260,246$ hidden items.

- Then, for each hidden item i, we randomly select 1000 additional items from the test set S_{test}. Notice that most of those items selected are probably not interesting to user u.

- We predict the scores for the hidden item i and for the additional 1000 items, forming a ranking by ordering the 1001 items according to their scores. The best expected result is that the interesting item i_u to user u will precede the rest 1000 random items.

- Finally, we generate a *Top-N* recommendation list by selecting the N items with the highest score. If the test item i_u is in the *Top-N*, then we have a *hit*, otherwise we have a *miss*. We measure the quality by looking at the *recall* metric.

3.5.2 Evaluation Metric: Recall

Traditionally, collaborative filtering algorithms are evaluated by the accuracy of their predicted ratings. One commonly used performance metric for rating accuracy is the Mean Absolute Error (MAE).

However, we are interested in measuring Top-N recommendation performance and not in rating prediction. Therefore, we measure the quality by looking at the *recall* metric, also known as *hit rate*, which is widely used for evaluating Top-N recommender systems (e.g., [Deshpande and Karypis, 2004; Cremonesi et al., 2010]).

In our recommender systems setting, the *recall* metric is defined as follows:

$$\textbf{recall} := \frac{\sum_{u \in U_{test}} \mathbb{1}_{[i_u \in \text{Top-N}_u]}}{|U_{test}|} \quad , \tag{3.3}$$

where $\mathbb{1}_{[z]}$ is the indicator function that returns 1 if condition z holds, and 0 otherwise. A recall value of 1.0 indicates that the system was able to always recommend the hidden item, whereas a recall of 0.0 indicates that the system was not able to recommend any of the hidden items. Since the precision is forced by taking into account only a restricted number N of recommendations, there is no need to evaluate *precision* or *F1* measures, i.e., for this kind of scenario, precision is just the same as recall up to a multiplicative constant.

3.5.3 Experiment I: RMFO Evaluation

We implemented the three variations of our model `RMFO-SP`, `RMFO-UB` and `RMFO-SRV`, and evaluated them against two other competing models:

(1) Trending Topics (TT). This model sorts all hashtags based on their popularity, so that the top recommended hashtags are the most popular ones, which represent the trending topics overall. This naive baseline is surprisingly powerful, as crowds tend to heavily concentrate on few of the many thousands available topics in a given time frame. We evaluate the TT from the whole training set and the ones from the last four weeks before the evaluation.

(2) Weighted Regularized Matrix Factorization (WRMF). This is a state-of-the-art matrix factorization model for item prediction introduced by Hu et al. [Hu et al., 2008]. WRMF is formulated as a regularized Least-Squares problem, in which a weighting matrix is used to differentiate the contributions from observed interactions (i.e., positive feedback) and unobserved ones. WRMF outperforms neighborhood based (item-item) models in the task of item prediction for implicit feedback datasets, and therefore is considered as a more robust contender. Please note that this reference model is computed in *batch mode*, i.e., assuming that the *whole stream* is stored and available for training. WRMF setup is as follows: $\lambda_{\text{WRMF}} = 0.015$, $C = 1$, $epochs = 15$, which corresponds to a regularization parameter, a confidence weight that is put on positive observations, and to the number of passes over *all* observed data, respectively. The hyperparameters were set using grid search through a subset of powers of 10, e.g., 10^{-4}, 10^{-3}, ..., 1. We have observed that WRMF is not so sensitive to changes in the hyperparameters, the most important aspect is the number of iterations before early stopping, i.e., epochs=15 [Hu et al., 2008].

For all variations of `RMFO` we simulate the stream receiving one instance at the time based on the tweets' publication dates. Tweets without hashtags were ignored.

For RMFO-UB, we want to explore the effect of the user's buffer size b on the recommendation performance, we vary $b \in \{2^m \mid m \in \mathbb{N}, 1 \leq m \leq 9\}$, i.e., from 2 to 512.

For RMFO-SRV, we vary the reservoir size $|R| \in \{0.5, 1, 2, 4, 8\}$ million, and compute the model using 15 epochs over the reservoir only. We set regularization constants $\lambda_W = \lambda_{H^+} = \lambda_{H^-} = 0.1$, learning rate $\eta_0 = 0.1$, and a learning rate schedule $\alpha = 1$, and find that the setting gives good performance. We are currently investigating how to efficiently perform a grid search on stream data to tune-up the hyperparameters dynamically.

We divide the seven-month Twitter activity of our dataset by choosing the first six months for training. We use the remaining month, i.e., December, to build 10 independent test sets following the evaluation protocol described previously in this section. We compute the recall metric for Top-N recommendations, where $N \in \{1, 5, 10, 15, 20, 30, 40, 50\}$. The performance is evaluated on the test set only, and the reported results are the average over 10 runs.

Results

We found that recent topics are more valuable for recommendations: trending topics from the previous four weeks achieve a recall@10 of 7.8%, compared to 6.77% from the ones corresponding to the whole training period (6 months). The performance exhibited by this recommender, based on the crowd behavior in Twitter, largely outperforms a random model, whose recall@10 is under 1%. In the rest of the discussion we focus only on the recent trending topics.

Figure 3.8 shows the recommendation quality in terms of recall@10 for RMFO-SP, and RMFO-UB with varied user buffer sizes. We can see that recall@10 for RMFO-SP is 14.69%, 88.3% better than the overall trend.

We also observed that having a per-user buffer improves the performance. However if the buffer is small (e.g., 2 or 4), RMFO-UB achieves low recall. Although increasing the buffer size boosts the recommendation

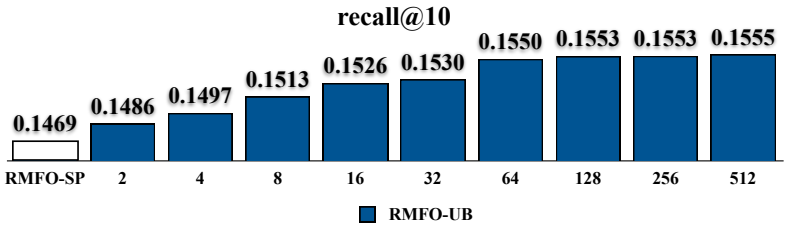


Figure 3.8: `RMFO-SP` and `RMFO-UB` Top-*10* performance for different sizes of user buffer.

quality, we found that as the quality reaches a plateau (see Figure 3.8), the buffer size provides limited improvements.

Figure 3.9 shows that `RMFO-SRV` achieves the best performance over all methods evaluated when the reservoir size is greater than 4 million, which corresponds to 11.32% of the entire number of transactions in the dataset.

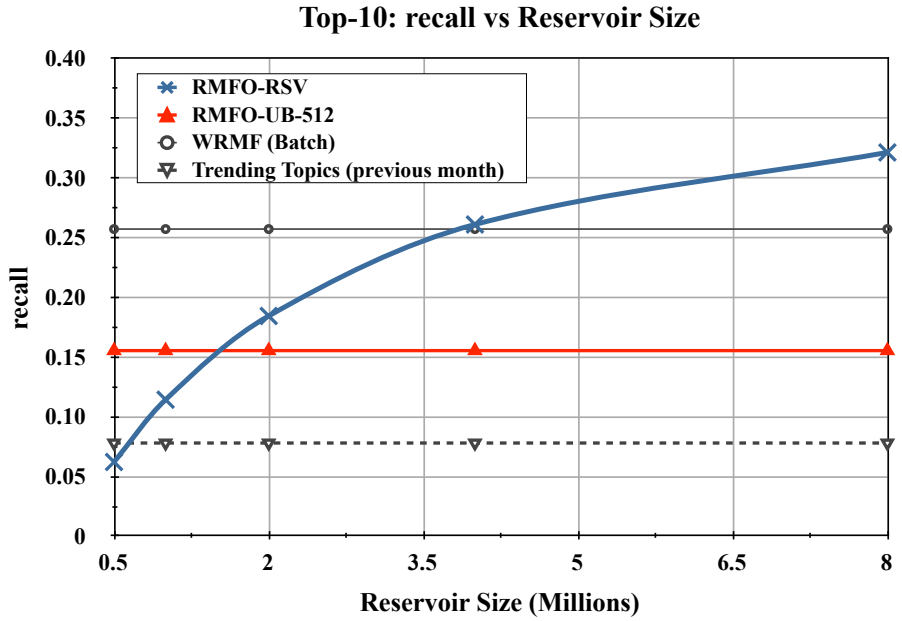


Figure 3.9: Recommendation performance for different sizes of the reservoir.

We summarize in Figure 3.10 the best performance achieved by the methods evaluated for different Top-N recommendations.

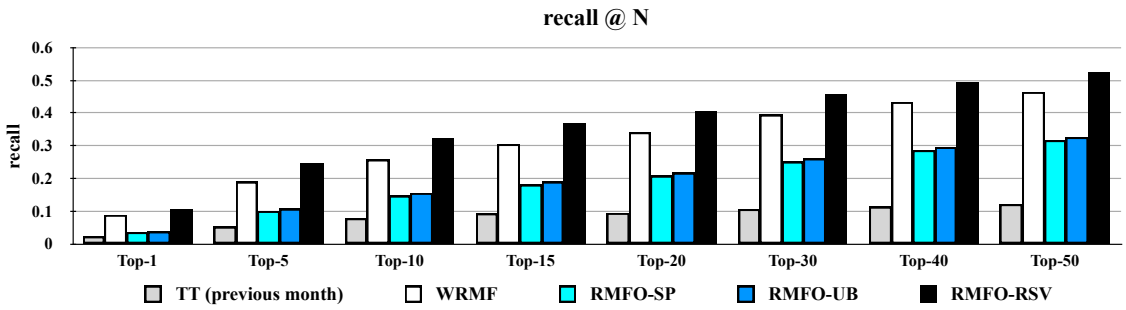


Figure 3.10: **Recommendation performance for different Top-N recommendation lists.**

With a fixed reservoir size of 8M, we also explored the impact of model dimensionality over the recommendation quality for `RMFO-RSV`. The results are presented in Figure 3.11. From the figure, we see that the 16-factor low-rank approximation given by `RMFO-RSV` exhibits a better recall@10 than WRMF computed in batch mode using 128 factors.

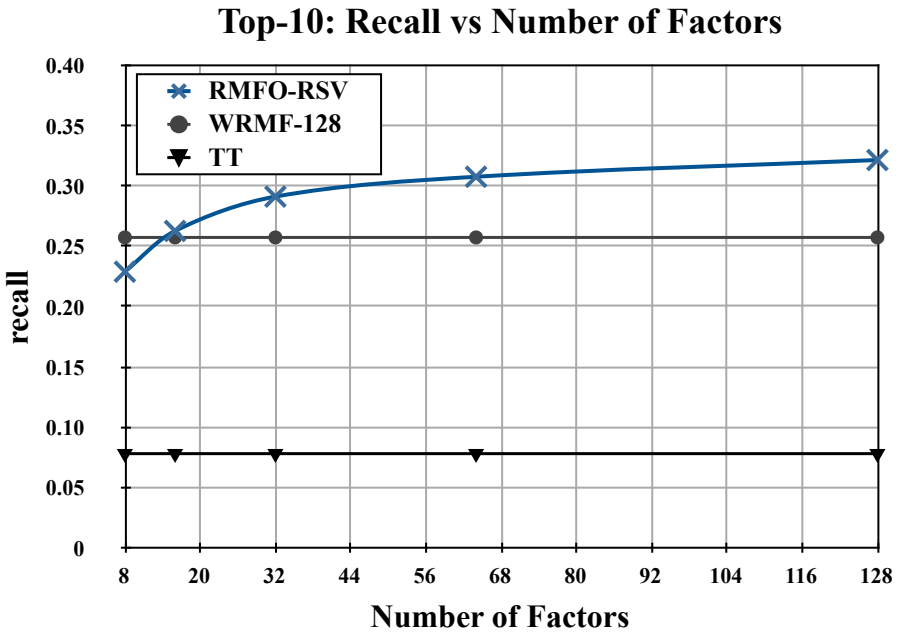


Figure 3.11: **Performance for different number of factors.**

Time and Space Savings vs. Recommendation Quality

We report in this section the CPU training times and space required for the best performing variation of our online approach: `RMFO-RSV`, and the ones for the strongest baseline: WRMF. Please remember that running times heavily depend on platform and implementation, so they should be only taken as relative indicators.

All variations of `RMFO` were implemented in Python. `RMFO` ran on a Intel Xeon 1.87GHz machine. For WRMF, we used the C# implementation provided by *MyMediaLite* library [Gantner et al., 2011]. The baseline WRMF was run on a machine with a slightly faster CPU (Intel Xeon 2.27GHz). None of the methods was parallelized and therefore used a single CPU for computations. GNU/Linux 64-bit was used as OS.

In Table 3.2, we can observe the gains in speed of our approach over the baseline for all the evaluated reservoir sizes. For reservoir sizes of 4M and 8M, `RMFO-RSV` is not only faster and space efficient, but also exhibits a better recommendation performance with respect to WRMF, for example, `RMFO-RSV` with a reservoir of size 8M is over 36 times faster and uses 77% less space than WRMF, and yet it delivers a recommendation performance almost 25% better than the state-of-the-art baseline. As a reference, we also include the performance of `RMFO-RSV INF`, which uses an *infinite* reservoir, e.g., one that is able to remember all observed transactions.

Method (128 factors)	Time (seconds)	recall@10	Space	Gain in speed	Gain in recall
WRMF (Baseline)	23127.34	0.2573	100.00%	–	–
RMFO-RSV 0.5 M	47.97	0.0621	1.41%	482.16	-75.85%
RMFO-RSV 1 M	89.15	0.1143	2.83%	259.42	-55.56%
RMFO-RSV 2 M	171.18	0.1845	5.66%	135.11	-28.30%
RMFO-RSV 4 M	329.60	0.2611	11.32%	70.17	+1.49%
RMFO-RSV 8 M	**633.85**	**0.3215**	**22.63%**	**36.49**	**+24.95%**
RMFO-RSV INF	1654.52	0.3521	100.00%	13.98	+36.84%

Table 3.2: **Time, Space and Performance Gains.** RMFO-RSV **is compared against WRMF for different reservoir sizes. The dimensionality of all methods is 128 factors.** RMFO-RSV INF **uses an *infinite* reservoir, e.g., one that is able to store all observed transactions and is provided as a reference. Time is measured in seconds and corresponds to the training time of one single epoch. Space is measured as the percentage of all transactions is the dataset, which is 100%=35.35M. Positive values of recommendation gain indicate a better performance of** RMFO-RSV **over the baseline. Observe, for example the line highlighted:** RMFO-RSV **with a reservoir of size 8M is over 36 times faster and uses 77% less space than WRMF, and yet it delivers a recommendation performance almost 25% better than the state-of-the-art baseline.**

3.5.4 Experiment II: RMFX Evaluation

We implemented our RMFX, and evaluated it against the following competing models, which are defined in Section 3.5.3:

1. RMFO-RSV**: Reservoir Sampling**
2. RMFO-SP**: Single Pass**
3. **Trending Topics (TT)**
4. **Weighted Regularized Matrix Factorization (WRMF)**

We simulate the stream receiving one instance at the time based on the tweets' publication dates. Tweets without hashtags were ignored.

For RMFX, RMFO-RSV, and RMFO-SP we set regularization constants $\lambda_W = \lambda_{H^+} = \lambda_{H^-} = 0.1$, learning rate $\eta_0 = 0.1$, and a learning rate

schedule $\alpha = 1$, and find that the setting gives good performance. We are currently investigating how to efficiently perform a grid search on stream data to tune up the hyperparameters dynamically. Moreover, the number of iterations is set to the size of the reservoir for both `RMFX` and `RMFO-RSV`.

WRMF setup is the same as in Experiment I: $\lambda_{\text{WRMF}} = 0.015$, $C = 1$, $epochs = 15$. Also as in Experiment I, we divided the seven-month Twitter activity of our dataset, by choosing the first six months (from first of June, 2009 to end of November, 2009) for training. We use the remaining month, i.e., December, to build 10 independent test sets following the evaluation protocol described previously in this section. The models `RMFX` and `RMFO-RSV` are built on the sketch of the stream available just before the evaluation period, i.e., end of November, 2009. For TT we use as predictors the most popular hashtags from the last four weeks before the evaluation, i.e., TT of November, 2009.

We restricted the analysis to a short list of recommendations and computed the recall metric for Top-N recommendations, where $N \in \{1, 5, 10\}$. The value of the metric for a particular Top-N is denoted as *recall@N*. The performance is evaluated on the test set only and the reported results are the average over 5 runs. All the differences reported are statistically significant (two-sample t-test, $p < 0.015$).

Results

Figure 3.12 summarizes the recommendation performance for `RMFX`, the baselines and the upper bound given by WRMF. We can observe that `RMFX` is superior with respect to the online methods `RMFO-SP` and `RMFO-RSV`, and largely outperforms the trending topics (TT). Please note that the trending topics from the previous four weeks achieve a recall@10=7.8%, this performance based on the crowd behavior in Twitter is much better than a random model, whose recall@10 is under 1%.

Table 3.3 shows that `RMFX` achieves the best performance over all online methods evaluated with reservoir sizes 2, 4 and 8 million. As expected, the offline method WRMF sets an upper bound for the online

approaches achieving a recall@5 of 18.96%, but RMFX is still competitive with a recall@5=16.58% and the advantage of real-time updates.

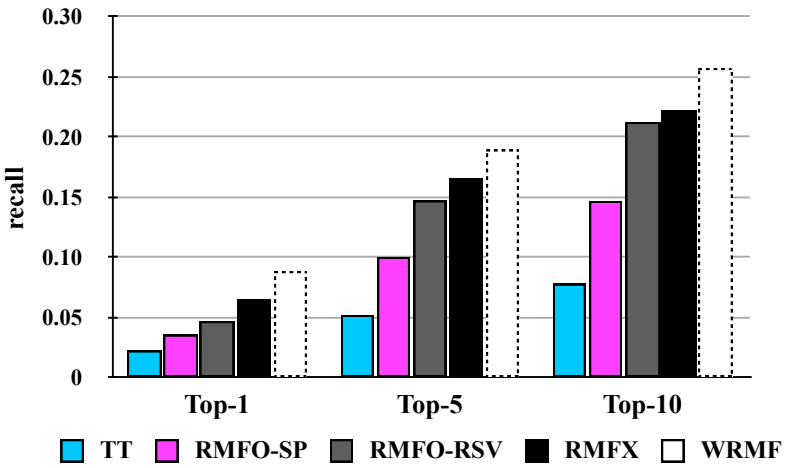


Figure 3.12: **Recommendation performance in terms of recall for Top–{1, 5, 10}. The reservoir size for RMFO-RSV and our method RMFX is set to 8M. The number of factors for the matrix factorization models is set to 128. The batch mode WRMF provides an upper bound reference for the online methods' performance.**

Method	Reservoir Size	Recall@1	Recall@5	Recall@10
RMFX	**8M**	**6.50%**	**16.58%**	**22.25%**
RMFO-RSV	8M	4.70%	14.72%	21.25%
RMFX	4M	4.16%	11.24%	15.84%
RMFO-RSV	4M	2.82%	9.02%	13.70%
RMFX	2M	1.95%	5.59%	8.41%
RMFO-RSV	2M	1.68%	4.89%	7.36%
RMFO-SP	–	3.57%	10.03%	14.69%
TT	–	2.26%	5.22 %	7.80%
WRMF	(Batch)	8.85%	18.96%	25.73%

Table 3.3: **Recommendation performance for different sizes of the reservoir. The number of factors is set to 128 for the online methods RMFX, RMFO-RSV, as well as for the offline approach WRMF.**

Finally, with a fixed reservoir size of 8M, we also explored the impact of model dimensionality over the recommendation quality for RMFX. The results are presented in Figure 3.13 and Table 3.4. From the figure, we see that RMFX consistently outperforms the baseline TT and the online competitors for 32, 64 and 128 dimensions.

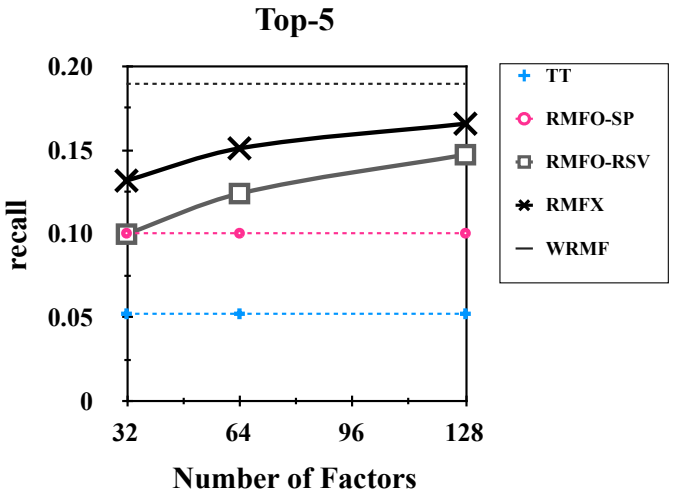


Figure 3.13: **Recall@5 for 32, 64 and 128 factors.**

Method	Factors	Recall@1	Recall@5	Recall@10
RMFX	**64**	**5.57%**	**15.10%**	**20.79%**
RMFO-RSV	64	3.79%	12.40%	18.89%
RMFX	32	4.32%	13.18%	18.96%
RMFO-RSV	32	2.69%	9.98%	16.07%

Table 3.4: **Recommendation performance for 32 and 64 factors, with a fixed reservoir of size 8M (the information for 128 factors is shown in Table 3.3).**

Time and Space Savings vs. Recommendation Quality

We report in this section the CPU training times and space required for the best performing variation of our online approach: `RMFX`, and the ones for the strongest baseline: WRMF. We also discuss the trade-off between time and space savings against the recommendation performance.

`RMFX` is implemented in the Python programming language using SciPy[2]. We ran `RMFX` on a Intel Xeon 1.87GHz machine. For WRMF, we used the implementation provided by *MyMediaLite* [Gantner et al., 2011], which is implemented in C#. The baseline WRMF was run on a machine with a slightly faster CPU (Intel Xeon 2.27GHz). The experiments were conducted using GNU/Linux 64-bit as operating system. None of the methods was parallelized and therefore used one single CPU for computations. Please remember that running times heavily depend on platform and implementation, so they should be only taken as relative indicators.

In Table 3.5, we can observe the gains in speed of our approach over the baseline for all the evaluated reservoir sizes. We can observe that for all reservoir sizes `RMFX` is faster and more space efficient than WRMF. For example, `RMFX` with a reservoir size 8M is approximately 43 times faster and uses 77% less space than WRMF, and yet it delivers a highly competitive recommendation performance corresponding to 86.47% of the state-of-the-art baseline computed offline.

[2]**SciPy :** scipy.org .

Method (128 factors)	Time (hh:mm:ss)	recall@10	Space	x times faster than WRMF	Recommendation Quality w.r.t WRMF
WRMF (Batch) [Baseline]	96:21:50.093	0.2573	100.00%	–	100%
RMFX 2M	0:33:12.898	0.0841	5.66%	174.07	32.69%
RMFX 4M	1:07:10.570	0.1584	11.32%	86.07	61.52%
RMFX 8M	**2:14:32.355**	**0.2225**	**22.63%**	**42.98**	**86.47%**

Table 3.5: **Time, Space and Recommendation Quality Comparison. RMFX is compared against WRMF for different reservoir sizes: 2, 4 and 8 million. The dimensionality of all methods is 128 factors. Time is measured in seconds and corresponds to the training time of 15 epochs in case of WRMF and one single epoch in case of RMFX, the gain is computed w.r.t. WRMF's duration. Space is measured as the percentage of all transactions in the dataset, which is 100%=35.35M. Recommendation performance quality is computed with respect to WRMF's recall, which is set as an upper-bound.**

3.6 Related Work

Our work is mainly related to the fundamental research areas of *Matrix Factorization for Collaborative Filtering*, *Online Learning*, and *Learning to Rank*.

Matrix Factorization for Collaborative Filtering

Collaborative Filtering (CF) [Goldberg et al., 1992] analyzes users behavior and their relationships and interactions with items of interest in order to discover new user-item associations. CF techniques are domain free and can capture patterns of the data that are often elusive and difficult to model explicitly, for example, using the items' content or their attributes.

Latent factor models are one of the most successful techniques for CF due to their prediction quality and effectiveness [Koren et al., 2009]. These models consider that only a small number of *latent* factors can influence the user preferences. Some of the most successful realizations of latent factor models are based on *matrix factorization*.

Traditional latent factor models for CF aim at predicting the ratings the users will give for a specific item, a task called *rating prediction* (e.g., [Koren et al., 2009; Srebro et al., 2005]). In contrast, we focus

on another task which is *item prediction*, also known as Top-N recommendation task, i.e. predict Top-N items that are interesting to the user. This problem is often formulated in the literature as a ranking problem. For example, approaches to CF that are conceptually similar to ranking (e.g., [Shi et al., 2010; Rendle et al., 2009b; Liu et al., 2009; Srebro et al., 2005]) have been presented in the literature, thus addressing the item prediction problem as well.

Another approach based on matrix factorization that deals with implicit feedback is Weighted Regularized Matrix Factorization (WRMF), introduced by Hu et al [Hu et al., 2008]. In their work, the authors model the MF as a regularized Least-Squares problem, in which a weighting matrix is used to differentiate the contributions from the implicit feedback. More recently, Rendle et al. introduced a Bayesian Personalized Ranking framework (BPR) [Rendle et al., 2009b], which provides a loss function that can be interpreted as a smooth version of the Area Under the ROC-Curve ranking evaluation measure (AUC). BPR also specifically tackles the problem where only positive observations about users are available, i.e, *implicit feedback*, as in our problem setting. Similar to this work, we also guide the matrix factorization process using an ordinal loss, but instead of the maximum posterior estimator from a Bayesian analysis offered by BPR, we explored a pairwise ranking approach based on the SVM loss.

The key difference between the aforementioned methods and our work is that their training procedure is performed in a *batch* mode, that is, they assume that all training examples are available before the learning task begins. Our work, however deals with a more realistic scenario where observations come from a stream with a high temporal dynamics. This means that batch approaches need to be often retrained to cope with the changes in the data over time, which make batch models unsuitable for some real world scenarios where the training instances arrive sequentially, as in the case of social web stream applications. One solution to this problem is to resort to online learning methods.

Online Learning

Online learning of matrix factorization methods for rating prediction have been investigated by Rendle and Schmidt-Thieme in [Rendle and Schmidt-Thieme, 2008]. They propose online update rules on a stochastic gradient descent style based on the *last* example observed. However, `RMFO-RSV` (the best performing variant of `RMFO`) and `RMFX` maintain a *reservoir* with a representative set of previously seen data points from the stream, which provides a significant boost in performance compared to the one obtained when only the last example is considered (e.g., `RMFO-SP`). The technique of random sampling with a reservoir is widely used in data streaming [Vitter, 1985], and recently has also been exploited by Zhao et al. in the context of binary classification [Zhao et al., 2011].

Learning of large-scale recommender systems for dealing with dynamic and fast changing content has been addressed before, for instance in the context of the *Google News* system [Das et al., 2007]. However the problem setting in [Das et al., 2007] is different from the one addressed here, since their work does not deal with a continuous stream of user generated data, but instead provides recommendations to users based on offline models.

Online learning techniques have also been applied to the problem of user generated content classification [Li et al., 2011]. In this work, the authors propose an online classification method that first builds a global model from the content generated by a group of users, and then leverages it to build a personalized classification model for individual users. Their experiments include a small scale evaluation on a Twitter dataset (7131 tweets) that targets to classify the microblog posts. We, on the other hand, are focused on a learning to rank setting for personalized item prediction on stream data containing millions of tweets.

Haghani et al. [Haghani et al., 2010] address the problem of continuously processing large number of user defined subscription queries, which they called *user profiles*, over a Web 2.0 stream of documents arriving at high rates. They concentrate their analysis on a strategy that filters user profiles, and intelligently avoids inserting all incoming

documents in the results lists using a skyline based method. Their model is based on the content of the incoming documents and the user profiles (queries). Their evaluation centers on efficiency of the online processing of incoming blog posts, and reports CPU time as main measure, without reporting on accuracy measures, since all the algorithms they evaluated actually reported the same Top-N results. We are also interested in Top-N recommendations, but our approach is an online CF setting, which does not require any text analysis, or the computation of TF/IDF vectors, which can be expensive to process online under stream data. Furthermore, the short and sparse text of Twitter documents impose additional challenges on traditional language models, whose performance heavily relies upon the content of the documents. Our online CF approach does not require a language model of the tweets, it only needs to recognize the hash tags from the text and the author of the tweet, in order to derive high quality Top-N recommendations in real-time.

Since we deal with pairwise classification from positive-only data, negative examples must be sampled. The sampling of the 59 negative examples for each positive one has been proposed, discussed and proved in [Ertekin et al., 2007]. Whereas they do it for active learning, we adapt it for our online learning to rank scenario, idea embodied in our `RMFX` algorithm.

Learning to Rank

In recent years, the task of *Learning to Rank* for Information Retrieval has drawn a lot of interest in machine learning, several methods have been applied to learn ranking functions and promising results have been obtained [Liu, 2011]. Existing methods include pairwise approaches that take pairs of objects and their relative preference as training instances, and transform the ranking problem into a binary classification on document pairs, e.g., each pair is classified as either correctly or incorrectly ranked. Typical methods include Ranking SVM [Joachims, 2002], RankBoost [Freund et al., 2003], RankNet [Burges et al., 2005]. Other methods include, pointwise and listwise approaches. Pointwise

approaches associate each training instance with a rating, i.e., absolute relevance. The learning task is to find a model that can predict the rating of a given instance, refer to [Crammer and Singer, 2002] for a typical example of this approach. The listwise approaches use a list of ranked objects as training instances and learn to predict the list of objects, (e.g., ListNet [Cao et al., 2007]). Common characteristics of these works are: (i) they are unpersonalized, and (ii) they are trained offline in a batch mode. Our work follows a pairwise approach to minimize an ordinal loss. Our formalization extends the work of Scully [Sculley, 2010] for unpersonalized learning to rank, to an online collaborative filtering setting, in order to learn a matrix factorization model for collaborative filtering in presence of streaming data.

An overview of the field is given in [Liu, 2011] and future directions in learning to rank are discussed in [Chapelle et al., 2011]. One of such directions is *Online Learning to Rank*, and we consider that our work represents a contribution towards this path.

3.7 Discussion

This chapter provides an example of integrating large-scale collaborative filtering with the real-time nature of Twitter.

We proposed `RMFO` and `RMFX`, approaches for recommending topics to users in presence of streaming data. Our online setting for collaborative filtering captures: "what is interesting to *me* right now within the social media stream", going beyond existing one-size-fits-all solutions.

Our methods receive instances from a microblog stream, and update a matrix factorization model following a pairwise learning to rank approach for dyadic data. At the core of `RMFO` and `RMFX` is stochastic gradient descent which makes our algorithms easy to implement and efficiently scalable to large-scale datasets. From the `RMFO`'s variants explored in this work, we found that the one using reservoir sampling technique performed the best.

`RMFX` represents a novel principled approach for online learning from streams. It builds upon the ideas of `RMFO` and extends them to consider

a strategy that selects a subsample of the observed data, based on the objective function gradients, and uses this information to guide the matrix factorization.

Observe that `RMFO` is simpler to implement, since it does not require a selective model update as `RMFX`, but `RMFO` requires more iterations over the reservoir to achieve a competitive recommendation performance, when compared to batch models. We found that `RMFX`, using a single pass over the interactions captured in the reservoir, achieves a better performance than `RMFO` with a single iteration. The selection of which approach to use depends on the concrete application scenario.

For example, if the time spent by `RMFO`, while learning the model using several iterations, does not impact the timeliness of the information recommended, then its ease of implementation can be a strength.

On the other hand, when timeliness is compromised, then `RMFX` becomes a better option. Measuring the recommendation performance online is specially challenging in scenarios of streaming data, which opens the doors to more research dedicated to tackle such issues.

Our empirical study used Twitter as test bed and showed that our approaches worked well relative to matrix factorization models computed in batch mode, in terms of recommendation quality, speed and space efficiency. Note that the experiments presented in this chapter were conducted over a single time split of the dataset. This allowed us to consider a large volume of the data so as to evaluate not only the recommendation quality, but also time and space performance. Even though we conducted the measurements over several rounds using different test sets (following a leave-one-out protocol), the use of one single split may lead to overfitting. In future work, we plan to extend our experimental evaluation and consider multiple splits of the data over time and conduct an online user study in order to evaluate the user experience of a system based on our approaches.

Finally, we are also interested in investigating how the frequency of repeated events (i.e. users using the same hashtag or listening to the same song) can be incorporated to the model to generate more accurate predictions.

4 Swarm Intelligence for Ranking and Recommendation

This chapter is composed by two parts. In the first one, we present an approach to automatically optimize the retrieval quality of ranking functions. Taking a Swarm Intelligence perspective, we present a novel method, *SwarmRank*, which is well-founded in a Particle Swarm Optimization framework.

SwarmRank learns a ranking function by optimizing the combination of various types of evidences such as content and hyperlink features, while directly maximizing Mean Average Precision, a widely used evaluation measure in Information Retrieval. Experimental results on well-established Learning To Rank benchmark datasets show that our approach significantly outperformed standard approaches (i.e., BM25) that only use basic statistical information derived from document collections, and is found to be competitive with Ranking SVM and RankBoost in the task of ranking relevant documents at the very top positions.

In the second part, we focus on the item prediction task of Recommender Systems and present *SwarmRankCF*, a method to automatically optimize the performance quality of recommender systems using a Swarm Intelligence perspective. Our approach, which is well-founded in a Particle Swarm Optimization framework, learns a ranking function by optimizing the combination of unique characteristics (i.e., features) of users, items and their interactions. In particular, we build feature vectors from a factorization of the user-item interaction matrix, and directly optimize Mean Average Precision metric in order to learn a linear ranking model for personalized recommendations. Our experimental evaluation, on a real world online radio dataset, indicates that

our approach is able to find ranking functions that significantly improve the performance of the system for the Top-N recommendation task.

4.1 Introduction

Although the most popular Information Retrieval (IR) model for document retrieval is still the vector space model [Baeza-Yates and Ribeiro-Neto, 2011], in recent years supervised learning-based methods have been proposed to automatically learn an effective ranking model based on training data and then applied to unseen test data (e.g., [Joachims, 2002; Freund et al., 2003; Cao et al., 2007]). This task is referred to as "Learning To Rank for IR" in the field.

The first part of this chapter presents an approach to learning ranking functions by analyzing important and unique characteristics (i.e., features) of a given text document collection so that the learned function is more effective than any general solution. To accomplish this, we use Particle Swarm Optimization (PSO), a Swarm Intelligence (SI) technique inspired by the social behavior of biological organisms, specifically the ability of some animal species to work as a whole in locating desirable positions in a given area, e.g., bird flocking or fish schooling [Eberhart and Kennedy, 1995; Kennedy and Eberhart, 1995; Poli et al., 2007].

To evaluate the quality of our SI approach, SwarmRank, we performed experiments on the well-established benchmark dataset collections provided by LETOR [Liu et al., 2007]. Results indicate that our approach is able to find ranking functions that significantly outperform the BM25 baseline, and that performs competitively compared with Ranking SVM and RankBoost in the task of ranking relevant documents at the very top positions to get a high MAP.

Observe that learning to rank relies upon pre-engineered features that characterize the items to be ranked. Such a set of features can be difficult to compute and maintain.

In the absence of high quality *explicit features*, recommender systems can infer *latent factors* about users and items using matrix fac-

torization algorithms for Collaborative Filtering (CF), as discussed in Section 2.1.

In the second part of this chapter, we present an approach to learning ranking models for recommender systems that exploits collaborative latent factors as features. To accomplish this, instead of manually creating an item feature vector, we factorize a matrix of user-item interactions, and use these collaborative latent factors as input to our SwarmRank approach. We focus on the item prediction or Top-N recommendation problem in the context of recommender systems, which we consider a harder prediction problem than rating prediction, as only positive feedback is available.

PSO, which is at the core of SwarmRank and SwarmRankCF, was chosen because it is a global non-linear optimization algorithm that does not require, nor approximate, gradients of the cost function, and due to its resilience to local minima. These characteristics allow us to directly optimize non-smooth Information Retrieval (IR) measures, such as MAP (Mean Average Precision) [Baeza-Yates and Ribeiro-Neto, 2011], a desirable property to tackle the ranking and top-N recommendation problems.

Contributions. The main contributions of this chapter are as follows:

- To the best of our knowledge, SwarmRank is the first that applies PSO to the learning to rank for IR task. The closest works to ours are the evolutionary computation methods based on Genetic Programming that have been applied to learn ranking functions, which are discussed in the related work section.

- We present SwarmRankCF, a model suitable for the personalized item recommendation task, that seamlessly integrates a SI learning to rank method with a collaborative filtering approach, to derive a personalized ranking function optimized for the ranking task.

- We present an empirical study on the public datasets *LETOR* and *Last.fm Dataset – 1K users* demonstrating the superior ranking and recommendation performance of our SI approaches.

4.2 Swarming to Rank for Information Retrieval

Inspired by the success of PSO as a common heuristic technique for global optimization, we propose in this chapter *SwarmRank*, a PSO-based method to handle the task of learning to rank for IR. The goal of SwarmRank is the discovering of good ranking formulas more adapted to the particularities of a specific collection, which are also able to generalize well beyond the training data. The procedure examines important information retrieval features extracted from query-document pairs instances and learns a linear function that combines them optimally.

SwarmRank is basically an iterative process based on PSO, that learns linear ranking functions of the form:

$$f(q,d) = \vec{p_g} \cdot \phi(q,d) \quad , \tag{4.1}$$

where $\vec{p_g}$ denotes a weight vector. In ranking for query q_i the model associates a score to each of the documents d_j as their degree of relevance with respect to query q_i using $f(q_i, d_j)$ and sorts the documents based on their scores (see Section 2.2).

Note that $\vec{p_g}$ is the weight vector corresponding to the best global solution found by the *swarm* in PSO. The function learned applying PSO, which represents a linear combination of the query-document pairs feature vectors, associates a real value to each query-document pair as its degree of relevance.

SwarmRank takes as an input the query-document pairs and their corresponding feature vectors. As an instance of PSO, SwarmRank, quantifies the optimality of a solution by means of a fitness function, which serves as a criteria to update the particle and global best solutions. As output, the procedure returns a linear ranking function whose coefficients are given by the dimension values corresponding to the global best solution vector found by the swarm. SwarmRank is summarized in Procedure (1).

The rest of the section discusses the key components of our approach and its implementation.

Fitness Function

Since our particles represent solutions in terms of weight vectors to be used in a document ranking function, our fitness function must measure the quality of the ranking generated by a given particle. We take advantage of the non-linear optimization capabilities of PSO and directly optimize a non-smooth IR measure, namely Mean Average Precision – MAP [Baeza-Yates and Ribeiro-Neto, 2011] (as defined in Section 2).

Best Particle Selection

In addition to the training set T, we consider a validation set V, also provided by LETOR, that helps us to reduce *overfitting*, i.e., to avoid selecting solutions that work well in the training set but do not generalize well for unseen query-document pairs. The choice of the best solution is accomplished by considering the performance of a particle in both the training and validation sets, which are used symmetrically in this work. Hence, a fitness function, as defined in Eq. (4.2), is exploited to select the best particle.

$$fitness(\vec{x_i}, T, V) = MAP(\vec{x_i}, T) + MAP(\vec{x_i}, V) \tag{4.2}$$

where the first and second terms of the function correspond to the performance of the particle i in the training and validation set, respectively.

Implementation

SwarmRank has been implemented as a simulation component using the optimization framework provided by GenOpt [Wetter, 2011], which is a platform independent, open and extensible simulation environment, written in Java, that implements a variety of optimization algorithms, including several variants of PSO. Specifically, in our experiments, we used the PSO with Constriction Coefficient algorithm (PSOCC) available in the framework.

Procedure 1 The proposed SI-based learning procedure: **SwarmRank**

Input: (i) A training set T of query-document pairs with their feature vectors. (ii) A validation set V of query-document pairs with their feature vectors.

Output: A ranking function $f(q, d)$ that associates a score $s \in \mathbb{R}$ to a document d as their degree of relevance w.r.t the query q

1: **for** each time step t **do**
2: **for** each particle i in the swarm **do**
3: update position $\vec{x_i^t}$ using Eqs. 2.4 & 2.5
4: calculate particle fitness $fitness(\vec{x_i^t}, T, V)$ (Eq. (4.2))
5: update $\vec{p_i}$, $\vec{p_g}$
6: **end for**
7: **end forreturn** $f(q, d) = \vec{p_g} \cdot \phi(q, d)$

4.2.1 Experiments on LETOR 2.0

In this part of the experimental evaluation, we used the LETOR (version 2.0) benchmark datasets [Liu et al., 2007]: TD2003 and TD2004. The datasets details are presented in Section 2.5.

SwarmRank Parameters and Setup

The parameter settings for SwarmRank are summarized in Table 4.1. For the PSO algorithm the parameters values correspond to the most commonly found in the literature, (e.g., [Wetter, 2011; Poli et al., 2007; Bratton and Kennedy, 2007]). Each particle has exactly 44 dimensions, corresponding to the number of features provided by the datasets. The population of particles is initialized with random positions and velocities. Furthermore, the value of each dimension k of particle i position vector $\vec{x_i}$ is restricted to the interval $[-1, +1]$, i.e., $x_{i,k} \in [-1, +1]$.

Data Normalization. Since the absolute values of a feature for different queries might not be comparable, we conducted query-based normalization for each feature before applying SwarmRank. Suppose that $L = \{d_j^i \in D | j = 1, \cdots, |L|\}$ is a list of documents for a given

query q_i in the dataset, then the normalized value of a feature $\phi_k(q_i, d_j) \in [0, 1]$ of a document $d_j^i \in L$ is given by Eq. (4.3):

$$\phi_k(q_i, d_j) = \frac{\phi_k(q_i, d_j) - min\{\phi_k(q_i, d_l)\}}{max\{\phi_k(q_i, d_l)\} - min\{\phi_k(q_i, d_l)\}} \tag{4.3}$$

where $min\{\phi_k(q_i, d_l)\}$ and $max\{\phi_k(q_i, d_l)\}$ are the minimum and the maximum value of $\phi_k(q_i, d_j)$ respectively for all $d_l \in L$.

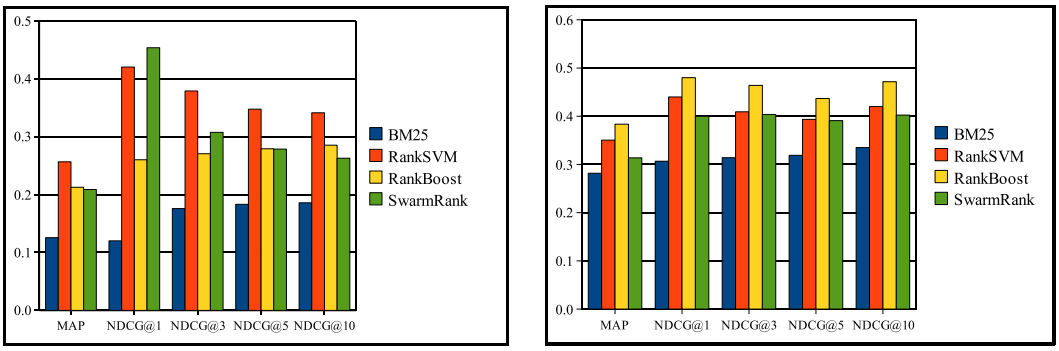
Evaluation and Baselines

We compared the retrieval results of our swarm intelligence approach with (i) Okapi BM25 [Robertson, 1997], a non-learning baseline, and two state-of-the-art learning-based baseline methods: (ii) Ranking SVM [Joachims, 2002] (denoted as RankSVM) and (iii) RankBoost [Freund et al., 2003]. The evaluation results from the baselines reported in this chapter are excerpted from LETOR [Liu et al., 2007]. Please refer to LETOR for details on the methods and parameter settings of these baselines.

We conducted 5-fold cross validation experiments, following the guidelines of LETOR [Liu et al., 2007]. For each fold, we used three subsets for training, one subset for validation, and the remaining one for testing. The test set is used to report the ranking performance of the function learned by SwarmRank. The reported performance in this chapter is the average over the five folds.

Parameter	Value
SI Algorithm	PSO with Constriction Coefficient
Swarm size	100
# of Generations	100
Neighborhood Topology	Local Best (*lbest*)
Neighborhood Size	10
Cognitive Acceleration	$c_1 = 2.05$
Social Acceleration	$c_2 = 2.05$
Constriction Coefficient	$\chi = 0.72984$
# Dimensions	44 (one per feature)

Table 4.1: **SwarmRank parameter settings.**



(a) TD2003 (b) TD2004

Figure 4.1: **Ranking accuracies on LETOR 2.0 data.**

Results

Table 4.2 and Figure 4.1 detail the ranking performance of SwarmRank and the baselines on the TD2003 and TD2004 datasets. We conducted pairwise t-tests on the improvements between SwarmRank and the baselines for all measures reported. The *statistically significant* differences are highlighted in Table 4.2. As we can see, SwarmRank outperforms BM25 with respect to all measures for both datasets. For TD2003, SwarmRank reached a mean average precision of 20.9%, corresponding to gain of 66.25% over BM25. On TD2004, SwarmRank achieved a mean average precision of 31.4%, outperforming BM25 by 11.35%.

We also observe in Figure 4.1 that on TD2003, SwarmRank improves over the best baseline, RankSVM, for P@1 and NDCG@1. The relative improvement is 7.94% in both cases. Similarly, we can observe that our approach outperforms RankBoost on P@1-3 and NDCG@1-4 on the same dataset. However, SwarmRank does not perform equally well on P@n and NDCG@n values when n is greater than 3. The results suggest that SwarmRank is better at ranking relevant documents at the very top positions to get a high MAP.

The results presented in Table 4.2 also show that SwarmRank and RankSVM perform similarly on the TD2004 dataset. From the table we observe that RankBoost outperforms SwarmRank and RankSVM in terms of all measures.

However, on TD2003 and TD2004 t-tests show that, in most of the cases, there is no statistically significant differences among the performance of SwarmRank, RankSVM and RankBoost; the small number of queries in both datasets clearly influences this result. This is a common problem for major available datasets. Experiments on additional collections are left to future work. In general, the performance of SwarmRank seems to be comparable to that of RankSVM, which can be explained by the type of ranking function learned by both methods, i.e., linear.

Evaluation Measures	TD2003						
	Baselines			SwarmRank			
	BM25	RankSVM	RankBoost	*Measure Value*	*Gain over BM25*	*Gain over RankSVM*	*Gain over RankBoost*
MAP	0.126	0.256	0.212	0.209	**+66.25%**	-18.63%	-1.81%
P@1	0.120	0.420	0.260	0.453	**+277.78%**	+7.94%	**+74.36%**
P@2	0.130	0.350	0.270	0.330	**+153.85%**	-5.71%	+22.22%
P@3	0.160	0.340	0.240	0.269	**+68.13%**	-20.92%	+12.04%
P@4	0.145	0.300	0.230	0.223	+54.02%	-25.56%	-2.90%
P@5	0.148	0.264	0.220	0.207	+39.64%	-21.72%	-6.06%
P@6	0.140	0.243	0.210	0.188	+34.20%	-22.83%	-10.58%
P@7	0.129	0.234	0.211	0.185	+43.77%	-21.14%	-12.61%
P@8	0.120	0.233	0.193	0.173	+44.44%	-25.45%	-9.96%
P@9	0.116	0.218	0.182	0.164	+42.32%	-24.49%	-9.76%
P@10	0.109	0.206	0.178	0.151	+38.72%	-26.86%	-15.36%
NDCG@1	0.120	0.420	0.260	0.453	**+277.78%**	+7.94%	**+74.36%**
NDCG@2	0.140	0.370	0.280	0.343	**+145.24%**	-7.21%	+22.62%
NDCG@3	0.176	0.379	0.270	0.307	+74.87%	-18.86%	+13.63%
NDCG@4	0.174	0.363	0.272	0.284	+63.31%	-21.82%	+4.24%
NDCG@5	0.183	0.347	0.279	0.278	+52.12%	-19.84%	-0.19%
NDCG@6	0.184	0.341	0.280	0.271	+46.96%	-20.45%	-3.17%
NDCG@7	0.184	0.340	0.287	0.273	+47.95%	-19.64%	-5.09%
NDCG@8	0.185	0.345	0.282	0.270	+45.96%	-21.78%	-4.39%
NDCG@9	0.186	0.342	0.282	0.267	+43.83%	-21.81%	-5.34%
NDCG@10	0.186	0.341	0.285	0.263	+41.44%	-22.99%	-7.89%
Evaluation Measures	**TD2004**						
	Baselines			**SwarmRank**			
	BM25	**RankSVM**	**RankBoost**	*Measure Value*	*Gain over BM25*	*Gain over RankSVM*	*Gain over RankBoost*
MAP	0.282	0.350	0.384	0.314	+11.35%	-10.29%	**-18.23%**
P@1	0.307	0.440	0.480	0.400	+30.44%	-9.09%	**-16.67%**
P@2	0.293	0.407	0.447	0.380	**+29.55%**	-6.56%	-14.93%
P@3	0.258	0.351	0.404	0.351	+36.21%	0.00%	-13.19%
P@4	0.243	0.327	0.347	0.317	**+30.14%**	-3.06%	-8.65%
P@5	0.229	0.291	0.323	0.296	**+29.07%**	+1.83%	-8.26%
P@6	0.224	0.273	0.304	0.278	+23.76%	+1.63%	-8.76%
P@7	0.210	0.261	0.293	0.253	+20.91%	-2.92%	**-13.64%**
P@8	0.192	0.247	0.277	0.235	+22.61%	-4.73%	**-15.06%**
P@9	0.182	0.236	0.262	0.221	**+21.14%**	-6.29%	**-15.82%**
P@10	0.175	0.225	0.253	0.215	**+22.90%**	-4.73%	**-15.26%**
NDCG@1	0.307	0.440	0.480	0.400	+30.44%	-9.09%	-16.67%
NDCG@2	0.327	0.433	0.473	0.413	**+26.53%**	-4.62%	-12.68%
NDCG@3	0.314	0.409	0.464	0.404	+28.63%	-1.33%	-12.98%
NDCG@4	0.315	0.406	0.439	0.393	+24.65%	-3.24%	-10.58%
NDCG@5	0.319	0.393	0.437	0.391	**+22.38%**	-0.67%	-10.52%
NDCG@6	0.325	0.397	0.448	0.394	+21.40%	-0.77%	-11.91%
NDCG@7	0.326	0.406	0.457	0.392	+20.33%	-3.42%	-14.26%
NDCG@8	0.324	0.410	0.461	0.396	+22.10%	-3.55%	-14.10%
NDCG@9	0.332	0.414	0.464	0.397	+19.79%	-4.17%	-14.47%
NDCG@10	0.335	0.420	0.472	0.402	+20.13%	-4.21%	-14.68%

Statistically significant differences are highlighted (p-value < 0.05).

Table 4.2: **Ranking performance for TD2003 and TD2004.**

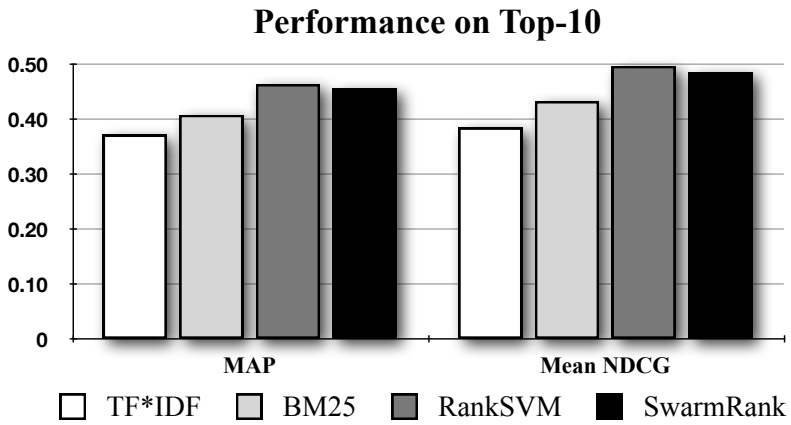


Figure 4.2: **Ranking performance in terms of MAP and mean NDCG on LETOR 4.0.**

4.2.2 SwarmRank on LETOR 4.0

The experiments presented so far were performed on a limited number of queries using the benchmark dataset LETOR version 2.0, specifically, on the TD2003 and TD2004 datasets, wich include only 50 and 75 queries, respectively. We present here an extended evaluation on LETOR 4.0, the most recent version of the dataset, which uses the Gov2 web page collection (∼25M pages) and two query sets from Million Query track of TREC 2007 and TREC 2008, denoted as MQ2007 and MQ2008 for short. There are about 1700 queries in MQ2007 with labeled documents and about 800 queries in MQ2008 with labeled documents. Further details about LETOR 4.0 dataset can be found in Section 2.5.

Results

We conducted 5-fold cross validation experiments, following the guidelines of LETOR [Liu et al., 2007]. For each fold, we used three subsets for training, one subset for validation, and the remaining one for testing. The test set is used to report the ranking performance of the function learned by SwarmRank. The reported performance in this chapter is the average over the five folds.

The experimental results are presented in Figure 4.2. The figure shows ranking performance in terms of MAP and mean NDCG[1] for a list of ten top-10 results. We can observe that SwarmRank is superior than TF*IDF and BM25, and performs competitive against RankSVM. SwarmRank achieves a MAP of 45.6% and a mean NDCG of 48.6%, while RankSVM's scores are slightly better (MAP = 46.4% and NDCG = 49.7%). The results suggest that RankSwarm keeps its good performance when scaling up to larger datasets.

4.3 Swarming to Rank for Recommender Systems

In this section we present our collaborative ranking approach for learning to rank. The goal of learning to rank is to automatically learn a ranking model from training data, such that the model can sort objects (e.g., documents, songs, movies) according to their degrees of relevance, preference, or importance as defined in a specific application.

In the general learning to rank scenario, a retrieval function is learned from rankings of documents associated to queries. Each document is represented by a vector of predefined features that characterize it, for example, text documents are usually characterized by their term frequency, inverse document frequency, document length and other low-level content characteristics, as well as high-level content features as BM25 scores. In the case of web pages, hyperlink features are also used, such as PageRank or HITS.

In CF, on the other hand, item and user features can be learned based on user-item interactions (e.g, ratings or relevance scores) using a matrix factorization method, which learns for each item (columns of $\mathbf{X}$), a k-dimensional feature vector (rows of $\mathbf{I}$), where each row of $\mathbf{U}$ is a feature vector that captures the latent factors of the users, and can be considered as a linear predictor, predicting the entries in the corresponding row of $\mathbf{X}$, based on inner products in a k-dimensional space (Section 2.1).

[1]Mean NDCG is the average value of NDCG at list positions 1 to 10.

Algorithm 2 : SwarmRankCF

Input: (i) Matrix $\mathbf{X}$ of user-item interactions and (ii) The number of factors k

Output: A ranking function $f(u, i) = \vec{p_g} \cdot \phi(u, i)$

1: Extract k collaborative features (i.e., latent factors) $\mathbf{I}$ by factorizing matrix $\mathbf{X}$, e.g, by applying SVD (Chapter 2).

2: Apply SwarmRank to learn a ranking function $f(u, i)$, by optimizing Mean Average Precision (MAP), using U as queries Q, items I as documents D, the item feature vectors $\mathbf{I}$ as $\phi(u, i)$, and $\mathbf{X}$ as the relevance scores Y (Chapter 2). **return** $f(u, i)$

We propose to learn a collaborative ranking model using SwarmRank. To this end, we cast the item recommendation task as a learning to rank problem, where users U can be considered as queries Q, items I as documents D, feature vectors $\phi(u, i)$ as $\mathbf{I}$ (i.e, the latent factors)[2]. Finally, the relevance scores Y are given by $\mathbf{X}$, that corresponds to ratings or implicit feedback information that measures user u's preferences for item i. The approach, named *SwarmRankCF* is summarized in Algorithm (2).

4.3.1 Experiments and Evaluation

We evaluate the recommendation performance of our approach, SwarmRankCF, on a public dataset obtained from *Last.fm*, a major social media Internet radio station. In our evaluation, we empirically compared the recommendation quality of SwarmRankCF to: (i) *PureSVD*: a matrix factorization algorithm based on SVD and proposed for Top-N recommendation tasks [Cremonesi et al., 2010], (ii) *Most Popular*: a non-personalized method that recommends the most popular artists to every user, and (iii) *Random*: a baseline method, that recommends random artists to users.

[2]Please note that since we are learning a linear ranking function, the model cannot make use of the user u features, because such features are the same for all items of user u within his rankings.

Dataset

We conducted experimentation, parametrization and algorithmic fine tuning on the "real-world" public dataset *Last.fm Dataset – 1K users* (Section 2.5). Specifically, we used in our evaluation a 5-core subset of the dataset, i.e., every user listened to at least 5 songs and each song was listened by at least 5 users. The 5-core statistics are as follows: 242,103 user-item interactions (transactions), 844 unique users and 37,315 unique items (artists).

Evaluation Methodology

Our goal is to evaluate the system performance when it suggests Top-N items to a user. For example, recommending the user a few specific artists which are supposed to be the most attractive to him. That is, to find the relative position of these interesting items within the total order of items ranked for a specific user.

To evaluate the recommenders we used a variant of the leave-one-out protocol followed in Section 3.5.1. The only variation is that in this case we predict the scores for the hidden item i and for 100 additional items, forming a ranking by ordering the 101 items according to their scores. The best expected result is that the interesting item i to user u will precede the rest 100 random items. Note that we used the last four weeks in the dataset as our test set[3].

We generate a Top-N recommendation list by selecting the N items with the highest score. If the test item i is in the Top-N, then we have a *hit*, otherwise we have a *miss*. We measure the quality by looking at the *recall* metric as defined in Equation 3.3. Remember that following this protocol, precision is forced by taking into account only a restricted number N of recommendations, that is, precision is just the same as recall up to a multiplicative constant. Therefore, there is no need to evaluate *precision* or *F1* explicitly.

[3]We also experimented with a larger test set, e.g., twelve weeks, observing similar results.

Experimental Setting

For SwarmRankCF, we used SVD as factorization method to learn the collaborative feature vectors. Furthermore, we set $k = 46$ as the number of latent factors for SwarmRankCF and PureSVD. The rationale behind using the particular value of $k = 46$ is to have an experimental setting with a dimensionality similar to the one used in non-personalized learning to rank tasks, e.g., the number of features in the benchmark dataset LETOR [Liu, 2011].

Observe that in this experimental evaluation we compare the performance of our approach against PureSVD and not to WRMF, as we did in the Chapter 3. The reason is that we consider more illustrative to demonstrate how the re-ranking performed by SwarmRankCF improves the recommendation quality, with latent features inferred using a more widespread method, such as SVD, which is not directly optimized for ranking. For example, to show recommender system practitioners that the features they have already computed using SVD can be leveraged to obtain a better recommendation quality for the the top-N recommendation task.

SwarmRankCF has been implemented in the Java programming language using the optimization framework provided by GenOpt [Wetter, 2011]. Specifically, in our experiments we used the PSO with Constriction Coefficient algorithm (PSOCC) with parameter values that are most commonly found in the literature (e.g., [Poli et al., 2007]). The parameter setting for SwarmRankCF is summarized in Table 4.3.

Parameter	**Value**
SI Algorithm	PSO with Constriction Coefficient
Swarm size	50
Neighborhood Topology	Local Best (*lbest*)
Neighborhood Size	10
Cognitive Acceleration	$c_1 = 2.05$
Social Acceleration	$c_2 = 2.05$
Constriction Coefficient	$\chi = 0.72984$
# Dimensions	46 (one per latent feature)

Table 4.3: **SwarmRankCF parameter settings.**

Results

Recall was evaluated for different recommendation list sizes: Top-N, where $N \in \{1, 5, 10, 20\}$. The results are presented in Figure 4.3. SwarmRankCF clearly delivers the best recommendations, achieving significantly better results than the other methods (two-sample t-test, $p < 0.025$). As expected, SwarmRankCF and PureSVD largely outperform the non-personalized baselines. The reader should note that both SwarmRankCF and PureSVD use the latent factors learned by factorizing the user-item matrix using SVD, the strong performance of SwarmRankCF may be attributed to its better ability to model ranking semantics based on those features.

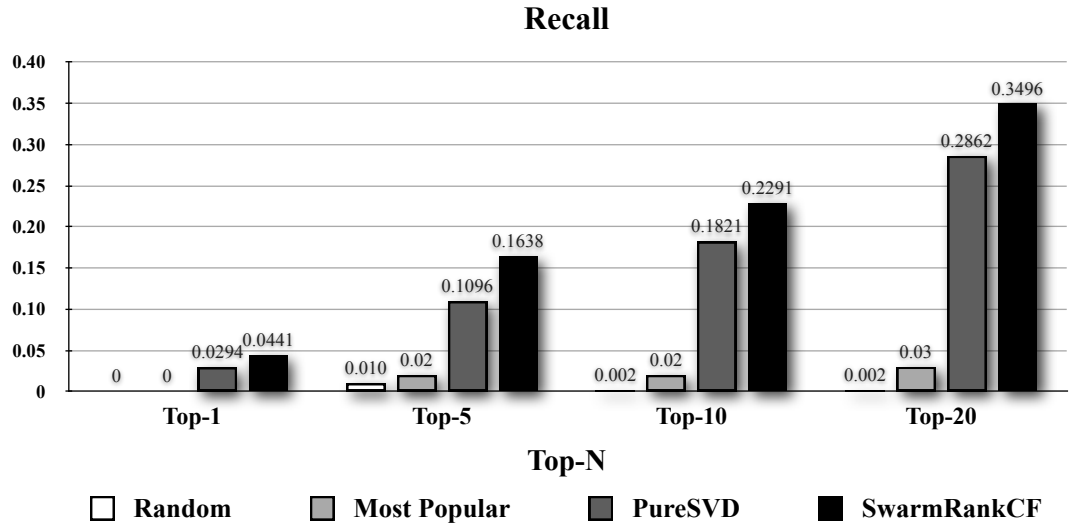


Figure 4.3: **Recommendation performance in terms of recall measure.**

4.4 Related Work

In recent years, the task of learning to rank has drawn a lot of interest in machine learning and several methods have been applied to learn ranking functions and promising results have been obtained. Typical methods include Ranking SVM [Joachims, 2002; Herbrich et al., 2000], RankBoost [Freund et al., 2003] and RankNet [Burges et al., 2005]. These three methods minimize loss functions that are loosely related

to the evaluation measures such as MAP and NDCG. Given the nature of our approach, we are able to directly optimize such measures.

Different approaches to discover ranking functions based on genetic algorithms and genetic programming have been proposed in the literature. Fan et al. [Fan et al., 2000, 2004, 2005] propose an approach to generate term weighting strategies for different contexts. The authors used an expression tree data structure to represent a term weighting formula and applied genetic programming (GP) to select the best performing function. The work in [de Almeida et al., 2007] also presents a GP approach, the authors successfully combine components or portions of well-known, significant ranking functions from different retrieval systems to generate completely new ranking formulas using genetic programming.

In [Yeh et al., 2007] a learning method that employs genetic programming to learn ranking functions is introduced. The authors also evaluate their method using the LETOR benchmark datasets obtaining promising results.

PSO shares many similarities with evolutionary computation techniques such as Genetic Programming. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GP, PSO has no evolution operators such as crossover and mutation. Furthermore, while the GA is inherently discrete, i.e. it encodes the design variables into bits of 0's and 1's, PSO is inherently continuous.

Learning to rank algorithms rely upon a manually selected set of features to build the ranking models. In contrast, our approach swarmRankCF, characterizes the user and items using feature vectors based on the collaborative latent factors, that are learned directly from the user-item interactions

In the field of recommender systems, a lot of attention was directed to the rating prediction task, motivated by the Netflix challenge (e.g., [Koren et al., 2009]), standard sparse regression and classification methods proved to be successful tackling the challenge, but they cannot be directly applied to the item recommendation problem, as only positive

observations are made. State-of-the-art methods for item recommendation are based on matrix factorization models that infer latent factors from the user-item interaction matrix, which we exploit to learn ranking functions for item prediction.

4.5 Discussion

In the first part of the chapter, we presented a novel approach to address the task of learning to rank for Information Retrieval, based on Swarm Intelligence techniques. Cornerstone of our approach is the use of Particle Swarm Optimization for learning a ranking function that improves its retrieval performance automatically.

Our approach, *SwarmRank*, directly optimizes Mean Average Precision, an evaluation measure used in IR, in comparison with many learning to rank algorithms, which minimize a loss function loosely related to the IR measures.

Experimental results on the LETOR benchmark datasets show that SwarmRank performs well in practice, successfully improving the ranking accuracies compared to standard approaches (i.e., BM25) that only use basic statistical information derived from documents collections. Furthermore, the results show that SwarmRank is competitive to Ranking SVM and RankBoost ranking relevant documents at the very top positions. An interesting research direction for future work is to explore the learning process of non-linear functions, which would require alternative particle representations, not just as weight vectors, but possibly as traversal of an expression tree representing a non-linear ranking function.

In the second part of the chapter, we proposed SwarmRankCF to address the item recommendation task in the context of recommender systems.

SwarmRankCF is a collaborative learning to rank algorithm based on swarm intelligence. While learning to rank algorithms use hand-picked features to represent items, we learn such features based on user-item interactions, and apply a PSO-based optimization algorithm that directly maximizes Mean Average Precision.

Our experimental study demonstrates the recommendation performance gain of SwarmRankCF for the Top-N recommendation task.

For future work, we plan to extend SwarmRankCF to handle stream data. Most learning to rank algorithms work in batch mode, but social media streams, require ranking models to be updated online. Swarm intelligence can help to evolve models when new data enters the system. Another future research direction is to learn the latent factors using swarm intelligence as well, and at the same time optimize the ranking measure.

Although our investigation has provided promising results, we believe that our contribution is an initial step in the study of SI techniques for Learning to Rank and Recommender Systems, additional research in this field is still to be explored.

5 Automatic Tagging

In this chapter we propose α-*TaggingLDA*, a method for automatic tagging sparse and short textual resources. In the presence of a new resource, our method creates an ad hoc corpus of related resources, and then applies Latent Dirichlet Allocation (LDA) to elicit latent topics for the resource and the associated corpus. This is done in order to automatically tag the resource based on the most likely tags derived from the latent topics identified.

In the first part of the chapter, we evaluate our approach using an offline analysis on publicly available BibSonomy[1] dataset (Section 2.5) and through an online study, showing the effectiveness of α-*TaggingLDA* for the tag recommendation task.

In the second part, we apply our approach to address *the cold start problem* in collaborative learning environments. An online presence is gradually becoming an essential part of every learning institute. As such, a large portion of learning material is becoming available online. Incongruently, it is still a challenge for authors and publishers to guarantee accessibility, support effective retrieval, and the consumption of learning objects. One reason for this is that non-annotated learning objects pose a major problem with respect to their accessibility. Non-annotated objects not only prevent learners from finding new information; but also hinder a system's ability to recommend useful resources. To address this problem, commonly known as the cold start problem, we automatically annotate specific learning resources using α-*TaggingLDA*. We performed a user evaluation with 115 participants to measure the usability and effectiveness of our approach in a collaborative learning environment.

[1] **BibSonomy:** bibsonomy.org .

The results show that automatically generated tags were preferred 35% more than the original authors' annotations. Further, they were 17.7% more relevant for users, in terms of recall. The implications of these results is that automatic tagging can facilitate effective information access to relevant learning objects.

5.1 Introduction

Tagging has proven to be an intuitive and flexible Web 2.0 mechanism to enhance the users' online experience. Tags are capable of facilitating search, easing navigation (e.g., tag clouds), and improving personalization in collaborative tag recommendations and across disparate media types.

Consider for instance the case of collaborative learning systems, where digital collections of educational materials or Learning Objects (LOs), such as, lecture videos, notes, and presentations, are made available in online repositories. Online learners are not only able to browse or search for LOs, but also enrich this content with value-added metadata. Learning object enrichment is crucial within a collaborative setting. For example, consider a scenario in a collaborative environment where a user wants to retrieve specific documents related to his interests and uses tags to navigate to the associated resources. Ideally, if the system can effectively provide good tag coverage over the resources, the user can better navigate through document objects and be steered to the relevant resources in the system. On the contrary, if tags are either unclear, not specific for the resource, noisy, or ambiguous, then users cannot retrieve or easily locate resources. Unfortunately, the latter situation is all too common. Since users typically only tag a small fraction of the documents, most of the other documents have no associated metadata. Furthermore, newly added resources, which have not yet been tagged, are hard to be located or associated with related objects. This dilemma is well known and referred to as the *new item cold start problem.*

As outlined in the aforementioned scenario, an important prerequisite for realizing the benefit of tags in a collaborative learning setting, is that a LO actually has to have at least a minimum number of tags associated with it. When a learning resource has no associated tags, the collaborative learning system cannot provide useful recommendations, nor does any descriptor exist in the tag cloud to help support navigation to the orphan resource.

One way to address the cold start problem in collaborative systems is by using automatic tagging, which associates tags with untagged resources. State-of-the-art work in this area relies upon latent data models to make explicit, some hidden, underlying "context" (i.e., set of keywords or tags). The untagged resources are treated as a new resource, for which inferencing can be performed to bring the new resource into a known context where it can inherit an appropriate set of tags. Little has been done in the area of latent model based automatic tagging for learning objects repositories. Moreover, the usability and effectiveness of such automatic tagging has not been assessed by the learners themselves.

In total, the key contributions of this chapter are:

- $\alpha-TaggingLDA$: an approach in which an untagged item is annotated by exploiting the content from similar resources found outside the boundaries of a single site.

- A detailed experimental evaluation of our automatic tagging approach, which shows how to efficiently address the cold start problem in collaborative learning environments by relying on content from resources in an auxiliary domain, i.e., one that lies outside of the content repository of the untagged LO.

In the rest of the chapter, we introduce our model, then we first perform an evaluation on data from the social bookmarking system BibSonomy, and after that, we come back to the collaborative learning system scenario and explore in detail the application of our approach. Finally, we present related work and discuss the lessons learned.

5.2 The Proposed LDA-based Method

Consider a *folksonomy* as a four-tuple, $\mathbb{F} := (U, T, R, Y)$ [Jäschke et al., 2008], where:

- U, T and R are finite sets, whose elements are called users, tags and resources, respectively, and
- Y a ternary relation between them, i.e. $Y \subseteq U \times T \times R$, whose elements are called tag assignments.

The set of all tags that user $u \in U$ has assigned to resource $r \in R$ is defined as $T(u, r) := \{t \in T \mid (u, t, r) \in Y\}$ and the set of all *posts* of the folksonomy as $P := \{(u, T(u, r), r) \mid u \in U, r \in R, T(u, r) \neq \emptyset\}$.

The goal of *automatic tagging* consists of automatically annotating a given resource $r \in R$, with a set of tags $\widetilde{T}(SYS, r) \subseteq T$, where $SYS \in U$ is a special user representing the system.

Our approach, α-*TaggingLDA*, is based on the probabilistic topic model: Latent Dirichlet Allocation (LDA) [Blei et al., 2003], which is a generative probabilistic model for collections of discrete data such as text corpora. The basic idea of LDA is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over terms. Please refer to Section 2.4 for a brief introduction on LDA.

For example, an LDA model might have topics that can be labeled as Education and Entertainment [2]. Furthermore, a topic has probabilities of generating various words such as *school*, *students*, and *teacher*, which can be classified and interpreted as Education. Naturally, the word *education* itself will have high probability given this topic. The Entertainment topic likewise has high probability of generating words such as *film*, *music*, and *theater*.

In order to illustrate the method with an example, consider a novel LO entitled *Knowledge Technologies in Context*, as shown in Figure 5.1, this resource is new to the collaborative learning system and does not

[2]Please note that these labels are arbitrary. The algorithm does not automatically assign any particular label to the latent topics.

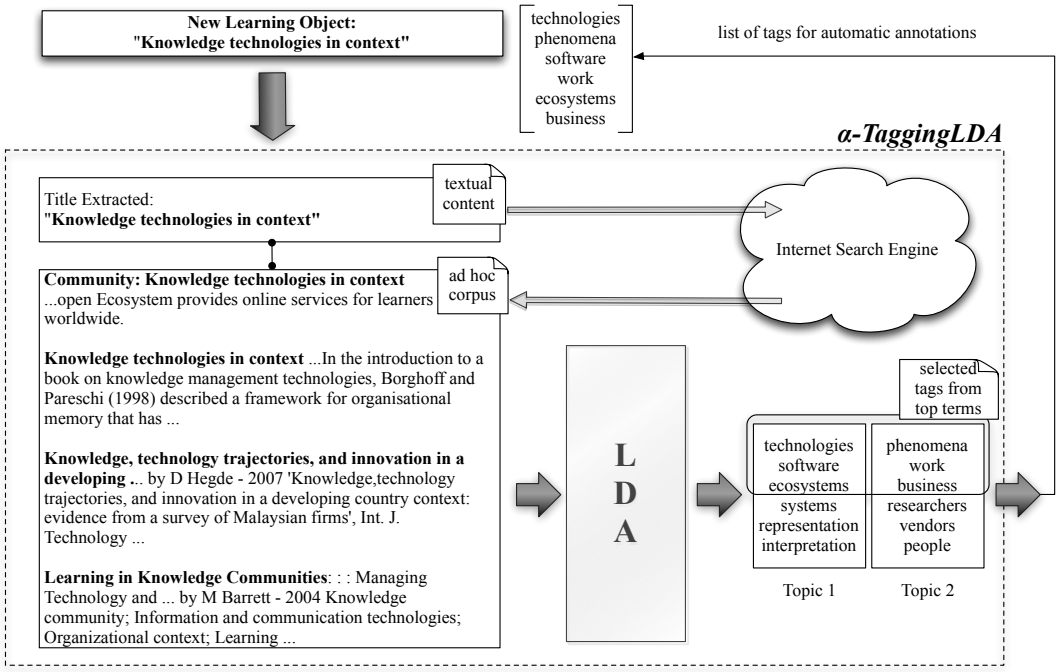


Figure 5.1: ***α-TaggingLDA* is applied to annotate a new LO: *"Knowledge Technologies in Context"*, with a list of six tags:** $TopN_{tags}(LO) =$ { ***technologies, phenomena, software, work, ecosystems, business*** }**, based on two LDA topics.**

have any tag annotations assigned. The absence of tags makes it difficult for the system to consider it as candidate for recommendations, for instance.

α-TaggingLDA first extracts relevant LO's *textual content*, such as the title, description or metadata (e.g., author) and creates a document denoted as d_{LO}. Then, the LO is associated to a set of 'similar' documents, which we refer to as an *ad hoc corpus* for the LO, represented as $corpus_{LO}$.

Note that the *α-TaggingLDA* method does not impose any restriction on the similarity measure used to associate the corpus with the LO. The similarity measure could be specified based on the nature of the resources, (e.g., text documents, multimedia items) and the textual content or metadata available. For example, a particular implementation might rely upon a computationally inexpensive similarity measure or on a more complex clustering algorithm.

In our particular example (Figure 5.1), the title of the LO is used to query an Internet search engine in order to retrieve the title and snippets of the n relevant results ($n = 4$, in this case). This subset corresponds to $corpus_{LO}$.

The LO's textual content is extracted and the subset of the top n results constitutes the text collection $D = \{d_{LO}\} \cup corpus_{LO}$, which is input to LDA, together with the number of topics required. In this example, the number of topics is set to two, i.e., $|Z| = 2$. The set of tags to be used to annotate the LO is denoted as $TopN_{tags}(LO)$, and its size is set to six for this particular case, i.e., $|TopN_{tags}(LO)| = 6$.

Table 5.1 presents an example of the output produced by LDA according to the setting described above. Topics are ordered based on the document-topic distribution $P(z \mid d)$, and within each topic, terms are ranked based on the topic-term $P(t \mid z)$ distribution.

For the construction of the final set of tags $TopN_{tags}(LO)$, α-*TaggingLDA* selects the first candidate tag from $Topic_1$'s top terms, the second tag from $Topic_2$'s top terms, the third tag, again from $Topic_1$'s top terms, and so forth. The final list of tag annotations for the LO in our example corresponds to $TopN_{tags}(LO) =$ { *technologies, phenomena, software, work, ecosystems, business* }.

$\boldsymbol{Topic_1}$		$\boldsymbol{Topic_2}$	
$P(z = 1 \mid d_{LO}) = 0.70$		$P(z = 2 \mid d_{LO}) = 0.30$	
Term t	$P(t \mid z = 1)$	Term t	$P(t \mid z = 2)$
technologies	0.45	phenomena	0.33
software	0.25	work	0.28
ecosystems	0.16	business	0.19
systems	0.11	researchers	0.15
representation	0.03	vendors	0.04
interpretation	0.01	people	0.01

Table 5.1: **Example of two topics output by LDA. Topics are ordered based on the document-topic distribution $P(z \mid d)$, and within each topic, terms are ranked based on the topic-term $P(t \mid z)$ distribution.**

Note that a LDA model is created *on-the-fly* for the resource and the associated similar documents, and it is discarded after the list of tag annotations is inferred. Figure 5.1 summarizes our α-*TaggingLDA* approach.

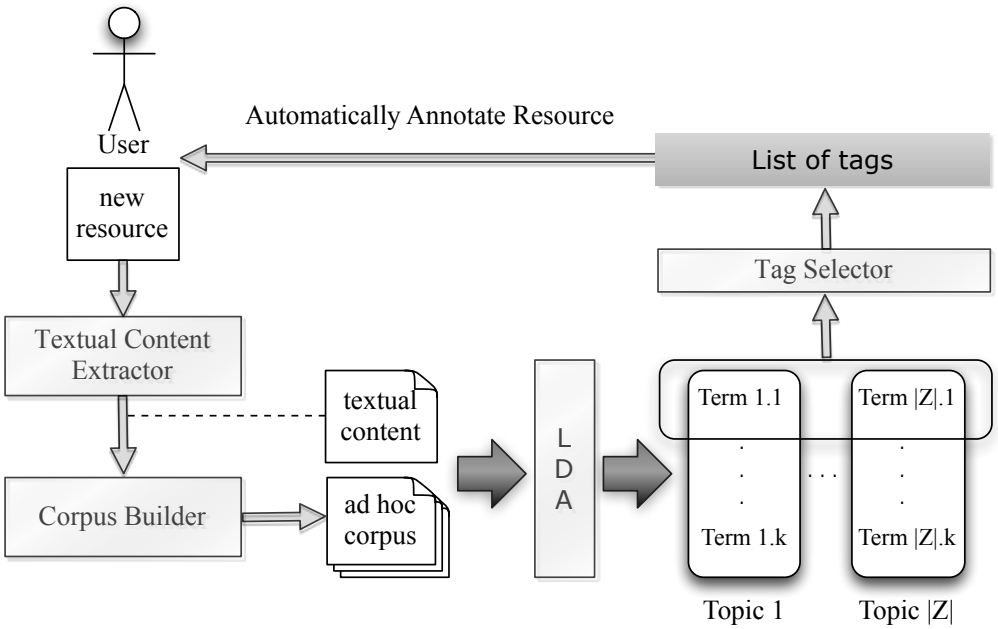


Figure 5.2: α-***TaggingLDA*** **general method.**

5.2.1 Concrete Realization

The concrete realization of α-*TaggingLDA* used in our experiments is implemented in Java. The corpus builder (Figure 5.2) is based on the search results obtained by querying Yahoo!'s open search web services platform (BOSS)[3]. The titles and short text summaries (snippets) of the top-10 results returned are used to create ten different textual documents. The final ad hoc corpus for the resource consists of these and the textual content of the resource. Then, by applying LDA on this corpus we extract the desired number of latent topics, and from them, the required tags are inferred. We use the LDA with Gibbs sampling implementation provided by the Machine Learning for Language Toolkit (MALLET) [McCallum, 2002].

[3]**Yahoo! BOSS:** developer.yahoo.com/search/boss/ .

5.3 Evaluation on BibSonomy

To evaluate the efectiveness of our approach, the problem of automatic tagging is cast as a recommender system task. In this section, we present two evaluations on data and services from BibSonomy.

5.3.1 Offline evaluation

First we evaluate our α-*TaggingLDA* method on a BibSonomy dataset from [Eisterlehner et al., 2009] and described in Section 2.5. This dataset is almost a complete dump of BibSonomy, i.e., all users, resources (publication references and bookmarks) and tags publicly available until December 31st, 2008. All tags are lowercased and a cleansing process was applied to the data. To use as textual resources we extract the url and description available from bookmarks and the following fields from bibtex entries: author, editor, title, abstract, journal, booktitle, notes and description.

Baselines

We compare the performance of our method against two baselines. The first one (*baselineMP*), relies on the most specific tags of a resource. For a given user $u \in U$, a given resource $r \in R$, and some $n \in \mathbb{N}$ the top-n *most popular tags by resource* are given by:

$$\widetilde{T}(u,r) := \underset{t \in T}{\overset{n}{\operatorname{argmax}}}(|Y_{t,r}|)$$

where $Y_{t,r} := Y \cap (U \times \{t\} \times \{r\})$, for $t \in T$ and $r \in R$, which corresponds to the frequency of tag assignments on r having the tag t. When resources have just a few number or zero tag assignments, we complement the set with the *most popular tags of the folksonomy*:

$$\widetilde{T}(u,r) := \underset{t \in T}{\overset{n}{\operatorname{argmax}}}(|Y_t|)$$

where Y_t is the set of all tag assigments having tag $t \in T$, and is defined as $Y_t := Y \cap (U \times \{t\} \times R)$.

The second baseline corresponds to a LDA-based tag recommender introduced in [Krestel et al., 2009] and evaluated on the same datasets and splits as ours in [Krestel and Fankhauser, 2009], the values of their evaluation on the corresponding test splits are reported here as baseline and are identified as $baselineLDA$.

Observe that strategies based on tensor factorization have proved to be highly competitive for the tag recommendation task, e.g., [Rendle et al., 2009a; Rendle and Lars, 2010]. Since these approaches do not exploit any content information, their ability to deliver high-quality recommendations depends on the user interactions within the system and the density of the dataset. Therefore, the performance of such methods suffers in cold start scenarios, e.g., where a very limited number of resources is tagged by users. We tackle the cold start problem with our approach.

Measures

For the evaluation, we used the test data splits provided also by [Eisterlehner et al., 2009]. For a given user $u \in U$ and a given resource $r \in R$, the test data consists of a set of posts without tag assignments, i.e., $P^{test} := \{(u, S, r) \mid u \in U, r \in R, S = \emptyset\}$. The system has to compute the set of tags for this posts $S = \widetilde{T}(u, r)$ to complete the tag assignments. Some statistics about the test data splits are presented as follows:

$\lvert U^{test} \rvert$	$\lvert R^{test} \rvert$	$\lvert R^{test} \setminus R \rvert$	$\lvert P^{test} \rvert$
1,591	43,002	39,070	43,002

In the evaluation, the list of recommendations consists of five different tags, i.e., $|\widetilde{T}(u, r)| = 5$. The number of iterations of the underlying LDA algorithm is set to 100.

As performance measures we use precision and recall and F1 measure which are standard in such scenarios [Herlocker et al., 2004].

For each post $(u, T(u,r), r)$ we compute precision and recall as defined in Equations 5.1 and 5.2, respectively:

$$\textbf{precision} := \frac{|T(u,r) \cap \widetilde{T}(u,r)|}{|\widetilde{T}(u,r)|} \tag{5.1}$$

$$\textbf{recall} := \frac{|T(u,r) \cap \widetilde{T}(u,r)|}{|T(u,r)|} \tag{5.2}$$

We then average these values over all posts in the given set and compute the F1 measure as follows:

$$\textbf{F1} := 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.3}$$

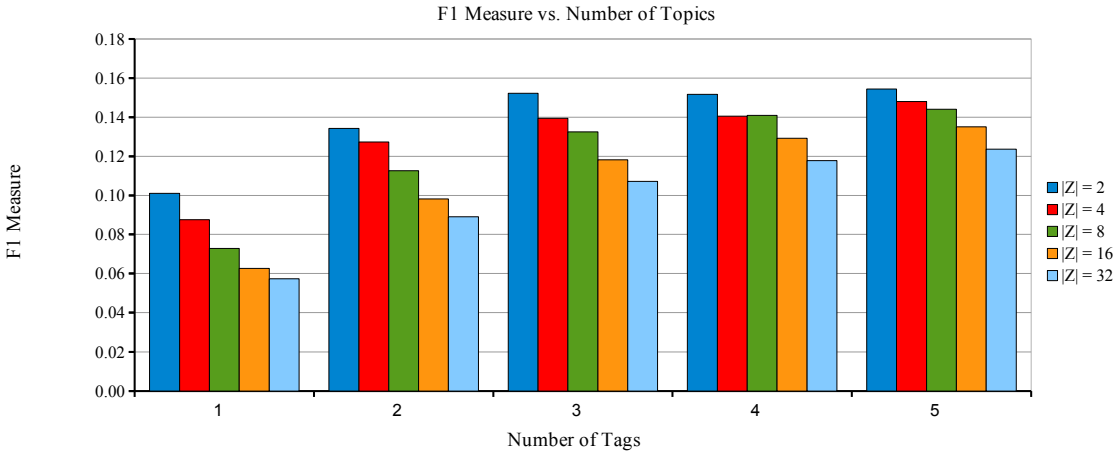
5.3.2 Online Evaluation

We deployed our implementation as a recommender system on the BibSonomy recommendation framework according the guidelines described in [Eisterlehner et al., 2009; Jäschke et al., 2009].

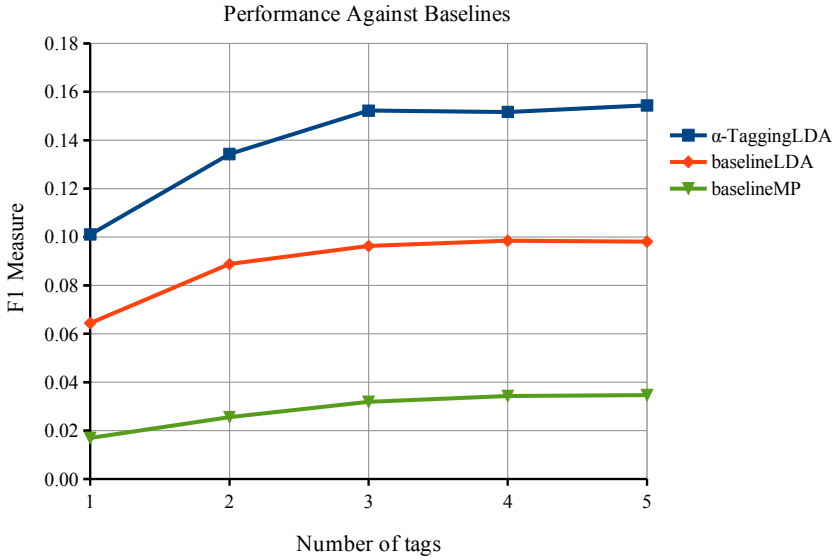
The online evaluation took place from July 27^{th}, 2009, until September 1^{st}, 2009. More than 200 users received recommendations. The recommendations consisted of a list of 5 tags. The number of posts for we delivered tag recommendations is 11,102. For the online evaluation, we set the LDA parameter to produce two *general* topics and fixed the number of iterations to 50.

5.3.3 Results and Discussion

The behavior of our method varying the number of topics(|Z| = 2, 4, 8, 16, 32) is shown in Figure 5.3a. As can be seen in the figure, performance decreases with the LDA topic size. A solution with few topics will typically result in broad topics whereas a solution with too many topics will result in uninterpretable topics that pick out idiosyncratic tags. The results on the datasets explored, suggest that such broad topics have higher chance to produce tags general enough to explain the limited document collection, leading to a higher recall and precision.



(a) F1 measure for different number of topics and tags.



(b) F1 measure against baselines($|Z|$=2).

Figure 5.3: **F1 measure.**

Offline Evaluation

The prediction quality of *α-TaggingLDA* with two general topics is clearly superior to the one of the baselines (Table 5.2, Figure 5.3b) achieving a F1@5= 15.43% (i.e., F1 measure evaluating 5 tags). Given the high number of unseen resources in this dataset, a solution based on relational information only, such as the most popular tags by resource, is expected not to perform well, in this case, the *baselineMP* achieves a of F1@5= 3.5%. Surprisingly, the LDA baseline method only achieves a F1@5 of 9.8%. This can be explained on how this method represents

the resources of the system. Each resource is considered as a *bag of tags*, without exploiting any content feature. A LDA model is built using the whole corpus of available resources, i.e., bag of tags, in the training set, the model is then applied on test resources to infer the tag recommendations. For unseen resources, i.e., without tags, the method fails to produce a suitable representation and performs suboptimal in this, more realistic, sparse dataset.

#tags	baselineMP			baselineLDA			α – taggingLDA		
	recall	precision	F1	recall	precision	F1	recall	precision	F1
1	0.01075	0.03995	0.01694	0.04113	0.14797	0.06436	0.0654	0.2218	**0.1010**
2	0.01961	0.03674	0.02557	0.06876	0.12523	0.08878	0.1058	0.1837	**0.1342**
3	0.02756	0.03780	0.03188	0.08723	0.10738	0.09626	0.1395	0.1675	**0.1522**
4	0.03257	0.03616	0.03427	0.10196	0.09518	0.09845	0.1578	0.1459	**0.1516**
5	0.03528	0.03408	0.03467	0.11358	0.08630	0.09808	0.1768	0.1369	**0.1543**

Table 5.2: **Precision, recall and F1 against baselines.**

Online Evaluation

In the online setting, the average time period the recommender needs for delivering a list of tag recommendations is 1630.58 milliseconds[4]. The results obtained during the online evaluation are shown in Table 5.3.

# tags	recall	precision	F1
1	0.06875	0.22500	0.10532
2	0.10625	0.19000	0.13629
3	0.13125	0.18000	0.15181
4	0.15000	0.16500	0.15714
5	0.15469	0.15000	0.15231

Table 5.3: $\boldsymbol{\alpha}$**-***TaggingLDA* **online performance.**

[4]The results presented in this work ignore any possible timeouts in the process.

5.4 Automatically Tagging Learning Objects

Cold start is a common problem in many user-centric systems that seek to improve information access. Specifically for the collaborative learning environment setting, a new LO is introduced into the domain, but it has no (or incomplete) associated user-defined metadata or annotations. Automatic enhancement of LO metadata is an alternative to address this problem.

In this section, we apply α-*TaggingLDA* to automatically associate tag annotations to untagged LOs by exploiting the content from different, but similar resources, found outside the boundaries of a single content repository. In doing so, we will address the following research questions:

> **Q1:** To what extent do the LO's annotations assigned by the authors agree with the ones assigned automatically?
>
> **Q2:** From the user perspective, how relevant are the automatic generated tags comparing to the ones provided by experts?
>
> **Q3:** In a social tagging recommendation scenario for LOs, are the automatic annotations better candidate terms for assisting users in the tagging process than the keywords assigned by the LO's author?

To answer these questions, we conducted three distinct evaluations, first, an experimental evaluation followed by two user studies. The rest of the section describes each evaluation settings. We based our experiments on a dataset sampled from the OpenScout project collection [Niemann et al., 2010], which is described in Section 2.5.

5.4.1 Evaluation I: Author's Keywords and Automatic Tags

Our first evaluation consists of an offline study that measures the agreement between the keywords assigned to the LOs by its author and the

tag annotations provided by α-*TaggingLDA*. In order to quantify such agreement, we consider a recommender system setting, where the author's keyword assignments constitute our test set. The task of the collaborative learning environment is to recommend $TopN_{tags}$ relevant tags for a given LO.

In this experiment we also use recall, precision and F1 to assess the performance. In this context, the metrics are defined as follows:

- Recall for a given author u and a learning object i is defined as:

$$\textbf{recall} := \frac{|Keywords(u,i) \cap TopN_{tags}(i)|}{|Keywords(u,i)|} \quad , \; and \tag{5.4}$$

- Precision for a given author u and a learning object i is defined as:

$$\textbf{precision} := \frac{|Keywords(u,i) \cap TopN_{tags}(i)|}{|TopN_{tags}(i)|} \quad , \tag{5.5}$$

where $Keywords(u,i)$ is the set of keywords assigned by the author u to the learning object i and $TopN_{tags}(i)$ is the set of size N corresponding to the tags automatically assigned by α-*TaggingLDA* to the given LO. In this experiment we set $N = 10$.

For the dataset, we averaged these values over all the authors. The aggregated values of recall and precision are then used to compute their harmonic mean or F1 measure as defined according to Equation 5.3.

5.4.2 Evaluation II: Guided Choice User Study

The goal of this experiment was to compare the automatically generated tags against the ones provided by experts.

This evaluation is a user study in which each participant was presented with basic information regarding a learning object, namely, the title and an abstract that varies from 20 up to 200 words (see Figure 5.4). The format of the original resource (e.g., video, image, presentation or document) was not made known to the participant in order to align the nature of the evaluation and to avoid biased judgments of the tag relevance based on non computer-understandable information.

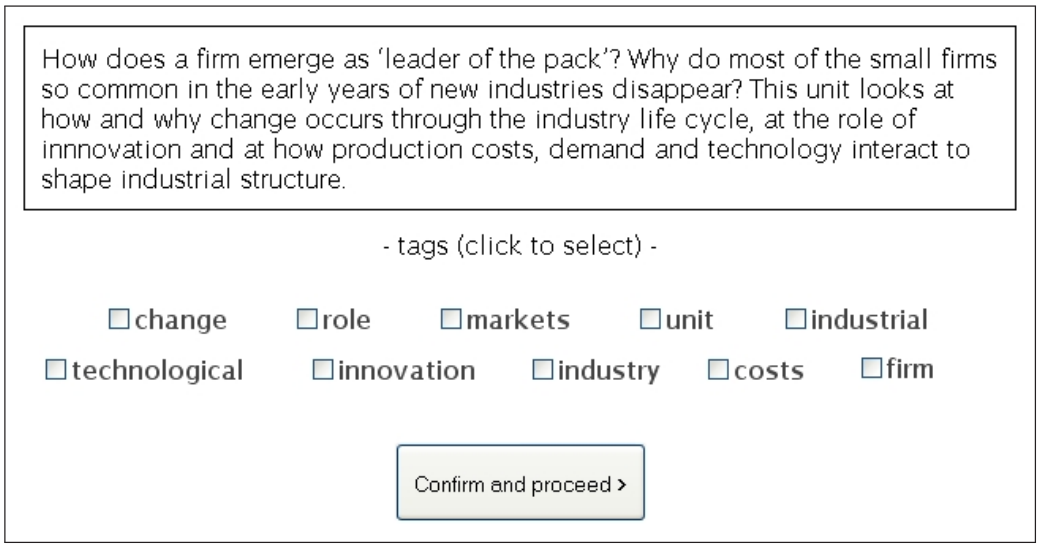


Figure 5.4: **Evaluation II: Guided Choice User Study Interface. Each participant was instructed to choose at least three tags from the set of ten suggested tags. Five tags were originally added by the expert/author of the content, while the remaining five were automatically generated. The tags were presented in a random order and their origin was not disclosed to the participants.**

In addition to that, the participants were presented with ten tags to be evaluated. From the ten tags presented, five tags were originally added by the expert/author of the content, while the remaining five were the top ranked automatically generated ones. The tags were presented in a random order and their origin was not disclosed to the participants.

Each participant was then instructed to read the title and the description of the learning object and finally choose at least three tags from the set of ten suggested tags. Once the submission of the form is completed the participant was presented with a new object to be evaluated. We kindly asked each participant to repeat the process for at least ten objects, however, we did not limit the maximum of their contribution to the study.

In order to compare the automatically generated tags against the ones provided by experts, we designed this experiment as a recommendation task, and used the recall measure, which is widely used to assess the recommendation quality [Herlocker et al., 2004] of recommender systems.

In this case *recall* for a given author u and a learning object i is defined in Equation 5.6, as follows:

$$\textbf{recall} := \frac{|Tags(u,i) \cap TopN_{tags}(i)|}{|Tags(u,i)|} \quad , \tag{5.6}$$

where $Tags(u,i)$ is the set of tags assigned by the user u to the learning object i and $TopN_{tags}(i)$ is the set of size N corresponding to the tags recommended to the user for learning object i, either based on α-*TaggingLDA* or on the author's keywords. In this experiment we set $N = 5$.

As in Evaluation I, we averaged the values over all the participants. Using a fixed number of recommendations, precision is just the same as recall up to a multiplicative constant and thereby there is no need to evaluate precision.

5.4.3 Evaluation III: Free Choice User Study

The goal of this study was to collect evidence to evaluate if the automatic annotations are better candidate terms for assisting users in the tagging process than the keywords assigned by the LO's author.

Similarly to Evaluation II, in this user study, each participant was presented with the title and an abstract of a learning object. Once again, due to same reasons as presented before, the format of the original resource (e.g., video, image, presentation or document) was not disclosed to the participants.

Each participant was then instructed to read the title and the description of the learning object and finally input five tags she thinks to be relevant for describing the object, as depicted in Figure 5.5. Once the submission of the form was completed, the participant was presented with a new object to be evaluated. Each participant was asked to repeat the process for at least ten objects.

As in Evaluation II, in order to evaluate this experiment we cast it as a recommendation task and evaluate the recall measure. In this case $Tags(u,i)$ corresponds to the set of tags that would be recommended to the participant u for the given LO i.

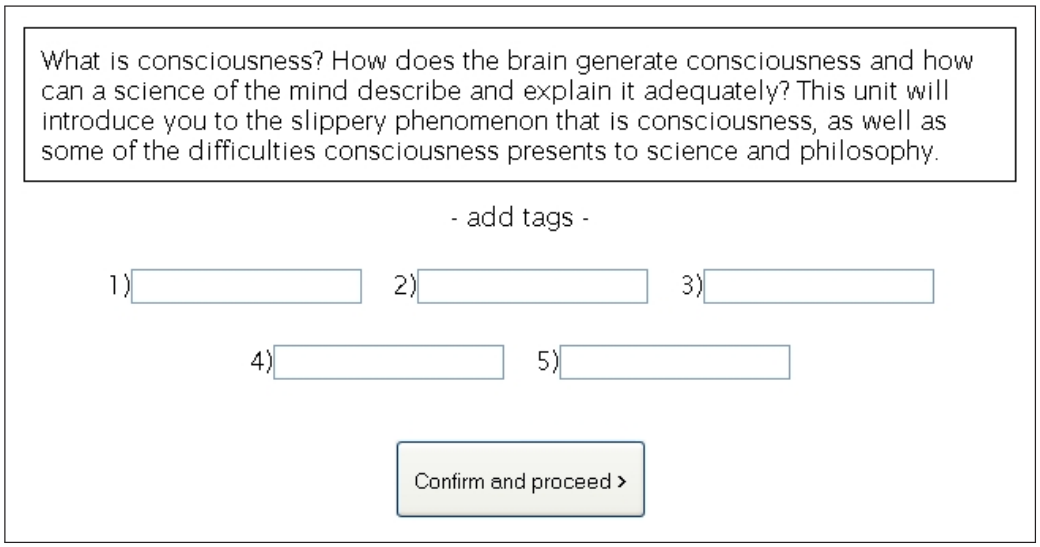


Figure 5.5: **Evaluation III: Free Choice User Study Interface. Participants were instructed input five tags they think to be relevant for describing the LO.**

Note that, even though, the set of tags is not presented to the participant, it helps us to measure which terms are better for assisting users in the tagging process.

5.4.4 Results

Evaluation I aims to answer the research question *'Q1: At what extent do the LO's annotations assigned by the authors agree with the ones assigned automatically?'*. As an outcome for $|TopN_{tags}| = 10$, we obtained the following results, recall=0.26 precision=0.13 and F1=0.18. Table 5.4 shows the F1 measure for different sizes of $TopN_{tags}$.

$\boldsymbol{\|TopN_{tags}\|}$	1	2	3	4	5	6	7	8	9	10
F1	0.04	0.07	0.09	0.10	0.10	0.11	0.12	0.13	0.16	0.18

Table 5.4: **F1 measure for different sizes of $TopN_{tags}$.**

From the user study in Evaluation II we collected the feedback of 115 participants (43 female and 72 male), 100 of them explicitly stated to be students. Their average age was 24, ranging from 20 to 53 years old. In total the participants evaluated 1,134 objects covering 478 unique ones.

Also, in total 4,035 tags were chosen to represent the documents, in average each participant picked 3.56 tags per document. As explained in the setup of this evaluation, the tags exposed to the participants were originated from two different sources, the expert who created the learning material and a second set from the automatic tagging method.

It's important to note that the tags from each group were always presented to the participants in an equal distribution to preserve the fairness of the study. Additionally, when a participant chose a tag that was in both α-*TaggingLDA* set and the Experts' set we computed this choice as two tag assignments as outlined in Section 5.4. Although the participants chose 4,035 tags, in our experiments, we used a total of 4,939 tag assignments. Out of the 4,939 tag assignments, 67.5% of them were originated by α-*TaggingLDA* and 32.5% by the experts (Table 5.5).

The most straightforward analysis of these results shows a clear preference of the participants for the tags that were automatically added. Thus, it is also reasonable to conclude that these tags are more relevant to the participants, which answers our second question: '*Q2: From the user perspective, how relevant are the automatic generated tags comparing to the ones provided by experts*?'. The main reason is that the underneath approach generates tags that represents better the learners tagging behavior. Through the outcomes of this evaluation we interpret that the automatic generated tags are, in general, more descriptive and more useful for the learners than experts' tags. Additionally, it is reasonable to assume that these learners, when searching for one of these documents would (with a higher probability) use a tag that was automatically generated rather than the experts' tags. The same assumption is valid for the case of browsing resources in a hierarchical classification or in a facet browsing interface.

To validate the significance of the results achieved, for each participant, we took the averages of the distribution of α-*TaggingLDA* and Expert's tag sets. With two groups of 115 samples we performed a two-tailed t-test that confirmed a statistically significant difference of α-*TaggingLDA* mean (68.2%) and Expert's keywords mean (31.8%).

		Experiment II: Guided Choice		Experiment III: Free Choice	
	Participant's TAS	4939	-	4745	-
Sets	α-*TaggingLDA*	3336	67.5%	1824	38.4%
	Experts	1603	32.5%	983	20.7%

Table 5.5: **Tag Assignment (TAS) results for the user evaluations. The *Sets* rows show the number of TAS that were chosen by the participants that overlapped with TAS given by the experts, or with α-*TaggingLDA* method and the respective recall measure.**

For Evaluation III, the participants evaluated 832 objects covering 454 unique ones. In this phase, where the participants were instructed to freely choose terms that best classify the objects, 4,745 tags were generated (1,868 unique tags).

Using these data we now have three different sets of tags: α-*TaggingLDA* tags, experts' tags and learners' tags. By validating the learners' generated tags against the other sets we found an overlap of 38.4% with automatic generated α-*TaggingLDA* tags and 20.7% with the experts' tags (Table 5.5). Additionally, in only 8.9% of the cases, a tag occurs in all three sets. At this point we are just considering the whole sets of tags and not the precision of them regarding each resource. These results complement Evaluation II by firmly stating that on average the automatic generated tags are closer to the ones used by learners.

By considering only the results from those 100 participants that stated to be students, the numbers do not change significantly. The overlap with automatic generated tags increases slightly to 39.04% while the overlap with the experts' tags remains on the same levels (20.3%). These results help us to answer the third question – Q3, as the automatic annotations turned out to be the best candidate terms for assisting users in the tagging process.

The values of recall and precision of Evaluation I (Section 5.4.4) suggest that the information captured by the automatic tag annotations partially agrees with the expert keywords assigned to the LO. The values are not exceptionally high, which suggest that the automatic tag annotations tend to capture different information than the expert keyword assignments. The user studies conducted in Evaluation II

and III exposed how additional information captured by the automatic annotations are perceived by the learners, and explore the usability improvements of the collaborative learning system.

The results from the first user study setup (Evaluation II) clearly demonstrate the preference of the participants for tags produced by the automatic tagging method. This means that, the produced tags reflect better the participants' preferences in comparison to the experts' keyword assignments. The most probable reasons are, first, the aforementioned problem that a tag assignment is not always clear to users other than its creator. Second, learners usually have a viewpoint that differs from the experts, thus they are more prone to avoid terms that are too specific or that they would probably not remind later. Finally, the terms given by the automatic tagging, extracted from search results' snippets, represent better the wisdom of the crowd since these results are originally extracted from multiple resources. It is also important to remark that the search results themselves are consequence of ranking algorithms that exploit collective knowledge and preferences.

In principle, the results of the second user study (Evaluation III) support the same benefits. The goal of this phase was to prevent any possible biases in the first evaluation. We hypothesize that, when asking a participant to tag a learning object, we are implicitly observing which tags the participants would use in a collaborative social learning environment, and indirectly potential terms to query or browse for a learning object.

Bearing in mind the overall results obtained in the experiments, the most important consideration to highlight is the potential benefits produced by the information delivered by the automatic tagging method evaluated.

5.5 Related Work

In this section, we present related works in two main areas, namely Automatic Tagging and Learning Object Enrichment.

Automatic Tagging

Latent data models have been used to expose some hidden stucture or "context" to suggest tags for enhanced information access and collaborative tag recommendations. By context we refer to some meaningful aggregation of resources such as: association rules [Heymann et al., 2008] or user/system defined clusters[Abel et al., 2007; Song et al., 2008]. In each of these cases, properties of aggregated resources increase overlap, which can be exploited to derive tag information about the resource on the Web.

Latent approaches treat the automatic suggestion of tags by relying upon dimensionality reduction: such as Latent Dirichet Allocation. In [Krestel et al., 2009; Krestel and Fankhauser, 2009] resources annotated by many users and thus having a relatively stable and complete tag set are exploited to overcome the cold start problem. They build an LDA model from tags which have been previously assigned by users. In this way, a resource in the system is represented with tags from topics discovered by LDA. For a new resource with few or no annotations, they expand the latent topic representation with the top tags of each latent topic. The work of [Phan et al., 2008] external knowledge from a large, so called "Universal Dataset" is used to address textual sparseness in the classification of short segments of text such as chat messages, or news feeds. They learn a LDA topic model from both a small set of labeled training data and the universal dataset. The model is then exploited to discover a set of latent topics which are subsequently used as the target in a multi-class classifier for the original sparse text.

In contrast to model based systems, instance based approaches do the association between users and annotations on-the-fly. For example, in Cross-Tagging [Stewart et al., 2009], information accesses is enhanced for a non-folksonomy user, such as a music blogger, by exploiting the

tag assertions made by (similar) users of folksonomies. The overlap between the mention of tracks in a music blog, and the tracks in LastFM is determined. The user-resource-tag triples are modeled with a tensor; exploiting the underlying latent semantic structure in the tensor to form multi-way correlations between users, tags, and resources.

In these auto tagging systems, the performance of the aforementioned approaches highly relies on the assumption of a dense set of data upon which the model can be built. To overcome this issue, we introduced in this chapter α-*TaggingLDA*, a method for automatic tagging resources with sparse and short textual content.

Learning Object Enrichment

Lohmann et al. [Lohmann et al., 2008] demonstrate the importance of additional metadata to learning resources visibility and reusability and suggest design guidelines for automatic tagging approaches. The authors suggest (i) the use of a stable set of tags for agreed description of resources, (ii) to guide the tagging process (e.g., with tag recommendations), (iii) to use text extracted from resources for starting set of tags, and (iv) the use of a small set of selectable tags for tags convergence. The two user evaluations we present in this work align with these guidelines.

Another recent system, namely, ReMashed [Drachsler et al., 2010] takes advantage of tagged and rated data of combined Web 2.0 sources, integrating the metadata from decentralized sources of content. Their work addresses the *new user cold start scenario* and shows that a recommender system that exploits already tagged resources can mitigate the lack of user information. Our work, on the other hand, focuses on the *new item cold start problem*, and aims to annotate untagged LOs. Once objects are tagged, it is possible to improve the performance of a recommender system [Stewart et al., 2009].

Abel et al. [Abel et al., 2009] introduce the LearnWeb 2.0 environment,which supports sharing, discovering, and managing learning resources, which are spread across different Web 2.0 platforms, between learners and educators.

LearnWeb 2.0 aggregates resources and enhances their metadata using functionalities from ten different Web 2.0 services. Furthermore, in order to support collaborative searching, the authors are provided with an automatic resource annotation service. Once a search result is displayed in the environment, it is automatically tagged with the corresponding query terms. This mechanism assumes that the system has enough information to retrieve the item as relevant for a given query. Furthermore, it requires an initial user interaction, i.e., search, in order to be able to annotate the resource. This is not necessarily the case for resources with sparse text, or multimedia resources with little or no metadata available. Our approach is content based and does not require any user interaction to automatically annotate the LOs.

In contrast to previous work, we aim to evaluate the usability and effectiveness of this automatic tagging approach, and address its potential to generate metadata for novel resources in the context of a collaborative learning environment.

5.6 Discussion

In this chapter, we presented an approach to addressing the dynamics associated with online environments where novel items appear rapidly. We show the ability of our method, α-*TaggingLDA*, to enrich sparse and limited textual information by means of exploiting the resource redundancy and latent topic overlap between similar resources found in an auxiliary domain.

We empirically evaluate, both offline and online, the effectiveness of our approach addressing the cold start problem on a collaborative tagging recommender scenario.

The online deployment of our method demostrates its efficiency and scalability delivering high quality recommendations in real time, without requiring any expensive offline model computation or updates. We believe that our approach would be ideally suited as part of a complementary solution for bootstrapping Web 2.0 social information systems.

We have empirically demonstrated through a series of evaluations that the proposed α-*TaggingLDA* method produces quality metadata enhancement for the learning objects. First, by experimentally comparing against existing authors' tag annotations. Second, by running a user study comparing the participants' preference for automatically produced tags against the authors' tags. Finally, a last user study that demonstrated that α-*TaggingLDA* tags are the best candidate terms for assisting users in the tagging process.

The additional metadata that was automatically generated by our method can improve personal recommendations of learning objects and most notably it overcomes the 'cold start' for objects that are not tagged, consequently isolated from the rest of the folksonomy.

Our approach faces some limitations, since we depend on the external resources provided by a search engine. One implication of this shortcoming is that the collection of the documents retrieved may not contain enough meaningful text for good topics to be generated. Another implication is that in some cases, there may be valuable documents for enriching the learning resources, but the documents may not be available if they are buried in the "Hidden-Web", i.e. documents that are not indexed by search engines. One potential solution to, at least, mitigate this limitation is to use multiple and heterogeneous sources for building the topic model. Heterogeneity would include the use of multiple search engines, and open information sources such as wikipedia. Future experiments are needed to examine heterogeneous sources, and we consider this in future work.

Note that our automatic tagging approach, as in the case of recommender systems, can have a bias effect towards a particular algorithm, affecting the folksonomy evolution and the emergence collective intelligence. But in dynamic scenarios where novel items appear rapidly, e.g., news articles, and in domains where it is difficult to reach a critical mass of users and a stable set of tags, approaches like ours represent a powerful alternative to overcome data sparsity and cold start problems. This represents a fundamental trade-off that should be consider by practitioners.

We plan to evaluate how our approach could be further refined to assist authors in tasks of keyword assignment by recommending them relevant terms for the LO. We are interested in exploring how automatically added tags can be incorporated in user's profiles and to what extent it can improve recommendations and discovery of new items. Although our work focuses on approaching the cold start problem, we are also interested in running an evaluation with learning objects that have already been enriched by an active community. This would provide us valuable insights to compare the automatic generated tags based on the general wisdom of the crowd and the focused learning community.

In addition, we intend to evaluate how the tags we suggest can help with regard to recommending resources to the users, and we plan to evaluate our approach on items that are not textual like photos, video, music and other multimedia resources using the metadata. Early results in this direction suggest this to be challenging not only given the sparseness of such metadata, but the difficulty with which topics can be found, even after the enrichment from an auxiliary domain.

6 Social Media Analytics

The high rate at which users share their opinions on blogs, forums, and social networking sites, such as Facebook or Twitter, makes this kind of media attractive to measure collective behavior towards current affairs. Real-time access to the large amount of user generated content available can provide the tools to social researchers, public health agencies, and citizens in general, to monitor the *pulse* of the society towards specific topics of interest, a task traditionally accomplished only through costly analysis procedures and opinion polls, which are also time consuming to conduct, and therefore frequently limited to small sample sizes. Our goal in this chapter is to explore two application domains for Social Media Analytics: (i) Epidemic Intelligence and (ii) Political Emotion Detection.

Tracking Twitter for public health has shown great potential. However, most recent work has been focused on correlating Twitter messages to influenza rates, a disease that exhibits a marked seasonal pattern. In the presence of sudden outbreaks, how can social media streams be used to strengthen surveillance capacity? In the first part of this chapter, we seek to address the issues that can help deliver a public health surveillance system based on Twitter, by taking into account two important stages in epidemic intelligence: *Early Outbreak Detection* and *Outbreak Analysis and Control*.

In the second part, the chapter provides evidence of the potential uses of twitter in emerging regions. Considering that Latin America is not the exception of Twitter's global adoption, we provide an example of integration of sentiment analysis in Spanish and the real-time nature of Twitter using Latin America as test bed, that is, automatically detecting in the social media streams the sentiments and emotions towards political figures in this region.

6.1 Introduction

Epidemic Intelligence (EI) encompasses activities related to early warning functions, signal assessments and outbreak investigation. Only the early detection of disease activity, followed by a rapid response, can reduce the impact of epidemics. Recently, modern disease surveillance systems have started to also monitor social media streams, with the objective of improving their timeliness to detect disease outbreaks, and producing warnings against potential public health threats (e.g., [Corley et al., 2010]). The real-time nature of Twitter makes it even more attractive for public health surveillance. Recent works have shown the potential of using Twitter for public health. However, these works have either focused on: the text classification and filtering of tweets [Sofean et al., 2012; Sriram et al., 2010]; or finding predictors for diseases that exhibit a seasonal pattern (i.e., influenza-like illnesses) by correlating selected keywords with official influenza statistics and rates [Culotta, 2010; Lampos and Cristianini, 2010; Signorini et al., 2011]. Still others have focused on mining Twitter content for topic [Paul and Dredze, 2011a,b] or sentiment analysis [Chew and Eysenbach, 2009]. Furthermore, these existing approaches have all focused on countries where the tweet density is known to be high (e.g., the UK, or U.S.) [Semiocast, 2012].

In the first part of this chapter, we seek to address the issues that can help deliver a public health surveillance system based on Twitter, and take up the following questions:

1. *Early Outbreak Detection*: Is it possible, by only using Twitter, to find early cases of an outbreak before well established systems?

2. *Outbreak Analysis and Control*: Is it possible to use Twitter to understand the potential causes of contamination and spread?, and how can we provide support for public health official to analyze and assess the risk based on the available social media information?

In contrast to the aforementioned studies, ours focuses on a sudden outbreak of a disease that does not involve any seasonal pattern. Moreover, our work shows the potential of Twitter in countries where the tweet density is significantly lower, such as Germany. The contributions of this study are summarized as follows:

- We provide an example of the application of standard surveillance algorithms on Twitter data collected in real-time during a major outbreak of EHEC/HUS in Germany, and provide insights showing the potential of Twitter for early warning.
- For outbreak analysis and control, many studies have been made for systems that return documents in response to a query, little effort has been devoted to exploiting learning to rank in a personalized setting, specially in the domain of epidemic intelligence. This chapter presents an innovative personalized ranking approach that offers decision makers the most relevant and attractive tweets for risk assessment, by exploiting latent topics and social hash-tagging behavior in Twitter.

In the second part of this chapter, we study how social media streams can help monitoring people's emotions towards political figures. The explosion in Twitter's global user adoption over the past years also impacts Latin America. Brazil, Mexico, Venezuela, Colombia, Argentina, and Chile are among the top-20 countries in terms of Twitter accounts, as reported by a recent study by *Semiocast*, a provider of consumer insight and brand management solutions [Semiocast, 2012]. Twitter and other social media services allow users to express themselves, share their emotions and discuss their daily life affairs in real-time, covering a variety of different points of view and opinions, including political and event-related topics such as immigration, economic issues, tax policy or election campaigns. On the other hand, traditional methods tracking public opinion still heavily rely upon opinion polls, which are usually limited to small sample sizes and can incur in significant costs in terms of time and money. We leverage state-of-the-art techniques of sentiment analysis for real-time political

emotion tracking. In particular, we analyze mentions of personal names of 18 presidents in Latin America, and measure each political figure's effect in the emotions reflected on the social web. In summary the contributions of this study are:

- We present an extensive sentiment analysis of the political landscape of Latin America. To the best of our knowledge, this is the first study of this nature with a coverage of eighteen countries in the region.

- For polarity and emotion detection, many studies have been made in sentiment analysis. This chapter presents, not only the extracted emotions and polarity, but also goes a step forward and quantifies which combination of emotions explains better the public's opinion.

6.2 Epidemic Intelligence Based on Twitter

In May 2011, an outbreak of enterohaemorrhagic *Escherichia coli* (EHEC) occurred in northern Germany. It was one of the largest described outbreaks of EHEC/HUS worldwide and the largest in Germany [Frank et al., 2011].

Day 1: May 19, 2011, the Robert Koch Institute (RKI), Germany's Federal Public Health Authority, was invited by the Health and Consumer Protection Agency in Hamburg to assist in the investigation of three cases of Hemolytic-uremic syndrome (HUS), a life-threatening illness caused by EHEC. Day 2: May 20, alarmed by the type of persons affected and the rapid spread of EHEC, an investigation was initiated by RKI, involving all levels of public-health and food-safety authorities to identify the cause of the outbreak, and to prevent further cases of disease. On day 5: May 23, RKI asked *all* health departments to expedite procedures, by immediately forwarding all case reports of suspected or confirmed EHEC/HUS, to the Federal Public Health Authority, relying directly on the diagnoses of notifying clinicians [Frank et al., 2011; Robert Koch Institute (RKI), 2011].

Based on this five-day timeline of EHEC/HUS 2011 outbreak in Germany, one can see that public health officials are faced with new challenges for outbreak alert and response. This is due to the continuous emergence of infectious diseases and their contributing factors such as demographic change, or globalization. Early reaction is necessary, but often communication and information flow through traditional channels is slow. *Can additional sources of information, such as social media streams, provide complements to the traditional epidemic intelligence mechanisms?*

6.2.1 Twitter for Early Warning

The continuous emergence of infectious diseases and their contributing factors impose new challenges to public health officials. Early reaction is necessary, but often communication and information flow through traditional channels is slow. Additional sources of information, such as social media streams, provide complements to the traditional reporting mechanisms.

For example, if we observe Figure 6.1, we can see two plots, one of them corresponds to the relative frequency of EHEC cases as reported by RKI [Robert Koch Institute (RKI), 2011], and the other to the relative frequency of mentions of the keyword “EHEC" in the tweets collected during the months of May and June 2011. We can appreciate the high correlation of the curves, which corresponds to a Pearson correlation coefficient of 0.864. We can also observe the *inertia* of the crowd that continued tweeting about the outbreak, even though the number of cases were already declining (e.g., June 5 to 11).

Twitter has shown potential as a source of information for public health event monitoring (e.g., [Paul and Dredze, 2011b; Sofean et al., 2012]), but could it be possible to generate an early warning signal before well established systems by only tracking Twitter?

In this section, we have a closer look to the time period of the EHEC/HUS outbreak in Germany, and address this question.

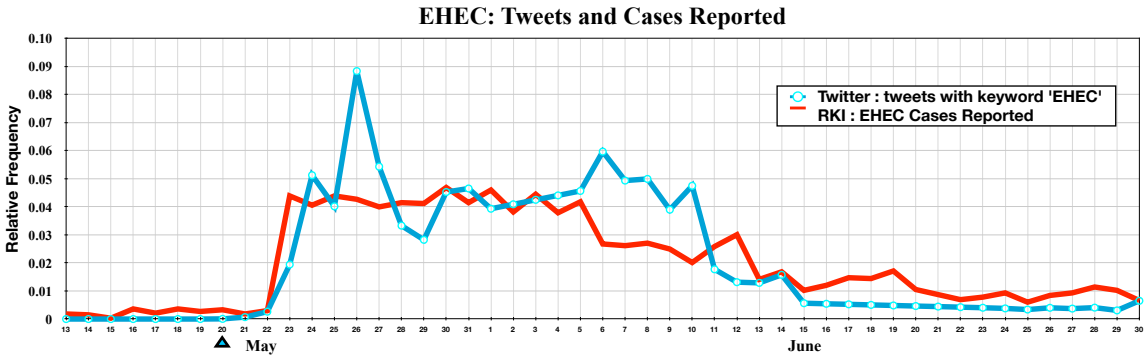


Figure 6.1: **Relative frequency of cases reported to RKI and the number of tweets mentioning the name of the disease: *EHEC*. The Pearson correlation coefficient is 0.864. Monitoring Twitter allowed us to generate the first signal on Friday, May 20th, 2011, using standard biosurveillance methods, before well established early warning systems (triangle on the time axis).**

Data Collection

We incrementally collected tweets using Twitter's API, currently we monitor over 500 diseases and symptoms, which include "EHEC". One of the challenges we face collecting data from Twitter, besides the API restrictions, is the level of *noise* with respect to medical domain content. Straightforward techniques relying on regular expressions, even though they exhibit high recall, are difficult to maintain and prone to high false positive rates. For example, consider the following two tweets collected by a combination of regular expressions, and a dictionary of diseases that includes the medical conditions *EHEC* and *fever*:

1. *RKI warns against north German vegetables: Experts looking feverishly*
   EHEC *source http://bit.ly/itGpJx*

2. *I've definitely Bieber-**fever**. There's no doubt. but who hasn't got bieber **fever**? @justinbieber is soo damn rawwwr*

Tweet number one is of obvious importance for epidemic intelligence, but number two is not.

Description	Amount
Number of tweets collected related to medical conditions during May and June, 2011	7,710,231
Tweets extracted related to the EHEC/HUS outbreak out of the ones collected	456,226
Distinct users that produced the tweets related to the outbreak	54,381

Table 6.1: **Data collected from Twitter related to the EHEC/HUS outbreak in Germany during May and June, 2011.**

Instead of simple keyword matching to filter out irrelevant tweets, our data collection strategy includes text classification methods and a multi-level filtering based on supervised learning, following the approach of Stewart et al. [Stewart et al., 2011b].

Table 6.1 summarizes the data collected related to the outbreak that was used in our analysis.

Detection Methods

The surveillance algorithms we used are well documented in the disease aberration literature e.g. [Khan, 2007; Hutwagner et al., 2003; Basseville and Nikiforov, 1993]. The objective of these algorithms is to detect aberration patterns in time series data when the volume of an observation variable exceeds an expected threshold value. In our case, for example, the observation variable corresponds to mentions of medical condition “EHEC" withing the tweets.

The five biosurveillance algorithms we used for early detection are: the Early Aberration Reporting System (EARS) (1) C1, (2) C2, and (3) C3 algorithms, (4) F-statistic, and (5) Exponential Weighted Moving Average (EWMA), which are briefly described below, for a detailed introduction please refer to [Khan, 2007].

The **Early Aberration Reporting System (EARS)** (1) **C1**, (2) **C2**, and (3) **C3** algorithms [Hutwagner et al., 2003] compute a test statistic, T_t, on day t as follows:

$$T_t = \max(0, (X_t - (\mu_t + k \cdot \sigma_t))/\sigma_t) \tag{6.1}$$

where X_t is the count on day t, k is the shift from the mean to be detected, and μ_t and σ_t are the mean and standard deviation of the counts during the baseline period. For C1, the baseline period is $(t-7,\ldots,t-1)$; for C2 the baseline is $(t-9,\ldots,t-3)$. The test statistic for C3 is the sum of $T_t + T_{t-1} + T_{t-2}$ from the C2 algorithm (Eq. 6.1). The constant k determines how sensitive is the algorithm to generate a signal. With a lower value of k, the algorithm becomes more sensitive as it will trigger an alarm with less of a deviation from the mean of the process.

(4) The **F-statistic** [Burkom, 2005] is computed as:

$$T_t = \sigma_t^2 + \sigma_b^2$$

where σ_t^2 approximates the variance during the testing window and σ_b^2 approximates the variance during the baseline window. Their calculation is as follows:

$$\sigma_t^2 = \frac{1}{n_t} \sum_{test}^{n_t} (X_t - \mu_b)^2$$

$$\sigma_b^2 = \frac{1}{n_b} \sum_{test}^{n_b} (X_t - \mu_b)^2$$

(5) **Exponential Weighted Moving Average (EWMA)** model [Khan, 2007], provides for a non-uniformly weighted baseline by down-weighting counts that are on days further from the target day. The smoothed daily counts were calculated as:

$$Y_1 = X_1; \;\; Y_t = \omega\, X_t + (1-\omega)\, Y_t - 1$$

and the test statistic was calculated as

$$T_t = (Y_t - \mu_t)/[\sigma_t * (\omega/(2-\omega))^{1/2}]$$

where $0 > \omega > 1$ is the smoothing constant, and μ_t and σ_t are the mean and standard deviation for the baseline window, which was set to $(t-15,\ldots,t-5)$.

We signal an alarm if the test statistic reported by the detection methods exceeds a threshold value, which is determined experimentally. The larger the amount by which the threshold is exceeded, the greater the severity of the alarm. Table 6.2 summarizes the alarm dates and detection methods parametrization, which follows the guidelines of N. Collier [Collier, 2010] and Khan [Khan, 2007].

Detection Method	**Parametrization** [Khan, 2007; Collier, 2010]	**Alarm Dates**
C1	Training window = 15 days; buffer = 5 days; upper control limit = $\mu + 3\sigma$	May 20 to May 28
C2	Training window = 15 days; buffer = 5 days; upper control limit = $\mu + 3\sigma$; alarm threshold=0.2	May 20 to May 28
C3	Training window = 15 days; buffer = 5 days; upper control limit = $\mu + 3\sigma$; alarm threshold=0.3	May 20 to May 24
F-statistic	Training window =15 days; buffer = 5 days; alarm threshold=0.6	May 20 to June 30
EWMA	Training window =15 days; buffer = 5 days; alarm threshold=4, ω = 0.24	May 20 to May 30

Table 6.2: **Detection method parameters and alarm dates.**

Using any of the detection methods (Table 6.2), a daily count less than five tweets was enough to signal an alert on May 20th, 2011. The Early Warning and Response System (EWRS) [1] of the European Union received a first communication by the German authorities on Sunday May 22. MedISys [2] detected the first media report in the German newspaper *Die Welt* [3] on Saturday May 21 [Linge et al., 2011] and ProMED-mail [4] and all other major early alerting systems (e.g., ARGUS, Biocaster, GPHIN, HealthMap, PULS) covered the event on Monday May 23.

[1] **EWRS**: ewrs.ecdc.europa.eu .
[2] **MedISys**: medusa.jrc.it/medisys .
[3] **Die Welt**: welt.de .
[4] **ProMED-mail**: promedmail.org .

Why was this early detection possible with respect to well established early warning systems? We tracked only Twitter as source of information, in contrast to MedISys for example, that tracks hundreds of news sources on the Internet. We consider Twitter's *diversity* was the key element that helped in the earlier detection of the event.

Twitter is a diverse stream of *multiple sources*. In Twitter converges the *contribution from the crowd* – millions of individual users obscure and renown; big and small media outlets; global and local newschapters, etc. Our work and that of MedISys focus on an analysis at a national level, but there are cases where support for the *local perspective* is important, for example local and smaller news chapters reaching a broader audience through Twitter.

A closer look to day May 20, reveals that the first alarm was triggered based on five tweets, the actual messages are shown in Table 6.3, all of them generated from sources not far from where the first cases of the outbreak were reported. Those users acted as *local sensors*, producing tweets that spread the news faster than major newschapters.

6.2.2 Twitter for Outbreak Analysis and Control

For public health officials, who are participating in the investigation of an outbreak, the millions of documents produced over social media streams represent an overwhelming amount of information for risk assessment.

To reduce this overload we explore to what extent recommender systems techniques can help to filter information items according to the public health users' context and preferences (e.g., disease, symptoms, location). In particular, we focus on a personalized learning to rank approach that ultimately offers the user the most relevant and attractive tweets for risk assessment. In this section, we introduce our approach and report an experimental evaluation on the EHEC/HUS dataset collected from Twitter.

No.	Tweet	User	Location
1	Hannover: Gefahr durch EHEC-Erreger http://bit.ly/l2bJwN	BS_Zeitung	Braunschweig
2	Twitter Hannover: Gefahr durch EHEC-Erreger http://bit.ly/l2bJwN	WN_Wolfsburg	Wolfsburg
3	20. Mai 2011, 19:48 Uhr - Mehrere Hamburger mit EHEC-Erreger infiziert: In Hamburg haben sich mehrere Menschen m... http://bit.ly/jEfoHw	Hamburg_	Hamburg
4	Mehrere Hamburger mit EHEC-Erreger infiziert: Hamburg (dpa/lno) - In Hamburg haben sich mehrere Menschen mit dem... http://bit.ly/lRM5Kr	Lokales_Hamburg	Hamburg
5	Hannover: Gefahr durch EHEC-Erreger http://bit.ly/l2bJwN	SZ_Zeitung	Salzgitter
6	Mehrere Hamburger mit EHEC-Erreger infiziert: Hamburg (dpa/lno) - In Hamburg haben sich mehrere Menschen mit dem... http://bit.ly/m7ZQWp	inselhiddensee	Insel Hiddensee
7	Twitter Mehrere Hamburger mit EHEC-Erreger infiziert http://bit.ly/jt9uHA	schleswigbiz	Schleswig

Table 6.3: **The early tweets gathered related to the outbreak on May 20, 2011. These few tweets were enough to trigger an alert using a *moving average* biosurveillance detection method. The Early Warning and Response System (EWRS) of the European Union received a first communication by the German authorities on Sunday May 22.**

Our Approach: Ranking Tweets for Epidemic Intelligence

We propose to use the user *context* as implicit criteria to select tweets of potential relevance, that is, we will rank and derive a short list of tweets based on the user context.

The user context C_u is defined as a triple

$$C_u = (t, MC_u, L_u) \ , \tag{6.2}$$

where t is a discrete time interval, MC_u the set of medical conditions, and L_u the set of locations of user interest.

We define three concepts that will help us to discuss our approach in the rest of this section:

Medical Condition is a string that describes a human medical condition, such as a disease, disorder or syndrome. We represent the set of medical conditions as MC.

Location is a string that is used to identify a point or an area on the Earth's surface, which can be mapped to a specific pairing of latitude and longitude. The set of locations is denoted as L.

Complementary Context is defined as the set of nouns, which are neither Locations nor Medical Conditions. Complementary Context may include named entities such as names of persons, organizations, affected organisms, expressions of time, quantities, etc. We denote the set of named entities that represents the complementary context as CC, where $CC \cap (L \cup MC) = \emptyset$.

Our Personalized Tweet Ranking for Epidemic Intelligence algorithm or PTR4EI is shown in Algorithm 3. The algorithm extends a learning to rank framework by considering a personalized setting that exploits user's individual context (see Section 2.2). Table 2.1 gives a summary of notations used in learning to rank.

More precisely, we consider the context of the user, C_u, and prepare a set of queries, Q, for a target event (e.g., a disease outbreak). We first compute LDA [Blei et al., 2003] on an indexed collection $\mathcal{T}$ of tweets for epidemic intelligence, where not all tweets are necessarily interesting for the target event.

We also extract the hash-tags that co-occur with the user context by considering the medical conditions and locations in C_u as hash-tags themselves, and find which other hash-tags co-occur with them within a tweet, and how often they co-occur, which will help us to select the most representative hash-tags for the target event.

Algorithm 3 Personalized Tweet Ranking algorithm for Epidemic Intelligence (PTR4EI)

Input: User Context $C_u = (t, MC_u, L_u)$, Inverted index $\mathcal{T}$ of tweets collected for epidemic intelligence before time t, and subset of the tweets $D_y \subset D$.

Output: Ranking Function f_{C_u} for User Context C_u.

1: Compute LDA topics ($topicsLDA$) on $\mathcal{T}$

2: Consider each $mc \in MC_u$ as a hash-tag, and extract from $\mathcal{T}$ all co-occurring hash-tags: $coHashTags$

3: Classify the terms in $topicsLDA$ and the hash-tags in $coHashTags$ as Medical Condition MC_x, Location L_x or Complementary Context CC_x

4: Build a set of queries as follows:
$$Q = \{q \mid q \in MC_u \times \mathcal{P}(\{L_u \cup MC_x \cup L_x \cup CC_x\})\}$$

5: For each query $q_i \in Q$ obtain tweets D from the collection $\mathcal{T}$

6: Elicit relevance judgments Y on the given subset $D_y \subset D$

7: For each tweet $d_j \in D$, obtain the feature vector $\phi(q_i, d_j)$ w.r.t. $(q_i, d_j) \in Q \times D$

8: Apply learning to rank to obtain a ranking function for the user context C_u:
$$f_{C_u}(q, d) = \vec{w} \cdot \phi(q, d)$$

9: **return** $f_{C_u}(q, d)$

The set Q is constructed by expanding the original terms in C_u with the ones in the LDA topics and co-occurring hash-tags, which are previously classified as medical condition, location or complementary context.

We build the set D of tweets by querying index $\mathcal{T}$ using $q \in Q$ as query terms. Next, we elicit judgments from experts on a subset of the tweets retrieved, in order to construct $D_y \subset D$.

We then obtain for each tweet $d_j \in D$ its feature vector $\phi(q_i, d_j)$ with respect to the pair $(q_i, d_j) \in Q \times D$.

Finally and with these elements, we apply a learning to rank algorithm to obtain the ranking function for the given user context.

In the rest of the section, we evaluate our approach considering as event of interest the EHEC/HUS outbreak in Germany, 2011.

Experiments and Evaluation

To support users in the assessment and analysis during the EHEC/HUS outbreak, we set the user context (Eq. 6.2) as $C_u = (t, MC_u, L_u) =$ ([2011-05-23; 2011-06-19], {"EHEC"}, {"Lower Saxony"}), in this way, we are taking into account the main period of the outbreak [5], the disease of interest, and the German state with more cases reported.

Following Algorithm 3, we computed LDA and extracted the co-occurring hash-tags using the indexed collection $\mathcal{T}$ described in Section 6.2.1. Table 6.4 shows four LDA topics for each week of the time period of interest, and Table 6.5 presents the hash-tags co-occurring with *#EHEC*.

We asked three experts: one from the Robert Koch Institute and the other two from the Lower Saxony State Health Department (NLGA) [6] to provide their individual judgment on a subset D_y of 240 tweets, evaluating for each tweet, if it was relevant or not to support their analysis of the outbreak. Any disagreement in the assigned relevance scores was resolved by majority voting.

We selected these tweets from the index $\mathcal{T}$ as follows: 30 were obtained using as query the term "EHEC", i.e., MC_u, together with the medical conditions identified using LDA, and 30 using the medical conditions from the hash-tags. We used a similar procedure combining query "EHEC" with the locations and complementary context extracted from LDA and hash-tag co-occurrence, obtaining 30 tweets at every step, for a total of 120 tweets. For the rest 60, we used the query term "EHEC" alone, then we ordered the result set chronologically based on the tweets' publication date, and selected the most recent ones.

[5] Please note, that even though the main period of the outbreak is considered for the evaluation, nothing prevents us for building the model during the ongoing outbreak, and recompute it periodically (e.g., weekly).

[6] **NLGA**: nlga.niedersachsen.de .

Week 21			
EHEC *(MC)*	fever *(MC)*	EHEC *(MC)*	EHEC *(MC)*
cucumbers *(CC)*	pain *(MC)*	casualty *(-)*	pathogen *(MC)*
Spain *(L)*	headache *(MC)*	women *(CC)*	Northern Germany *(L)*
tomatoes *(CC)*	sniff *(MC)*	intestinal germ *(MC)*	diarrhea *(MC)*
salad *(CC)*	pain *(MC)*	panic *(MC)*	dead *(MC)*
Week 22			
EHEC *(MC)*	EHEC *(MC)*	EHEC *(MC)*	EHEC *(MC)*
dead *(MC)*	intestinal germ *(MC)*	cucumbers *(CC)*	cucumbers *(CC)*
Germany *(L)*	source *(-)*	pathogen *(MC)*	salad *(CC)*
people *(-)*	search *(-)*	Spain *(L)*	pain *(MC)*
live *(-)*	Hamburg *(L)*	farmers *(CC)*	women *(CC)*
Week 23			
EHEC *(MC)*	headache *(MC)*	EHEC *(MC)*	EHEC *(MC)*
cucumber *(CC)*	pain *(MC)*	cucumbers *(CC)*	sprout *(CC)*
eu *(CC)*	fever *(MC)*	sprout *(CC)*	source *(-)*
crisis management *(-)*	people *(-)*	pathogen *(MC)*	suspicion *(-)*
farmers *(CC)*	cough *(MC)*	salad *(CC)*	hus *(MC)*
Week 24			
EHEC *(MC)*	headache *(MC)*	stomach ache *(MC)*	pain *(MC)*
germ *(MC)*	fever *(MC)*	sniff *(MC)*	bellyache *(MC)*
sprout *(CC)*	slept *(-)*	pain *(MC)*	cough *(MC)*
health *(MC)*	sniff *(MC)*	regions *(-)*	throat *(CC)*
all-clear *(CC)*	head *(CC)*	examined *(-)*	sniff *(MC)*

Table 6.4: **Four LDA topics (columns) computed weekly during the main period of the outbreak: from May 23 to June 19, 2011. We classify terms within each topic as *Medical Condition (MC)*, *Location (L)*, or *Complementary Context (CC)*.**

We prepared five binary features for each tweet as follows:

Feature	Value = True
F_{MC}	If a medical condition is present in the tweet
F_{L}	If a location is present in the tweet
$F_{\#\text{-tag}}$	If a hash-tag is present in the tweet
F_{CC}	If a complementary context term is present in the tweet
F_{URL}	If a URL is present in the tweet

For learning the ranking function, we used Stochastic Pairwise Descent (SPD) algorithm [Sculley, 2009], which solves the same optimization problem as Ranking SVM [Joachims, 2002], but using stochastic gradient descent, whose characteristics make it more appealing to scale to larger datasets (e.g., [Bottou, 2010]).

Medical Condition	Location	Complementary Context	
Week 21			
bacteria	bremen	cucumber_salad	cdu
diarrhea	cuxhaven	cucumbers	edeka
ehec_victim	hamburg	ehec_vegetable	fdp
hus	münster	tomatoes	merkel
intestinal_infection	northern_germany	vegetables	rki
Week 22			
bacteria	berlin	cucumbers	bild
diarrhea	germany	obst	fdp
ehec_pathogen	hamburg	salad	n24
hus	lübeck	terror	rki
intestinal_infection	spain	tomatoes	rtl
Week 23			
bacteria	bavaria	cucumbers	ehec_free
diarrhea	berlin	salad	fdp
ehec_pathogen	germany	sojasprout	merkel
hus	hamburg	sprout	n24
intestinal_infection	lower_saxony		rki
Week 24			
bacteria	lower_saxony	donate_blood	
died		ehec_free	
health		sojasprout	
hus			

Table 6.5: **Hash-tags co-occurring with *#EHEC* during May 23 and June 19, 2011, the main period of the outbreak. The hash-tags are classified as entities of type *Medical Condition*, *Location*, or *Complementary Context*, hash-tags out of these categories are discarded.**

We compared our approach, that expand the user context with latent topics and social generated hash-tags, against two ranking methods:

- **RankMC**: It learns a ranking function using only the medical condition feature F_{MC}. Note that this baseline also considers related medical conditions to the ones in MC_u, which makes it stronger than non-learning approaches, such as BM25 or TF-IDF scores, that use only the MC_u elements as query terms.

- **RankMCL**: It is similar to RankMC, but besides the medical conditions, it uses a local context to perform the ranking (i.e., features: F_{MC} and F_L). We expect this method to perform better than RankMC, since it does not only take into account the spatial information from the user context, but also additional locations in the collection.

We randomly split the dataset into 80% training tweets, which will be used to compute the ranking function, and 20% testing tweets. To reduce variability, we performed the experiment using ten different 80/20 partitions. The test set is used to evaluate the ranking methods. The reported performance is the average over the ten rounds.

Evaluation Measures

For evaluation, we used three measures widely used in information retrieval, namely precision at position n ($P@n$), mean average precision (MAP), and normalized discount cumulative gain (NDCG). Their definitions are given in Section 2.6.

Results

The ranking performance in terms of precision is presented in Table 6.6, MAP and NDCG results are shown in Figure 6.2. As we can appreciate PTR4EI outperforms both baselines. Local information helps RankMCL to beat RankMC, for example MAP improves from 71.96% (RankMC) up to 81.82% (RankMCL). PTR4EI, besides local features, exploits complementary context information and particular Twitter features, such as the presence of hash-tags or URLs in the tweets, this information allows it to improve its ranking performance even further, reaching a MAP of 91.80%. A similar behavior is observed for precision and NDCG, where PTR4EI also outperforms RankMC and RankMCL.

Method	P@1	P@3	P@5	P@10
RankMC (baseline)	90 %	73.34 %	64 %	69 %
RankMCL (baseline)	90 %	83.33 %	88 %	85 %
PTR4EI	**100 %**	**90 %**	**94 %**	**96 %**

Table 6.6: **Ranking performance in terms of P@{1, 3, 5, 10}.**

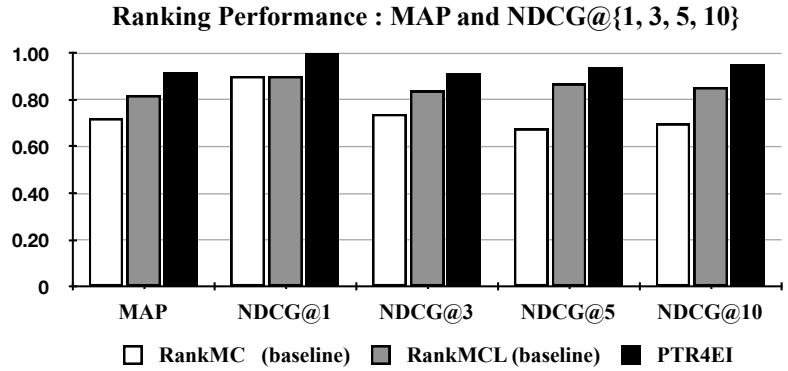


Figure 6.2: **MAP and NDCG results.**

6.3 Detecting Political Emotions in Social Web Streams

Real-time analysis of social media streams allows for discovery of latent patterns in public opinion, which can be exploited to improve decision making processes. For example, automatically detecting emotions such as joy, sadness, fear, anger, and surprise in the social web has several practical applications, for instance, tracking the popularity of political figures or public response to new released products. This is the field of sentiment analysis, which involves determining the opinions and private states (beliefs, feelings, and speculations) of the speaker towards a target entity [Wiebe, 1994].

Our goal in this work is to explore the sentiments and emotions towards political figures in Latin America. To this end, we analyze mentions on Twitter and blogs of eighteen Latin American presidents, between October 1, 2011 and April 1, 2012. The names of the presidents and their respective country are listed in Table 6.7. By making use of an emotion lexicon, we study the emotions evoked by each president. While this approach is standard in many applications (e.g., [Dodds

et al., 2011; Mohammad, 2011; Demartini et al., 2011]), we felt that a study on political emotion detection via the social web, covering Latin America in particular, was necessary.

ID	Country	President	ID	Country	President
P1	Argentina	Cristina Fernández	P10	Guatemala	Otto Pérez Molina
P2	Bolivia	Evo Morales	P11	Honduras	Porfirio Lobo
P3	Brazil	Dilma Rouseff	P12	Mexico	Felipe Calderón
P4	Chile	Sebastián Piñera	P13	Nicaragua	Daniel Ortega
P5	Colombia	Juan Manuel Santos	P14	Panama	Ricardo Martinelli
P6	Costa Rica	Laura Chinchilla	P15	Paraguay	Fernando Lugo
P7	Dominican Republic	Leonel Fernández	P16	Peru	Ollanta Humala
P8	Ecuador	Rafael Correa	P17	Uruguay	José Mujica
P9	El Salvador	Mauricio Funes	P18	Venezuela	Hugo Chávez

Table 6.7: **Presidents of Latin America considered in the analysis (listed alphabetically by country name).**

6.3.1 Taking the Pulse of Political Emotions

Our approach consists of the following steps:

1. Data collection process
2. Emotion and polarity analysis
3. Pattern recognition from the sentiment analysis

In this section, we first present how we collected the set of documents used in the study and the dataset statistics. Second, we explain the preprocessing techniques on the dataset, and finally, we explain the approach for emotion analysis employed in this work. In Section 6.3.2, we present the results and discuss the patterns discovered from the sentiment analysis that can help explain the popularity observed in opinion polls.

Data Collection Process

We perform our study on a collection of 165,484 documents, from them, 155,280 are 140-character Twitter messages or *tweets*, and 10,204 are

snippets of weblog posts. The total of documents was produced by 55,013 distinct users during the six-month period between 1st of October, 2011 and 1st of April, 2012. We chose this period of time because it allowed us to discuss and contrast our findings against an independent opinion poll published in April 2012 [Consulta Mitofsky – www.consulta.mx, 2012]. The same procedure and analytic techniques discussed in this work can be directly applied on real-time data streams, as illustrated in Figure 6.3. Both, tweets and blog posts are in Spanish[7] and they were collected as follows:

- *Twitter messages* were retrieved using Topsy[8], a Twitter search engine that indexes and archives messages posted on Twitter[9]. For each president name listed in Table 6.7, we issued a query against Topsy using its API. We forced an exact match on the name by enclosing it in double quotes. We also included in the parameters the corresponding start and end date of interest.

- *Blog posts* were fetched using *Google News* RSS Feeds[10]. Similarly as in the case of tweets, we used as query term the name of the president and forced an exact match. We restricted the sources of information to be exclusively blogs in the Spanish language. Again, the time range was specified to the period under analysis. In this case, we consider as *document* the post's title and the short snippet of text (∼300 characters) contained in the item's *description* tag of the RSS result as returned by Google.

[7]For the president of Brazil, we analyzed a total of 18,933 documents (tweets and blog posts) in Spanish. Future work includes also the analysis of documents in the Portuguese language.

[8]**Topsy:** topsy.com .

[9]Please note that Twitter's search API only allows to retrieve recent tweets (between 6-9 days old) (https://dev.twitter.com/docs/using-search).

[10]**Google News:** news.google.com .

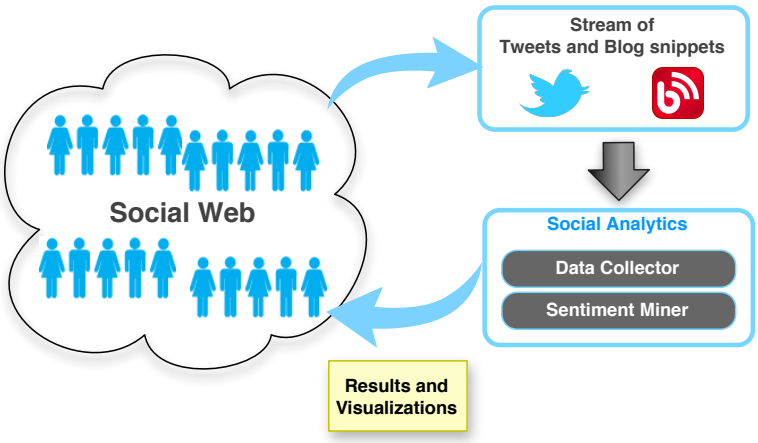


Figure 6.3: **Social Analytics Process. The process monitors in real-time the interactions of people in the Social Web, e.g., the content they generate and exchange. The data is analyzed and global-patterns are discovered. Finally, the loop is closed, and the results are given back to the web of people.**

Emotion Model and Polarity Analysis for Political Figures

Our objective is to identify the emotions reflected in the Social Web towards a political figure, in our case, a particular Latin American president. To this end, we analyze tweets and blog post snippets of maximum 140-character and 300-character long, respectively. Given the short text of the documents, we assume that words close to the president's name convey the emotion to be captured. In particular, we focus our analysis on *nouns* and *adjectives*.

Our emotion detection approach comprises the following procedure:

1. Create a *profile* for each president

2. Extract the terms from the profile

3. Associate to each term an *emotion* and *polarity* based on an emotion lexicon

4. Compute the *emotion vector* and *polarity* for each president

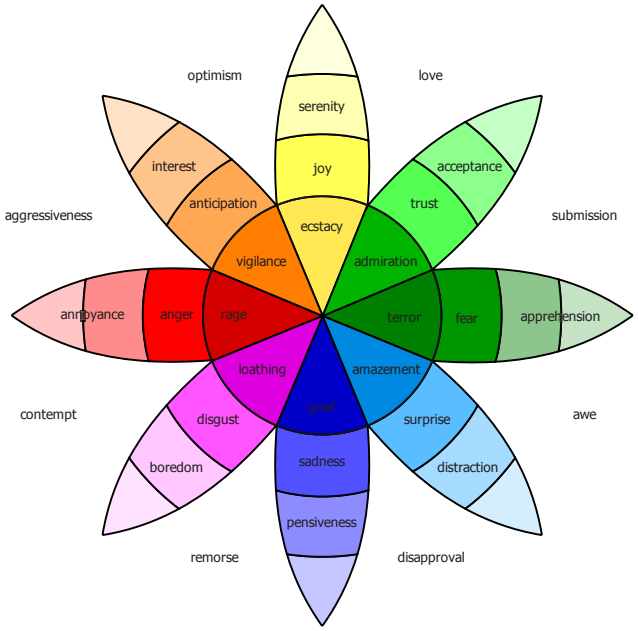


Figure 6.4: **Plutchik's wheel of emotions** (image taken from Wikimedia Commons).

First, we build a *profile* for each of the 18 presidents. The profile consists of all tweets and blog post snippets collected for the corresponding president. After building the profiles, we use *TreeTagger* to perform part-of-speech tagging on each of them [Schmid, 1994]. Then, based on the output of TreeTagger, we extract the *nouns* and *adjectives*. Finally, we use a term-based matching technique to associate each term with emotion and polarity values.

We used in our study the NRC Emotion Lexicon (EmoLex), a large set of human-provided word emotion association ratings. EmoLex was created by crowdsourcing to Amazon's Mechanical Turk, and it is described in [Mohammad and Turney, 2011].

The lexicon contains 3292 distinct words[11] annotated according Plutchik's psychoevolutionary theory of eight basic emotions, which form four opposing pairs, joy–sadness, anger–fear, trust–disgust, and anticipation–surprise [Plutchik, 1980]. This emotion contrast is shown in Figure 6.4 by the spatial opposition of these pairs. In addition, EmoLex also includes positive and negative sentiments associated to the words. For instance, the word *friend* has the emotion *joy* and a

[11]We used EmoLex version 0.5 for academic institutions.

positive polarity associated to it, whereas the word *violent* has associated the emotions of *anger, disgust, fear, surprise*, and *trust*; and a *negative* polarity.

Sentiment Analysis and Multilingualism

Note that the terms in the lexicon are in English, however, the profile of the presidents contains text in Spanish. One approach to address this issue is to use machine translation to translate the text of each document into English, and conduct the analysis in this language, for example in [Tumasjan et al., 2010] tweets are translated from German to English to extract the emotions. However, our objective is to process the social stream in real-time, and translating each and every microblog post would be costly. Instead, we propose to machine translate the terms in the lexicon from English to Spanish, in this way, the process is performed once, offline, and the resulting terms are used to perform the analysis in the same language of the posts. To this end, we translated the terms in EmoLex using three different services: *Google Translate*[12], *Bing Translator*[13], and *Yahoo! Babel Fish*[14]. The resulting terms in Spanish were associated to the corresponding English term's emotions and polarity.

President's Emotional Vector

We define the emotional vector, e_p, for president p as follows: Let T_p be the set of terms extracted from the president's profile p, and T_m the set of all terms in EmoLex annotated with emotion m, where $m \in M$; $M := \{joy, sadness, anger, fear, trust, disgust, anticipation, surprise\}$, i.e., Plutchik's eight basic emotions. Then, the m^{th} dimension of emotional vector $e_p \in \mathbb{R}^{|M|}$ is given by:

$$e_p[m] := \sum_{t \in T_p} \mathcal{I}_m(t)$$

[12] **Google Translate:** translate.google.com .
[13] **Bing Translator:** bing.com/translator .
[14] **Yahoo! Babel Fish:** babelfish.yahoo.com .

where $\mathcal{I}_m(t)$ is an indicator function that outputs 1 if the term $t \in T_p$ is associated to emotion m, and 0 otherwise. Finally, we normalize vector e_p to produce a probability vector

$$\hat{e_p} = \frac{e_p}{N_M}$$

where N_M is a normalization constant that corresponds to the total number of terms $t \in T_p$ associated to an emotion.

For example, the emotional vector for the president of El Salvador over dimensions [joy, sadness, anger, fear, trust, disgust, anticipation, surprise], corresponds to : [0.08, 0.18, 0.14, 0.21, 0.15, 0.17, 0.03, 0.04] (see Figure 6.7i). Note that the components of the probability vector add up to 1, and each of them is a positive number between 0 and 1.

President's Polarity

Similarly as in the case of emotions, we compute the polarity tuple $(positive,\ negative)_p$ of president p as follows:

$$(\text{positive, negative})_p = \frac{(\sum_{t \in T_p} \mathcal{I}_+(t), \sum_{t \in T_p} \mathcal{I}_-(t))}{N_{polarity}}$$

where the indicator functions are defined analogously as in the case of the emotions, and the normalization constant $N_{polarity}$ is the sum of terms with a polarity value assigned.

6.3.2 Results

In this Section, we present the results of our investigation. First, we explore the polarity of the terms in each president's profile. Second, we will analyze the emotions associated to each president, and the extracted patterns that could explain the degree of acceptance of each political figure.

As reference we use a survey report published in April 2012 by the Mexican analytics group *Consulta Mitofsky* [Consulta Mitofsky – www.consulta.mx, 2012]. The report is a compilation of the results of individual opinion polls of Latin American presidents. Note that the

president of Guatemala was not included in this survey report, hence, for this particular case we used the results of another poll published in December 2011 [CID-Gallup – www.cidgallup.com, 2011].

Polarity Detection

The polarity detected for each president is shown in Figure 6.5. Polarity detection provides a quick overview of the sentiment conveyed by the terms co-occurring with the presidents' name, but it is too coarse grained, and as we will discuss later in this section, it does not fully explain the popularity as measured by the opinion poll.

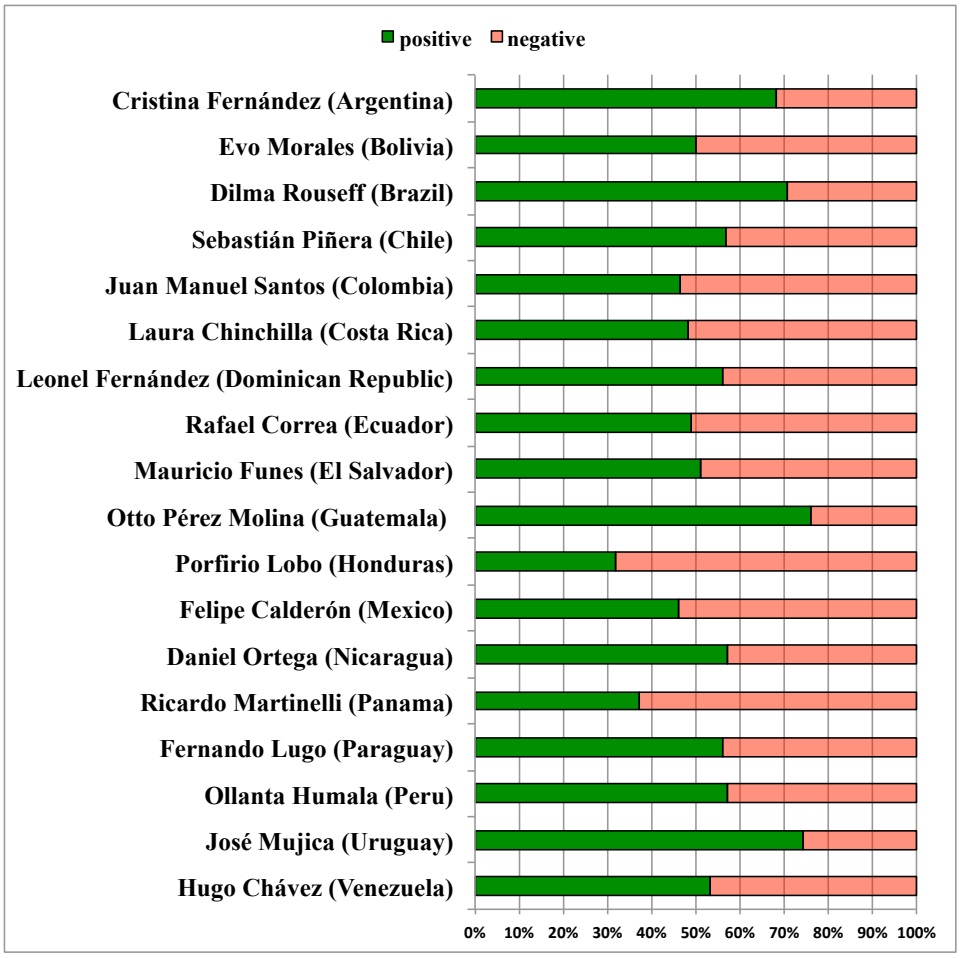


Figure 6.5: **Polarity analysis.**

Emotion Detection

In order to illustrate the terms behind each of the emotions extracted, we present in Figure 6.6 a *tag cloud* per each emotion considered. Each emotion tag-cloud includes the top-25 most frequent terms aggregated over all presidents[15].

(a) Joy (b) Trust (c) Fear (d) Surprise

(e) Sadness (f) Disgust (g) Anger (h) Anticipation

Figure 6.6: **Tag clouds for each emotion. The terms presented in each tag cloud correspond to the top-25 most frequent terms aggregated over all presidents. For (a) *joy*, the salient terms are *happy, friend, special, truly, and powerful* (feliz, amigo, especial, verdadero, poderoso); for (b) *trust*, the terms *important, friend, truly, and main* (importante, amigo, verdadero, principal) are the most mentioned; the most popular terms in the emotion (c) *fear* are *bad, military, urgent, terrorist, and dictator* (malo, militar, urgente, terrorista, dictador); for (d) *surprise*, the main terms are *urgent, crazy, terrorist, violent, and different* (urgente, loco, terrorista, violento, diferente); for (e) *sadness* the salient terms are *bad, and dead* (malo, muerto); for (f) *disgust*, the terms *bad, strong, prisoner, false, and powerful* (malo, fuerte, preso, falso y poderoso) are the most mentioned; the most popular terms in the emotion (g) *anger* are *bad, false, powerful, terrorist, and expensive* (malo, falso, poderoso, terrorista, caro); for (h) *anticipation*, the main terms are *urgent, early, powerful, superior, and successful* (urgente, pronto, poderoso, superior, exitoso).**

[15]The tag clouds were created using **Wordle:** wordle.net .

The president emotional vectors are visualized as pie charts in Figure 6.7.

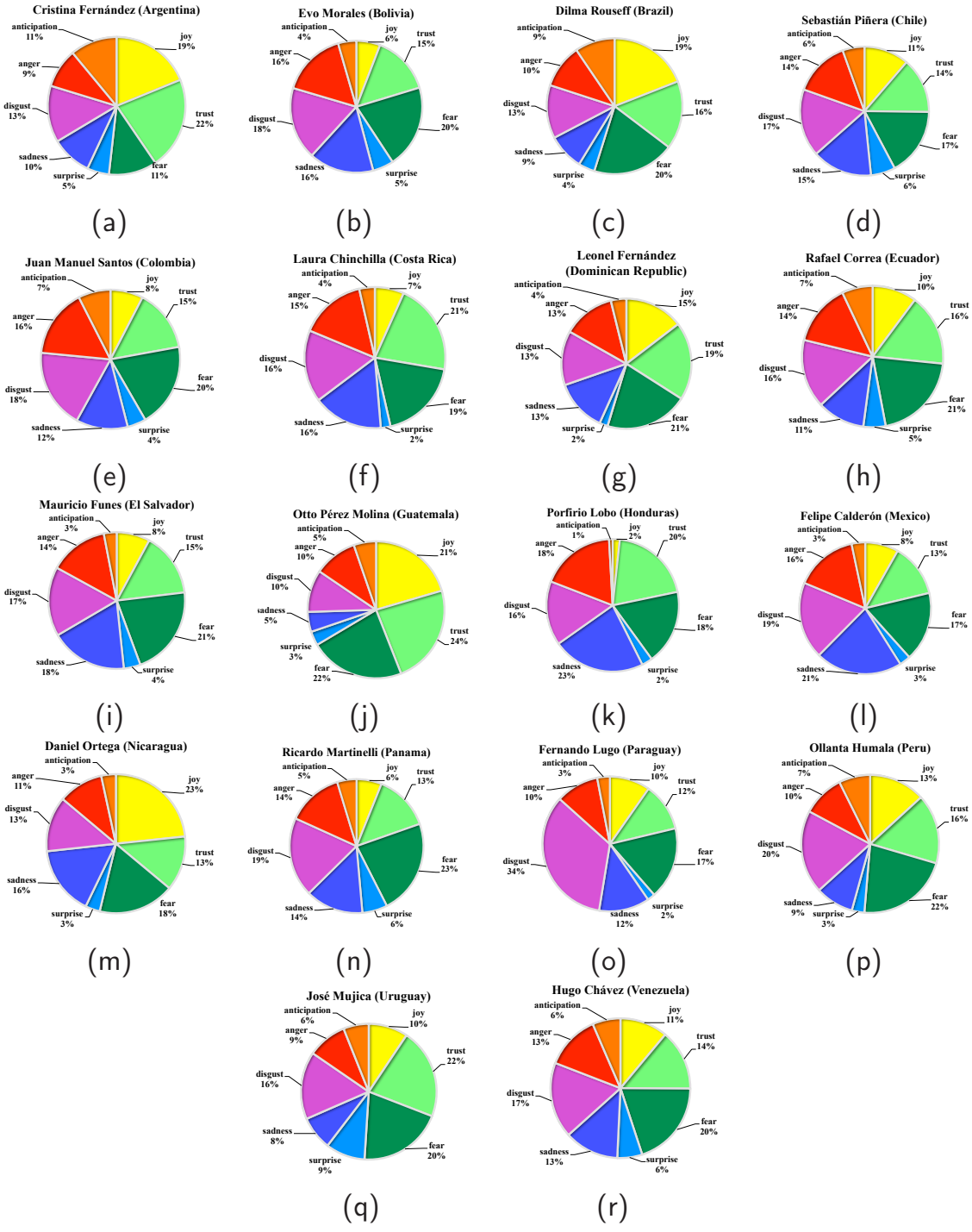


Figure 6.7: **Emotions detected for each president.**

We can observe that the emotion analysis provides more insights on the perception of the presidents in the social stream, than the polarity value alone. For example, in the case of the Mexican president F. Calderón, he has a negative polarity value of 54%, which can be better qualified by the predominant emotions extracted from his profile, namely: sadness, anger, fear, and disgust.

Emotional Pattern Analysis and Opinion Poll

Can the polarity or emotions, alone or combined, explain the (un-)popularity of a particular political figure? and what are the most influential emotions that could help predict the outcome of a traditional opinion poll? In this section, we seek to shed some light on these questions.

In Table 6.8, we contrast the opinion poll results, polarity (positive–negative), and Plutchik's eight basic emotions opposing pairs: joy–sadness, anger–fear, trust–disgust, and anticipation–surprise. The opinion poll reflects the percentage of people's approval with respect to the corresponding president's job performance [Consulta Mitofsky – www.consulta.mx, 2012; CID-Gallup – www.cidgallup.com, 2011]. if the percentage of approval is strictly over 50% we depict a black-filled triangle pointing up (▲), otherwise an empty-filled triangle pointing down is presented (▽). In the case of polarity and emotion pairs, for each pair A–B, if the emotional dimension A is grater than emotional dimension B, i.e., $A > B$, then a '▲' is shown, if $A == B$, then a symbol equal ('=') is presented, and finally, a '▽' is depicted if $A < B$.

We can observe that neither the polarity extracted, nor the single emotion pairs alone can fully explain the results of the opinion poll.

In order to analyze this outcome more in depth, we cast the problem as a binary classification problem [Bishop, 2007], where the objective is to predict whereas the public would approve (▲) or disapprove (▽) the president's job, we associate the numerical *label* values of $+1$ and -1, correspondingly.

ID	Country	opinion poll	positive–negative	joy–sadness	anger–fear	trust–disgust	anticipation–surprise
P1	Argentina	▲	▲	▲	▽	▲	▲
P2	Bolivia	▽	▲	▽	▽	▽	▽
P3	Brazil	▲	▲	▲	▽	▲	▲
P4	Chile	▽	▲	▽	▽	▽	=
P5	Colombia	▲	▽	▽	▽	▽	▲
P6	Costa Rica	▽	▽	▽	▽	▲	▲
P7	Dominican Republic	▲	▲	▲	▽	▲	▲
P8	Ecuador	▲	▽	▽	▽	=	▲
P9	El Salvador	▲	▲	▽	▽	▽	▽
P10	Guatemala	▽	▲	▲	▽	▲	▲
P11	Honduras	▽	▽	▽	=	▲	▽
P12	Mexico	▽	▽	▽	▽	▽	=
P13	Nicaragua	▲	▲	▲	▽	=	=
P14	Panama	▽	▽	▽	▽	▽	▽
P15	Paraguay	▽	▲	▽	▽	▽	▲
P16	Peru	▲	▲	▲	▽	▽	▲
P17	Uruguay	▽	▲	▲	▽	▲	▽
P18	Venezuela	▲	▲	▽	▽	▽	=

Table 6.8: **Contrast of opinion poll results, polarity (positive–negative), and Plutchik's eight basic emotions opposing pairs: joy–sadness, anger–fear, trust–disgust, and anticipation–surprise. (Presidents are listed alphabetically by country name).**

As training vectors we will use president's emotional vectors based on the emotion opposing pairs, plus the polarity dimensions, i.e., the training instance, X_p, for president p, is defined as follows:

$$X_p := [F_{\text{positive–negative}}, F_{\text{joy–sadness}}, F_{\text{anger–fear}}, F_{\text{trust–disgust}}, F_{\text{anticipation–surprise}}] \quad ,$$

where $F_{\text{A–B}}$, corresponds to the binary feature for polarity or emotion pair, A–B. $F_{\text{A–B}}$ is equal to $+1$, if $A > B$; equal to 0, if $A == B$; and equal to -1, if $A < B$.

For example, the corresponding training instance for the Argentinian president, is as follows: $X_{P1} = [1, 1, -1, 1, 1]$, and the corresponding label is $+1$, and for the president of Mexico, the training instance corresponds $X_{P12} = [-1, -1, -1, -1, 0]$, with a label -1 (see Table 6.8).

The predicted value of the poll is given by:

$$poll(p) = \vec{w} \cdot \vec{X}_p$$

where $\vec{w}$ is the weight vector to be learned. Our objective is to fit a linear model, e.g., using a Support Vector Machine (SVM), that is able to predict the binary value of the opinion poll. Please note that given the small amount of training data, we do not expect that this model will generalize well for unseen data, our objective is to discover what are the emotional features that are more influential. To this end, we will analyze the learned weights $\vec{w}$.

As metric, we use the Area Under ROC Curve (AUC) [Fawcett, 2006] to measure how well the linear model fits the data. The AUC value will always be between 0.0 and 1.0, being the best classifiers the ones with a higher AUC value. A random guess has an AUC equal to 0.5 .

We train the SVM model using Stochastic Gradient Descent [Bottou, 2010], with a learning rate equal to 0.001 and 5 epochs of 10000 iterations each, the regularization parameter is set to 0.0, since as we mentioned before, we are interested in fitting the model to the observed data. The averaged model weights obtained after repeating the procedure 100 times, are as follows:

$$\vec{w} = [0.249, 1.748, 0.265, -1.192, 1.694] \quad .$$

This particular model achieves an AUC = 0.81. If we use only polarity scores as predictors, the AUC drops significantly to 0.61. This indicates that polarity analysis is limited for popularity prediction, and a combination of emotions is a better approach for short-term popularity forecasts.

Roughly speaking, a high positive (resp. negative) weight indicates that presidents with these emotional features should be approved (resp. disapproved) by the people. The features corresponding to the emotional pairs *joy–sadness* and *anticipation–surprise* are the dominant terms of the expression, with weights 1.748 and 1.694, respectively.

The pair of emotions *sadness–fear* has a weight still over the polarity score pair. Note that the negative weight for the pair *trust–disgust* suggests a negative correlation with the opinion poll results.

6.4 Related Work

We present in this section related work to the problems discussed in the chapter, namely, Social Media Analytics for Epidemic Intelligence and Emotion Detection.

Social Media Analytics for Epidemic Intelligence

In order to detect public health events, supervised [Stewart et al., 2011b], unsupervised [Fisichella et al., 2011] and rule-based approaches have been used to extract public health events from social media and news. For example, PULS [Steinberger et al., 2008] identify the disease, time, location and cases of a news-reported event. It is integrated into MedISys, which automatically collects news articles concerning public health in various languages, and aggregates the extracted facts according to pre-defined categories, in a multi-lingual manner.

Other systems have sought to use the web and social media as a predictor to monitor and gauge the seasonal patterns of influenza. These systems correlate the queries used in search behavior with the infection rates of influenza-like illnesses statistics [Polgreen et al., 2008; Ginsberg et al., 2009].

Monitoring analysis has also been carried out on Twitter. The work of Chew et al. focused on the use of the terms “H1N1" and “swine flu" during the H1N1 2009 outbreak [Chew and Eysenbach, 2009]. They showed that the concise and timely nature of tweets can provide health officials with the a means to become aware, and respond to concerns raised by the public.

Culotta applied text classification to filter out tweets that are not reporting about influenza-like illnesses. Further, they modeled influenza rates by regression models and compared to U.S. Center of Disease Control statistics [Culotta, 2010].

Lampos and Cristianini also presented a monitoring tool for social media that is based on the textual analysis of micro-blog content [Lampos and Cristianini, 2010], [Lampos et al., 2010]. Their study focused on influenza-like illnesses in the UK and showed a correlation with data from the Health Protection Agency. Another study of Twitter content concentrated on influenza-like illnesses in the U.S. [Signorini et al., 2011]. Paul and Dredze [Paul and Dredze, 2011a,b] introduced a new aspect topic model for Twitter that associates symptoms, treatments and general words with diseases. Their focus is on general public health, not necessarily infectious diseases or disease outbreaks.

In contrast to these systems, we seek to not only detect and monitor potential public health threats, but also provide support for public health officials to asses the potential risk associated with the volume of information that is available within Twitter streams. Moreover, our proposed approach shows the potential of using Twitter for monitoring non-seasonal outbreaks in and geo-spacially sparse tweet locations.

Our work is similar to that of [Linge et al., 2011], were media reports on the 2011 EHEC outbreak in Germany are tracked. Although in their work no early warning was possible, they identified key aspects of developing outbreak stories. In contrast to this work, our approach exploits social media data and we show that a system can help to get early warnings on public health threats.

Although some works exist that address the task of ranking tweets, little effort has been devoted to explore personalized ranking of tweets in the domain of epidemic intelligence. For example Duan et al. rank individual generic tweets according to their relevance to a given query [Duan et al., 2010]. The features used include content relevance features, Twitter specific features and account authority features. In contrast, our is a personalized learning to rank approach for epidemic intelligence, that exploits an expanded user context by means of latent topics and on social hash-tagging behavior.

Social Media Analytics for Emotion Detection

Social media provides a rich and diverse source of users' opinions that has shown great potential for political analysis. For example, Demartini et al. [Demartini et al., 2011] apply sentiment and time series analysis techniques on blog data to estimate the temporal development of opinions for two candidates, Obama and McCain, during the US presidential elections in 2008. In contrast, our work applies sentiment analysis on tweets and small blog post snippets referring to Latin American presidents to estimate the effect that each of them has in the emotions reflected on user generated content. Another contrasting aspect is that we do not limit the analysis to candidates within the same country, but we study the presidents of eighteen Latin American countries.

Andranik Tumasjan et al. [Tumasjan et al., 2010], examine whether Twitter is a vehicle for online political deliberation by looking at how people use microblogging to exchange information about political issues. The authors evaluate whether Twitter messages reflect the current offline political sentiment in a meaningful way and analyze whether the activity on Twitter can be used to predict the popularity of parties or coalitions in the real world. Similarly, we analyze tweets and blog snippets as a source of political sentiment, however, our aim is to show how from these data, it is possible to build a profile of political figures, but in particular, current Latin American presidents. Furthermore, our analysis is performed outside elections period, which gives us interesting insights of how users feel with respect to actual presidents on a daily basis. The geographical area of our study and the language of the analyzed tweets and snippets are contrasting aspects as well, since we consider the area of Latin America and we analyze content written in Spanish.

Johan Bollen et al. [Bollen et al., 2011] explore how public mood patterns relate to fluctuations in macroscopic social and economic indicators in the same time period. The authors perform a sentiment analysis of Twitter data using an extended version of a psychometric instrument, the Profile of Mood States (POMS). Our work, on the contrary, focuses on how to exploit Social Media content (tweets and

snippets) to build a profile of each one of the Latin American presidents based on what people write about them and the sentiment expressed on the used vocabulary. Yet another contrast is the fact that we do not limit our analysis to the emotion or polarity extraction, but also exploit Plutchik's four opposing emotions pairs to train a linear model using SVM that provides insights on which combination of emotions explains the outcome of people's opinion.

6.5 Discussion

In the first part of the work presented in this chapter, we showed the potential of Twitter for early warning, we focused on the recent EHEC/HUS outbreak in Germany, and monitored the social stream. We applied several biosurveillance methods on a set of tweets collected in real time during the time of the event using Twitter API. All the detection methods triggered an alarm on May 20, a day ahead of well established early warning systems, such as MedISys.

In this particular case of the EHEC/HUS outbreak in German, we observed a time series with low oscillation beginning of the outbreak onset. Figure 6.1 shows that the observed variable for the event noticeably peaks within the outbreaks period, we note that this is generally the case of food-borne diseases that cause high international and media coverage due to the ease with which they spread. Since the number of tweets is zero outside of the outbreak period, it is expected that none of the detection methods would have major trouble generating a signal for this kind of pattern.

In other cases algorithm support is more critical, for example, in the presence of time series patterns with high oscillation and high magnitudes, exhibited by a disease that occurs continuously in a country, such as mumps or leptospirosis; or by diseases whose name is highly ambiguous, such as 'anthrax', which can also denote the Rock band. The number of early (false) alarms in these cases is too large to survey manually, and it is not trivial to identify significant aberrations. More research work in this direction is necessary in order to be able to detect

outbreak events for general disease patterns using the abundant but noisy social media data available.

After the detection of the outbreak, authorities investigating the cause and the impact in the population were interested in the analysis of micro-blog data related to the event. Thousands of tweets were produced every day, which made this task overwhelming for the experts. We proposed in this work a Personalized Tweet Ranking algorithm for Epidemic Intelligence (PTR4EI) that provides users a personalized short list of tweets that meets the context of their investigation. PTR4EI exploits features that go beyond the medical condition and location (i.e., user context), but includes complementary context information, extracted using LDA and the social hash-tagging behavior in Twitter, plus additional Twitter specific features. Our experimental evaluation showed the superior ranking performance of PTR4EI.

We are currently working closely with German and global public health institutions to help them integrate the monitoring of social media to their existing surveillance systems.

As future work, we plan to scale up our experiments, and to apply techniques of online ranking in order to update the model more efficiently as the outbreak develops.

We have shown the potential of Twitter to trigger early warnings in the case of sudden outbreaks and how personalized ranking for epidemic intelligence can be achieved. We believe our work can serve as a building block for an open early warning system based on Twitter, and hope that this chapter provides some insights into the future of epidemic intelligence based on social media streams.

We also explore a different application scenario for Social Web Analytics in the second part of the chapter, were we showed how the real-time nature of social media streams, in particular, Twitter, can be leveraged to take the pulse of political emotions in emerging regions of the world, namely: Latin America. We performed a sentiment analysis of tweets and brief blog posts over a period of six months. We applied a term-based method to detect the polarity and emotions associated to eighteen Latin American presidents.

We conclude that the extracted polarity and isolated emotions alone, are not good predictors for the opinion captured in an independent opinion poll. But the linear combination of basic emotions opposing pairs and polarity, achieved a prediction performance of 81% in terms of AUC, whereas using polarity alone the AUC dropped to 61%. We also noticed that the pair of emotions *joy–sadness* dominated the model.

Future work includes an online evaluation of our approach, where the global patterns discovered in the stream are immediately fed-back to the social web. We are interested in measuring how this information and awareness can affect individual and collective behavior.

All the 165,484 documents analyzed were in Spanish, but the emotion lexicon was in English. Previous approaches dealing with multilingualism in sentiment analysis use machine assisted translation to translate all documents to the target language, usually English. However, our aim is to process the social media stream in real-time, therefore the translation of documents arriving at high speed becomes problematic. Instead, we propose to translate the emotion lexicon itself to the original language of the documents, from English to Spanish. The translation of the lexicon is done offline and only once. This methodology allowed a more flexible architecture and the possibility of processing the documents more efficiently in real-time and in their original language.

Our study has limitations that have to be considered for future extensions, for example, the data collection process can be refined to consider more sophisticated named entity recognition, and not just the exact match of the names, moreover, the term-based sentiment analysis techniques that we applied are fast, but they do not consider complex context or senses, e.g., irony and sarcasm. For instance, the term *bad* (in Spanish: *malo*) has several emotions associated, namely, anger, disgust, fear, sadness, and a negative polarity (see Figure 6.6). More sophisticated techniques are necessary in order to understand better what is the context in which the term was used and its meaning in such situation. This represents an interesting and challenging research direction.

Additionally, a lexicon specialized in political sentiment, in the target language studied, could bring benefits in the precision of the sentiment extracted, such a lexicon could be built using crowdsourcing techniques, as in the case of EmoLex, the one we used in this study.

Finally, we hope that this work provides some insight into the future of short-term forecasting of disease outbreaks and political figures popularity, both key applications that could help us, as individuals and society, to gain a better understanding of the reality that surrounds us.

7 Conclusions and Further Research

Undoubtedly, analytics methods for the social web are becoming increasingly necessary to make sense out of the huge amount of user generated content and reduce the complexity for human user understanding. Current research greatly benefits from cross-disciplines, including machine learning, recommender systems, and computational social science. The integration of interdisciplinary evidence also represents an important ingredient of this work, and our research on collaborative filtering in social media streams and collective intelligence, expands focus into new directions, namely that of the emerging science of the Web and *Living Analytics*.

The main objective of this work is to provide a set of tools to capture people's interactions from a highly dynamic stream data, automatically annotate resources on the Web, and to understand and predict user actions and preferences. The objective of such *Living Analytics* models is to assist people with the overwhelming amount of choices they face as they consume goods, services, social media and leisure time [Schwartz, 2004].

The main problem is not the actual access to the content (e.g., *Twitter* or *YouTube*), rather the problem is to transform this huge mass of data into useful insight. Effective recommendation systems and personalized rankings are key elements in this scenario. In Chapter 3 of this work, we tackle the problem of personalized ranking, that is, instead of creating one global ranking, the rankings should reflect the individual taste of the users. Based on these ideas, we have introduced `RMFO`, a method that in the presence of highly dynamic social media streams, creates in real-time user-specific rankings based on individual preferences that are inferred from users' past system interactions. We also proposed `RMFX`, a novel approach that follows a selective sampling

strategy to perform online model updates based on *active learning principles*, that closely simulates the task of identifying relevant items from a pool of mostly uninteresting ones. Both, `RMFO` and `RMFX` are online ranking approaches for collaborative filtering models based on matrix factorization. At their core stochastic gradient descent is used for optimization, which makes the algorithms easy to implement and efficiently scalable to large-scale datasets.

We demonstrated the usefulness of both methods for the task of recommending personalized topics to users. Having Twitter as test bed, we showed that our online approaches largely outperform highly competitive state-of-the-art matrix factorization techniques for collaborative filtering, not only in terms of recommendation quality, but also in terms of time and space savings.

In Chapter 4, we studied an approach to learn ranking functions that directly optimize non-smooth IR metrics using Swarm Intelligence. Our approach, *SwarmRank*, directly optimizes Mean Average Precision (MAP), an evaluation measure frequently used in IR, in comparison with many learning to rank algorithms, which minimize a loss function loosely related to the IR measures.

In addition, we extended our ideas to address the item recommendation task in the context of recommender systems, to this end, we presented *SwarmRankCF*, a collaborative learning to rank algorithm based on swarm intelligence. While learning to rank algorithms use hand-picked features to represent items, we learn such features based on user-item interactions, and apply a PSO-based optimization algorithm that directly maximizes MAP, which lead us to improve the recommendation performance for the Top-N recommendation tasks.

In Chapter 5, we addressed the *cold start problem* in online environments where novel items appear rapidly, we presented our approach: α-*TaggingLDA*, which is based on the probabilistic topic model: Latent Dirichlet Allocation (LDA). We showed the ability of our method to enrich sparse and limited textual information by means of exploiting the resource redundancy and latent topic overlap between similar resources found in an auxiliary domain. Our approach has the poten-

tial to facilitate search, ease navigation (e.g., tag clouds), and improve personalization.

Finally in Chapter 6 and in the scope of *Collective Intelligence* [MIT, 2012], we explored and demonstrated the potential of monitoring social media streams (e.g., Twitter) for early warning of disease outbreaks and presented an innovative personalized ranking approach that offers decision makers the most relevant and attractive tweets for risk assessment, by exploiting latent topics and social hash-tagging behavior in Twitter. In addition, we also presented an empirical study that shows how the real-time nature of social media streams can be leveraged to take the pulse of political emotions in emerging regions of the world, namely: Latin America. The study provided insights on which combination of emotions explains better the public's opinion.

7.1 Summary of Contributions

Our research and contributions made derive from issues that appear when analyzing the dynamics of the Social Web. These issues were outlined in Chapter 1. We then devised approaches to tackle some of these issues, e.g., online collaborative filtering for social media streams, personalized ranking, automatic tagging, etc., blending together into a synergetic set of tools for *Living Analytics*.

These contributions fall mainly into two categories:

- **Recommender Systems.** The models `RMFO` and `RMFX` presented in Chapter 3 advance the state-of-the-art in Online Collaborative Filtering. Moreover, we proposed a novel approach based on Swarm Intelligence (*SwarmRankCF*) to directly optimize ranking functions for item recommendations (see Chapter 4). Novel approaches to address the cold start problem in social recommender systems are discussed in Chapter 5. In addition, we offer a personalized ranking algorithm for epidemic intelligence in Chapter 6.

- **Collective Intelligence.** Our contributions in the field of computational social science are twofold. First, we explored how social media streams can be exploited for Epidemic Intelligence (see

Chapter 6) and showed its potential for early warning detection and outbreak analysis and control. Second, we also showed in Chapter 6 how the real-time nature of social media streams can be leveraged to take the pulse of political emotions.

The methods presented in this book constitute a set of tools to understand and analyze the social web addressing the *Living Analytics* challenges (Chapter 1). Note that the analytics *toolbox* presented is by no means complete, giving interesting opportunities for future research.

7.2 Outlook and Future Directions

We consider that a major strength of this work lies in its versatility and variety, making contributions in diverse fields. The single contributions are not necessarily confined to the *Living Analytics* scope, but also extends to other fields, for example, latent factor models for stream data, personalized ranking, and information filtering and retrieval.

There are several potential extensions and applications of the ideas presented in this work, particularly related to information filtering in the presence of highly dynamic data.

- **Better Learning Algorithms for Online Collaborative Filtering.** The success of collaborative filtering heavily relies upon the ability to translate the observed behavior to a meaningful cost function. We strongly believe that The Top-N recommendation task needs to be treated as a ranking problem as discussed in Chapter 3. The exploration of directly optimizing information retrieval metrics for personalized ranking has started and may significantly improve recommendation performance.

- **Prediction of Individual and Collective Behavior in Real-Time Context.** Modeling complex nonlinear dynamics and high-dimensional data, such as social media streams, is an active area of research in machine learning and recommender systems. Many of the existing models, such as matrix factorization and neighborhood based algorithms have been widely used

in practice. However, these models are limited in the types of structure they can model. What other methods could potentially capture nonlinear dynamics and also make multimodal predictions handling missing inputs?

- **Real-Time Experimentation.** How to conduct experimental evaluations at large scale in real-time networked settings involving users and their group interactions? A/B testing is a common practice in the industry to evaluate new project features and to support decision making processes, but such evaluations are expensive and time consuming. The exploration of new approaches that align long-term goals with the objective functions optimized by the learning models is an interesting research direction.

We have outlined several potential research directions. However, research on *Living Analytics* as unified field is very new, and there are many broad open questions to consider as outlined in the Introduction. We believe that answering many of those questions will allow us to build more intelligent systems for the benefit of society.

Bibliography

Abel, F., Diaz-Aviles, E., Henze, N., Krause, D., and Siehndel, P. (2010). Analyzing The Blogosphere For Predicting The Success Of Music And Movie Products. *International Conference on Advances in Social Network Analysis and Mining (ASONAM'10)*, pages 276–280.

Abel, F., Frank, M., Henze, N., Krause, D., Plappert, D., and Siehndel, P. (2007). Groupme! - Where Semantic Web Meets Web 2.0. In *ISWC/ASWC*.

Abel, F., Marenzi, I., Nejdl, W., and Zerr, S. (2009). Sharing Distributed Resources In Learnweb2.0. In *EC-TEL*, pages 154–159.

Baeza-Yates, R. and Ribeiro-Neto, B. (2011). *Modern Information Retrieval*. Addison Wesley, 2nd edition.

Basseville, M. and Nikiforov, I. (1993). *Detection Of Abrupt Changes: Theory And Application*. Prentice Hall.

Bell, R. M. and Koren, Y. (2007). Scalable Collaborative Filtering With Jointly Derived Neighborhood Interpolation Weights. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*.

Bhattacharya, I. and Getoor, L. (2006). A Latent Dirichlet Model For Unsupervised Entity Resolution. In Ghosh, J., Lambert, D., Skillicorn, D. B., and Srivastava, J., editors, *SDM*. SIAM.

Bishop, C. M. (2007). *Pattern Recognition And Machine Learning (Information Science And Statistics)*. Springer, 1 edition.

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

Bollen, J., Mao, H., and Pepe, A. (2011). Modeling Public Mood And Emotion: Twitter Sentiment And Socio-economic Phenomena. In *International AAAI Conference on Weblogs and Social Media*.

Bottou, L. (2010). Large-scale Machine Learning With Stochastic Gradient Descent. In Lechevallier, Y. and Saporta, G., editors, *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, Paris, France. Springer.

Bratton, D. and Kennedy, J. (2007). Defining A Standard For Particle Swarm Optimization. *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE.*

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning To Rank Using Gradient Descent. In *ICML '05: Proceedings of the 22nd international Conference on Machine learning*, pages 89–96, New York, NY, USA. ACM.

Burkom, H. (2005). Accessible Alerting Algorithms For Biosurveillance. In *National Syndromic Surveillance Conference 2005. USA.*

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. (2007). Learning To Rank: From Pairwise Approach To Listwise Approach. In *ICML '07: Proceedings of the 24th international Conference on Machine Learning*, pages 129–136, New York, NY, USA. ACM.

Celma, O. (2010). *Music Recommendation And Discovery In The Long Tail.* Springer.

Chapelle, O., Chang, Y., and Liu, T.-Y. (2011). Future Directions In Learning To Rank. *JMLR*, 14:91–100.

Chew, C. and Eysenbach, G. (2009). Pandemics In The Age Of Twitter: Content Analysis Of Tweets During The 2009 H1N1 Outbreak. *PLoS ONE*, 5(11):e14118.

CID-Gallup – www.cidgallup.com (2011). Encuesta De Opinión Pública Centro América Y República Dominicana. http://goo.gl/lc85v.

Collier, N. (2010). What's Unusual In Online Disease Outbreak News? *Journal of Biomedical Semantics*, 1(1):2.

Consulta Mitofsky – www.consulta.mx (2012). Aprobación De Mandatarios América Y El Mundo. http://goo.gl/8fFNU.

Corley, C. D., Cook, D. J., Mikler, A. R., and Singh, K. P. (2010). Text And Structural Data Mining Of Influenza Mentions In Web And Social Media. *Int J Environ Res Public Health*, 7(2):596–615.

Crammer, K. and Singer, Y. (2002). Pranking With Ranking. In *Advances in Neural Information Processing Systems 14*, pages 641–647. MIT Press.

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance Of Recommender Algorithms On Top-N Recommendation Tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, New York, NY, USA. ACM.

Culotta, A. (2010). Towards Detecting Influenza Epidemics By Analyzing Twitter Messages. In *First Workshop on Social Media Analytics*.

Das, A. S., Datar, M., Garg, A., and Rajaram, S. (2007). Google News Personalization: Scalable Online Collaborative Filtering. In *Proceedings of the World Wide Web*.

de Almeida, H. M., Gonçalves, M. A., Cristo, M., and Calado, P. (2007). A Combined Component Approach For Finding Collection-adapted Ranking Functions Based On Genetic Programming. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR Conference on Research and development in information retrieval*, pages 399–406, New York, NY, USA. ACM.

Demartini, G., Siersdorfer, S., Chelaru, S., and Nejdl, W. (2011). Analyzing Political Trends In The Blogosphere. In *International AAAI Conference on Weblogs and Social Media*.

Department, T. M., Minka, T., and Lafferty, J. (2002). Expectation-propagation For The Generative Aspect Model. In *In Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 352–359. Morgan Kaufmann.

Deshpande, M. and Karypis, G. (2004). Item-based Top-N Recommendation Algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177.

Diaz-Aviles, E., Drumond, L., Gantner, Z., Schmidt-Thieme, L., and Nejdl, W. (2012a). What Is Happening Right Now ... That Interests Me? Online Topic Discovery And Recommendation In Twitter. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12.

Diaz-Aviles, E., Drumond, L., Schmidt-Thieme, L., and Nejdl, W. (2012b). Real-time Top-N Recommendation In Social Streams. In

Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys '12, pages 59–66, New York, NY, USA. ACM.

Diaz-Aviles, E., Fisichella, M., Kawase, R., Nejdl, W., and Stewart, A. (2011a). Unsupervised Auto-tagging For Learning Object Enrichment. In *Proceedings of the 6th European Conference on Technology Enhanced Learning: Towards Ubiquitous Learning*, EC-TEL'11, pages 83–96, Berlin, Heidelberg. Springer-Verlag.

Diaz-Aviles, E., Georgescu, M., and Nejdl, W. (2012c). Swarming To Rank For Recommender Systems. In *Proceedings of the sixth ACM Conference on Recommender Systems*, RecSys '12, pages 229–232, New York, NY, USA. ACM.

Diaz-Aviles, E., Georgescu, M., Stewart, A., and Nejdl, W. (2010). Lda For On-the-fly Auto Tagging. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 309–312, New York, NY, USA. ACM.

Diaz-Aviles, E. and Kawase, R. (2012). Exploiting Twitter As A Social Channel For Human Computation. In *Proceedings of the First International Workshop on Crowdsourcing Web Search, Collocated with WWW'12. Lyon, France, April 17, 2012.*, CrowdSearch'12.

Diaz-Aviles, E., Nejdl, W., and Schmidt-Thieme, L. (2009). Swarming To Rank For Information Retrieval. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 9–16, New York, NY, USA. ACM.

Diaz-Aviles, E., Orellana-Rodriguez, C., and Nejdl, W. (2012d). Taking The Pulse Of Political Emotions In Latin America Based On Social Web Streams. In *LA-WEB '12: Proceedings of the 2012 Latin American Web Conference*. IEEE Computer Society.

Diaz-Aviles, E., Siehndel, P., and Naini, K. D. (2011b). Exploiting Social #-tagging Behavior In Twitter For Information Filtering And Recommendation (microblog Track). In *Proceedings of The Twentieth Text REtrieval Conference, TREC 2011, Gaithersburg, Maryland, USA, November 15-18, 2011*, TREC.

Diaz-Aviles, E. and Stewart, A. (2012). Tracking Twitter For Epidemic Intelligence. Case Study: Ehec/hus Outbreak In Germany 2011. In

Proceedings of the 4th ACM International Conference on Web Science, WebSci '12.

Diaz-Aviles, E., Stewart, A., Velasco, E., Denecke, K., and Nejdl, W. (2012e). Epidemic Intelligence For The Crowd By The Crowd. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012*.

Diaz-Aviles, E., Stewart, A., Velasco, E., Denecke, K., and Nejdl, W. (2012f). Towards Personalized Learning To Rank For Epidemic Intelligence Based On Social Media Streams. In *Procedings of the 21st International Conference Companion on World Wide Web*, WWW '12 Companion, pages 495–496, New York, NY, USA. ACM.

Dodds, P. S., Harris, K. D., Kloumann, I. M., Bliss, C. A., and Danforth, C. M. (2011). Temporal Patterns Of Happiness And Information In A Global Social Network: Hedonometrics and Twitter. *PLoS ONE*, 6(12):e26752.

Drachsler, H., Rutledge, L., van Rosmalen, P., Hummel, H. G. K., Pecceu, D., Arts, T., Hutten, E., and Koper, R. (2010). Remashed - An Usability Study Of A Recommender System For Mash-ups For Learning. *iJET*, 5(S1):7–11.

Duan, Y., Jiang, L., Qin, T., Zhou, M., and Shum, H.-Y. (2010). An Empirical Study On Learning To Rank Of Tweets. In *Coling 2010*.

Eberhart, R. and Kennedy, J. (1995). A New Optimizer Using Particle Swarm Theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, Nagoya, Japan. Piscataway, NJ: IEEE Service Center.

Eisterlehner, F., Hotho, A., and Jäschke, R., editors (2009). *Ecml Pkdd Discovery Challenge 2009 (dc09) Http://www.kde.cs.uni-kassel.de/ws/dc09*, volume 497 of *CEUR-WS.org*.

Ertekin, S., Huang, J., Bottou, L., and Giles, L. (2007). Learning On The Border: Active Learning In Imbalanced Data Classification. In *Proceedings of the ACM CIKM Conference*.

Fan, W., Gordon, M. D., and Pathak, P. (2000). Personalization Of Search Engine Services For Effective Retrieval And Knowledge Management. In *ICIS '00: Proceedings of the twenty first international*

Conference on Information Systems, pages 20–34, Atlanta, GA, USA. Association for Information Systems.

Fan, W., Gordon, M. D., and Pathak, P. (2004). Discovery Of Context-specific Ranking Functions For Effective Information Retrieval Using Genetic Programming. *IEEE Trans. on Knowl. and Data Eng.*, 16(4):523–527.

Fan, W., Gordon, M. D., and Pathak, P. (2005). Genetic Programming-based Discovery Of Ranking Functions For Effective Web Search. *J. Manage. Inf. Syst.*, 21(4):37–56.

Fawcett, T. (2006). An Introduction To Roc Analysis. *Pattern Recognition Letters*, 27(8):861 – 874.

Fisichella, M., Stewart, A., Cuzzocrea, A., and Denecke, K. (2011). Detecting Health Events On The Social Web To Enable Epidemic Intelligence. In *String Processing and Information Retrieval*, volume 7024 of *Lecture Notes in Computer Science. SPIRE'11*, pages 87–103. Springer Berlin Heidelberg.

Frank, C., Werber, D., Cramer, J. P., Askar, M., Faber, M., an der Heiden, M., Bernard, H., Fruth, A., Prager, R., Spode, A., Wadl, M., Zoufaly, A., Jordan, S., Kemper, M. J., Follin, P., Müller, L., King, L. A., Rosner, B., Buchholz, U., Stark, K., and Krause, G. (2011). Epidemic Profile Of Shiga-toxin–producing Escherichia Coli O104:h4 Outbreak In Germany. *New England Journal of Medicine*, 365(19):1771–1780.

Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y. (2003). An Efficient Boosting Algorithm For Combining Preferences. *J. Mach. Learn. Res.*, 4:933–969.

Gantner, Z., Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2011). Mymedialite: A Free Recommender System Library. In *Proceedings of the ACM RecSys Conference*.

Gemulla, R., Nijkamp, E., Haas, P. J., and Sismanis, Y. (2011). Large-scale Matrix Factorization With Distributed Stochastic Gradient Descent. In *Proceedings of the 17th ACM SIGKDD international Conference on Knowledge discovery and data mining*, KDD '11, pages 69–77, New York, NY, USA. ACM.

Giles, J. (2012). Computational Social Science: Making The Links. *Nature*, 488(7412):448–450.

Ginsberg, J., Mohebbi, M. H., Patel, R. S., Brammer, L., Smolinski, M. S., and Brilliant, L. (2009). Detecting Influenza Epidemics Using Search Engine Query Data. *Nature*.

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using Collaborative Filtering To Weave An Information Tapestry. *Commun. ACM*, 35:61–70.

Griffiths, T. L. and Steyvers, M. (2004). Finding Scientific Topics. *Proc Natl Acad Sci U S A*, 101 Suppl 1:5228–5235.

Haghani, P., Michel, S., and Aberer, K. (2010). The Gist Of Everything New: Personalized Top-k Processing Over Web 2.0 Streams. In *Proc. of the 19th ACM international Conference on Information and knowledge management*, CIKM '10, pages 489–498, New York, NY, USA. ACM.

Herbrich, R., Graepel, T., and Obermayer, K. (2000). Large Margin Rank Boundaries For Ordinal Regression. In Smola, A., Bartlett, P., Schölkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pages 115–132, Cambridge, MA. MIT Press.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.*, 22:5–53.

Heymann, P., Ramage, D., and Garcia-Molina, H. (2008). Social Tag Prediction. In *31st Annual International ACM SIGIR Conference (SIGIR'08)*.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative Filtering For Implicit Feedback Datasets. In *Proceedings of the IEEE ICDM Conference*.

Hutwagner, L., Thompson, W., Seeman, G. M., and Treadwell, T. (2003). The Bioterrorism Preparedness And Response Early Aberration Reporting System (EARS). *Journal of urban health bulletin of the New York Academy of Medicine*, 80(2 Suppl 1):i89–i96.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated Gain-based Evaluation Of Ir Techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Jäschke, R., Eisterlehner, F., Hotho, A., and Stumme, G. (2009). Testing And Evaluating Tag Recommenders In A Live System. In *RecSys '09*. ACM.

Jäschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., and Stumme, G. (2008). Tag Recommendations In Social Bookmarking Systems. *AI Communications*, pages 231–247.

Joachims, T. (2002). Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the eighth ACM KDD Conference*.

Karimi, R., Freudenthaler, C., Nanopoulos, A., and Schmidt-Thieme, L. (2011). Towards Optimal Active Learning For Matrix Factorization In Recommender Systems. In *IEEE ICTAI Conference*.

Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE Press.

Khan, S. A. (2007). Handbook Of Biosurveillance. *Journal of Biomedical Informatics*.

Kim, T.-K., Kim, H., Hwang, W., and Kittler, J. (2005). Component-based Lda Face Description For Image Retrieval And Mpeg-7 Standardisation. *Image Vision Comput.*, 23(7):631–642.

Kleinberg, J. (1999). Authoritative Sources In A Hyperlinked Environment. *J. ACM*, 46(5):604–632.

Koren, Y. (2008). Factorization Meets The Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceeding of the 14th ACM SIGKDD international Conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, New York, NY, USA. ACM.

Koren, Y. (2009). Collaborative Filtering With Temporal Dynamics. In *Proc. of the 15th ACM International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 447–456, New York, NY, USA. ACM.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix Factorization Techniques For Recommender Systems. *Computer*.

Krestel, R. and Fankhauser, P. (2009). Tag Recommendation Using Probabilistic Topic Models. In *ECML/PKDD Discovery Challenge (DC'09)*.

Krestel, R., Fankhauser, P., and Nejdl, W. (2009). Latent Dirichlet Allocation For Tag Recommendation. In *Proceedings of the third ACM Conference on Recommender Systems*, RecSys '09, pages 61–68, New York, NY, USA. ACM.

Lampos, V., Bie, T. D., and Cristianini, N. (2010). Flu Detector Tracking Epidemics On Twitter. In *ECML PKDD 2010*.

Lampos, V. and Cristianini, N. (2010). Tracking The Flu Pandemic By Monitoring The Social Web. In *IAPR 2nd Workshop on Cognitive Information Processing (CIP 2010)*.

LARC (2012). Living Analytics Research Center. http://www.larc.smu.edu.sg/.

Li, F.-F. and Perona, P. (2005). A Bayesian Hierarchical Model For Learning Natural Scene Categories. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 524–531. IEEE Computer Society.

Li, G., Chang, K., Hoi, S. C., Liu, W., and Jain, R. (2011). Collaborative Online Learning Of User Generated Content. In *Proc. of the 20th ACM international Conference on Information and knowledge management*, CIKM '11, pages 285–290, New York, NY, USA. ACM.

Lin, J., Snow, R., and Morgan, W. (2011). Smoothing Techniques For Adaptive Online Language Models: Topic Tracking In Tweet Streams. In *Proceedings of the 17th ACM SIGKDD international Conference on Knowledge discovery and data mining*, KDD '11, pages 422–429, New York, NY, USA. ACM.

Linge, J., Mantero, J., Fuart, F., Belyaeva, J., Atkinson, M., and Van Der Goot, E. (2011). Tracking Media Reports On The Shiga Toxin-producing Escherichia Coli O104:H4 Outbreak in Germany. In *ICST Conference on eHealth, 2011*.

Liu, B. (2012). *Sentiment Analysis And Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Liu, N. N., Zhao, M., and Yang, Q. (2009). Probabilistic Latent Preference Analysis For Collaborative Filtering. In *Proc. of the 18th ACM CIKM*, CIKM '09, pages 759–766, New York, NY, USA. ACM.

Liu, T.-Y. (2011). *Learning To Rank For Information Retrieval*. Springer.

Liu, T. Y., Xu, J., Qin, T., Xiong, W., and Li, H. (2007). Letor: Benchmark Dataset For Research On Learning To Rank For Information Retrieval. In *SIGIR '07: Proceedings of the Learning to Rank workshop in the 30th annual international ACM SIGIR Conference on Research and development in information retrieval*.

Lohmann, S., Thalmann, S., Harrer, A., and Maier, R. (2008). Learner-generated Annotation Of Learning Resources - Lessons From Experiments on Tagging. In *International Conference on Knowledge Management (I-KNOW 08), Graz, Austria, September 2008*.

McCallum, A. K. (2002). Mallet: A Machine Learning For Language Toolkit. http://mallet.cs.umass.edu.

MIT (2012). Handbook Of Collective Intelligence (wiki).

Mohammad, S. (2011). From Once Upon A Time To Happily Ever After: Tracking Emotions In Novels And Fairy Tales. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 105–114, Portland, OR, USA. Association for Computational Linguistics.

Mohammad, S. M. and Turney, P. D. (2011). Crowdsourcing A Word-emotion Association Lexicon. *Computational Intelligence*.

Muthukrishnan, S. (2005). *Data Streams: Algorithms And Applications*. Now Publishers.

Nie, L., Davison, B. D., and Qi, X. (2006). Topical Link Analysis For Web Search. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR Conference on Research and development in information retrieval*, pages 91–98, New York, NY, USA. ACM.

Niemann, K., Schwertel, U., Kalz, M., Mikroyannidis, A., Fisichella, M., Friedrich, M., Dicerto, M., Ha, K.-H., Holtkamp, P., and Kawase, R. (2010). Skill-based Scouting Of Open Management Content. In Wolpers, M., Kirschner, P. A., Scheffel, M., Lindstaedt, S. N., and Dimitrova, V., editors, *EC-TEL*, volume 6383 of *Lecture Notes in Computer Science*, pages 632–637. Springer.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The Pagerank Citation Ranking: Bringing Order To The Web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

Paul, M. J. and Dredze, M. (2011a). A Model For Mining Public Health Topics From Twitter. Technical report, Johns Hopkins University.

Paul, M. J. and Dredze, M. (2011b). You Are What You Tweet: Analyzing Twitter For Public Health. In *ICWSM'11*.

Phan, X.-H., Nguyen, L.-M., and Horiguchi, S. (2008). Learning To Classify Short And Sparse Text & Web With Hidden Topics From Large-scale Data Collections. In *WWW '08*. ACM.

Plutchik, R. (1980). *A General Psychoevolutionary Theory Of Emotion*, pages 3–33. Academic press, New York.

Polgreen, P. M., Chen, Y., Pennock, D. M., Nelson, F. D., and Weinstein, R. (2008). Using Internet Searches For Influenza Surveillance. *Clinical Infectious Diseases*, 47(11):1443–48.

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle Swarm Optimization. *Swarm Intelligence*, 1(1):33–57.

Qin, T., Liu, T.-Y., Zhang, X.-D., Chen, Z., and Ma, W.-Y. (2005). A Study Of Relevance Propagation For Web Search. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR Conference on Research and development in information retrieval*, pages 408–415, New York, NY, USA. ACM.

Rendle, S., Balby Marinho, L., Nanopoulos, A., and Lars, S.-T. (2009a). Learning Optimal Ranking With Tensor Factorization For Tag Recommendation. In *Proceedings of the 15th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 727–736, New York, NY, USA. ACM.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009b). BPR: Bayesian Personalized Ranking From Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States. AUAI Press.

Rendle, S. and Lars, S.-T. (2010). Pairwise Interaction Tensor Factorization For Personalized Tag Recommendation. In *Proc. of the third ACM international Conference on Web search and data mining*, WSDM '10, pages 81–90, New York, NY, USA. ACM.

Rendle, S. and Schmidt-Thieme, L. (2008). Online-updating Regularized Kernel Matrix Factorization Models For Large-scale Recommender Systems. In *Proceedings of the ACM RecSys Conference*.

Resnick, P. and Varian, H. R. (1997). Recommender Systems. *Commun. ACM*, 40(3):56–58.

Robert Koch Institute (RKI) (2011). Final Presentation And Evaluation Of Epidemiological Findings In The EHEC O104:H4 Outbreak, Germany 2011. Technical report, RKI. `http://goo.gl/9tciB`.

Robertson, S. E. (1997). Overview Of The Okapi Projects. *Journal of Documentation*, 53(1):3–7.

Schmid, H. (1994). Probabilistic Part-of-speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.

Schwartz, B. (2004). *The Paradox Of Choice : Why More Is Less*. Harper Perennial.

Sculley, D. (2009). Large Scale Learning To Rank. In *NIPS 2009 Workshop on Advances in Ranking*.

Sculley, D. (2010). Combined Regression And Ranking. In *Proceedings of the ACM KDD Conference*.

Semiocast (2012). Countries On Twitter. http://goo.gl/RfxZw.

Shakery, A. and Zhai, C. (2003). Relevance Propagation For Topic Distillation Uiuc Trec 2003 Web Track Experiments. In *TREC*, pages 673–677.

Shi, Y., Larson, M., and Hanjalic, A. (2010). List-wise Learning To Rank With Matrix Factorization For Collaborative Filtering. In *Proc. of the 4th ACM RecSys*, RecSys '10, pages 269–272, New York, NY, USA. ACM.

Signorini, A., Segre, A. M., and Polgreen, P. M. (2011). The Use Of Twitter To Track Levels Of Disease Activity And Public Concern In the U.S. During The Influenza A H1N1 Pandemic. *PLoS ONE*.

Smola, A. J. and Schölkopf, B. (2000). Sparse Greedy Matrix Approximation For Machine Learning. In *Proceedings of the International Conference on Machine Learning*.

Sofean, M., Stewart, A., Denecke, K., and Smith, M. (2012). Medical Case-driven Classification Of Microblogs: Characteristics And Annotation. In *ACM IHI 2012*.

Song, Y., Zhuang, Z., Li, H., Zhao, Q., Li, J., Lee, W.-C., and Giles, C. L. (2008). Real-time Automatic Tag Recommendation. In *SIGIR '08*. ACM.

Srebro, N., Rennie, J., and Jaakkola, T. (2005). Maximum-margin Matrix Factorization. *Advances in neural information processing Systems*, 17:1329–1336.

Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., and Demirbas, M. (2010). Short Text Classification In Twitter To Improve Information Filtering. In *ACM SIGIR'10*, New York, NY, USA.

Steinberger, R., Fuart, F., Goot, E. V. D., and Best, C. (2008). Text Mining From The Web For Medical Intelligence. *Mining massive data sets for security Advances in data mining search social networks and text mining and their applications to security*, 19.

Stewart, A., Diaz-Aviles, E., and Nanopoulos, A. (2011a). Self-supervised Detection Of Disease Reporting Events In Outbreak Reports. In *IEEE International Conference on Information Reuse and Integration (IRI'11). August 2011, Las Vegas, Nevada, USA*, pages 416–421.

Stewart, A., Diaz-Aviles, E., and Nejdl, W. (2008). Mining User Profiles To Support Structure And Explanation In Open Social Networking. *International Workshop on Interacting with Multimedia Content in*

the Social Semantic Web (IMC-SSW 2008). Collocated with the 3rd International Conference on Semantic and Digital Media Technologies (SAMT 2008). Koblenz, Germany, Dec. 03 2008.

Stewart, A., Diaz-Aviles, E., Nejdl, W., Marinho, L. B., Nanopoulos, A., and Schmidt-Thieme, L. (2009). Cross-tagging For Personalized Open Social Networking. In *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*, HT '09, pages 271–278, New York, NY, USA. ACM.

Stewart, A., Smith, M., and Nejdl, W. (2011b). A Transfer Approach To Detecting Disease Reporting Events In Blog Social Media. In *ACM HT '11*.

Tong, S. and Koller, D. (2002). Support Vector Machine Active Learning With Applications To Text Classification. *J. Mach. Learn. Res.*, 2:45–66.

Tumasjan, A., Sprenger, T., Sandner, P., and Welpe, I. (2010). Predicting Elections With Twitter: What 140 Characters Reveal About Political Sentiment. In *International AAAI Conference on Weblogs and Social Media*.

Vapnik, V. N. (1995). *The Nature Of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Vitter, J. S. (1985). Random Sampling With A Reservoir. *ACM Trans. Math. Softw.*, 11:37–57.

Wetter, M. (2011). Generic Optimization Program – Genopt. *User Manual,User Manual Version 3.1.0. Lawrence Berkeley National Laboratory.*

Wiebe, J. M. (1994). Tracking Point Of View In Narrative. *Comput. Linguist.*, 20(2):233–287.

Xing, D. and Girolami, M. (2007). Employing Latent Dirichlet Allocation For Fraud Detection In Telecommunications. *Pattern Recogn. Lett.*, 28(13):1727–1734.

Xue, G.-R., Yang, Q., Zeng, H.-J., Yu, Y., and Chen, Z. (2005). Exploiting The Hierarchical Structure For Link Analysis. In *SIGIR*

'05: Proceedings of the 28th annual international ACM SIGIR Conference on Research and development in information retrieval, pages 186–193, New York, NY, USA. ACM.

Yang, J. and Leskovec, J. (2011). Patters Of Temporal Variation In Online Media. In *Proceedings of the ACM WSDM Conference*.

Yeh, J.-Y., Lin, J.-Y., Ke, H.-R., and Yang, W.-P. (2007). Learning To Rank For Information Retrieval Using Genetic Programming. In *SIGIR 2007 Workshop on Learning to rank for information retrieval (LR4IR 2007)*, New York, NY, USA. ACM.

Yu, H. (2005). SVM Selective Sampling For Ranking With Application To Data Retrieval. In *Proceedings of the eleventh ACM KDD Conference*.

Zhai, C. and Lafferty, J. (2004). A Study Of Smoothing Methods For Language Models Applied To Information Retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214.

Zhao, P., Hoi, S., Jin, R., and Yang, T. (2011). Online Auc Maximization. In *Proceedings of the International Conference on Machine Learning*.

Zubiaga, A. (2012). Real-time Analysis And Mining Of Social Streams. In *International AAAI Conference on Weblogs and Social Media*, ICWSM'12.

Ernesto Diaz-Aviles is a Web Research Scientist whose current research interests focus on supervised machine learning, social computing, and applications to problems in large-scale recommender systems and social media. Ernesto holds a Doctor of Natural Sciences degree (Dr. rer. nat.) from the L3S Research Center at the Leibniz University of Hannover, Germany, a MSc in Computer Science from the University of Freiburg, Germany, and a BSc in Electrical Engineering from the Central American University (UCA), El Salvador. He is also an entrepreneur and has served as IT consultant providing advice to a range of public and private organizations.