

UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA  
Dipartimento di Informatica Sistemistica e Comunicazione  
Dottorato di Ricerca in Informatica - XXVII Ciclo



# Advancing Recommender Systems from the Algorithm, Interface and Methodological Perspective

**Marco Rossetti**  
Ph.D. Thesis

Supervisor:  
Prof. Fabio Stella  
Tutor:  
Prof. Francesca Arcelli

Coordinator:  
Prof. Stefania Bandini

ANNO ACCADEMICO 2013-2014



# Acknowledgments

First of all I want to thank my supervisor, Prof. Fabio Stella. I can say that I was lucky to meet him in my life. Not only has he taught me how to do research, but with his example he also taught me important life lessons. Thanks to him I learnt to question my actions and to challenge accepted norms, even if it seemed the way to follow. I think he was one of people who had the most positive influence on my life.

I also want to thank Prof. Markus Zanker, whose help was fundamental in my exploration of the field of recommender systems. It was always a pleasure to speak with him and he gave me important suggestions through the last years of my PhD.

During these years I also spent 6 months in Sydney thank to Prof. Longbing Cao who hosted me at his institute. That period enriched me with many experiences in a completely different environment, and I want to thank Prof. Cao for giving me this opportunity. Furthermore, I want also to thank the Australian government, which awarded me with the Endeavour Research Fellowship.

In the past three years I also had the opportunity to know and to work with great people, such as Alessandro, Simone, Davide, Daniele and Marco. I was very lucky to meet these guys, and I hope that our friendship will continue even after this period.

Finally, I want to thank my family, especially my parents Mauro and Rosa, who supported me throughout. They are great parents and I hope they can be proud of me.

And last but not least, thank you Cristina, your love and your kindness have helped me in every moment. Without you I would not have accomplished half of the things that I have done.



# Contents

<b>Abstract</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivations . . . . .	1
1.2 Thesis Structure . . . . .	5
1.3 Contributions . . . . .	6
1.4 Publications . . . . .	7
<b>2 State of the Art in Recommender Systems</b>	<b>9</b>
2.1 Ratings and side data . . . . .	9
2.1.1 Social Networks . . . . .	10
2.1.2 User-Contributed Information . . . . .	11
2.2 Algorithms . . . . .	11
2.2.1 Collaborative recommendation . . . . .	12
2.2.2 Content-based recommendation . . . . .	18
2.2.3 Knowledge-based recommendation . . . . .	21
2.2.4 Hybrid recommendation . . . . .	22
2.3 Interfaces . . . . .	23
2.3.1 Explanations . . . . .	24
2.3.2 Conversational RS . . . . .	26
2.3.3 Persuasive aspects of the interaction with RS . . . . .	27
2.4 Evaluation . . . . .	28
2.4.1 Offline evaluation . . . . .	28
2.4.2 Online evaluation . . . . .	32
<b>3 State of the Art in Topic Models</b>	<b>35</b>
3.1 Introduction . . . . .	35
3.2 Methodology . . . . .	37
3.3 Extensions . . . . .	39
3.4 Topic model evaluation . . . . .	40
3.5 Topic model and recommender systems . . . . .	41

---

<b>4</b>	<b>Integrating Concepts and Time in Recommender Systems</b>	<b>43</b>
4.1	Introduction . . . . .	44
4.2	Integrating Concepts in Large Content Networks . . . . .	46
4.2.1	The Object-Object Network and the Topic-Topic Network	46
4.2.2	Experiments . . . . .	48
4.3	Tracking Hot Topics . . . . .	52
4.3.1	Topic Models for Hot Topic Tracking . . . . .	52
4.3.2	Experiments . . . . .	53
4.4	Learning CTBNC Using MapReduce . . . . .	57
4.4.1	CTBN and MapReduce . . . . .	57
4.4.2	Experiments . . . . .	60
4.5	Conclusions . . . . .	62
<b>5</b>	<b>Analyzing User Reviews with Topic Models</b>	<b>63</b>
5.1	Introduction . . . . .	63
5.2	The Topic-Criteria model . . . . .	64
5.2.1	Motivating Example . . . . .	64
5.3	Empirical evaluation . . . . .	69
5.3.1	Scenario 1: Rating Prediction and Recommendation . . .	70
5.3.2	Scenario 2: Analytics and Interpretation . . . . .	71
5.3.3	Scenario 3: Suggest Ratings for Review . . . . .	72
5.4	Conclusions . . . . .	72
<b>6</b>	<b>Explaining Latent Factors with Topic Models</b>	<b>75</b>
6.1	Introduction . . . . .	75
6.2	Research Questions and Design . . . . .	76
6.3	Implementation . . . . .	78
6.3.1	MovieLens Non-negative Matrix Factorization (Step 1) . .	78
6.3.2	Topic Extraction from ML Movies (Step 2) . . . . .	78
6.3.3	Combine latent factors and topics (Step 3) . . . . .	79
6.3.4	Collection of Survey Data (Step 4) . . . . .	80
6.3.5	Evaluation (Step 5) . . . . .	81
6.4	Evaluation . . . . .	82
6.4.1	Evaluation Methods . . . . .	82
6.4.2	Datasets . . . . .	82
6.4.3	Results . . . . .	83
6.4.4	Limitations and Future Work . . . . .	85
6.5	Conclusions . . . . .	86
<b>7</b>	<b>Validity of Accuracy Measurements for Offline Recommendations</b>	<b>87</b>
7.1	Introduction . . . . .	87
7.2	Related Work . . . . .	89
7.3	Study design . . . . .	90
7.3.1	Overview . . . . .	90
7.3.2	MovieLens Dataset . . . . .	91

- 7.3.3 Second phase: online evaluation . . . . . 93
- 7.3.4 Algorithms . . . . . 94
- 7.4 Discussion on measurement methodology . . . . . 97
- 7.5 Results . . . . . 99
- 7.6 Conclusions . . . . . 103
  
- 8 Conclusions . . . . . 105**
- 8.1 Future Works . . . . . 106





# Abstract

Recommender systems are software components that assist users in finding what they are looking for. They have been applied to all kinds of domains, from e-commerce to news, from music to tourism, exploiting all the information available in order to learn user's preferences and to provide useful recommendations.

The broad area of recommender systems has many topics that require a deep understanding and great research efforts. In particular, three main aspects are: *algorithms*, which are the hidden intelligent components that compute recommendations; *interfaces*, which are the way in which recommendations are shown to the user; *evaluation*, which is the methodology to assess the effectiveness of a recommender system.

In this dissertation we focus on these aspects guided by three considerations. First, textual content related to items and ratings can be exploited in order to improve several aspects, such as to compute recommendations, provide explanations, understand user's tastes and item's capabilities. Second, time in recommender systems should be considered as it has a great influence on popularity and tastes. Third, offline evaluation protocols are not fully convincing, as they are based on accuracy statistics that do not always reflect real user's preferences.

Following these motivations six contributions have been delivered, broadly divided in the integration of concepts and time in recommender systems, the application of the topic model to analyze user reviews and to explain latent factors, and the validation of offline recommendation accuracy measurements.



# Chapter 1

## Introduction

In this chapter a general overview of the Thesis is presented. Motivations and goals of this work are discussed, as well as the Thesis structure and list of related publications.

### 1.1 Motivations

Internet diffusion in the past decades has pushed more and more companies towards the Web to attract as many customers as possible, due to the fact that old communication and sales channels are no longer enough. A recent Nielsen report on online shopping has measured that from 2011 to 2014 online purchase intention rates have doubled for more than half of the shopping categories considered (Nielsen, 2014). In this scenario, the amount of data available is growing everyday: beyond classical web sites, traditional newspapers publish their news on the web, every kind of product is available on e-commerce websites, movies and music can be bought and enjoyed online and more and more people are using social networks.

The huge amount of data generates big challenges, as people can have difficulties finding what they really need, a problem usually referred to as *information overload*. While at the beginning the only way to find something useful was through search engines, a growing paradigm is trying to automatically present the user what he/she is looking for or assists him/her to search it. This paradigm is known as a recommender system (Jannach et al., 2010; Ricci et al., 2011). Recommender systems are software components that sug-

gest “*items*” to “*users*” based on their interests. Item is a general term that indicates the entity to recommend, i.e. in an e-commerce web site it indicates a product, while in a news web site it indicates a news article. For instance, if a person browses an e-commerce website and visits the product page of a TV, the system usually shows a section of the web page with product suggestions, such as “*customers who bought this product also bought...*”. After the customer has seen or bought many products, the system learns his/her interests and can automatically compute personalized recommendations.

Recommender systems have been applied with success to many domains. E-commerce was one of the first domains in which recommender systems were applied, due to the natural need to replace the human salesman with an automatic shopping assistant. As a human employee does in a retail shop, a recommender system suggests products on the basis of user interests. For instance, if a user looks for a camera, the system can suggest different cameras or different items related to the camera, such as a memory card or lenses. Recommender systems have been applied to the tourism industry, one of the biggest economic sectors worldwide. User preferences on locations, such as hotels or restaurants, have been combined with contextual information like distance, weather and season, to provide useful recommendations. Other popular domains are the movie and music domains: music portals, such as *Pandora*<sup>1</sup> or *Last.fm*<sup>2</sup>, and movie portals, such as *NetFlix*<sup>3</sup>, incorporate in their websites powerful recommender systems that can suggest music or movies, exploiting the data collected about millions of users. Nonetheless, these companies are active in the research on recommender systems and publish datasets or promote competitions about RSs. Other areas in which recommender systems have been exploited in are the publishing industry and in research. News articles and research articles can be recommended to a reader based on what he/she read in the past or, in the case of research article recommendations, what he/she wrote. In this case the recommender system can exploit the content of the articles to produce better recommendations, looking for recurrent keywords or themes that can describe user interests. Another important domain is the social network area: recommender systems can suggest friends in networks like *Facebook*<sup>4</sup>, or people to follow in *Twitter*<sup>5</sup>. The social network can be additionally exploited to include relationships between users,

---

<sup>1</sup>[www.pandora.com](http://www.pandora.com)

<sup>2</sup>[www.last.fm](http://www.last.fm)

<sup>3</sup>[www.netflix.com](http://www.netflix.com)

<sup>4</sup>[www.facebook.com](http://www.facebook.com)

<sup>5</sup>[www.twitter.com](http://www.twitter.com)

such as trust or distrust. Finally, recommender systems can be applied to technical domains, such as financial services or life insurances, where the knowledge in the domain is essential for providing useful recommendations.

Research on recommender systems has produced a vast literature in many disciplines, such as computer science, information science, psychology and sociology. Jannach et al. (2012) present the analysis of the literature on recommender systems from 2006 to 2011. Papers considered come from journals or conference proceedings from computer science and information science areas. Most of the efforts from the computer science papers is towards algorithms, while the information science area is more focused on the user perspective and the interaction between the system and the user. On the other hand, psychology and sociology efforts are devoted to analyzing the persuasiveness of recommender systems, usually by the identification of the relationship between the system and the user as a social process (Yoo et al., 2012). Since research topics are very broad and cover different parts of a recommender system, Figure 1.1 shows a division into three main aspects:

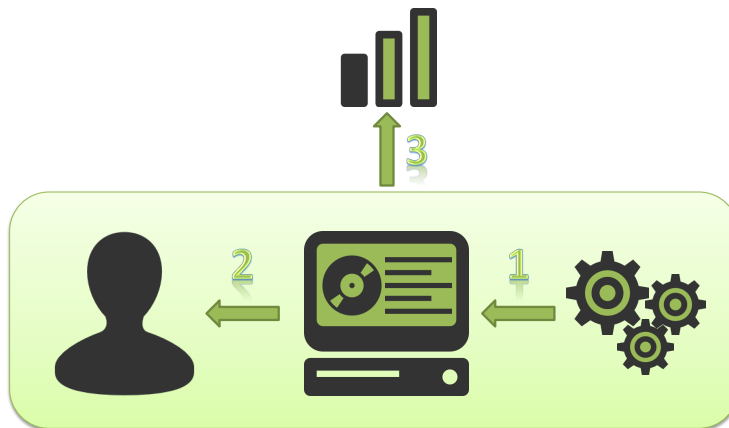


Figure 1.1: Main aspects of a recommender system.

1. **Algorithms:** The first aspect of a recommender system is the algorithm that computes the recommendations. This is the most popular aspect considered in the recommender system area, and most of the research efforts are devoted to this. The algorithm usually takes the data of users and items as input and produces recommendations as output. Given the kind

of data processed, the algorithms are divided into collaborative, content-based or knowledge-based algorithms.

2. **Interfaces:** After recommendations are computed, they must be presented to the user, usually on a computer or on a mobile device. The formulation of good explanations and human-machine dialogs are critical points that can drastically increase user satisfaction. Martin (2009) claims that while most of the research efforts are on the algorithmic aspect, this aspect has only 5% relevance in industrial applications, while the interface is the most effective component with 50% relevance. In the last few years the focus on interfaces has been growing with specific workshops on the topic, e.g. the *Interfaces and Human Decision Making for Recommender Systems (IntRS 2014)* at *RecSys 2014*.
3. **Evaluation:** Finally, the recommender system has to be evaluated to analyze and understand if recommendations are effective and which algorithm or interface is better. For this purpose several evaluation methods have been developed, mainly divided into offline and online approaches. Offline approaches rely on fixed dataset and apply methodologies typical of the data mining area. Online evaluations are based on user studies and involve real users who have to interact with the system.

The goal of the thesis is to cover as much as possible the whole recommender system area, bringing contributions to all three aspects. Every aspect presents many opportunities to innovate and extend recommender systems. In particular, research efforts in this thesis are guided by the following motivations:

- Extra-knowledge can be generated from textual data associated with items and exploited in recommender systems. In particular, item descriptions and reviews can be analyzed in order to find themes and concepts underlying the natural language text. This knowledge can be exploited in many ways, such as profiling user interests, defining strong and weak points of an item, discovering semantic connections between items, identifying popular topics and supporting user decisions.
- Time in recommender systems is an important factor and must be considered. User preferences may change over time, and topic popularity can fluctuate very quickly. Furthermore, in some cases even the duration of events has to be considered.

- Recommender systems evaluation protocols are the subject of intense discussions and controversies: offline evaluation is easy to assess and to reproduce, while it can be misleading. On the other hand, online evaluation deals with real users, but it is hard to perform.

## 1.2 Thesis Structure

The dissertation is structured in three parts. The first part, composed of Chapters 2 and 3, is a survey on the state of the art in recommender systems and topic models. The second part, composed of Chapters 4, 5, 6 and 7, presents the contributions of this work. The last part, Chapter 8, is the conclusion of the thesis. In particular, the dissertation is divided into the following chapters:

- Chapter 2 presents a brief state of the art in the recommender system area. First of all, preliminary notions about the kind of ratings and side data are given; after that, the three aspects of a recommender system, i.e. algorithms, interfaces and evaluation, are presented from the fundamentals to the latest developments.
- Chapter 3 introduces the topic model, a methodology that deals with natural language text. Since the topic model is exploited throughout the work, an introduction and presentation of the mathematical concepts behind it are given. Furthermore, topic model extensions are discussed, as well as topic model evaluation and application to the recommender system area.
- Chapter 4 discusses methodologies that can be exploited in the recommender system area. The first contribution is about large content networks: the topic model was applied to derive concepts and to discover semantic links between items, offering some insights on how recommendations can be provided. The second contribution is about the analysis of the temporal dimension with the topic model, specifically regarding labor-laws. This dimension is an important aspect of recommender systems, but in the literature is often overlooked. The third contribution is the definition of the MapReduce version of two Continuous Time Bayesian Network classifiers (CTBNC) training algorithms, a class of models that can deal with processes that occur in a continuous time frame. This models can

find application in specific recommender systems where the duration of events is a critical aspect.

- Chapter 5 introduces a new model to provide recommendations based on ratings and textual reviews. Reviews were analyzed in order to find topics discussed, and user and item profiles were computed as a degree of association between them and the topics discovered. On the basis of this analysis, items can be suggested to users combining their profiles. This work presents a contribution to the algorithm aspect of a recommender system, but can also provide useful knowledge on topics to exploit in the interface aspect: topics can be used as a representation of the strong and weak points of an item.
- Chapter 6 presents a contribution to the interface aspect. While matrix factorization techniques have been proved to be one of the best methods to compute recommendations, they do not provide a simple way to explain the recommendations provided. In this chapter textual content associated with items is exploited to provide an interpretation of latent factors. A user study was performed to evaluate how well the proposed method is able to predict the topics that users will select to explain their preferences.
- Chapter 7 questions the validity of the current evolution protocol of recommender systems. An online evaluation and an offline evaluation were conducted on the same dataset, and different metrics have been compared. The online evaluation was performed with a two round user study with more than two hundreds participants and one hundred reliable users. The study showed how the offline evaluation is not always able to emulate the online evaluation, since different metrics are significantly different in the two evaluation protocols.
- Finally, Chapter 8 concludes the thesis, discussing the contributions provided and exploring the possibility of future developments.

### 1.3 Contributions

This thesis brings the following contributions:

1. Definition of methods able to derive concepts and to discover semantic links in large content networks (Chapter 4, Rossetti et al. (2014)).



2. Analysis of hot topics in document corpora (Chapter 4, Pareschi et al. (2014)).
3. Definition of the MapReduce learning algorithms of two continuous time Bayesian network classifiers (CTBNC) (Chapter 4, Villa and Rossetti (2014)).
4. Definition of a model that combines ratings and reviews to provide recommendations and explain multi-criteria ratings (Chapter 5, Rossetti et al. (2015)).
5. Definition of a model that provides explanations for latent factor models (Chapter 6, Rossetti et al. (2013)).
6. Comparison of the offline and online evaluation protocol on a common dataset (Chapter 7, Rossetti and Zanker (2015)).

## 1.4 Publications

The research efforts presented in this dissertation are the summary of five international publications plus a working paper. The topics covered are discussed in more detail in the following papers:

### International Journals

- Rossetti, M., Arcelli, F., Pareschi, R., and Stella, F. (2014). Integrating concepts and knowledge in large content networks. *New generation computing*.
- Villa, S. and Rossetti, M. (to appear in 2014). Learning continuous time Bayesian network classifiers using MapReduce. *Journal of Statistical Software*.

### International Conferences

- Pareschi, F., Rossetti, M., and Stella, F. (to appear in 2014). Tracking hot topics for the monitoring of open-world processes. In *SIMPA*. Citeseer.
- Rossetti, M., Stella, F., Cao, L., and Zanker, M. (to appear in 2015). Analyzing user reviews in tourism with topic models. In *Information and Communication Technologies in Tourism 2015*. Springer.

### International Workshops

- Rossetti, M., Stella, F., and Zanker, M. (2013). Towards explaining latent factors with topic models in collaborative recommender systems. In *Database and Expert Systems Applications (DEXA), 2013 24th International Workshop on*, pages 162-167. IEEE.

#### **Working Papers**

- Rossetti, M. and Zanker, M. Assessing the validity of offline recommendation accuracy measurement.

## Chapter 2

# State of the Art in Recommender Systems

In this chapter the recommender systems area is presented. First, a discussion on the kind of ratings and side data usually available is presented to distinguish between different kinds of problems. After this, following the broad division given in the introduction, the three main components of a recommender system are presented. For each component the fundamental ideas and works are presented, together with the latest developments.

In this chapter we adopt the following notation:  $M$  is the number of users, while  $N$  is the number of items. Indices  $i$  and  $m$  are reserved for users, while  $j$  and  $n$  are reserved for items.  $R$  is the user-item matrix with  $M$  rows and  $N$  columns, and the specific rating given by user  $i$  to item  $j$  is indicated with  $R_{ij}$ .

### 2.1 Ratings and side data

The classical paradigm of a recommender system concerns three main aspects: users, items and relations between users and items. The most popular form of relation is the rating, which represents the feedback that a user gives on a specific item. Ratings can be explicit, i.e. when a user rates an item for the purpose of sharing his/her experience with it, or implicit, i.e. when a user buys or consumes an item and the system logs that piece of information. Ratings can take many forms (Schafer et al., 2007):

- numerical: ratings on a multi-point rating scale, usually from 1 to 5,

provided explicitly by the user;

- ordinal: ratings on a qualitative but ordinal rating scale, such as “strongly agree, agree, neutral, disagree, strongly disagree”, usually collected with questionnaires;
- binary: ratings such as good/bad, which can be *explicit* or *implicit* if the user behavior is analyzed (if the user listens to a song to the end it could be a “like”, if he/she skips the track immediately it could be a “don’t like”);
- unary: ratings only associated with positive items; also in this case the rating can be explicit, i.e. a “like” statement, or implicit, i.e. the user bought an item.

Since the most popular case is the numerical explicit rating from 1 to 5, in the following sections this kind of rating will be the adopted, unless explicitly expressed.

Apart from ratings, recommender systems can exploit side information to improve recommendation quality. The most important kinds of side information considered are *social networks* and *user-contributed information*.

### 2.1.1 Social Networks

Social networks in recommender systems are a useful source of user-user relationship that can be exploited to improve the quality of recommendations and their persuasive aspects. In particular, directed social networks have been studied as an evidence of trust and distrust relationships, as in *Epinions*<sup>1</sup> (Guha et al., 2004; Leskovec et al., 2010; Ma et al., 2009, 2008; Massa and Avesani, 2007) and *Twitter*<sup>2</sup> (Kwak et al., 2010). In the first case an explicit trust or distrust statement is specified, while in the second the *follow* relationship can be interpreted as a trust relationship. On the other hand, social networks can be undirected, as in *Facebook*<sup>3</sup>, where the friendship between users is considered as a positive relationship between them (Konstas et al., 2009). The fundamental assumption behind these relationships is that users who are in a positive relationship may share common interests.

---

<sup>1</sup>[www.epinions.com](http://www.epinions.com)

<sup>2</sup>[www.twitter.com](http://www.twitter.com)

<sup>3</sup>[www.facebook.com](http://www.facebook.com)

### 2.1.2 User-Contributed Information

With the introduction of Web 2.0 technologies, users have been able to share, comment and tag almost everything on the Internet. Ratings are just one type of user-contributed information, but several other sources exist, such as tags, geotags, and reviews and comments.

Tags are short textual labels that users assign to items with the purpose of describing them. They differentiate themselves from labels because they are freely created and assigned by users. Tags have been successfully exploited to improve recommendation accuracy (Sen et al., 2006; Tso-Sutter et al., 2008; Zhen et al., 2009).

Geotags are a special kind of tags related to the location of an item. This piece of information has started to be consistently present since the diffusion of mobile devices with GPS capabilities. Geotags are used especially in *context-aware RS* (Adomavicius and Tuzhilin, 2011), where the geographical position is considered as part of the context.

Reviews and comments are textual pieces of information in natural language that users write about items. This kind of side information is not only useful for improving recommendation accuracy, but even for understanding users' tastes and providing explanations (Levi et al., 2012; Moshfeghi et al., 2011; Jakob et al., 2009; Aciar et al., 2007).

## 2.2 Algorithms

Recommender system algorithms are usually classified into three categories based on the kind of data they use to compute recommendations. The most popular approach is *collaborative filtering (CF)* because it takes into account only the user-item matrix without any metadata on users and items. This makes the CF approach very flexible and suitable for almost every domain. The other categories consider features about items and users. In these cases the usual classification divides the approaches into *content-based*, when the features about an item are textual data, and *knowledge-based*, when the features carry some extra information such as in complex item domains. The boundaries between these two categories are not always well defined, and sometimes they are simply referred to as content-based. When different sources of data are taken into account and combined together, such as the user-item matrix and item features, approaches are called *hybrid*.

It is important to mention that the recommender system problem can be seen as a *prediction* problem, a *ranking* problem or a *classification* problem. When the task is prediction, the purpose of the system is the estimation of the exact rating. This is the most popular approach and most of the literature is focused on this task. While predicting a rating is important in order to create a system that can reproduce the human rating behavior, what the user sees is usually a recommendation list. For this reason, especially in the last few years, research efforts are moving towards the “*learning to rank (LETOR)*” paradigm (Fuhr, 1992), where the purpose of the system is to produce a personalized ranking of the items. It is important to mention that the prediction approach can be seen as a *point-wise LETOR* approach, where the estimated rating is used to rank items. Finally, if the item order is not important, the recommendation problem can be seen as a personalized classification problem, where items are classified as positive or negative. Even in this case it is important to note that the previously described approaches can be seen as classification approaches, taking a cutoff of the recommendation list. In the next section most of the models described are based on the prediction approach.

### 2.2.1 Collaborative recommendation

*Collaborative filtering (CF)* (Goldberg et al., 1992) is the most popular approach in the area of recommender systems. The main idea is to analyze the past user behavior to predict which items the user will like. In a pure CF scenario, the only information considered is the user-item matrix, which contains the ratings that the users gave to the considered items. Based on this data, the system computes recommendations for every user.

CF gained huge popularity in the last few years thanks to the Netflix Prize (Bennett and Lanning, 2007), a one million dollar prize for whoever would have been able to increase the accuracy on a given dataset. The prize was won in 2009 (Koren, 2009; Töscher et al., 2009; Pirotte and Chabbert, 2009), but the competition gave a huge push to the research on collaborative filtering. Many approaches have been developed in the last few years, which can be classified in different ways. One of the most popular classifications divides them as memory-based or model-based.

### Memory-based approaches

Memory-based approaches work directly on the user-item matrix. While they are easy to implement and to understand, they do not scale very well when the number of users and items grows.

The user-based nearest neighbor approach, which derives from the nearest neighbor classifier (Cover and Hart, 1967), is one of the earliest methods for recommending items (Herlocker et al., 1999). The main idea is based on similarities between users: if users had similar past rating behavior, they will keep a similar rating behavior in the future. To apply this approach to a given user, the first step is to find the set of users similar to him/her given the past rating behavior. After that, for every item that the user has not seen, a rating is estimated based on the ratings that the set of similar users gave to those items.

Table 2.1: Ratings on 5 books for 5 users.

	LOTR	TWL	HP	THG	ASOIAF
Alice	5	3	4	4	?
Bob	3	1	2	3	3
Carl	4	3	4	3	5
Daniel	3	3	1	5	4
Emily	1	5	5	2	1

Table 2.1 shows a user-item matrix with ratings given by 5 users on 5 books. The books considered are: The Lord of the Rings (LOTR), Twilight (TWL), Harry Potter (HP), The Hunger Games (THG) and A Song of Ice and Fire (ASOIAF). The objective of this example is the estimation of the rating that Alice will give to ASOIAF. The first step consists in the estimation of similarities between Alice and the other users. In the user-based nearest neighbor model the typical approach consists in the computation of Pearson's correlation coefficient, as defined in Formula 2.1, where  $P$  is the set of items rated by both  $i_1$  and  $i_2$  and  $\bar{r}_{i_1}$  is the average of the ratings given by user  $i_1$ .

$$\text{sim}(i_1, i_2) = \frac{\sum_{j \in P} (r_{i_1 j} - \bar{r}_{i_1})(r_{i_2 j} - \bar{r}_{i_2})}{\sqrt{\sum_{j \in P} (r_{i_1 j} - \bar{r}_{i_1})^2} \sqrt{\sum_{j \in P} (r_{i_2 j} - \bar{r}_{i_2})^2}}. \quad (2.1)$$

Correlation values between Alice and the other users are shown in Table 2.2. Alice seems to be similar to Bob and Carl, while she has opposite tastes with respect to Emily. Equation 2.2 defines how to estimate the rating for a user-item

Table 2.2: Pearson correlation values between Alice and the other users.

	Alice
Bob	0.8528
Carl	0.7071
Daniel	0.0000
Emily	-0.7921

pair, where  $\mathcal{N}_i$  is the set of user  $i$ 's neighbors.

$$pred(i, j) = \bar{r}_i + \frac{\sum_{m \in \mathcal{N}_i} sim(i, m) * (r_{mj} - \bar{r}_m)}{\sum_{m \in \mathcal{N}_i} sim(i, m)}. \quad (2.2)$$

The number of neighbors to consider is set by a parameter. If  $|\mathcal{N}_i| = 2$ , the rating for the couple Alice-ASOIAF is computed considering only Bob and Carl as neighbors, as shown in Equation 2.3

$$pred(Alice, ASOIAF) = 4 + \frac{0.8528 * 0.6 + 0.7071 * 1.2}{0.8528 + 0.7071} = 4.872. \quad (2.3)$$

Another kind of approach that belongs to this group is the item-based nearest neighbor approach (Sarwar et al., 2001). In this case the idea of computing similarities is transferred from users to items. The first step consists in the computation of similarities between items based on the users who rated them: two items are similar if the users have a similar rating behavior on them. After this the prediction for an unseen item is computed using the similarity values between the unseen item and the items the user rated in the past.

In the item-based approach the cosine similarity replaces the Pearson correlation because it has been shown to produce the most accurate recommendations. Since the cosine similarity does not take into account the average rating of an item, the adjusted cosine similarity between two items is defined as in Equation 2.4, where  $U$  is the set of users who rated both items.

$$sim(j_1, j_2) = \frac{\sum_{i \in U} (r_{ij_1} - \bar{r}_{j_1})(r_{ij_2} - \bar{r}_{j_2})}{\sqrt{\sum_{i \in U} (r_{ij_1} - \bar{r}_{j_1})^2} \sqrt{\sum_{i \in U} (r_{ij_2} - \bar{r}_{j_2})^2}}. \quad (2.4)$$

Table 2.3 shows the adjusted cosine similarity values between ASOIAF and the other items. ASOIAF seems to be very similar to LOTR and THG. Equation 2.5 defines how to predict a rating for a couple user-item, where  $I$  is the set of



Table 2.3: Adjusted cosine similarity values computed between ASOIAF and the other items.

	LOTR	TWL	HP	THG
ASOIAF	0.9695	-0.4781	-0.4276	0.5817

items rated by user  $i$ .

$$pred(i, j) = \frac{\sum_{n \in I} sim(j, n) * r_{in}}{\sum_{n \in I} sim(j, n)}. \quad (2.5)$$

Equation 2.6 shows the prediction of the rating for the couple Alice-ASOIAF when the neighborhood of an item, in this case the item ASOIAF, is considered with size equal to 2.

$$pred(Alice, ASOIAF) = \frac{0.9695 * 5 + 0.5817 * 4}{0.9695 + 0.5817} = 4.625. \quad (2.6)$$

The main advantage of the item-based CF is from the computational point of view. Although user-based CF has been applied in different domains, it does not scale well when the number of users and items grows. This problem is due to the fact that the user-based CF has to compute user similarities online using the full user-item matrix: if the matrix has several millions of rows and columns, as in popular e-commerce websites, the computation of similarities cannot be performed online. On the other hand, the item-based CF offers the possibility to compute the item similarities offline and to exploit them online to compute recommendations. This possibility is given by the fact that item similarities are more stable and are not influenced by the rating choices of a single user, therefore they can be computed offline, while user similarities can significantly change based on the user behavior during an online session.

It is important to note that in the literature several extensions and modifications have been proposed, such as the introduction of different similarity measures, the analysis of significance and variance, and neighborhood selection techniques. More details can be found in Ricci et al. (2011).

### Model-based approaches

Model-based approaches create an offline model based on training data and apply it online to compute recommendations, which usually leads to greater accuracy and stability in recommendations (Koren, 2008). Although a great variety

of models belong to this group, such as association rule mining, probabilistic recommendation approaches and graph-based approaches, the most popular are the matrix factorization approaches, also called latent factor models.

Matrix factorization models analyze the user-item matrix in order to find latent factors, which can be described as latent features that characterize the user-item relationships (Koren, 2008; Bell et al., 2007; Mnih and Salakhutdinov, 2007). For instance, in the movie recommendation domain a latent factor could map the movie genre, while another can map the age of the users that like a specific group of movies. Since latent factors are computed in a non-supervised way, their interpretation is not always trivial, and in some cases it is not possible. The computation of latent factors is performed through matrix factorization techniques: the user-item matrix is decomposed into two smaller matrices, the product of which is an approximation of the original matrix. Items are recommended to users if they are close in the latent factor space.

Since the user-item matrix is usually sparse, matrix factorization usually consists in the minimization of a loss function, such as the one defined in Equation 2.7, where  $U_i$  and  $V_j$  are latent factor vectors for user  $i$  and item  $j$ , while  $\lambda_U$  and  $\lambda_V$  are regularization parameters to avoid data overfitting.

$$\min_{U,V} \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^N I_{ij} (R_{ij} - U_i^\top V_j)^2 + \frac{\lambda_U}{2} \|U\|_F^2 + \frac{\lambda_V}{2} \|V\|_F^2. \quad (2.7)$$

$I_{ij}$  is an indicator function that is equal to 1 if user  $i$  rated item  $j$ , and 0 otherwise. In this way latent factors are computed only exploiting observed ratings.

To illustrate how matrix factorization works, the *probabilistic matrix factorization (PMF)* (Mnih and Salakhutdinov, 2007) was applied to the data presented in Table 2.1. PMF is a special kind of MF with a probabilistic perspective. The PMF framework models the conditional probability of latent factors given the observed ratings.

The result of the PMF is shown in Table 2.4. Users and items are mapped in the latent factor space. The affinity between a user and an item can be evaluated by the proximity of the mapping on the latent space. For instance, Daniel and Bob are very close to THG because their interests match the latent features of the item, while Alice and Carl are close to LOTR and ASOIAF for the same reason. To better illustrate the mapping of users and items in the latent factor space, Figure 2.1 shows the projection of them in a two dimensional

Table 2.4: User and item mapping on the two latent factors.

<b>U</b>	Alice	Bob	Carl	Daniel	Emily
Factor 1	-0.03	0.95	-0.22	1.20	-1.40
Factor 2	-1.52	0.37	-0.96	0.07	1.10

<b>V</b>	LOTR	TWL	HP	THG	ASOIAF
Factor 1	0.31	-1.07	-1.57	0.77	0.45
Factor 2	-1.25	0.16	-0.42	-0.35	-1.66

space, where the X-axis represents factor 1 and Y-axis represents factor 2. It is important to note that the latent space does not have a given interpretation of the dimensions' meaning; factor 1 and factor 2 represent concepts hidden in the user-item matrix, thus their interpretation is often not obvious. The rating

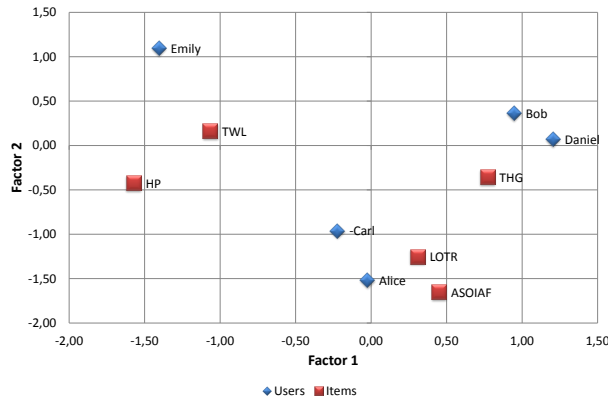


Figure 2.1: Latent factor space and users and items collocation.

for user  $i$  and item  $j$  can be estimated as shown in Equation 2.8, where  $\bar{r}$  is the average dataset rating and  $U_i$  and  $V_j$  are the columns of matrices  $U$  and  $V$  corresponding to user  $i$  and item  $j$ .

$$\text{pred}(i, j) = \bar{r} + U_i^\top V_j. \quad (2.8)$$

Following the previous examples, the rating for the couple Alice-ASOIAF has been estimated in Equation 2.9

$$\text{pred}(\text{Alice}, \text{ASOIAF}) = 3.21 + (-0.03 * 0.45) + (-1.52 * -1.66) = 5.71. \quad (2.9)$$

The idea of extracting latent factors comes from the information retrieval

area, where singular value decomposition (SVD) was used to discover latent factors in documents (Deerwester et al., 1990). SVD and principal component analysis (PCA) was soon applied to the RS area (Sarwar et al., 2000; Goldberg et al., 2001). Since then, with the contribution of the Netflix Prize, several different approaches have been developed including additional information such as demographic data, temporal information and users and items biases (Koren et al., 2009), social information (Ma et al., 2011) and context-aware data (Baltrunas et al., 2011).

### **Limitations**

Collaborative filtering approaches suffer from two main issues. First, the user-item matrix is usually very sparse: in many cases the number of co-rated items between users is low, as users rate only a few items, while collaborative filtering methods need many ratings to perform better. A typical solution to this issue, which moves towards hybrid recommenders, is to take into account user and item meta-data, such as gender, age or item categories (Pazzani, 1999). A second typical issue is the cold start problem: when recommender systems are at the first stages of their development, there are not enough ratings to compute personalized recommendations. Furthermore, when a new user or a new item is introduced, the system is not able to provide recommendations for them. A simple solution is to adopt non-personalized recommender systems as a backup recommendation method. Another strategy is to force the user to rate a minimum number of items before starting to use the system (Rashid et al., 2002).

### **2.2.2 Content-based recommendation**

*Content-based filtering* exploits item content to provide recommendations based on what the users liked in the past. The difference with respect to CF is that in this case the system is not taking into account relationships between users or between items, but it exploits information about items that a user rated in the past. For instance, if a user liked fantasy books, the system will suggest other books of the same kind. The recommendation process consists in the definition of a user profile, based on past user preferences and in the comparison between the user profile and the item profile. While items often have technical descriptions of their features, the challenging task is to extract significant features that can guide the recommendation process. Following the previous example, Table

2.5 shows an example of item features. Keywords have been selected as the most representative for the items considered.

Table 2.5: Item features that can be exploited for the content-based approach.

Title	Genre	Author	Nationality	Keywords
LOTR	Fantasy	Tolkien	British	fantasy hobbit ring wizard elves
TWL	Fantasy	Meyer	American	vampire fantasy love teenage werewolf
HP	Fantasy	Rowling	British	fantasy students wizard magic friends
THG	Science	Collins	American	science post-apocalyptic districts battle death
ASOIAF	Fantasy	Martin	American	fantasy throne dragons war power

The most common technique to define a user profile from the descriptions of the items he/she liked is to build a list of words that describes such items. In this case the main issue is the definition of the criteria to apply for selecting the most important words. A first approach consists in the selection of the most frequent words that appear in the item descriptions. However, these words can be frequent not only in the items liked by the user, but in every item. For instance, in the movie domain, every item description will contain terms like “movie”, “director” and “actor”. A popular approach to overcome this limitation is offered by the *TFIDF* encoding format (Salton et al., 1975). This method takes into account the *term frequency* (TF), but it also considers the *inverse document frequency* (IDF), which means that the importance of a word is inversely proportional to its frequency in all the item descriptions. TF is defined as in Equation 2.10 (Chakrabarti, 2003): the frequency of the word  $i$  in document  $j$  is divided by the maximum TF value computed in the document  $j$ :

$$TF(i, j) = \frac{freq(i, j)}{maxFreq(j)}. \quad (2.10)$$

The second component decreases the weight of words that appear in most of the documents. IDF for the word  $i$  is defined as in Equation 2.11, where  $n(i)$  is the number of documents that contain item  $i$ , and  $N$  is the total number of documents:

$$IDF(i) = \log \frac{N}{n(i)}. \quad (2.11)$$

The product of the TF for the word  $i$  in document  $j$  and the IDF of word  $i$  is

the TFIDF measure (Equation 2.12):

$$TFIDF(i, j) = TF(i, j) * IDF(i). \quad (2.12)$$

The representation of items as vectors of TFIDF values is called *Vector Space Model (VSM)*. To increase the performances of the vector space model, pre-processing techniques can be applied, such as stop words removal, stemming, size cutoffs and the use of *n-grams* instead of simple terms (Baeza-Yates and Ribeiro-Neto, 1999).

As long as items are represented in the VSM, recommendations can be computed in different ways. A k-nearest neighbor (kNN) method similar to the item-based CF consists in the computation of item-item cosine similarities and in the recommendation of items similar to the items the users liked in the past (Allan et al., 1998). Another method consists in the acquisition of a user profile based on words with an high TFIDF value in the items that the user liked, and in the training of a classifier that filters the items of interest (Billsus and Pazzani, 1999). In some cases, especially in the news domain, it is important to adopt two approaches for short-term and long-term interests. In Billsus et al. (2000) short-term interests are analyzed with the kNN approach, while a classifier is trained to monitor long-term interests. A different approach consists in the exploitation of the user feedback to learn a better profile. The Rocchio relevance feedback method (Salton, 1971) explicitly asked the user to rate the items proposed, and based on the relevance feedback the user profile was adapted and the results were refined.

Recent approaches that deal with textual information exploiting topic models are described in Section 3.5.

### Limitations

The main issue with content-based recommendations is based on the similarity between items. In some domains, recommending items similar to the items already consumed is not a good strategy. For instance, a news recommender system that recommends articles on the same story already read by the user is not a good recommender. This issue, called *overspecialization*, can be solved increasing the *serendipity* of the system, which is the ability to recommend unexpected items. A simple strategy consists in filtering out not only dissimilar ones, but even too similar items (Billsus and Pazzani, 1999), while another approach is based on the inclusion of random items (Shardanand and Maes,

1995).

### 2.2.3 Knowledge-based recommendation

*Knowledge-based recommender systems* have been developed to handle specific recommendation tasks for which collaborative filtering and content-based filtering are not suitable. For instance, in domains like housing or automotive ones, users do not usually have many ratings, and the two previously described approaches are not applicable. Another problem is related to the temporal dimension: for instance, in the electronic domain, five year old ratings on electronic devices are not likely to be significant today. Furthermore, in CF and content-based filtering users cannot explicitly specify their needs to the system, but can only receive recommendations based on their past behavior. Knowledge-based systems can handle these problems because they guide the user interactively to items that satisfy their needs (Burke, 2000). In this sense, knowledge-based recommender systems are very different from the other two approaches because they do not filter items based on past user interests, but assist the user in the navigation of the item catalog.

The main idea of knowledge-based recommender systems is to exploit the detailed knowledge on items to provide an effective way to navigate through them. Users can express their needs as constraints on different features on the items desired, i.e. “*the car has to run 20km with 1 liter of gasoline*”. The requirements expressed are used to select items in two different ways: constraint-based approaches (Felfernig and Burke, 2008; Felfernig et al., 2006; Zanker et al., 2010) and case-based approaches (Bridge et al., 2005; Burke, 2000).

Knowledge-based recommender systems provide a three step process: in the first step the user expresses his/her preferences and needs, usually with a web-based form; the system scans the item catalog looking for products that satisfy the requirements and presents them to the user; the user can refine the search in case he/she is not satisfied, and the system can optionally offer some advice on requirements to be changed.

#### Constraint-based recommender systems

Constraint-based systems try to solve a *constraint satisfaction problem (CSP)* (Tsang, 1993), which can be described by a tuple  $(V, D, C)$  where  $V$  is a set of variables,  $D$  is a set of finite domains for the variables and  $C$  is a set of constraints that describes the combination of values the variables can simulta-

neously take. To solve a CSP, a value to each variable in  $V$  has to be assigned in a way that all constraints are satisfied. In constraint-based recommender systems  $V$  is composed of customer properties ( $V_C$ ) and product properties ( $V_{PROD}$ ), while  $C$  is composed of compatibility constraints ( $C_R$ ), filter conditions ( $C_F$ ) and product constraints ( $C_{PROD}$ ). Compatibility constraints ( $C_R$ ) define a way to translate customer properties to product properties (i.e. cheap camera means that it costs under \$ 100), filter conditions ( $C_F$ ) specify which properties customers want (i.e. I want a cheap camera) and product constraints ( $C_{PROD}$ ) list which properties products have (i.e. camera.1 costs \$99). Each solution to the CSP is an actual recommendation.

### Case-based recommender systems

On the other hand, case-based recommendation approaches are based on the computation of a similarity measure between the desired features and the products, which usually is defined as in Equation 2.13 (McSherry, 2003).  $p$  is the product considered,  $REQ$  is the set of requirements and  $sim(p, r)$  is the similarity between the product  $p$  and the requirement  $r$ .  $w_r$  is a weight that defines the importance of a specific requirement.

$$similarity(p, REQ) = \frac{\sum_{r \in REQ} w_r * sim(p, r)}{\sum_{r \in REQ} w_r}. \quad (2.13)$$

The similarities between a requirement and a product can be defined in several ways, based on the kind of property that is considered. For some properties, such as price, *less is better (LIB)*, while for others, such as the resolution of the digital camera, *more is better (MIB)*. A third case concerns the situation when only the distance from the requirement is important in both direction, with no preference for more or less.

### 2.2.4 Hybrid recommendation

In the past sections the three main approaches to recommendations have been presented. Since every approach has its own strength and weakness, especially depending on the domain considered, a combination of different approaches can exploit the advantages that every method offers. These recommenders are called *hybrid recommenders* (Burke, 2002).

Hybrid approaches can be implemented in several ways. Many methods have been developed taking into account different kinds of features merged in



a single recommender, like models that exploit collaborative filtering features, user features and item features merged together. Another approach consists in the application of different kinds of recommender systems one after another: a general recommender filters the set of items available, while another more specialized processes the filtered items.

The most popular kind of hybridization design consists in the application of two or more recommender systems and in the combination of recommendations as a post-processing step. The aggregation can be a linear combination, as in Claypool et al. (1999): a collaborative filtering method and a content-based method are combined with equal weights, which are then adjusted based on recommendations accuracy. Another aggregation method is based on majoring voting: items that receive more votes from different algorithms are pushed towards the top of the recommendation list (Pazzani, 1999). The assumption behind these kinds of combination schema is that different methods perform in a uniform way over all the items and the users. This assumption is not always true, as the number of ratings per item/user or the quality of features can influence the accuracy of the recommenders.

## 2.3 Interfaces

Interfaces are the way to deliver recommendations to the user. While most efforts in research are dedicated to recommender system algorithms, interfaces have a bigger impact on the user. Francisco Martin from the company *Strands*<sup>4</sup>, during his industry keynote delivered at the ACM RecSys conference in New York in 2009 (Martin, 2009), made the provocative statement that 50% of the success of a recommender system depends on interfaces, while algorithms account only for 5%. In this section three key aspects of interfaces are presented: explanations, i.e. pieces of information used to explain why an item is recommended, conversational RS, i.e. systems designed to provide recommendations in a guided and conversational interaction, and persuasive aspects, i.e. which are psychological and sociological aspects that influence how recommendations are perceived.

---

<sup>4</sup>[www.strands.com](http://www.strands.com)

### 2.3.1 Explanations

Artificial intelligence systems try to mimic human behavior and to perform complex tasks that are the result of hidden reasoning and computations. In the recommender systems area, the system suggests items to the user, as a salesman can do in a retail shop. When this happens in the real world, the salesman usually motivates the suggestion with *explanations* about why the item has been suggested (Brewer et al., 1998). In recommender systems explanations are pieces of information associated with recommendations with different purposes, such as transparency, validity, trustworthiness, persuasiveness, effectiveness, efficiency, satisfaction, relevance, comprehensibility and education (Tintarev, 2007; Tintarev and Masthoff, 2007).

Explanation approaches are tied to the kind of algorithm that computed recommendations. Therefore, different recommender systems follow different explanation strategies.

#### Explanations in knowledge-based recommenders

Constraint-based recommenders, as discussed in Section 2.2.3, are recommender systems that ask users to specify a set of requirements in order to find the desired products. In this case the explanations can belong to two classes: if the system asks to specify a requirement, the user can be interested in understanding why the system needs that piece of information; in another case, if the system presents a product, the user could want to understand how the proposed product was chosen, which features make that specific product more desirable than others. In the first case the explanation is called a *why-explanation*, while in the second is a *how-explanation* (Buchanan and Shortliffe, 1984).

Explanations for constraint-based systems are generated through *abduction* (Console et al., 1991): given a constraint satisfaction problem (CSP), a set of constraints is an explanation if the solution to the CSP is a (logical) consequence of it. For instance, if a smartphone recommender system asks the user whether he/she likes HD videos, the explanation for this question can be that if he/she likes HD videos, the system has to select smartphones with a good camera; in this case the constraint “*if you want HD videos, you need a good camera*” is a logical consequence of the question. If constraints and variables in the CSP are associated with descriptions, natural language explanations can be automatically derived (Buchanan and Shortliffe, 1984).

Case-based recommenders (see Section 2.2.3) recommend products that best

fit user requirements. The difference between this kind of recommender and constraint-based recommenders is that in this case products are selected even if they do not fulfill all the requirements, but they are the best products similar to what the user wishes.

A typical explanation is the why-explanation: if the system proposes a product, the user wants to understand why the specific product has been proposed to him/her. The usual approach is to highlight the user requirements and product features with respect to the specified requirements. In this way the user can understand which requirements are fulfilled and which are not (McSherry, 2003).

Recently, the *Interfaces and Human Decision Making for Recommender Systems (IntRS 2014)* workshop at *RecSys 2014* tackled the interface and explanation problem in recommender systems. Zanker and Schoberegger (2014) formulated explanations as arguments that provide structured reasoning, joining a conclusion that follows from one or more premises with the conjunction “therefore”. The study showed that a structured explanation has a stronger impact than just presenting the facts without structure. On the other hand, synthetic explanations with facts connected by “therefore” were preferred to natural language explanations. Lanche et al. (2014) provides interactive explanations in a mobile shopping recommender system. While typically explanations are used to transmit pieces of information from the system to the user, the proposed approach generates personalized interactive explanations using user preferences and the mobile context, providing a conversational channel to the user.

### **Explanations in collaborative filtering recommenders**

The main difference between explanations in collaborative filtering and knowledge-based approaches is that in CF there is no knowledge about the items recommended that can be exploited to produce explanations. In this scenario there are three main steps that are considered to compute recommendations and that can be exploited to provide explanations. The first step is represented by the fact that users rate items: ratings are a good source of information for explanations, while other sources of information are not considered. For example, a good explanation can be based on the fact that the user rated a similar book, while the fact that he/she bought a similar book can be misleading if the user bought the book as a present. The second step is about the identification of neighbors: if a user knows that users similar to him/her have appreciated an

item, he/she can be more inclined to accept the recommendation. The third step is about the rating predictions: if the user knows that the recommendation is computed based on a solid number of ratings, confidence in the system can increase.

One of the best ways to explain recommendations has been shown to be based on rating behaviors of neighbors. In Herlocker et al. (2000) users were asked to evaluate different kinds of explanations to understand which was the best. The most appreciated explanation was based on an histogram that divided neighbors into three groups: neighbors who rated the item with high ratings, average ratings or low ratings. Other good explanations were based on a simple statement, such as “*You liked 80% of the past recommendations provided*” or on content-related arguments, while too much information had a negative impact. Recently, Bridge and Dunleavy (2014) defined *explanation rules* in a user-based collaborative filtering recommender system, providing explanations in the form “*People who liked movie X also liked movie Y*”. They performed a user study and found that nearly 50% of the participants found their explanations helpful. Another kind of explanation strategy based on graphs is presented in Verbert et al. (2014). A clustermap and a Venn diagram were used to provide explanations for a conference recommender system, and a user study was performed to compare the two methods: while the clustermap was more effective, it was also too complex for non-technical users who prefer the simplicity of the Venn diagrams.

### 2.3.2 Conversational RS

Standard recommender systems are affected by a general limitation: they are intrinsically designed to collect input data at the beginning, produce recommendations as output and terminate their job. However, it can be useful to refine the recommendations and to guide the user to the best items for him/her. In order to apply this approach, the user has to provide more pieces of information to the system in a sort of dialog. This kind of recommender systems is called *conversational RSs* (Carenini et al., 2003; Thompson et al., 2004; McGinty and Smyth, 2006; Mahmood and Ricci, 2009). The critical step is the design of the interactive dialog between the user and the system: the interaction has to finish with the identification of the desired item, and it has to take a small number of steps.

*Critiquing systems* are one variant of conversational recommender systems

that establish an interactive dialog with their users. The popularity of this approach is due to the balance between the effort required by the user and the information value that this effort provided. For instance, the specification of a critique such as “*more items like X, but different in terms of feature Y*” has a lower cost than the process of elicitation of all the required features. Furthermore, critiquing helps users without deep knowledge of the domain because they give him/her the possibility to explore the product options available and to become more familiar with them (Payne et al., 1993).

A user usually has two ways of defining a critique: directional or replacement (Mcsherry, 2005). Directional critiques consist in the increase or decrease of a numerical feature, such as price, while replacement critiques let the user substitute the feature value required. The typical *recommendation cycle* starts with the presentation of an item to the user (Burke et al., 1996). The user has to accept the recommendation or to critique one of the features. The process continues until a recommendation is accepted or all the possibilities are exhausted.

### 2.3.3 Persuasive aspects of the interaction with RS

One of the key aspects in the RS area is that recommendations have to be considered by users in order to be effective. If the recommender system is not trusted, recommendations will be useless, even if predictions are very accurate. Users tend to prefer recommendations from credible sources (Metallinos, 1991). A way through which the system can increase its credibility is to leverage social aspects (Fogg, 2002; Nass and Brave, 2005). In this sense, some recommender systems act as social actors as they assist users in their tasks. Users who perceive social characteristics such as benevolence and integrity when they interact with an online recommender agent are more likely to trust the system (Benbasat and Wang, 2005). In addition, if the user perceives that the system generates recommendations in a way similar to user’s decision-making process, he/she is more likely to be persuaded by the system (Aksoy et al., 2006). These considerations show that social aspects in recommender systems are crucial in order to develop systems that are better human-computer communicators.

## 2.4 Evaluation

The evaluation of a recommender system is not an easy task. The main issue is that the quality of a recommender system is related to objectives that are difficult to assess. For instance, a recommender system has to improve the quality of the interaction between the user and the system and to provide useful recommendations with respect to relevance, novelty and serendipity. How can we measure the achievement of these objectives? Traditionally, recommender systems have been evaluated with offline experimentation. This approach is very convenient because it can be repeated as many times as needed, and it gives an impartial way to compare different algorithms. On the other hand, offline experiments do not really deal with the user because they are just based on a historical dataset collected on the past user behavior. Online experiments overcome this limitation because they deal with real human beings that have to evaluate the system they are using. The drawbacks of this approach are the difficulty to perform a user study with a large number of users and the impossibility to repeat the experiments several times.

### 2.4.1 Offline evaluation

The offline evaluation is similar to the evaluation adopted for classification systems. Broadly speaking, a dataset with users, items and ratings is split into training and validation sets. The model is trained on the training set, and then tested on the validation set. Performances are evaluated on the validation set with several metrics.

#### Methodologies

The methodology used for evaluating recommender systems is one of the steps more often overlooked. Since the metrics used to evaluate the recommendations can have different results depending on the methodology used, it is important to use and to clearly explain the evaluation procedure.

Let us assume that the dataset is a sparse user-item matrix that contains the ratings given by users to items. The most popular approach consists in the *N-fold cross-validation*: for every user the ratings are split into  $N$  folds, as shown in Figure 2.2. After that at every iteration,  $N - 1$  folds are used for training and one fold for validation. If a specific number of ratings is required for training purposes, the approach is called *given-N*. In this case, for every



Figure 2.2: Cross-validation, every fold is indicated with a different color.

user the ratings are split into  $N$  training ratings and the remaining as validation ratings (Figure 2.3). Since the cross-validation is not feasible in this situation, the usual methodology is the *repeated holdout*, that consists in the repetition of the sampling procedure to increase the stability of the results. On the other hand there is the *all-but- $N$* , where  $N$  ratings are reserved for the validation phase. The choice of using the *given- $N$*  or *all-but- $N$*  is crucial because it affects



Figure 2.3: Given-n, 3 training ratings for every user (orange).

the values of the evaluated metrics. A fixed training set has the advantage that the models learns from the same amount of data for every user, while a validation set of the same size makes the metrics more comparable.

While the methodologies described are based on the splitting of ratings, a different kind of splitting is based on users. Cross-validation is applied to split users in folds (Figure 2.4). As in the previous case,  $N - 1$  folds are used as the training set, while one fold is used as the validation set, but in this case there is a further step: how much data is given to the model for every validation user in order to predict the remaining ratings? A popular approach is the *all-but-one* validation, where all the user's ratings except one are given to the model, and the remaining one is predicted, repeating the procedure for every rating. While this approach guarantees the fairest evaluation methodology, it is also the most computationally expensive because for every validation user the model has to be tested for every user's rating. A different methodology consists in the *all-but- $N$*  or *given- $N$*  validation.

		Items									
		5	4			2	3		1		
			3	4			5	3		1	2
			5	3		2			4		4
Users			3		5	2	4			3	1
			3	4		3	4	5	1	2	
		5		3	4		2			5	
			5		5		5	4		3	
			2		3			5	5		5

Figure 2.4: Users Cross-validation, every fold is indicated with a different color.

### Metrics

The accuracy of the recommendations computed has to be assessed based on the ratings in the validation set. The metrics are divided into three categories, depending on the kind of task they measure (Herlocker et al., 2004). The tasks are prediction, i.e. the ability of the system to predict the exact rating, classification, i.e. the ability of the system to produce a set of positive items, and ranking, i.e. the ability of the system to predict a top- $N$  list of positive items.

The metrics applied to evaluate the prediction task are *RMSE* (Equation 2.14) and *MAE* (Equation 2.15).  $T_u$  indicates the test set of user  $u$ .



$$RMSE = \frac{\sum_{u \in U} \sum_{i \in T_u} (\hat{r}_{ui} - r_{ui})^2}{\sum_{u \in U} |T_u|}. \quad (2.14)$$

$$MAE = \frac{\sum_{u \in U} \sum_{i \in T_u} |\hat{r}_{ui} - r_{ui}|}{\sum_{u \in U} |T_u|}. \quad (2.15)$$

These metrics measure the difference between the predicted ratings and the actual ratings. The difference between RMSE and MAE is that RMSE penalizes errors greater than 1, while it puts less emphasis on small errors.

Since in a real recommender system the user deals with a recommendation list and not with numeric values, Cremonesi et al. (2010) highlighted that prediction metrics are not able to measure the real accuracy of a system. To evaluate the classification task at a specific cutoff of the recommendation list the considered measures are *precision* (Equation 2.16), *recall* (Equation 2.17) and *F1* (Equation 2.18).

$$P_u = \frac{H_u}{L_u}. \quad (2.16)$$

$$R_u = \frac{H_u}{T_u}. \quad (2.17)$$

$$F1_u = \frac{2 \cdot P_u \cdot R_u}{P_u + R_u}. \quad (2.18)$$

Precision is the ratio between the number of items recommended that are liked by the user ( $H_u$ ) to the number of items recommended ( $L_u$ ). Recall is the ratio of items recommended that are liked by the users ( $H_u$ ) to the total number of items liked by the users ( $T_u$ ). Finally, the F1 measure is the harmonic average between precision and recall.

These classification metrics have a main problem: if there is only one hit in the top-10 recommendation list, it does not matter if the hit is at the first position or at the last position, the precision will still be 0.1. To measure ranking accuracy other metrics have been proposed: *rank score* (Breese et al., 1998), *lift index* (Ling and Li, 1998) and *discount cumulative gain (DCG)* (Järvelin and Kekäläinen, 2000). All the three metrics give less importance to a hit if it appears in a later position in the recommendation list. Equation 2.19 describes how to compute the DCG:

$$DCG_{pos} = rel_1 + \sum_{i=2}^{pos} \frac{rel_i}{\log_2 i}. \quad (2.19)$$

where  $pos$  denotes the position up to which relevance is accumulated and  $rel_i$  returns the relevance of the recommendation at position  $i$ . The relevance can be 1 if hits are considered or the actual value of the rating. The DCG can be normalized (*normalized NDC (nDCG)*) in the interval  $[0, 1]$  dividing the DCG by the *ideal DCG (iDCG)*, which is the value of the DCG when all the hits are in the top positions.

Other metrics that fall in the ranking metrics category are *Area Under Precision-Recall Curve (AUPR)* and *Mean Average Precision (MAP)*. When plotting precision versus recall values, the AUPR is the area under the curve and lies in the interval  $[0, 1]$ . The curve can be plotted computing precision and recall at every position of the recommendation list. On the other hand, MAP is usually computed at a specific cutoff of the recommendation list as the average precision computed at every hit, as described in Equation 2.20, where  $H_i$  is the number of hits up to position  $i$  and  $rel_i$  returns 1 if there is a hit at position  $i$ .

$$MAP_{pos} = \frac{1}{H_{pos}} \sum_{i=1}^{pos} rel_i \frac{H_i}{i}. \quad (2.20)$$

### 2.4.2 Online evaluation

Although the offline evaluation is widely used due to the possibility to perform the evaluation with available resources, some considerations have to be made. First of all, in a typical historical dataset ratings are divided into training and validation ratings. Therefore, if the system recommends an item that does not have a rating in the validation set, this is seen as an incorrect prediction, while maybe the user could even like that item, but he/she is not aware of it. Cremonesi et al. (2012) investigated this subject comparing offline and online evaluations. Another common issue is due to the fact that usually the temporal dimension is not considered, and ratings are split into training and validation sets without considering the collection sequence. Campos et al. (2011) analyzed this issue and proposed a time-aware evaluation protocol, which unfortunately is rarely used.

Online evaluation can overcome these limitations by performing *online experiments* with real users. In these experiments some variables are considered

independent, i.e. the demographic features of the users, while some others are considered dependent, such as user satisfaction or click-through rate.

In an *experimental research design* at least one of the variables is manipulated in order to observe the effect of the manipulation on the dependent variables, while users are randomly assigned to different levels or categories of the manipulated variable (Pedhazur and Schmelkin, 2013). In the recommender system area the manipulated variable is usually the recommendation algorithm, and users have to be randomly assigned to different algorithms. The observed variables have to be measured before and after the interaction of the user with the system, usually with questionnaires or observing user behavior. It is also important to distribute users with different independent variables to all the considered algorithms in order to avoid an abnormal concentration of users with similar characteristics on a single algorithm.

On the other hand, in a *quasi-experimental design* subjects decide on their own about the algorithms they want to use. In this case the conclusions must be carefully considered: for instance, if users can decide if they want to use the recommender system or not, and the experiment shows that users who use the recommender system buy more items, the tendency to buy more can derive from the fact the users who are more inclined to buy are willing to use the recommender system, and not vice versa.

Finally, the *nonexperimental design* includes all other kinds of experiments where quantitative and qualitative variables are observed with questionnaires or with the implicit observation of their behavior. An interesting case is the *longitudinal search* in which the user is observed during his/her interaction with the system over time (Zanker et al., 2006).

User studies can be also divided into *lab studies* and *field studies*. In lab studies users are “recruited” to participate in a controlled environment where all the considered variables can be controlled. A frequent issue in this kind of experiment is the real commitment of the users who participate, as they could be motivated only by rewards offered for participation. In field studies this kind of problem does not exist because users are already using the tested system. A popular kind of field study is the A/B test, where users are divided into two groups that are presented with different versions of the system (Dixon et al., 2013).



## Chapter 3

# State of the Art in Topic Models

In this chapter the topic model methodology is described. The main idea behind this model is presented, and the main methodological aspects are discussed. Popular topic model extensions to address specific problems are presented and the topic model evaluation approaches are discussed. Finally, topic model applications in the recommender system area are illustrated.

### 3.1 Introduction

A *topic model (TM)*, as described in Blei (2012), is a machine learning statistical model that aims to discover and annotate large text corpora with thematic information. The main purpose of these algorithms is the analysis of words in natural language texts in order to discover themes represented by sorted lists of words.

To illustrate how topic models work, we borrow the example from Blei (2012). The article in Figure 3.1, titled “*Seeking Life’s Bare (Genetic) Necessities*”, is about using data analysis to determine the number of genes that an organism needs to survive (in an evolutionary sense). We should expect that this article would be about different themes, such as *computer science*, *evolutionary biology* and *genetics*. Every word in the article can be assigned to one of these themes, based on the context of the document itself and all the other documents analyzed. For instance, we can imagine that the word “computational”

is about computer science, while “organism” is about biology. If all the words in the document are labeled, except for unmeaningful words such as conjunctions or prepositions, we can calculate how much the document is about the themes considered. Topic models aim to do automatically what we did manually in the example: assign every word to a specific topic, compute topic proportions for every document and produce list of words as topic representation.

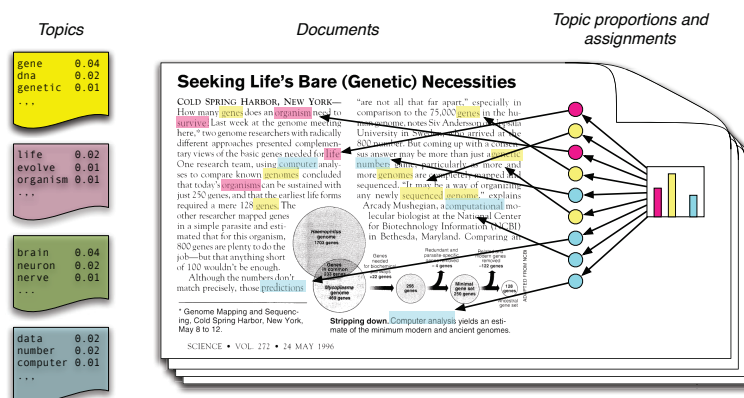


Figure 3.1: The words in a document can be about different themes; their assignment to different topics defines the document topic proportions (Blei, 2012)..

The first approach that tried to discover latent semantic themes in a document corpus was the *Latent Semantic Indexing technique (LSI)* (Deerwester et al., 1990), which consists in the decomposition of the TF-IDF matrix with matrix factorization techniques. An evolution of this model was offered by the *probabilistic LSI (pLSI)* (Hofmann, 1999), also known as *aspect model*, which introduced a probabilistic generative approach for the text: every word is generated by a *mixture model*, where mixture components are multinomial random variables that can be considered as a topic. The next step, which represents the main milestone for topic models, was the introduction of the *Latent Dirichlet Allocation technique (LDA)* (Blei et al., 2003), which describes a full generative model for topics and text. The basic idea is that every document is about several topics and that each word in the document can be associated with one of these topics.

## 3.2 Methodology

Let us define  $D$  as the number of documents,  $K$  as the number of topics and  $N_d$  as the number of words in document  $d$ . The following variables are defined:

- $\Phi_{1:K}$ : topic distributions over the vocabulary, where  $\Phi_k$  is the distribution for topic  $k$ ;
- $\theta_{1:D}$ : document distributions over topics, where  $\theta_d$  is the distribution for document  $d$ ;
- $z_{1:D,1:N_d}$ : topic assignments for each document, where  $z_{dn}$  is the assignment of position  $n$  of document  $d$  to a specific topic;
- $w_{1:D,1:N_d}$ : word occurrences for each document, where  $w_{dn}$  is the word that occurs in position  $n$  of document  $d$ .

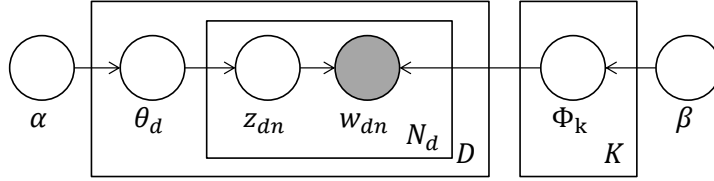


Figure 3.2: Plate notation of the LDA model.

The generative model is illustrated in Figure 3.2. Every word in every document is generated from the combination of the variables  $\theta$  and  $\Phi$ . The generative process of a generic document  $d$  consists in the following steps:

1. a topic distribution  $\theta_d$  is randomly generated;
2. for each position  $n = 1, \dots, N_d$ :
  - a topic  $k$  is extracted from  $\theta_d$  and  $z_{dn} = k$ ;
  - a word  $m$  is extracted from  $\Phi_k$  and  $w_{dn} = m$ ;

The aim of the LDA model is to invert the generative process: the occurrences of words in the documents are the observed variables ( $\mathbf{w}$ ), while the topic structure is hidden (Figure 3.3). The generative process gives the possibility to define a joint probability on observed and hidden variables (Equation 3.1), which is

used to compute the conditional probability of hidden variables given the observed variables, namely the posterior probability distribution (Equation 3.2). Variables in bold indicate the full set of variables for every index.

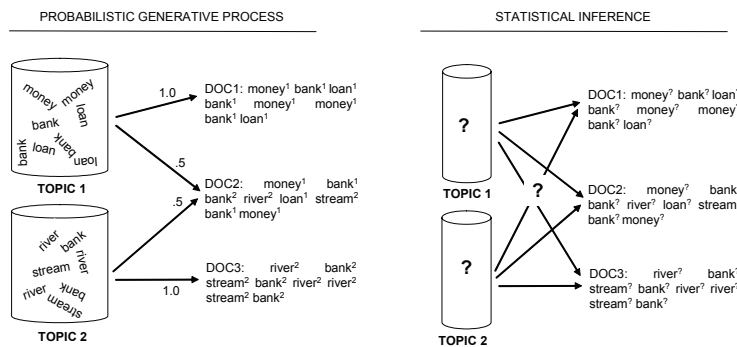


Figure 3.3: From the generative process to the statistical inference (Steyvers and Griffiths, 2007).

$$P(\Phi, \theta, \mathbf{z}, \mathbf{w}) = \prod_{i=1}^K P(\Phi_i) \prod_{d=1}^D P(\theta_d) \prod_{n=1}^{N_d} P(z_{dn} | \theta_d) P(w_{dn} | \Phi, z_{dn}). \quad (3.1)$$

$$P(\Phi, \theta, \mathbf{z} | \mathbf{w}) = \frac{P(\Phi, \theta, \mathbf{z}, \mathbf{w})}{P(\mathbf{w})}. \quad (3.2)$$

The exact estimation of these distributions requires the computation of the probability to observe every word of every document assigned to every possible topic, which is computationally infeasible. To tackle this problem, two different approaches have been proposed: sampling algorithms and variational algorithms.

The most popular sampling algorithm for topic models is the Gibbs sampling Geman and Geman (1984). This approach consists in the definition of a Markov chain on the set of hidden variables and in the sampling of the value of each variable, given the value of the others. The process is iterated many times, and samples are periodically collected in order to approximate the stationary distribution of the Markov chain. Samples collected are used to estimate the posterior distribution. In particular,  $\Phi$  and  $\theta$  are computed as shown in Equation 3.3 and Equation 3.4, where  $C_{kw}^{\Phi}$  is the number of times that the word  $w$  is assigned to topic  $k$ ,  $C_{dk}^{\theta}$  is the number of times that a word in the document  $d$



is assigned to topic  $k$ ,  $C_k^\Phi$  is the number of words assigned to topic  $z$  and  $C_d^\theta$  is the number of words in document  $d$ . This approach was proposed by Griffiths and Steyvers (2004).

$$\hat{\Phi}_{kw} = \frac{C_{kw}^\Phi + \beta}{C_k^\Phi + W\beta}. \quad (3.3)$$

$$\hat{\theta}_{dk} = \frac{C_{dk}^\theta + \alpha}{C_d^\theta + Z\alpha}. \quad (3.4)$$

Variational algorithms are the deterministic alternative to sampling algorithms. They exploit distribution families to estimate the posterior distribution, transforming the inference problem into an optimization problem. This approach was applied to the LDA model in Blei et al. (2003).

### 3.3 Extensions

LDA is a powerful model that is extremely useful for discovering hidden themes in big textual corpora. One of the main advantages of the probabilistic formulation is the possibility to adapt and enrich it in order to design more complex models and to solve different kinds of problems.

A first extension of topic models can be obtained removing the “*bag of words*” assumption, which consists in the idea that the occurrence of words is important, but not the order in which they occur. In Wallach (2006) the generative process is modified so that a word is not only generated based on the topic sampled, but even based on the previous word. In Griffiths et al. (2004) LDA was merged with the Hidden Markov Model, where the former is responsible for the word generation, while the latter defines the syntax class of the word. The major drawback of these approaches is that they significantly increase the dimension of the parameter space, but the modeling of natural language text is better than in the standard LDA.

A second extension is obtained considering the ordering of documents that in LDA is not taken into account. In Blei and Lafferty (2006b) documents are divided into time frames, and a special kind of topic model, called *Dynamic topic model*, discovers topics and their evolution through time.

A third important extension concerns the number and structure of topics. In the LDA model the number of topics is a parameter, and the topic structure is flat. Teh et al. (2006) introduced the Bayesian nonparametric topic model,

where the number of topics is automatically determined during the inference phase, and new documents can cause the creation of new topics. This approach was extended in Blei et al. (2010), where the topic structure is assumed to be hierarchical, and the number of subtrees and the depth of each subtree is determined automatically. This model is able to discover relations of specialization and generalization between topics, finding high-concept topics close to the root and specialized topics towards the leaves.

Topic models have been extended in many other ways: correlated topic model (Blei and Lafferty, 2006a) and the Pachinko allocation machine (Li and McCallum, 2006) are designed to find correlations between topics. The author-topic model (Rosen-Zvi et al., 2004), the author-recipient-topic model (McCallum et al., 2005) and the relational topic model (Chang and Blei, 2009) are able to discover topics in structured corpus where the author and/or the recipient of a document are known or when documents have connections between them.

### 3.4 Topic model evaluation

Once a set of topic have been extracted from a document corpus by applying LDA, their semantic coherence must be validated. The first approach proposed to evaluate topic models was based on the *perplexity* measure on held out documents. Perplexity indicates the uncertainty in predicting a word given a model: a lower perplexity value means that the model explains the natural language text in a good way. In topic models it simply means that the topic structure in the training set is the same as the one in the validation set.

To improve this simple technique, Wallach et al. (2009) summarized several evaluation techniques based on tools from language modeling. In particular, the *Chibb-Style estimation* and the *left-to-right evaluation* have been proved to perform better on real-world corpora. On the other hand, Chang et al. (2009) measured topic coherence using human judgments. Judges recruited through *Amazon Mechanical Turk*<sup>1</sup> were asked to evaluate a set of topics: every topic was presented as a list of words in which one word was manually included to act as an intruder. The judges had to find the intruder in the topics evaluated. Based on the success ratio, topics were evaluated for their coherence because if the intruder was easy to detect it means that the other words had a strong coherency. Unfortunately, metrics computed for this study have a negative

---

<sup>1</sup>[www.mturk.com](http://www.mturk.com)

correlation with metrics usually applied to evaluate topics. Other works show that some problems exist when using supervised classification predictive metrics. To overcome this limitation, Ramirez et al. (2012) proposed a procedure based on the comparison of the extracted topics with the results obtained through alternative models. Numerical experiments show that the proposed approach for topic model validation is effective. However, the problem of topic model validation is still an open issue.

### 3.5 Topic model and recommender systems

Topic models were also adapted and introduced in recommender systems for recommending textual items.

Wang et al. (2010) proposed a probabilistic generative model similar to LDA applied to textual reviews on hotels to estimate aspect ratings, a problem defined as *Latent Aspect Rating Analysis (LARA)*. Each review is split into sentences, and each sentence is supposed to be about a specific aspect. The proposed generative model assumes that for each sentence a user decides which aspect he/she wants to write and chooses the words to write, carefully based on the decision made. To assign one or more aspects to each sentence a bootstrap procedure is defined: an initial seed of aspect keywords is provided, and based on this sentences are assigned to different aspects. The empirical experiments show that the proposed method is able to estimate aspect ratings, discovering interesting cases where the overall ratings are the same, but aspect ratings are different. Furthermore, review analysis opens a range of possible applications, such as aspect opinion summarization, ranking of entities based on aspect ratings, and analysis rating behavior of reviewers.

Agarwal and Chen (2010) introduced a matrix factorization method for recommender systems where items have a natural bag-of-word representation, called *fLDA*. Topics extracted from item descriptions and user metadata are used as priors to regularize item and user latent factors. The posterior distribution of item and user factors depends on both the prior and user ratings on items, since the LDA model is exploited to regularize item latent factors, and the Gaussian linear regression regularizes user latent factors. The proposed model is accurate and able to deal with cold-start and warm-start scenarios. Furthermore, it provides interpretable latent factors that can explain user-item interactions.

Wang and Blei (2011) defined an extension of LDA for recommending scientific articles called *collaborative topic regression (CTR)*. Topic model and matrix factorization are merged in a single method, where item latent factors are obtained adding an offset latent variable to the item topic distribution. The latent variable is optimized with an EM algorithm, together with LDA and MF parameters. This method is capable of providing in-matrix and out-of-matrix predictions, where the former case consists in the prediction of items already rated by some users, while the latter consists in the prediction of new items. Even in this case the method is able to provide interpretable latent factors that can be used to profile users and items.

Recently, McAuley and Leskovec (2013) merged matrix factorization and topic models in order to estimate the ratings from textual reviews on different datasets. The *Hidden Factors as Topics (HFT)* consists in two steps: first, latent factors for rating prediction are fitted, and second, topic assignments are updated binding item topic distributions and item latent factors. In this work all the reviews associated with an item are merged into a single document. The proposed approach not only leads to more accurate predictions on recommendations, but can also solve side problems. First, it deals with the cold-start problem, exploiting item topics for items with only a few ratings. Second, it is able to discover and automatically categorize items in different categories based on the topics discussed in the reviews. Third, it can identify representative reviews, which can be shown to users as a summary of item characteristics. The proposed approach was tested on a set of huge datasets scraped from the web: 35 millions review from *Amazon*<sup>2</sup>, 6 millions review from *ratebeer*<sup>3</sup> and 220 thousand reviews from *Yelp*<sup>4</sup>.

---

<sup>2</sup>[www.amazon.com](http://www.amazon.com)

<sup>3</sup>[www.ratebeer.com](http://www.ratebeer.com)

<sup>4</sup>[www.yelp.com](http://www.yelp.com)

## Chapter 4

# Integrating Concepts and Time in Recommender Systems

In this chapter three contributions to the algorithm component are presented with the discussion of different methodologies that can be applied to the recommender system area. The first idea is the integration and semantic connection of large content networks, such as Wikipedia or the web. The topic model is applied to discover concepts in the network and to create new semantic links between objects. These networks can be exploited to provide recommendations in content networks like the scientific literature or in the publishing industry. The second hint is the application of the topic model to analyze temporal fluctuations of themes in a document corpora. The temporal aspect in recommender systems is an important aspect that is often not considered. The third contribution is the definition of the MapReduce version of the Continuous Time Bayesian Network classifier training algorithm. This model can be applied to recommender systems where the duration of events is an important aspect to consider, while the MapReduce version is able to deal with Big Data.

## 4.1 Introduction

The first contribution of this chapter concerns large content networks, like the World Wide Web, which contain huge amounts of information that have the potential of being integrated because their components fit within common concepts and/or are connected through hidden, implicit relationships. While one attempt at the integration of semi-structured information sources is the program called the “Web of Data”, our approach aims to integrate unstructured information sources, as the vast majority of the information residing on the Web is in such form. For this purpose we exploit topic models in order to cluster Web pages into concepts (topics), which are then related through higher-level concept networks; we also make implicit semantic relationships emerge between single Web pages. While the applicative focus of the research reported here is on knowledge integration on the specific and relevant case of the WWW, this framework can be designed as a hybrid recommender system that exploits content information and semantic connections to provide useful recommendations. For instance, a researcher may use this framework to navigate the scientific literature through concepts, looking for the research area of interest and analyzing semantic connections between an area and other areas, visualizing border articles which connect different areas. Furthermore, based on other researchers’ interests, the system can automatically derive communities that have been proved to be effective in recommendation tasks (Vassileva, 2008). Another interesting application scenario is the publishing industry, where such recommenders can support “*word of mouth*”, recommending news articles (Webster and Vassileva, 2007).

Another important aspect in recommender systems is the temporal dimension, which is often not considered in most of the literature. The temporal aspect is obviously important in many domains, since user tastes may change over time. For instance, a teenager can be interested in action movies, while growing he/she may find also interesting other kinds of more serious movies. In such cases, preferences expressed several years before are no longer significant or may be less important than recent preferences. Many methods are trying to incorporate temporal dynamics in recommender systems, like Koren (2010) did to win the NetFlix prize or in Context-Aware recommender systems (Adomavicius and Tuzhilin, 2011). Our approach deals with the temporal dimension on document corpora, such as research papers or news articles. The proposed method hinges on the tracking of increased information flows, what we call hot

topics, around the evolution of a document corpus. These hot topics can be exploited in order to compute accurate recommendations that take into account user preferences as well as topic popularity.

In many cases the temporal dimension is just considered as a sequence, where the only relevant information is related to the previous states of the system, like in Dynamic Bayesian Networks and Hidden Markov Models. However, in many domains the temporal aspect has to be considered as it is, a continuous flow where the duration is a piece of crucial information. Continuous time Bayesian networks (CTBN) have been recently proposed to cope with continuous time stochastic processes (Nodelman et al., 2002). A continuous time Bayesian network is a graphical model whose nodes are finite state variables in which the state evolves continuously over time, and where the evolution of each variable depends on the state of its parents in the graph. Inference and learning algorithms for continuous time Bayesian networks and their classifiers have been presented in the literature (Nodelman, 2007; Stella and Amer, 2012), but they have a main limitation: when the data size grows the learning time becomes unacceptable. To overcome this limitation, the MapReduce framework (Dean and Ghemawat, 2008) can be used. This framework offers the possibility to implement a parallel application without focusing on the details of data distribution, load balancing and fault tolerance (Dean and Ghemawat, 2010). In this chapter we show how to implement the MapReduce version of the algorithms for parameter and structural learning of CTBN classifiers. These methods can be exploited by recommender systems that deal with continuous temporal dynamics. For instance, IPTV (Internet Protocol Television) and online news platforms can benefit from such models that can analyze the “examination duration” (Oard et al., 1998) and use it to learn user preferences (Song et al., 2012), while e-commerce websites can consider the duration between the item’s launch time and user’s purchase time (Lee et al., 2008).

The remainder of this chapter is as follows. Section 4.2 illustrates the applications of PTM to knowledge integration in the WWW via the clustering of Web pages into concept (topic-topic) networks and the explicitation of semantic links at the level of single Web pages through object-object networks, with quantitative and qualitative evaluations. Section 4.3 shows how to apply hot topic tracking to the monitoring of the evolution of themes in the context of the implementation of labor law. Section 4.4 shows the design of two training algorithms for continuous time Bayesian network classifiers following the MapReduce paradigm and the analysis of the speedups in different situations.

## 4.2 Integrating Concepts in Large Content Networks

Integrate concepts and knowledge in large content networks relies on Latent Dirichlet Allocation (LDA) (Blei and Jordan, 2003) as the basic component to develop a method for extracting concepts, i.e. topics, from existing information networks. Extracted concepts and their connections are then used to learn two semantic networks: i) a high-level topic-topic network where topics are related in terms of their semantic proximity; ii) a fine-grained object-object network where textual objects making up topics and accounting for their relationships are analyzed, with links that trespass topic boundaries.

This method has been tested through different datasets that were chosen on the basis both of their effective relevance in terms of available content and their suitability to act as meeting points for diverse knowledge sources, thus fitting well with the objective of demonstrating the potentials of knowledge integration on the World Wide Web.

### 4.2.1 The Object-Object Network and the Topic-Topic Network

Let  $K$  be the number of topics and  $N$  the number of objects. We reserve the  $j$  index to indicate objects and the  $z$  index to indicate topics. To indicate a matrix we use a letter without subscripts, while the single element is indicated with two subscripts. Full details about the construction of the *Object-Object Network* and the *Topic-Topic Network* are provided in Rossetti et al. (2014).

The *object-object matrix*  $\Gamma^{OO}$  describes the one step transition probability for each pair of textual objects. Therefore, the element  $\Gamma_{j_1 j_2}^{OO}$  is the one step probability  $P(j_2|j_1)$  of transitioning from the textual object  $j_1$  to the textual object  $j_2$ . By construction we have:

$$P(j_2|j_1) = \sum_{k=1}^K P(j_2|k)P(k|j_1) = \Gamma_{j_1 j_2}^{OO} \quad (4.1)$$

The *topic-topic matrix*  $\Gamma^{TT}$  describes the one step transition probability for each pair of topics. Therefore, the element  $\Gamma_{k_1 k_2}^{TT}$  of the *topic-topic matrix*  $\Gamma^{TT}$  is the one step probability  $P(k_2|k_1)$  of transitioning from topic  $k_1$  to topic  $k_2$ .



Also in this case by construction we have:

$$P(k_2|k_1) = \sum_{j=1}^N P(k_2|j)P(j|k_1) = \Gamma_{k_1 k_2}^{TT} \quad (4.2)$$

It is now possible to define the *object-object network* and the *topic-topic network*.

**Definition 1 Object-Object Network.** Given a textual object corpus  $\mathcal{O} = \{j_1, \dots, j_N\}$  and a set of topics  $\mathcal{Z} = \{z_1, \dots, z_Z\}$ , an *Object-Object Network* is a couple  $OON = \{G = (\mathcal{O}, E), \Gamma^{OO}\}$ , where  $G$  is a directed graph with nodes associated with textual objects belonging to the set  $\mathcal{O}$ ,  $E$  is the set of links between nodes where the link between node  $j_1$  and node  $j_2$  is associated with the weight  $\Gamma_{j_1 j_2}^{OO}$ .

**Definition 2 Topic-Topic Network.** Given a textual object corpus  $\mathcal{O} = \{j_1, \dots, j_N\}$  and a set of topics  $\mathcal{Z} = \{z_1, \dots, z_Z\}$ , a *Topic-Topic Network* is a couple  $TTN = \{G = (\mathcal{Z}, E), \Gamma^{TT}\}$ , where  $G$  is a directed graph with nodes associated with topics belonging to the set  $\mathcal{Z}$ ,  $E$  is the set of links between nodes where the link between node  $z_1$  and node  $z_2$  is associated with the weight  $\Gamma_{z_1 z_2}^{TT}$ .

Once the textual object corpus and the set of topics are transformed to the corresponding Object-Object Network  $OON = \{G = (\mathcal{O}, E), \Gamma^{OO}\}$  and Topic-Topic Network  $TTN = \{G = (\mathcal{Z}, E), \Gamma^{TT}\}$ , network analysis measures can be computed and community formation algorithms can be executed.

In particular, we applied the reverse Cuthill-McKee (RCM) algorithm (Cuthill and McKee, 1969). The main idea behind this approach is represented in Fig. 4.1: the RCM algorithm permutes a sparse matrix that has a symmetric sparsity pattern in a band matrix form with a small bandwidth. Broadly speaking, it reorders rows and columns of a sparse matrix with the aim to carry all non-zero elements towards the main diagonal.

Fig. 4.1(a) shows the result of the application of the RCM algorithm to the  $\Gamma^{OO}$  matrix. Documents are reordered in a way that brings documents one next to the other, which speak about the same topics. Squares on the main diagonal represent groups of documents that speak about the same topics. If we apply the permutation obtained from the  $\Gamma^{OO}$  to the adjacency matrix, which represents a hyperlink between documents with a 1 and no hyperlink with a

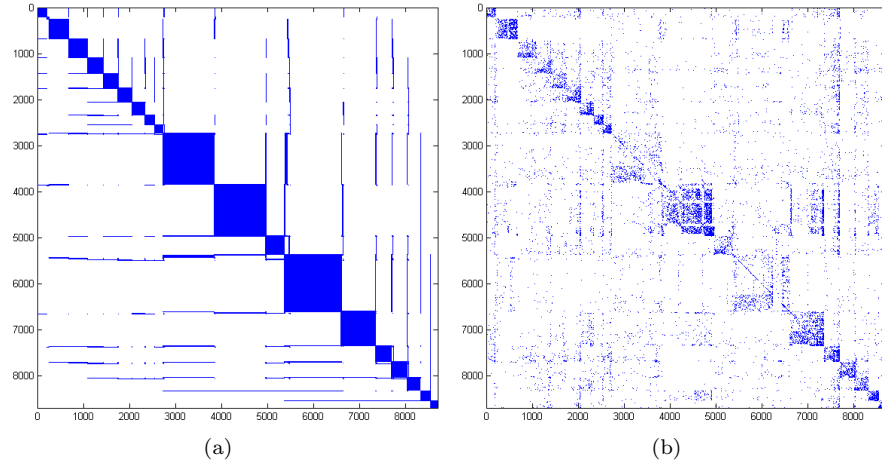


Figure 4.1: Chart (a) shows  $\Gamma^{OO}$  reordered with the RCM algorithm. Chart (b) shows the adjacency matrix reordered with the RCM permutation obtained from  $\Gamma^{OO}$ .

0, we obtain the matrix in Fig. 4.1(b). The formation of dense regions of hyperlinks corresponding to “topic-squares” area can be observed. We consider each one of this square as a document community.

### 4.2.2 Experiments

Numerical experiments are devoted to analyzing and comparing the behavior of the TT algorithm to the behavior of the following benchmarking algorithms: Infomap (Rosvall and Bergstrom, 2008) and Harel (Harel and Koren, 2001). Experiments are performed starting from different *core arguments*, each associated with a *textual object corpus* extracted from the YAGO ontology, Wikipedia and the web. Qualitative experiments have been also performed to show the validity of the proposed approach as a navigation tool.

#### Performance Measures

Performance measures have been selected to show that the TT algorithm, which exploits semantic information through the  $\Gamma^{OO}$  matrix but does not exploit any web page link data, performs well in terms of semantic and graph community similarity measures. Furthermore, selected performance measures allow to show that benchmarking algorithms which exploit only web page link data do not per-

form well in terms of semantic similarity and are almost always outperformed by the TT algorithm in terms of graph community similarity measures. Therefore, *YAGO similarity* and *symmetrized Kullback-Leibler similarity* have been selected to evaluate and compare semantic coherence through the computation of intra-community similarities based on an external ontology and on the latent topic structure. To evaluate the structural density and the quality of the partitioning of a graph structure the *modularity* and the *assortativity* performance measures have been selected. A full description of these measures is provided in Rossetti et al. (2014).

### Performance analysis and comparison

Performance analysis and comparison are based on the following core arguments: *Green Economy (GE)*, *City Cars (CC)*, *Touchscreen mobile phones (TMP)*, and *Terrorism Terr*). The size of each dataset is indicated in the label, i.e. *GE\_1K* means 1,000 pages. The main characteristics of numerical experiments, as well as *optimal values* for parameters  $K$  and  $\mu$  are reported in Rossetti et al. (2014).

Numerical experiments have been performed following an estimation scheme similar to *k-fold cross validation*, where each algorithm was applied to all possible combinations consisting of  $k - 1$  data subsets. Performance measure values achieved are summarized in Table 4.1. One-sided, two sample t-test with unequal variances has been used to test the TT algorithm against the other algorithms. Statistically significant results at 95% confidence level are bolded.

The TT algorithm outperforms Harel and Infomap algorithms for all core arguments and for almost all performance measures. Just in the case of the *Green Economy* core argument, the TT algorithm achieved a mean value of assortativity (ASS) which is not statistically greater than the mean value achieved by Harel and Infomap algorithms. Furthermore, for (GE\_5K) Harel is statistically better than TT.

Numerical experiments show that the TT algorithm outperforms Harel and Infomap algorithms with respect to both YAGO similarity (YAGS) and symmetrized KL similarity (SKLS) measures. Results on SKLS were somewhat expected since the TT algorithm is based on topics distributions.

However, good results achieved on the YAGO similarity (YAGS) measure mean the learned topics reflect the ontological structure of the analyzed textual objects. This confirms topics are useful to extract knowledge from textual data. Furthermore, communities discovered by exploiting topics are related to the on-

Table 4.1: Performance comparison for the *Green Economy* (GE.1K and GE.5K), *City Cars* (CC.10K), *Touchscreen mobile phones* (TMP.10K), and *Terrorism* (Terr.10K) core argument experiments. Mean value of performance measures for Harel, Infomap and TT algorithms.

Data set label	Algorithm	YAGS	SKLS	MOD	ASS
GE.1K	Harel	0.431	0.191	0.769	0.818
	Infomap	0.366	0.027	0.523	0.908
	TT	<b>0.442</b>	<b>0.428</b>	<b>0.862</b>	0.908
GE.5K	Harel	0.367	0.091	0.609	<b>0.941</b>
	Infomap	0.385	0.046	0.518	0.655
	TT	<b>0.571</b>	<b>0.450</b>	<b>0.733</b>	0.875
CC.10K	Harel	0.397	0.130	0.522	0.799
	Infomap	0.398	0.040	0.364	0.674
	TT	<b>0.632</b>	<b>0.425</b>	<b>0.789</b>	<b>0.823</b>
TMP.10K	Harel	0.384	0.133	0.526	0.798
	Infomap	0.379	0.030	0.252	0.701
	TT	<b>0.613</b>	<b>0.394</b>	<b>0.894</b>	<b>0.928</b>
Terr.10K	Harel	0.503	0.165	0.556	0.595
	Infomap	0.423	0.020	0.633	0.728
	TT	<b>0.665</b>	<b>0.383</b>	<b>0.874</b>	<b>0.899</b>

tological structure of the analyzed textual objects. Results achieved by TT on modularity (MOD) and assortativity (ASS) are encouraging. It achieves performance values which are better than those achieved by the Harel and Infomap algorithms, both designed to exploit the graph structure.

### Topic-Topic network navigation

Topic-Topic network is navigated to discover useful details and relations between the learned topics.

Figure 4.2 depicts the navigation of the Topic-Topic network associated with the *Terrorism* core argument. Every node is associated with a topic: green nodes are associated with topics related to the core argument, while red nodes are associated with topics related to green nodes. An edge between topics means that there are textual objects associated with both topics. The graph is directed, since an edge from topic A to topic B means that textual objects which are about topic A usually are also about topic B, but not viceversa. The size of the edge is proportional to the number of textual objects that are about the two topics.

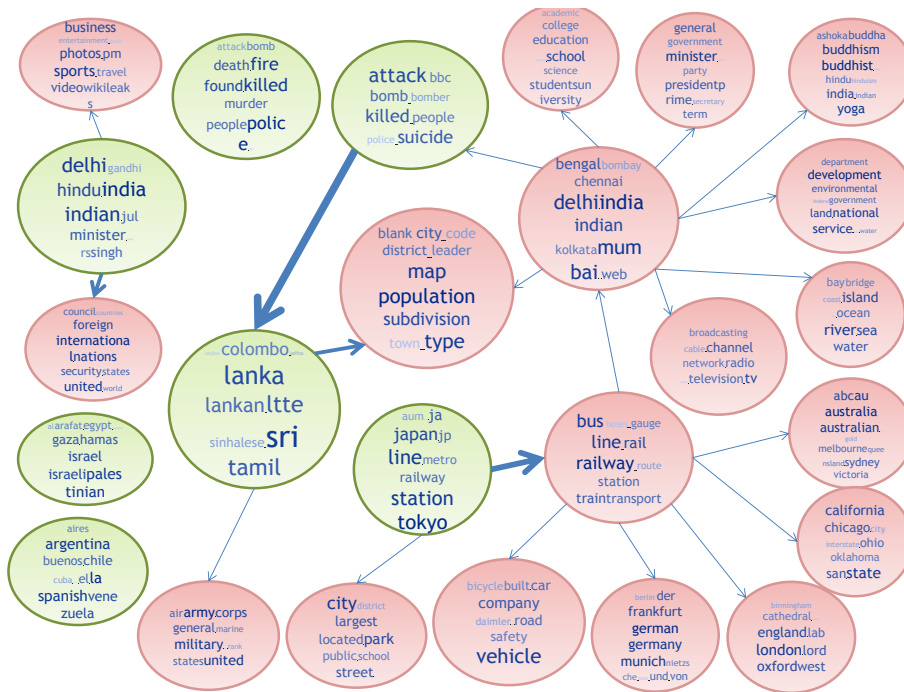


Figure 4.2: Topics Navigation.

In this example we assume there is interest in terrorism on public transportation. To better illustrate the navigation process, navigated topics are labeled with the corresponding topic number. Topic 162 is one of the topics related to the *Terrorism* core argument, and it seems to be also related to public transportation. The analysis of pages related to this topic shows that it is about the Sarin gas attack on the Tokyo subway in 1995. The topic is connected with topic 158, which is about public transportation but not terrorism. Navigating the graph we find several topics connected with this last one, and every one is about a specific location or geographic area related to public transportation. In the end we find that topic 80 is about *India*, and it is connected to one of the most explicit topics about *Terrorism*, topic 97. In this way we have found a navigation path that seems to be interesting for our purpose. In addition we can further analyze this navigation path looking at the textual objects which are about these topics with an additional graph (see Rossetti et al. (2014) for more details).

### 4.3 Tracking Hot Topics

The aim of this contribution is to study hot topics in text corpora. Hot topics are extremely interesting for recommender systems because they identify popular trends that must be separated from user interests. While a user can read news articles on popular topics, he/she may be not interested in that topic, but read it only to be updated on “general knowledge”. For instance, a popular topic in 2011 was the British Royal Wedding: it is very likely that everyone read at least one article about it, but this does not mean that a user is interested in that particular topic.

#### 4.3.1 Topic Models for Hot Topic Tracking

To analyze hot topics we exploit LDA (Latent Dirichlet Allocation) (Blei et al., 2003) for the extraction of topics (concepts) from document corpora. The process of topic extraction returns the probability distribution  $p(w|z)$  of the words of the document corpora for each topic  $z$  and the probability distribution  $p(z|j)$  of the topics for each document  $j$ . The idea behind the use of LDA for monitoring hot topics is to exploit the flow of information to identify situations where topics that are more closely associated with contents become densely populated. We can take this as a signal that the topic is undergoing a critical phase and that the exchanged information may indeed contain key indicators about possible changes and re-adaptations. Conversely, once we observe that a topic has stabilized as far as the exchange of information is concerned, then we can assume that it has also reached a stable state in its definition.

To achieve this, we need to plot the evolution of the measured probability of each topic against time. Let us define  $t$  as the time frame considered,  $p(z|j)$  as the probability of topic  $z$  given the judgment  $j$ ,  $p(j|t)$  as the probability of judgment  $j$  given the time frame  $t$  and  $T_j$  as a function that associates the judgment  $j$  with the relative corresponding time frame. The empirical probability that an arbitrary judgment  $j$  issued a time period  $t$  was about topic  $z$  is indicated with  $p(z|t)$  and it is defined in Equation 4.3:

$$p(z|t) = \sum_{j:T_j=t} p(z|j)p(j|t) = \frac{1}{C} \sum_{j:T_j=t} p(z|j) \quad (4.3)$$

Since  $p(j|t)$  is the probability that the judgment is assigned to the time frame  $t$ , that term can be substituted with  $1/C$ , where  $C$  is the number of judgments

in the time frame  $t$ . The function  $T$  can be parameterized to yield time intervals corresponding to one month, two months, four months, six months and one year periods.

As a note on related work, a somewhat similar equation has been applied in Hall et al. (2008) with a different purpose, namely the statistical and quantitative reconstruction of the history of ideas in a variety of scientific areas, with a case study on the evolution of research directions in computational linguistics through the topic-based analysis of 12,500 articles published in major international conferences in the field between 1980 and 2005. Chen et al. (2007) addressed the notion of hot topic in a vein very similar to ours, but their formal and computational treatment falls completely outside PTM and LDA, and in fact is term-oriented rather than topic-oriented. It is important to note that extensions of PTM, such as dynamic topic models (Blei and Lafferty, 2006b), explicitly deal with topics evolution over time. In this case we are not interested in topic evolution as an analysis of how the discussion evolves about a specific topic over time, but we want to analyze topics that at a given point become very popular and widely discussed. For this reason, the standard LDA is more suitable.

### 4.3.2 Experiments

In order to show the ability of the proposed method to find hot topics we have employed LDA to classify 20,600 rulings issued by the Italian Court of Cassation in matters pertaining to labor law between 2009 and 2014. This period saw several innovations of Italian labor law, some of which are attributable to the implementation of European directives in the field, on topics such as contract flexibility, apprenticeship contracts, project contracts and supply contracts. We can therefore expect that processes, which typically involve businesses, trade unions and workers as stakeholders, made possible by these innovations have gone through a period of adjustment solved through the deliberating activity of the Court of Cassation; this activity can in turn be reconstructed by tracking hot topics within the corpus.

We ran LDA setting the number of topics to extract equal to 10, 20, 50, 100 and 200 in order to find the best granularity of topics. Domain experts, namely labor lawyers, chose the 50 topics experiment as the best candidate and specifically reviewed and graded the set of 50 topics. In all 18 topics turned out to be good performers, 14 were considered noise with the remaining ones

being somewhat uncertain. Of the 18 good performers a further selection can be made by taking out 4 topics that are so close to the other ones to correspond substantially to clones. Best performing topics are shown in Table 4.2.

Table 4.2: Best performing topics from the 50 topics extracted.

<b>Best performing topics</b>	<b>Relevant words</b>
Transfer of business	judge (giudice), conviction (convincimento), evidence (prova), irregularity (irregolarit), company (azienda)
Collective contract	collective (collettivo), contract (contratto), agreement (accordo)
Collective dismissal	employees (dipendenti), union (sindacali), criteria (criteri), mobility (mobilit), collective (collettivo)
Work injury	injury (infortunio), liability (responsabilit), damage (danno), insurance (assicurazione)
Dismissal for just cause	contestation (contestazione), sanction (sanzione), just cause (giusta causa), conduct (condotta), justified (giustificato)
Overtime work	compensatory rest (riposo compensativo), damage (danno), availability (reperibilit)
Journalistic job provision	provision (prestazione), activities (attivit), journalists (giornalisti), nature (natura), guarantee (garanzia)
Nature of the enterprise	cooperative (cooperativa), family (familiare), tax (tributario), administration (gestione), protection (tutela), shareholder (socio)
Fixed-term employment contracts at the Italian Post	contract (contratto), fixed-term (termine), Italian Post (Poste Italiane), damage (danno)
Impact on severance indemnities of overtime work	overtime work (lavoro straordinario), indemnities (trattamento), compensation (compenso), national collective labor contract (CCNL)
Notice and indemnity in the agency contracts	contract (contratto), agent (agente), indemnity (indennit), notice (preavviso)
Criteria of rotation in the extraordinary wages guarantee fund	extraordinary wages guarantee fund (CIGS), criteria (criteri), rotation (rotazione), agreement (accordo), Fiat
European directive on transfer of undertakings	transferee (cessionario), court of justice (corte di giustizia), European (europea), directive (direttiva), transfer (trasferimento), seniority (anzianit)
Duties and qualifications of company directors	national collective labor contract (CCNL), qualifications (mansioni), category (categoria), superiore (higher), director (dirigente)

The characteristics of a good performer, in the eyes of domain experts, can be summarily characterized in the ability to identify concepts specifically attributable to a particular legislative and / or decision-making context, e.g. “collective dismissal / union agreement / selection criteria” or “rotation / re-



dundancy funds / Fiat agreement”.

We have then applied Equation 4.3 to monitor the trends in the topics. Topic “Dismissal for just cause” (the fifth from the top of Table 4.2) makes for an interesting and a relevant case. The theme of the topic has been in fact substantially revised by the most recent labor reform in Italy, which entered in force in June 2012, among other things by introducing relevant modifications in the process related to the retaining of workers by businesses, in particular regarding so called small and medium enterprises (SMEs). We can therefore expect that immediately after that the topic would heat up. This could be related to the ability to make decisions on extant procedures, by taking into account the new norms. As a confirmation to that, Figure 4.3 shows a peak in the topic trend (probability evolution) during the second half of 2012, that on a bimonthly split can be exactly located in September 2012. After this peaking the topic progressively cools down, an indicator that the corresponding process has for the time being readjusted and stabilized.

However, the topic “Dismissal for just cause” is not the hottest topic among those that we have identified. In fact, the second last item from Table 4.2, namely “European directive on undertakings”, is hotter. We can compare how far hotter it is with respect to “Dismissal for just cause” by plotting the trends of the two topics one against the other as in the graphic in Figure 4.4, where we can also notice that the latter topic peaks up at its highest probability value during the second half of 2011 and then resurges sharply again for a longer

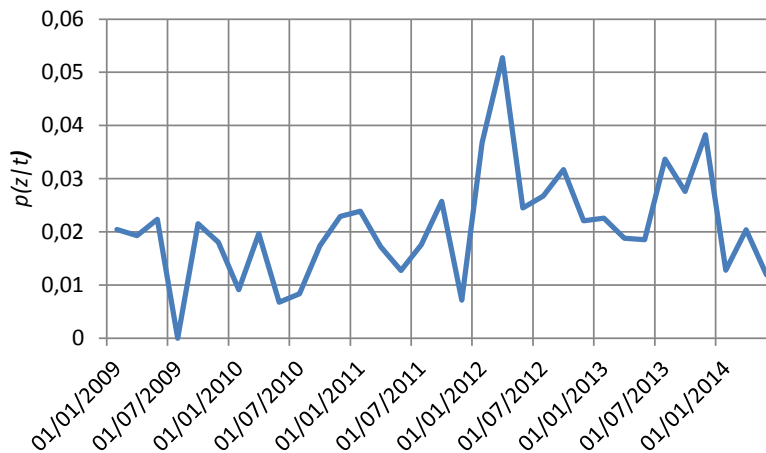


Figure 4.3: “Dismissal for just cause” topic evolution.

period, encompassing most of the second half of 2012 and of the first half of 2013. It is in this period that the Italian Court of Cassation issued a number of rulings that have become fundamental benchmarks in the Italian context for the implementation of the European directive on transfer of a business (or of a business unit). Topic “European directive on undertakings” is hotter as a topic than “Dismissal for just cause” because the scope of “European directive on undertakings”, that concerns companies of all sizes, and touches an issue of foremost importance (sometimes decisive for the fate of thousands of workers), is so much wider than the changes affecting the scope of “Dismissal for just cause”, where the actors mostly concerned are SMEs and the dealt cases are about individual workers. Eventually, as an example of a mid-flyer we can find topic “Overtime work”, dealing with the theme of compensation for overtime work, a subject that is well-known and established, but given its numerous interpretations and social relevance, is bound to heat up from time to time, with the Court of Cassation acting as an actor of arbitration and regulation for the diverse options open in the execution of the processes.

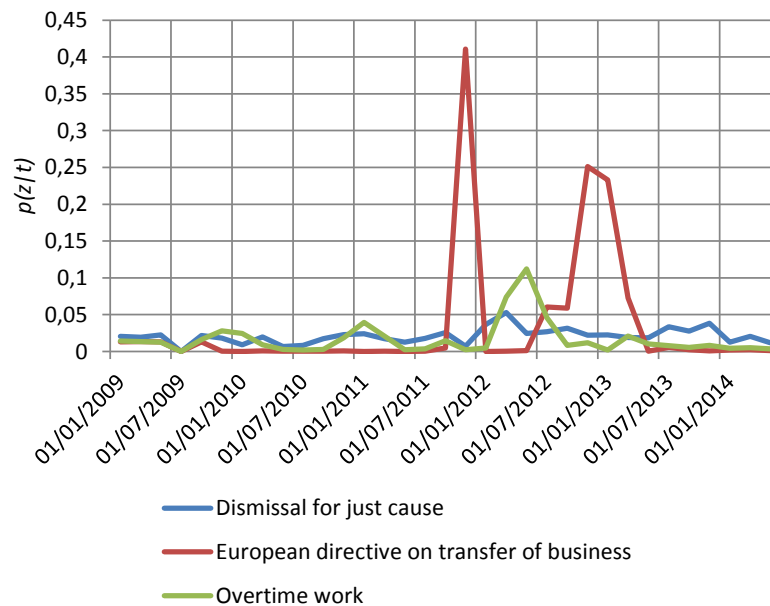


Figure 4.4: Topic evolution of three related topics.

## 4.4 Learning CTBNC Using MapReduce

This section presents the MapReduce learning framework of two continuous time Bayesian network classifiers (CTBNC): the naïve Bayes, and the tree augmented naïve Bayes, described in Friedman et al. (1997). Since the full picture goes beyond the objectives of this Thesis, a simple description of the work is given. Complete details can be found in Villa and Rossetti (2014). To the best of our knowledge, continuous time Bayesian networks haven't been applied to recommender systems yet. However, in specific domains where the duration of events is a critical feature, we think that CTBN can give a significant contribution.

### 4.4.1 CTBN and MapReduce

#### Continuous time Bayesian network

A continuous time Bayesian network is a graphical model whose nodes are finite state variables in which the state evolves continuously over time, and where the evolution of each variable depends on the state of its parents in the graph. This framework is based on homogeneous Markov processes, but utilizes ideas from Bayesian networks to provide a graphical representation language for these systems (Nodelman et al., 2002).

**Definition 3** *Continuous time Bayesian network (CTBN), (Nodelman et al., 2002). Let  $\mathbf{X}$  be a set of random variables  $X_1, X_2, \dots, X_N$ . Each  $X_n$  has a finite domain of values  $Val(X_n) = \{x_1, x_2, \dots, x_I\}$ . A continuous time Bayesian network  $\aleph$  over  $\mathbf{X}$  consists of two components: the first is an initial distribution  $P_{\mathbf{X}}^0$ , specified as a Bayesian network  $\mathcal{B}$  over  $\mathbf{X}$ , the second is a continuous time transition model specified as:*

- a directed (possibly cyclic) graph  $\mathcal{G}$  whose nodes are  $X_1, X_2, \dots, X_N$ ;
- a conditional intensity matrix,  $\mathbf{Q}_{X_n}^{Pa(X_n)}$ , for each variable  $X_n \in \mathbf{X}$ , where  $Pa(X_n)$  denotes the parents of  $X_n$  in  $\mathcal{G}$ .

Full details are provided in Villa and Rossetti (2014).

#### Continuous time Bayesian network classifiers

The continuous time Bayesian network model has been exploited to perform classification, a basic task in data analysis that assigns a class label to instances

described by a set of values, which explicitly represent the evolution in continuous time of a set of random variables  $X_n$ ,  $n = 1, 2, \dots, N$ . These random variables are also called attributes in the context of classification.

**Definition 4** *Continuous time Bayesian network classifier (CTBNC)*, (Stella and Amer, 2012). A continuous time Bayesian network classifier is a pair  $\mathcal{C} = \{\mathfrak{N}, P(Y)\}$  where  $\mathfrak{N}$  is a CTBN model with attribute nodes  $X_1, X_2, \dots, X_N$ , class node  $Y$  with marginal probability  $P(Y)$  on states  $Val(Y) = \{y_1, y_2, \dots, y_K\}$ , and  $\mathcal{G}$  is the graph, such that:

- $\mathcal{G}$  is connected;
- $Pa(Y) = \emptyset$ , the class variable  $Y$  is associated with a root node;
- $Y$  is fully specified by  $P(Y)$  and does not depend on time.

In this paper we focus on the continuous time version of two popular classifiers: the naïve Bayes, and the tree augmented naïve Bayes, described in Friedman et al. (1997). The first is the simplest classifier in which all the attributes  $X_n$  are conditionally independent given the value of the class  $Y$ . This assumption is represented by its simple structure depicted in Figure 4.5(a) where each attribute (leaf in the graph) is only connected with the class variable (root in the graph). Since the conditional independence assumption is often unrealistic, a more general classifier has been introduced in order to capture the dependencies among attributes. These dependencies are approximated by using a tree structure imposed on the naïve Bayes structure as shown in Figure 4.5(b).

### Learning in the MapReduce framework

We present the learning algorithms for continuous time Bayesian network classifiers in the MapReduce framework. Since structural and parameter learning rely on the computation of the sufficient statistics, we present one map function and one reduce function for both tasks. The primary motivation of using this programming model is that it simplifies large scale data processing tasks allowing programmers to express concurrent computations while hiding low level details of scheduling, fault tolerance, and data distribution (Dean and Ghemawat, 2008).

MapReduce programs are expressed as sequences of map and reduce operations performed by the mapper and the reducer respectively. A mapper takes as input parts of the dataset, applies a function (e.g., a partition of the data),

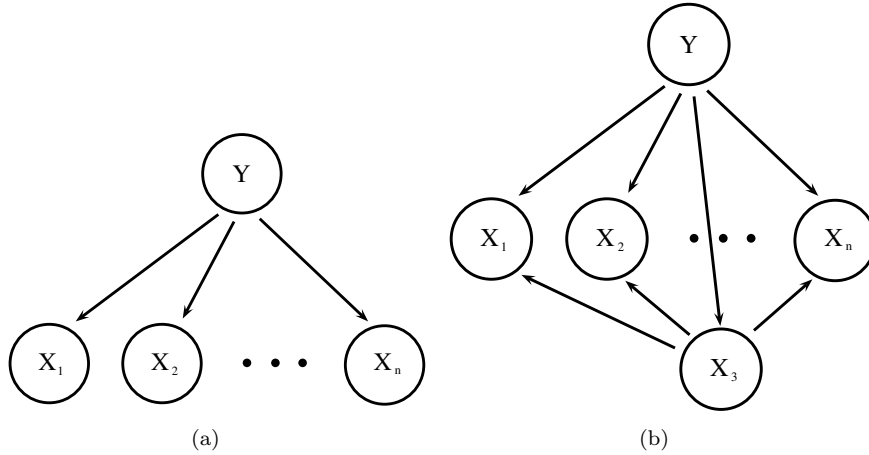


Figure 4.5: An instance of a continuous time naïve Bayes classifier (a) in which all the attributes  $X_n$  are conditionally independent given the value of the class  $Y$ ; and an instance of a continuous time tree augmented naïve Bayes classifier (b) in which the dependencies among attributes are approximated by using a tree structure imposed on the naïve Bayes structure.

and produces as output key-value pairs, while a reducer takes as input a list indexed by a key of all corresponding values and applies a reduction function (e.g., aggregation or sum operations) on the values. Once a reducer has terminated its work, the next set of mappers can be scheduled. Since a reducer must wait for all mapper outputs, the synchronization is implicit in the reducer operation, while fault tolerance is achieved by rescheduling mappers that time out.

The design of the learning algorithms is based on some basic patterns used in MapReduce (Lin and Dyer, 2010). The main idea is to exploit the peculiarities of the continuous time Bayesian network classifiers to parallelize the operations of structural and parameter learning. Through appropriate structuring of keys and values it is possible to use the MapReduce execution framework to bring together all the pieces of data required to perform the learning computation. In our case, the key-value pairs are constructed in order to encode all the information relevant for the description of the classifier, i.e., the marginal probability of the class, the structure, and the parameters associated with the structure.

We use the strips approach introduced by Lin (2008) to generate the output keys of the mapper, instead of emitting intermediate key-value pairs for each

interval. The mapper emits key-value pairs with text as keys and corresponding maps as values. The MapReduce execution framework guarantees that all associative arrays with the same key will be brought together in the reduce step. This last phase aggregates the results by computing the sufficient statistics and the estimation of the parameters of the conditional intensity matrix and of the Bayesian score. It is possible to further increase the performances by means of the use of combiners.

The main task of the map function is to count the transitions and the relative amount of time in the fully observed J-evidence-stream of each variable  $X_n$  given the instantiation  $pa(X_n)$  of its parents  $Pa(X_n)$ . In the case of structural learning, every possible combination of parents for each node must be computed subject to the constraint of the tree structure, while in the case of parameter learning the structure  $\mathcal{G}$  is defined as input. The task of the reduce function is to provide the basic elements for the description of the classifier, namely the class probability and the conditional intensity matrices.

As in Basak et al. (2012), we have tested the correctness of the algorithms by comparing the results generated by MapReduce against sequential versions.

#### 4.4.2 Experiments

We tested the proposed software in the parameter learning of a continuous time naïve Bayes classifier and in the structural learning of a continuous time tree augmented naïve Bayes. We made three types of experiments changing the dataset size, the number of Hadoop nodes, and the number of attributes. We compared the speedup of the proposed software versus the sequential version of the algorithm described in Stella and Amer (2012). The dataset is composed of a text file containing fully observed J-evidence-streams. These streams concern to high frequency transaction data of the Foreign Exchange market (Villa and Stella, 2014). Our tests are performed using M1 Large instances of Amazon EMR, while the training and output data are stored in Amazon S3.

##### Increasing the dataset size

In the first experiment, we measure the performance of the MapReduce algorithm in the case of parameter learning of a continuous time naïve Bayes classifier. We use 1 Master instance and 5 Core instances against the sequential algorithm using only one instance. The dataset consists of 1 binary class

attribute and 6 binary attributes. We increase the dataset size using 25K to 200K trajectories with step of 25K training samples to learn each classifier.

Figure 4.6(a) illustrates the learning time compared to the dataset size. The figure shows the time taken by the algorithms and the regression lines which interpolate the data points. Intuitively, the increase of the size of the training samples leads to increased training time because the MapReduce implementation has a computational overhead, which with a little data led to bad performance. Figure 4.6(a) shows that the MapReduce algorithm performs better than the sequential algorithm also with the smallest dataset, but when the number of trajectories increases, the gap between the two algorithms starts growing.

Figure 4.6(b) illustrates the speedup between the sequential and MapReduce algorithms. The points were calculated according to this equation:  $S_p = T_s/T_{mr}$ , where  $T_s$  is the sequential time and  $T_{mr}$  is the MapReduce time. As the data size increases, the speedup grows quickly at the beginning, while it become more stable when the data size is already big enough. For example, with 200K trajectories we have a speedup of about 3. In order to better understand this trend, the figure illustrates the speedup between the two regression lines (theoretical). This line shows very well how the speedup behaves in this case with a logarithmic trend.

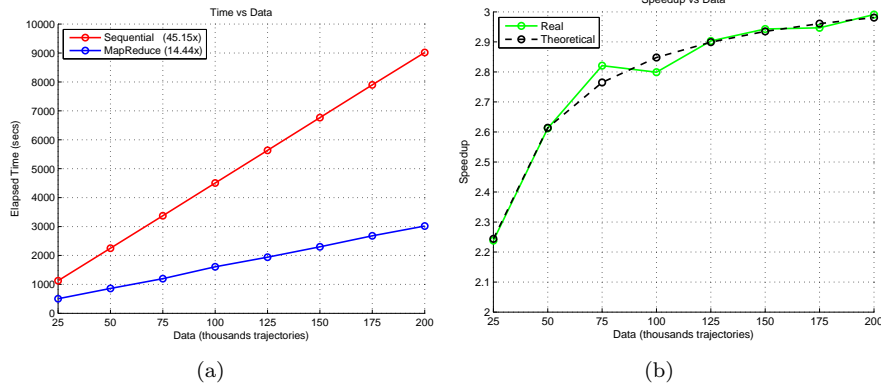


Figure 4.6: Chart (a) shows the elapsed time for the sequential and MapReduce algorithms with 5 nodes with respect to the data size in the case of CTBNC-NB learning. It also show the regression lines which represents the trend of the elapsed time used by the two algorithms. Chart (b) illustrates the real speedup between the sequential and MapReduce algorithms versus its expected theoretical value, in this case the speedup behaves with a logarithmic trend.

### **Increasing the Hadoop nodes and the number of attributes**

In the second experiment, we varied the number of Hadoop nodes to assess the parallel performance of the MapReduce algorithm in the case of parameter learning. As expected, increasing the number of Hadoop nodes significantly reduces the learning time, but this reduction is not equal to the ratio between the number of nodes. The real speedup is almost half the value of what theoretically expected, since the Hadoop overhead affect the MapReduce algorithm.

In the third experiment, we measured the performance of the MapReduce algorithm in the case of structural learning of a continuous time tree augmented naïve Bayes classifier varying the number of attributes. The learning time grows quadratically with respect to the number of attributes because we are testing the structural learning part of the algorithms. In this configuration, every possible parents combination given a variable is analyzed, for this reason we have two inner loops ranging over the number of variables for the map function). The quadratic coefficient of the polynomial regression for the MapReduce algorithm is 67.18 against 420.18 of the sequential version.

Full details of the experiments are provided in Villa and Rossetti (2014).

## **4.5 Conclusions**

We have shown how the application of topic models can provide a very effective way to integrate knowledge distributed over multiple sources in large content networks such as the Web. In this way distributed knowledge can both be clustered and organized into networks of higher-level concepts through topic-topic networks and navigated by creating object-object networks that link together formerly unrelated text documents, e.g. Web pages. The networks created can be exploited by a recommender system to discover useful links that connect user interests to new concepts that can enrich user knowledge.

The other contributions are about the temporal dimension in recommender systems. We have presented an interesting way to track hot topics in text corpora and a case study on a labor law dataset has been shown. Finally, we introduced continuous time Bayesian networks and the MapReduce framework, two important assets that can be exploited to deal with the temporal dimension and with Big Data.



## Chapter 5

# Analyzing User Reviews with Topic Models

In this chapter a contribution to the algorithm component is presented. User generated content in general and textual reviews in particular constitute a vast source of information for the decision making of users. This chapter explores different application scenarios for the topic model method to process these textual reviews in order to provide accurate decision support and recommendations as well as to build a basis for further analytics.

### 5.1 Introduction

Web 2.0 applications transformed the Internet from an information source to an opinion source (Dippelreiter et al., 2008; Schmallegger and Carson, 2008). Every piece of information, whether it is a product offered in an online store or a post in a social network, can be commented or rated in some way (Litvin et al., 2008; Xiang and Gretzel, 2010). In an economy heavily based on customer experience, such as tourism, individual decisions are strongly influenced by the written evidences of the experiences others already made - a.k.a. reviews (Pang and Lee, 2008; Zehrer et al., 2011; Ye et al., 2011).

From an IT perspective the automated exploitation of these opinions in order to provide advice and decision support led to tremendous research efforts in fields such as Machine Learning (ML) and Semantic Web (SemWeb). ML focuses on the construction and study of models that learn regularities and patterns from

known data in order to best possibly predict unknown data without a principal need to understand the semantics of the data. In contrast SemWeb focuses on capturing and modeling the semantics of the data on the web and tries to derive “unknown data” by principles of reasoning and logics. In this chapter we propose the application of the topic model method (Blei et al., 2003) to the task of analyzing user reviews. The topic model method is an approach that is clearly rooted in statistical ML and automatically extracts sets of terms with a coherent meaning (called topics) from document corpora such as reviews. Thus, although the method is agnostic of the semantics of the terms occurring in the documents themselves it automatically groups those terms where most presumably semantic ties exist between them. Therefore, it has the potential to, at least partly, bridge the gap between the two aforementioned principal research directions towards processing the data harvested from the Web.

The aim of this chapter is to explain the topic model method and describe its applicability to the tourism domain. Furthermore, we extend the method to derive interpretable user and item models that can be analyzed and exploited for deriving predictions and recommendations. In Section 5.2 we illustrate our contributions with a motivating example and continue with a more technical description. In Section 5.3 we present empirical evidence for the practical utility of our propositions by giving results from experimental evaluations, while in Section 5.4 the results obtained are discussed and future developments are planned.

## 5.2 The Topic-Criteria model

### 5.2.1 Motivating Example

The work presented in this chapter is motivated by the idea that reviews express different viewpoints or dimensions of the experience that a user made with an item and therefore extracting and interpreting these dimensions can be exploited to increase the accuracy of systems that automatically process these reviews in order to improve users’ experience on such platforms or to extract some form of business value from this review data. In the following we illustrate the propositions of this chapter with a fictitious example on reviews on accommodation services. Let’s assume Alice is a young woman who likes to travel around on a budget. She wrote the following two reviews on hotels she stayed at:

*Hotel 1: The hotel was right in the center of the city<sup>L</sup>, at walking*

*distance<sup>L</sup> from the city center<sup>L</sup>! Huge breakfast<sup>F</sup> with nice food<sup>F</sup>! Rating: 5*

*Hotel 2: I stayed in this hotel with my friends, the room was cheap, but the shower<sup>R</sup> was broken and the mattress<sup>R</sup> was very hard! Rating 2*

From these reviews we can get an idea of Alice’s taste and the “topics” she cares about, when staying in a hotel. In the literature a topic model (TM) (Blei, 2012) is a statistical machine learning approach that tries to extract thematic information from large corpora of natural language documents. Topics are defined as sorted lists of words with a coherent semantic meaning that can be extracted from documents. In Table 5.1 we provide examples for such lists of terms.

Table 5.1: An example of potential topics extracted from hotel reviews.

Topic Location	Topic Food	Topic Rooms	Topic Business
walking_distance	breakfast	Shower	executive_lounge
station	Service	bathroom	floor
city_center	restaurant	tub	executive_floor
metro	bar	bed	hilton
close	food	tv	conrad

Now Alice’s reviews can be mapped on these (pre-extracted) topics based on what she mentioned in the reviews. Note, that we are building user profiles solely based on the content of the user’s reviews that indicates what are the topics the user likes to talk about and ignore the specific rating values. In this small example Alice seems to care about the topics Location, Food and Rooms because the conditional probability of occurrence of the terms related to these topics are rather high in her reviews.

Another hotel (Hotel 3) received the following reviews from different users:

*User 1: The staff in the executive lounge<sup>B</sup> is very professional and the location<sup>L</sup> is very close<sup>L</sup> to the metro station<sup>L</sup>. Rating: 5*

*User 2: The room was nice, with a flat tv<sup>R</sup>, but the breakfast<sup>F</sup> was so poor! I didn’t have enough food<sup>F</sup>. Rating 3*

Now, given these reviews and ratings, we can compute scores for each topic and map items and users in the same “topic” space. Based on these two reviews the Hotel 3 might achieve a high rating w.r.t. the topics “Location” and “Business”, but only a low one for “Food” and “Rooms”.

How can the tourism domain benefit from applying this approach? First,

when Alice is looking for a hotel recommendation, the item profile of Hotel 3 can be matched against Alice’s profile in order to check if this item would be a plausible proposition. As Alice is, amongst others, interested in the topics “Food” and “Rooms” on which Hotel 3 is not scoring high on it might not be a formidable recommendation.

Second, the automated extraction of topics and the building of item profiles with scores on each topic is an opportunity to assess the strengths and weaknesses of each item as they are perceived by the users. This way item profiles based on collected reviews allow tourists to compare different service providers as well as provide a source for business analytics for management.

Third, based on analysis of what the user is writing we can estimate the rating the user would probably assign to the item. Such a scenario could either help to make rating values more consistent with reviews or enables a business analytics application to derive numeric scores from text, where no rating value is given (e.g. in posts on social networks or email feedback).

In this chapter we propose the Topic-Criteria (TC) model, which exploits the topic model method to extract latent features from textual reviews and discuss its application for several application scenarios in tourism. The difference between this approach and other approaches which use or extend topic model methods for Recommender Systems is that in this case the classic LDA is applied to process reviews and the extracted topics are exploited to define user and item profiles. This particular step makes the method very intuitive in its formulation as well as in the meaning of the computed information. Let us define  $R$  to be the set of ratings, and  $D$  to signify the set of textual reviews.  $r_{ij}$  is the rating given by user  $i$  to item  $j$ , while  $d_{ij}$  is its associated review. For simplicity, let  $R_i$  be the set of ratings given by user  $i$ , while let  $R^j$  be the set of ratings given to item  $j$ . The analogous notation is defined for reviews, i.e.  $D_i$  denotes the set of reviews given by user  $i$  and  $D^j$  reviews about item  $j$ . The probability of the topic  $z$  given the document  $d_{ij}$  is indicated with  $P(z|d_{ij})$ . Finally, we reserve the letter  $i$  to indicate users,  $j$  to indicate items and  $Z$  to define the number of topics. The user profile is constructed by aggregating the topic distributions of all the reviews written by the user, without considering the associated ratings. The main idea is that to profile a user we are only interested about what aspects of an, for instance, accommodation the user writes. Therefore, the user model consists of those topics that the user seems to care about in her/his reviews. The rating values are not needed for this purpose. The user profile is computed by aggregating the topic distributions of the user’s reviews, as shown in Figure

5.1. The user profile (UP) for user  $i$  is a numeric degree on each topic  $z$  (from 1 to  $Z$ ) that defines the relevance of topic  $z$  for user  $i$  (see Equation 5.1).

$$UP(i, z) = \frac{\sum_{d_{ij} \in D_i} P(z|d_{ij})}{|D_i|} \quad (5.1)$$

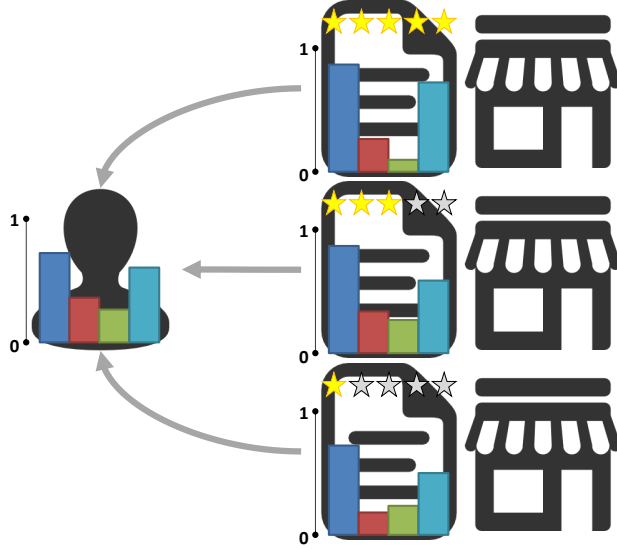


Figure 5.1: User profile creation from topic distributions.

Item profiles are built by using both: topic distributions and numeric ratings, because the topics signify the aspects the user cared about in her/his review and the rating values indicate how satisfied the user was with respect to these aspects. Thus, the main idea is that if an item has reviews that frequently mention a specific topic, we have to consider the ratings to understand if this topic is a strong or a weak point of this item. The item profile can be built as a numeric score from 1 to 5 for each extracted topic aggregating the topic distributions and the related ratings, as illustrated by Figure 5.2. As in the previous case, the item profile (IP) for item  $j$  can be computed as defined in Equation 5.2

$$IP(j, z) = \frac{\sum_{d_{ij} \in D_i} P(z|d_{ij}) * r_{ij}}{\sum_{d_{ij} \in D_i} P(z|d_{ij})} \quad (5.2)$$

Since user profiles define the interest of the users in different topics and item profiles indicate how well an item does with respect to each topic, the

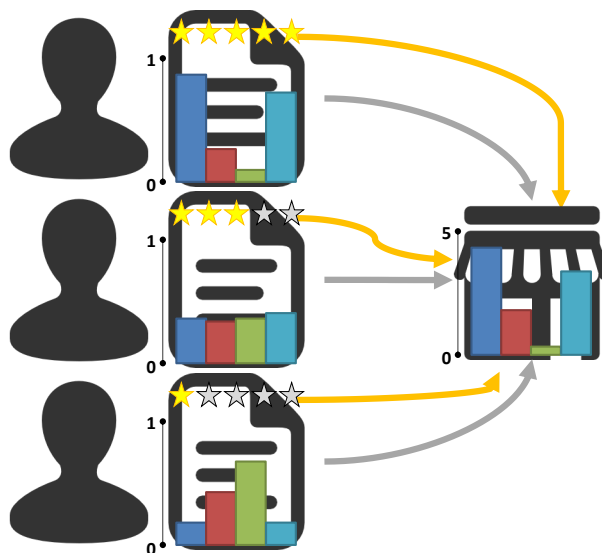


Figure 5.2: Item profile creation from topic distributions and ratings.

combination of both profiles should allow us to estimate a user rating for an unseen item. The match between a user and an item profile is computed by the sum of the products for each topic, as defined in Equation 5.3. In order to improve the prediction accuracy of the approach, a topic weight is added to assign more value to those topics that are more influential for the estimation of rating values.

$$r_{ij} = \sum_{z=1}^Z UP(i, z)IP(j, z)w_z \quad (5.3)$$

These weights are optimized by minimizing the loss function with the gradient descent approach, as shown in Equation 5.4. Note that  $\lambda$  is a regularization parameter that punishes more complex models in order to avoid data overfitting.

$$\min_w \sum_{r_{ij} \in R} \left( r_{ij} - \sum_{z=1}^Z UP(i, z)IP(j, z)w_z \right)^2 + \lambda \|w\|_F^2 \quad (5.4)$$

In this chapter we propose three different application scenarios for our approach and provide empirical evidence based on available data:

1. **Rating Prediction and Recommendation:** User profiles represent

the degree of interest of users in extracted topics. Item profiles express an item’s scoring on each topic. Thus, the match between both profiles indicates how appropriate an item might be for a user.

2. **Analytics and Interpretation:** The topic model method provides a natural characterization and interpretation of user and item profiles. When interpreting (selected) topics as item features or characteristics a system can transparently display to a user the model that is internally used for personalizing content. Furthermore, items can be compared to each other from several perspectives as if multi-criteria ratings from users would be known, where each item is assessed according to different dimensions such as quality of service, value for money, rooms, cleanliness or location (Jannach et al., 2014).
3. **Suggest Ratings for Review:** The proposed approach can also be exploited to suggest a rating given a textual review and a user profile. For instance, the system can propose a rating given what the user is writing in the review, or assess the coherence of the review and the rating given to the item.

## 5.3 Empirical evaluation

For assessing the proposed approach in the three scenarios two datasets were used: the *Yelp*<sup>1</sup> dataset and the *TripAdvisor*<sup>2</sup> dataset. The Yelp dataset is provided by Yelp for the *Yelp Dataset Challenge*<sup>3</sup> and it contains reviews and ratings given by users of the Yelp website to business activities, mainly restaurants. The TripAdvisor dataset (Jannach et al., 2014) was crawled from the popular website and it contains reviews about hotels in different cities. The TripAdvisor dataset contains also more fine-granular user feedback that not only encompasses an overall rating but also ratings on more specific dimensions such as value for money, cleanliness or rooms. In order to experiment with different levels of data sparseness (i.e. the share of unknown entries in the full user-item rating matrix) we identified data subsets that have at least n known ratings for each user and each item. This processing leads to the datasets described in Table 5.2.

---

<sup>1</sup>[www.yelp.com](http://www.yelp.com)

<sup>2</sup>[www.tripadvisor.com](http://www.tripadvisor.com)

<sup>3</sup>[www.yelp.com/dataset\\_challenge](http://www.yelp.com/dataset_challenge)

Table 5.2: Dataset summary.

	YELP-5-5	YELP-10-10	TA-3-3	TA-5-5
#Users	9382	3802	13048	1850
#Items	3733	2413	12342	1774
#Ratings	145735	101416	83395	14656
Sparsity	0.0042	0.0111	0.0005	0.0045

### 5.3.1 Scenario 1: Rating Prediction and Recommendation

The rating prediction accuracy was evaluated with the classic ML measure, the Root Mean Squared Error (RMSE). The proposed model was tested without considering topic weights (TC) and alternatively optimizing topic weights (TC-W). In the TC-W method we also kept track of the user average rating, subtracting it from the original rating in Equation 5.2 and adding it to the estimated rating in Equation 5.3. TC and TC-W were evaluated against three classic Collaborative Filtering (CF) algorithms: the K-Nearest Neighbor User Based (KNN-UB), the K-Nearest Neighbor Item Based (KNN-IB) and the Probabilistic Matrix Factorization (PMF). Neighborhood models, also known as memory-based models, are the most common approach to CF (Herlocker et al., 1999). In the user based case the idea is to suggest items which are liked by users with similar tastes, while in the item based one the system recommends items similar to the items liked by the user (Sarwar et al., 2001). Probabilistic Matrix Factorization (Mnih and Salakhutdinov, 2007) is a model-based approach which tries to factorize the user-item matrix with a probabilistic perspective. Although several extensions of this model have been developed, the classic PMF is still a good baseline for the CF.

Table 5.3: RMSE values for the different methods on the four datasets.

Algorithm	YELP-5-5	YELP-10-10	TA-3-3	TA-5-5
KNN-IB	1.0709	1.0249	1.0531	0.9601
KNN-UB	1.1088	1.0424	1.0715	<b>0.9447</b>
PMF	1.0956	1.0389	<b>1.0373</b>	0.9946
TC	1.0706	1.0247	1.0625	0.9719
TC-O	<b>1.0599</b>	<b>0.9955</b>	1.0916	0.9776

Table 5.3 shows RMSE values for the baselines and the two TC models. The proposed approach achieves a RMSE comparable to the classic CF approaches: on the YELP datasets the TC models achieve lower RMSE values than CF approaches, while on the Tripadvisor datasets they are not able to perform



so well. However, the advantage of the approach does not solely lie in being as accurate as or slightly better than other CF approaches, but in employing user models that are only based on review content and therefore offer ways to be made transparent to users or better explain users why a specific item is recommended.

### 5.3.2 Scenario 2: Analytics and Interpretation

The approach can be used to explain which topics are important to users or to analyze where the relative strengths and weaknesses of items are lying when compared to each other. Based on the Tripadvisor dataset we identified exemplary topics that can be related to the dimensional rating values. For instance, in case of low rating value for the “cleanliness” dimension, the topics associated with that dimension can provide hints about the reasons. On the other hand, in case of high ratings we can explore which topics the users particularly appreciated. To find which topics are important for a particular rating dimension we performed a non-parametric test to compare the overall rating distribution and the rating distribution of the top-k reviews strongly associated with a topic. A test rejecting the null hypothesis means that the presence of the topic has a positive (or negative) impact on the rating. We applied a two-sample Kolmogorov-Smirnov test with significance level equal to 5%.

Table 5.4: Illustrative examples for selected Topics related to multi-criteria dimensions.

<b>Topic related to..</b>	
<b>Cleanliness in reviews on Orlando hotels</b>	<b>Business in reviews on New York hotels</b>
dirty mold bugs smelled smell filthy carpet musty stained disgusting bed_bugs black mildew moldy stains bites dust musty_smell refund	internet_access wireless_internet business_center computers free_wireless business boarding gym center print free_internet_access

Table 5.4 shows two illustrative examples of topics strongly correlated with a rating dimension. For this analysis we split reviews based on a specific destination for the purpose of reducing the fragmentation of topics. Other subsamples can be extracted dividing the reviews by specific hotels or other tourism items or by user segment such as “senior couples” or “families on a budget” in analogy to Jannach et al. (2014).

### 5.3.3 Scenario 3: Suggest Ratings for Review

The third scenario refers to the ability of our approach to predict a rating given the textual review, for instance, interactively when the user just entered the review text. Such approach can be used to propose a rating right after the user finished writing his/her review. To estimate the rating of a review, topics are extracted from the text and the topic distribution is multiplied with the item profile in order to estimate the rating. Since the CF methods considered in Scenario 1 cannot predict ratings based on textual input, we only compute accuracy results RMSE for our proposed methods (see Table 5.5). It is interesting to notice that the RMSE values are only slightly higher than the ones already obtained for Scenario 1. The small difference can be explained by the fact that a user profile is more informative (aggregates several reviews) than the topic distribution of the single review and therefore it better represents user’s interests. Even in this Scenario the TC method without optimized weights achieves a lower RMSE on the Tripadvisor datasets.

Table 5.5: RMSE values for the Scenario 3.

	YELP-5-5	YELP-10-10	TA-3-3	TA-5-5
TC	1.0718	1.0258	<b>1.0663</b>	<b>0.9783</b>
TC-O	<b>1.0600</b>	<b>0.9976</b>	1.0932	0.9826

## 5.4 Conclusions

This chapter explores the application of the topic model method in the tourism domain. The chapter’s contribution is twofold; first, a novel Topic-Criteria is proposed that includes a novel way to model users based on their usage of different topics in their textual reviews that discloses their preferences or the criteria which they deem to be important for assessing a tourism product or service. In addition, also items are modeled by a rating for each topic that indicates how well they are performing with respect to each topic in the eyes of their customers. Such an approach shows not only the potential to increase the accuracy of different prediction mechanisms due to the exploitation of the content from textual reviews, but it also promises to deliver additional semantics and meaning when analyzing the big heaps of data that are continuously collected in present time. Second, we also contribute empirical evidence for the practical relevance of the proposed technical approach by describing the three

---

usage scenarios: Rating Prediction and Recommendation, Analytics and Interpretation and Suggest Ratings for Review and exploiting available datasets to compute the prediction accuracy of the approach. It remains to note, that the presented results constitute only a first step of our work agenda that will include hybridizing the method with other well-known techniques and developing the application scenarios further. Another possible extension of this work can be the application of the supervised LDA machine learning technique (Mcauliffe and Blei, 2008) selecting reviews as learning input based on user features or rating values. In this way the identification of topics will be guided by predefined criteria such as rating dimensions and will therefore be even better interpretable. Finally, another extension can also be the joint application of topic model and sentiment analysis (Lin and He, 2009) in order to extract topics explicitly based on the sentiment.



## Chapter 6

# Explaining Latent Factors with Topic Models

In this chapter a contribution to the interface component is presented. Latent factor models have been proved to be the state of the art for the Collaborative Filtering approach in a recommender system. However, latent factors obtained with mathematical methods applied to the user-item matrix can be hardly interpreted by humans. In this chapter we exploit topic models applied to textual data associated with items to find explanations for latent factors. Based on the MovieLens dataset and textual data about movies collected from Freebase we run a user study with over a hundred participants to develop a reference dataset for evaluating different strategies towards more interpretable and portable latent factor models.

### 6.1 Introduction

Recommender systems help users to identify items of interest from large collections, such as, for instance, books, movies or tourism services (Jannach et al., 2010; Ricci et al., 2011). Matrix factorization (MF) techniques that reduce the dimensionality of the input space by identifying latent factors have become popular methods. However, the *portability* of such parameterized models is rather low, i.e. the extent to which a model trained on a specific dataset, for instance, from the movie domain can be applied to another dataset with new movies and/or new users. Furthermore, it is hard to explain users how

a specific recommendation has been derived and why it matches to their presumed preferences when following a *white-box* explanations strategy (Friedrich and Zanker, 2011), i.e. their *interpretability* is rather low as opposed to explicit knowledge-representation formalisms such as constraints or logic.

The goal of this chapter is therefore to explore ways towards attaching semantics to the latent factors of a matrix factorization model, such that (parts of) these models can be applied to new users (i.e. users without or with only few known ratings) or can be exploited for explaining their recommendations. Typically, the parameters of matrix factorization models are determined based on single a optimization criterion such as minimized error rates. However, developing algorithms that would focus on a model’s portability and interpretability requires an appropriate data set.

This chapter therefore constitutes a first step towards this direction by making the following contributions:

1. Acquisition of a dataset with unary ratings via a user study that can serve as ground truth for the development of portable and interpretable MF models.
2. Empirical results on identifying topic related factors and predicting topical interests of participants in our user study.

The chapter is organized as follows. First, research questions and design are presented in Section 6.2. Section 6.3 details the implementation of applied techniques, while Section 6.4 gives empirical results. Finally, Section 6.5 discusses our findings.

## 6.2 Research Questions and Design

The goal of this work is to contribute to the development of better interpretable and portable latent factor models. Different application scenarios in the field of business intelligence and analytics would benefit from research progress into this direction, however this chapter focuses on the domain of movie recommendation as a first step. This work is based on the assumption that the textual description of movies, i.e. the movies’ content, explains at least some share of a user’s movie preferences. More specific research questions guiding this work are:

1. RQ1: Given the latent factors from a factorized ratings matrix, how can those latent factors be identified that are related to the items’ content

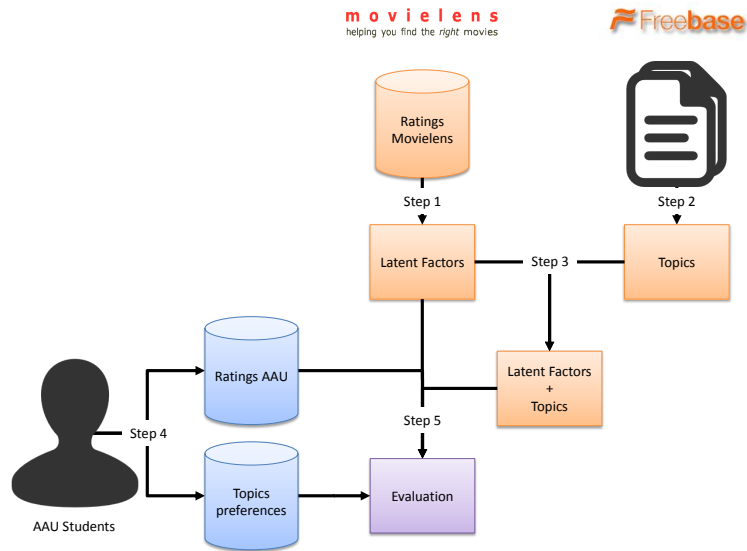


Figure 6.1: Design of the Research Approach.

description?

2. RQ2: Do MF models, consisting of better interpretable factors, produce more accurate topic predictions?

As an initial approach towards addressing the aforementioned research questions the authors initiated the collection of a dataset that can serve as ground truth for further research. Based on the set of movies rated in the MovieLens 1M dataset we started a user study with Alpen-Adria University students in order to collect users' preference information about the aforementioned movies and users' preferences with respect to movies' content. Figure 6.1 roughly outlines the research design. From MovieLens 1M dataset with around million ratings latent factors capturing common variances in the data have been derived (Step 1). Second, from movies' content description the most predominant topics are extracted (Step 2). After that, latent factors and topics are combined to address the research questions (Step 3). To evaluate the research questions a dataset is collected, where users not only disclose their movie preferences but also select those topics that best describe their interest in movies (Step 4). Finally, the research questions are evaluated against the collected data (Step 5).

## 6.3 Implementation

Let  $M^{ml}$  be the number of users in the MovieLens dataset. Furthermore, let  $N$ ,  $F$  and  $K$  be respectively the number of movies, latent factors and topics.

### 6.3.1 MovieLens Non-negative Matrix Factorization (Step 1)

Numerical experiments have been performed on the *MovieLens 1M* dataset. This dataset contains 1,000,209 ratings from 6,040 users and 3,706 different movies. Ratings are integer values from an ordinal scale ranging from 1 to 5, where 1 is the worst and 5 is the best feedback. A preprocessing step was necessary to transform this data into unary user feedback representing only *likes* judgements. Based on the average rating value for each individual user only 4 and 5 ratings that are equal or greater than the user's average rating score were transformed into unary *like* statements. After removing users and movies with no single remaining rating value the dataset consisted of 6,038 users and 3,086 movies. In addition content information about movies was required as will be explained in the next subsection. Therefore, the number of considered movies was actually further reduced to  $M^{ml}=3,077$ .

This transformed MovieLens dataset has been used to build a matrix  $R^{ml}$ . The *Alternating Least Squares non-negative matrix factorization* (Paatero and Tapper, 1994) algorithm was applied to decompose  $R^{ml}$  into  $U^{ml}$  and  $V$ , such that  $R^{ml} \approx (U^{ml})^T V$ . These matrices contains non-negative real values, which indicate the importance of a latent factor for a user or a movie.

### 6.3.2 Topic Extraction from ML Movies (Step 2)

The dataset to be used for topic extraction from ML movies was built by exploiting *FreeBase*. Unstructured data about more than three thousand movies matching with FreeBase items was collected and movies that could be matched with FreeBase items were removed.

As the textual description of movies contains proper names about actors, directors, characters, organizations and places, named entity recognition was applied in a preprocessing step. The identified named entities were not split into tokens but considered as single terms in the topic extraction step. We also addressed the co-reference problem by exploiting the aliases offered by FreeBase to transform different instances of the same entity into a standard form. As an



example, all the occurrences of “Walt Disney”, “Walter Elias Disney”, “Mr. Disney” and so on, were transformed into the standard entity “Walt\_Disney”.

The choice of the optimal number of topics in this range, was guided by the consideration that a high number of topics could bother the user in the evaluation phase. For this reason we set the number of topics equal to  $K = 30$ , which seemed to be a good compromise between topics’ granularity and the cognitive effort of users in order to select the appropriate topics. The LDA algorithm provides for every movie its’ probability distribution over topics. This set of probability values can be represented as a vector  $\theta_j \in \mathbb{R}^K$ . This vector is strictly non-negative,  $0 \leq \theta_{jk}$  per  $k = 1, \dots, K$ , and sum to 1,  $\sum_{k=1}^K \theta_{jk} = 1$ . All the probability distributions of movies have been organized as rows of the matrix  $\theta$ , which by construction is a *stochastic matrix*.

### 6.3.3 Combine latent factors and topics (Step 3)

The matrix  $V$  contains non-negative real values, which measure the strength between movies and latent factors. We divide each element of the matrix by the corresponding row sum to transform it into a stochastic matrix. The interpretation of such transformation is that a row of the normalized  $V$  matrix expresses the probability distribution of the movies given a latent factor.

The matrices  $V$  and  $\theta$  are both stochastic matrices and could now be combined to associate latent factors with topics.

$$\Psi = V\theta \tag{6.1}$$

The element  $\Psi_{fz}$  of the matrix  $\Psi$  is the probability of the topic  $z$  for the latent factor  $f$  (Equation 6.2).

$$P(z|f) = \sum_{j=1}^N P(z|j)P(j|f) = \Psi_{fz} \tag{6.2}$$

The computation of matrix  $\Psi$  is based on latent factors. However, it is obvious that not all latent factors can be explained by textual features. Therefore, we ranked latent factors with respect to topic sparsity. The rationale for this being that a factor which is strongly associated with few topics is probably the one which can be fruitfully explained by textual features. On the opposite, if a factor has a uniform distribution over all topics, it is very likely it has a low association degree with textual features. To evaluate the sparsity of the

distribution of latent factors over topics we used the Gini index, as described in Hurley and Rickard (2009). The general formula of the Gini index for a vector  $\mathbf{c}$ , in which the components are in increasing order  $c_1 \leq c_2 \leq \dots \leq c_N$ , is defined in Equation 6.3. Such index ranges in the  $[0..1]$  interval, where 0 indicates an homogeneous distribution and 1 indicates a concentration on only a single value.

$$S(\mathbf{c}) = 1 - 2 \sum_{k=1}^N \frac{c_k}{\|\mathbf{c}\|_1} \left( \frac{N - k + \frac{1}{2}}{N} \right) \quad (6.3)$$

The Gini index for every factor was computed and latent factors were ranked based on such values.

### 6.3.4 Collection of Survey Data (Step 4)

To evaluate how well the latent factors' distribution over topics helps in explanations, we collected user data with a specifically designed and implemented web application. We invited movie aficionados from *Alpen-Adria University's (aau)* mailing list to browse the platform and select their favorite movies among those forming the MovieLens dataset, as shown in Figure 6.2. Participants are

popularity	year	title	like
33	2000	Gladiator	<input type="checkbox"/>
70	2000	X-Men	<input checked="" type="checkbox"/>
81	2000	High Fidelity	<input type="checkbox"/>
95	2000	Chicken Run	<input checked="" type="checkbox"/>
104	2000	Erin Brockovich	<input type="checkbox"/>
129	2000	The Patriot	<input checked="" type="checkbox"/>
154	2000	Almost Famous	<input type="checkbox"/>
155	2000	Mission: Impossible II	<input type="checkbox"/>
244	2000	The Perfect Storm	<input type="checkbox"/>
259	2000	U-571	<input type="checkbox"/>
266	2000	Meet the Parents	<input type="checkbox"/>
276	2000	Frequency	<input type="checkbox"/>
315	2000	Shanghai Neon	<input type="checkbox"/>
366	2000	Best in Show	<input type="checkbox"/>
479	2000	The Cell	<input type="checkbox"/>
508	2000	Mission to Mars	<input type="checkbox"/>
515	2000	Boiler Room	<input type="checkbox"/>
516	2000	The Whole Nine	<input type="checkbox"/>
520	2000	American Psycho	<input type="checkbox"/>
539	2000	Wonder Boys	<input type="checkbox"/>
541	2000	Nurse Betty	<input type="checkbox"/>
573	2000	Scarv Movie	<input type="checkbox"/>
581	2000	Keeping the Faith	<input type="checkbox"/>
591	2000	Titan A.E.	<input type="checkbox"/>
599	2000	Pitch Black	<input type="checkbox"/>

Figure 6.2: Likes collection web interface.

incentivized to disclose their true preference as there is a lottery where participants can win DVDs of movies they like. After the movie selection phase, the user is asked to select which topics, under his/her judgment, best describe their movie preference. Topics are represented as tag clouds, with the dimension of

the tags being proportional to the probability value of topics over words. The collected poll data allowed to extract two binary matrices,  $R^{aa_u}$ , and  $T$ , which contain the selected movies and topics for  $M^{aa_u}$  users. We define the collected dataset as the AAU dataset.

### 6.3.5 Evaluation (Step 5)

Matrix  $R^{aa_u}$  allows to map every user to the latent factor space which has been computed from the MovieLens dataset (Equation 6.4).

$$U^{aa_u} = V(R^{aa_u})^\top \quad (6.4)$$

The matrix  $U^{aa_u}$  projects a user into the factor space of the MovieLens dataset. This mapping can be exploited for two purposes: predict ratings for unseen items and associate every user with topics.

The matrix  $U^{aa_u}$  can be multiplied by the matrix  $V$  to predict movies for users, which leads to the matrix  $\hat{R}$  (Equation 6.5).

$$\hat{R}^{AAU} = (U^{aa_u})^\top V \quad (6.5)$$

A threshold value can be applied to  $\hat{R}^{AAU}$  to compute the recommendation lists.

Once a user is mapped to the latent factor space, the matrix  $\Psi$  can be used to explain his/her preferences in term of topics. Similarly to what we have already done before, we normalize the matrix  $U^{aa_u}$  by dividing each element by the column sum. Each column of  $U^{aa_u}$  can be interpreted as the user probability distribution over latent factors. The product of matrices  $U^{aa_u}$  and  $\Psi$  lead to the matrix  $\hat{T}$ , which represents user preferences in terms of topics expressed through latent factors (Equation 6.6).

$$\hat{T} = (U^{aa_u})^\top \Psi \quad (6.6)$$

This matrix can be compared with the matrix  $T$ , which contains real user preferences about topics expressed on our poll platform.

## 6.4 Evaluation

### 6.4.1 Evaluation Methods

Predicting if a user likes a movie or has a preference for a specific topic are binary decision problems. Therefore, we compute Precision and Recall values for all levels of recommendation list lengths to evaluate the quality of the prediction models. The *Area Under Precision-Recall Curve* aggregates these Precision and Recall values and should not be mistaken as AUC, that is the area under the curve in ROC space, where the False Positive Rate is plotted against the True Positive Rate. See Davis and Goadrich (2006) for a discussion on the relationship between both curves. However, Precision and Recall do not consider the rank of correctly predicted items in a recommendation list. Therefore, *Normalized Discounted Cumulative Gain* measures the ranking quality and normalizes for different lengths of recommendation lists. Such measures are described in Subsection 2.4.1.

### 6.4.2 Datasets

Table 6.1 shows summary statistics for the MovieLens 1M dataset and the AAU dataset.

Table 6.1: Summary statistics for the MovieLens 1M and the AAU datasets.

	ML	AAU
Users	6,038	107
Likes	491,072	14,644
Min	1	14
Max	1,092	601
Average	81.33	136.86

In order to probe for the portability of the latent factor model from the MovieLens 1M dataset to the participants of our user study we computed *AUPR* on the MovieLens 1M dataset applying 10 folds-cross validation on users, where the all-but-one approach was applied to the validation fold. In the next step, the MovieLens MF model was applied to the AAU dataset, where the full MovieLens 1M dataset was used for training and all AAU data for testing. As can be seen from Table 6.2 *AUPR* values are smaller for the AAU dataset compared to the MovieLens, as we would expect due to the different user populations (online movie enthusiasts from the US vs. Austrian university students) with different

demographics and collected approximately 12 years later.

Table 6.2: *AUPR* of MF with different factors on MovieLens 1M and AAU dataset.

Factors	<i>MovieLens</i>	<i>AAU</i>
20	0,3737	0,2879
30	0,3808	0,2936
40	0,3786	0,2997
50	0,3724	0,2962
60	0,3661	0,2890
70	0,3590	0,2882
80	0,3523	0,2839
90	0,3477	0,2852
100	0,3423	0,2793

This assumption that more ratings per users lead to better accuracy results for AAU users when applying the MovieLens latent factor model is supported by the following observation. We split the dataset into three strata based on the *AUPR* error rate for each user, where Stratum 3 contains one third of all users with the highest *AUPR* values and Stratum 1 with the lowest. Table 6.3 gives descriptive statistics on these three subset of the AAU data. It is evident that the number of known *likes* drives accuracy and thus division into the three strata.

Table 6.3: Descriptive statistics for the three strata of the AAU dataset based on *AUPR*.

	Stratum 1	Stratum 2	Stratum 3
Users	36	36	35
Likes	2,131	4,571	7,942
Min	14	47	29
Max	188	335	601
Average	59.19	126.97	226.81

### 6.4.3 Results

As outlined in subsection III our research follows the goal to identify latent factors that are related to topics. Consequently, we present results on topic prediction and show empirically how latent factors are related to topics. As a first step into this direction we applied the Gini index (Hurley and Rickard, 2009) to rank latent factors by their nonuniformity of their distribution over

topics. For instance, Table 6.4 exemplifies Factor 11 as a good match between latent factor and topics. The table gives the best three topics and the best 10 movies for Factor 11 and the three most probable topics for each movie according to the topic model approach. It becomes obvious that a user scoring high with respect to Factor 11 should have an interest in cartoons and Disney productions. Knowing this relationship between latent factors and topics would allow a recommendation system to explain its users why a specific item has been proposed or could enable recommendations to users without any rated items but known topical preferences.

Table 6.4: Factor 11

Topics		
<b>3</b>	<b>4</b>	<b>7</b>
animated	script	musical
disney	production	adapted
feature	success	awards
walt_disney	budget	picture
classics	action	version
animation	reviews	selected
walt	filming	broadway
musical	wrote	culturally
walt_disney_pictures	television	documentary
video	character	german
Movies	Topics	
Lion King, The (1994)	<b>3</b>	6 <b>4</b>
Aladdin (1992)	<b>3</b>	<b>4</b> <b>7</b>
Snow White and the Seven Dwarfs (1937)	<b>3</b>	<b>7</b> 11
Little Mermaid, The (1989)	<b>3</b>	6 5
Cinderella (1950)	<b>3</b>	28 9
Mary Poppins (1964)	<b>3</b>	<b>7</b> 6
Lady and the Tramp (1955)	<b>3</b>	12 9
Fantasia (1940)	<b>3</b>	5 <b>4</b>
Dumbo (1941)	<b>3</b>	5 19

In order to investigate if the Gini coefficient is apt for distinguishing between topic-related latent factors and other latent factors we ranked factors according to the Gini index and split them into four quartiles. Note, that based on the sensitivity analysis of accuracy results we decided to use the MF model with 40 latent factors ( $F = 40$ ). When using these latent factors from the MovieLens dataset to predict the topic preferences of our AAU users, we hypothesize that

1. ...applying only higher ranked latent factors based on the Gini-coefficient

should lead to higher accuracy in topic preference prediction.

2. ...for users with higher *AUPR* results for movie prediction also topic prediction should be more accurate.

As can be seen from Table 6.5 both hypotheses are supported by the results of our offline computations on the AAU dataset. nDCG is consistently higher when exploiting only the top-ranked latent factors according to the Gini index (Hypothesis 1). Second Stratum 3 scores always higher than Stratum 1 and 2 (Hypothesis 2). Stratum 3 consists only of the top tier of study participants according to *AUPR* accuracy when predicting their movie preference.

Table 6.5: nDCG on topic predictions for the three strata with all latent factors and with the top 75%, the top 50%, and the top 25% of topic-related factors.

	All LF			Top 75% LF		
	S1	S2	S3	S1	S2	S3
1	0.0833	0.1111	0.2571	0.1944	0.1667	0.2571
2	0.1667	0.2639	0.4000	0.2083	0.3194	0.4714
3	0.2233	0.3005	0.4582	0.2816	0.3294	0.4577
4	0.2711	0.3413	0.4758	0.3117	0.3573	0.4616
5	0.2915	0.3553	0.4593	0.3141	0.3845	0.4745
6	0.3071	0.3623	0.4626	0.3275	0.3881	0.4836
7	0.3177	0.3638	0.4600	0.3394	0.3854	0.4711
8	0.3250	0.3843	0.4579	0.3456	0.3962	0.4662
9	0.3410	0.3909	0.4610	0.3617	0.4133	0.4692
10	0.3588	0.3973	0.4672	0.3806	0.4199	0.4861
	Top 50% LF			Top 25% LF		
	S1	S2	S3	S1	S2	S3
1	0.2778	0.2222	0.2571	0.5278	0.5278	0.4571
2	0.3611	0.3611	0.5286	0.3472	0.4167	0.5429
3	0.3466	0.3744	0.5012	0.3293	0.4367	0.5805
4	0.3419	0.3782	0.5164	0.3317	0.4163	0.5512
5	0.3513	0.3894	0.5019	0.3422	0.4124	0.5325
6	0.3539	0.4038	0.5153	0.3403	0.4199	0.5289
7	0.3642	0.4037	0.5044	0.3571	0.4234	0.5306
8	0.3712	0.4066	0.4896	0.3735	0.4331	0.5321
9	0.3952	0.4256	0.5104	0.3864	0.4423	0.5319
10	0.4138	0.4379	0.5150	0.3942	0.4498	0.5335

#### 6.4.4 Limitations and Future Work

The empirical results contribute to answering the first research question formulating in subsection III. However, as suggested in the chapters' title, these

results constitute only a first step towards explaining latent factors with topics and additional work will have to be pursued. For instance, we will apply differently parameterized MF models with varying degrees of interpretable factors to new user situations in order to research on RQ2. Furthermore, we plan to consider additional external knowledge sources besides movies' content description. Finally, varying the number of topics extracted from the content description and presenting the generated explanations to users is part of the authors' future work.

## 6.5 Conclusions

The chapter constitutes a first step towards (partly) explaining latent factors that are derived from a factorization of a user-item ratings matrix with topics learned from content descriptions. An interesting dataset for further development has been contributed and the Gini index was proposed to rank latent factors according to their nonuniformity of their probability distribution over topics. An evaluation of the dataset confirmed our hypothesis that higher-ranked latent factors lead to more accurate predictions of topical interests of users and thus is a reliable indicator for the aptness of the Gini index for this purpose.



## Chapter 7

# Validity of Accuracy Measurements for Offline Recommendations

In this chapter a contribution to the evaluation component is presented. Offline evaluation protocols that exploit Machine Learning techniques have been widely used in recommender systems' literature. However, their external validity has been investigated and criticized, especially in the last years. In this chapter we present, analyze and summarize the results of a user study comparing offline accuracy measurements computed on a 100 users dataset with answers provided by the same users in an online study. Such analysis shows that inconsistencies between offline and online evaluations exist, especially when the evaluation is performed on all items. On the other hand, statistics computed on long tail items, i.e. not popular items, seem to be a better indicator to rank algorithms with respect to their accuracy.

### 7.1 Introduction

While the first seminal paper proposing a personalized recommendation mechanism of news items dates back over 20 years (Resnick et al., 1994), we saw a thriving interest in recommendation systems both from academia as well as industry during the past decade (ACM, 2014). Besides traditional application

domains such as e-commerce, cultural goods or social media quite novel domains such as lifestyle, financial services or decisions in agriculture are starting to show up. From a methodological viewpoint research contributions focus mainly on novel methods, that mainly come with the promise of more accurately identifying relevant content in the diverse application domains. For instance, the survey of Jannach et al. (2012) gives evidence that validation of recommender systems research focuses mainly on offline evaluation scenarios. Thus, according to the paradigm of Machine Learning a predictive model is trained on a subset of the available data and it is optimized in order to correctly predict the withheld portions of the dataset. Several works have been focusing on the methodological aspects of this approach in recommender systems research. For instance, a recent contribution of Said and Bellogín (2014) identified several inconsistencies when recommendation methods and offline evaluation practices are compared across different framework platforms. They report that when comparing accuracy results for the same algorithms across different frameworks enormous differences occur that can be attributed to different parametrization of algorithms and evaluation procedures that are not clearly reported and therefore harm reproducibility of results. While this recent work of Said and Bellogín (2014) focuses on the internal validity of offline evaluation practices other authors focus on the external validity. Research questions targeting the external validity are, for instance, if and how confirmed results from an offline evaluation can be generalized to a recommender system's encounter with real users. Therefore, in addition to offline evaluation on dead data a user-centric evaluation approaches (i.e. user studies or field trials) are strongly advocated (Herlocker et al., 2004; Jannach et al., 2010; Ricci et al., 2011; Pu et al., 2011). Up to now several authors have reported results from comparing recommendation methods in an offline and an online evaluation setting (Cremonesi et al., 2012; Garcin et al., 2014; Ekstrand et al., 2014) and did find evidence that algorithm performance in offline and online evaluation settings differs. However, none of the research works so far has been able to demonstrate with a within user experimental design statistically significant differences in algorithm performance between both evaluation settings. This chapter's contribution therefore lies in presenting a novel evaluation design that researches algorithms' precision and their ability to present novel and relevant items in offline and online evaluation settings.

## 7.2 Related Work

Ekstrand et al. (2011) were also comparing the relative performance of different recommendation methods using distance-based versus rank-based accuracy metrics. They did not find evidence that ranking algorithms according to the distance-based error measure RMSE leads to other conclusions than ranking them according to the rank-based accuracy measure normalized discounted cumulative gain (nDCG). Cremonesi et al. (2012) investigated the persuasion potential of recommendation systems by conducting two empirical studies that assessed quality attributes such as novelty or perceived satisfaction with recommendation algorithms in between-users studies and compare these results to accuracy statistics computed on a subset of the Netflix dataset. While the first of the two studies was the first work to directly compare an offline evaluation with an online user study, there are some differences with respect to our approach. First of all, the online user study was conducted on a set of users, while the offline evaluation was performed on Netflix users. In our approach we are able to directly compare online and offline evaluations on the same set of users as we conducted the user study in two phases. Second, in their approach every user only evaluated one algorithm, leading to 30 users for each of the 7 algorithms considered, while in our case every user evaluated each algorithm, leading to 100 users for each algorithm. Garcin et al. (2014) report that in an offline setting a most popular strategy for their news recommendation system achieves the best performance while in an online field study this strategy performs poorest. This fact highlights the difference between offline and online evaluations in news recommendation: while on an historical dataset most popular recommendations are likely to be accurate, in an online setting a user does not want to read something that was already on the front page. Ekstrand et al. (2014) conducted a within-subjects user study in which each user has to evaluate two recommendation lists and to answer around 20 questions on accuracy, satisfaction, perceived personalization, novelty and diversity. Although the study was able to highlight important considerations, such that satisfaction is negatively dependent on novelty and positively dependent on diversity, their setting does not give the possibility to evaluate each recommendation but only recommendation lists, missing the computation of accuracy measurements to compare with the offline evaluation. Finally, a critical aspect in our work is the distinction between short head and long tail items, proposed in Cremonesi et al. (2010). Short head items are the most popular items, which own one third of

all the ratings even if they are usually a very little portion of the items (1.7% in Netflix and 5.5% in Movielens (Cremonesi et al., 2010)). In their work they compared the performance of top-N recommendation lists in terms of their ability to identify items from the long tail, suggesting that top-N recommendations on all items are highly influenced by popularity.

## 7.3 Study design

### 7.3.1 Overview

The goal of this research is to assess the external validity of a comparative evaluation of different recommendation methods on offline data. We would therefore like to answer the following research questions:

1. Does the relative ranking of algorithms based on offline accuracy measurement predict the relative ranking according to an accuracy measurement in a user-centric evaluation?
2. Does the relative ranking of algorithms based on offline measurements of the predictive accuracy for long-tail items produce comparable results to a user-centric evaluation?
3. Do offline accuracy measurements allow to predict the utility of recommendations in a user-centric evaluation?

For the purpose of answering these research questions we are employing a novel study design that compares offline and online measurements within the user experimental design that is depicted in Figure 7.1. We trained four different algorithms (most-popular, item-based nearest neighbors and matrix factorization with two different factor sizes) on the MovieLens 1M dataset and invited users to participate in an online experiment with two different phases. In the first phase we asked users to disclose their interests and movie preferences by browsing through an online catalog of movies and marking those movies they like. The catalog did not contain any recommendation function, thus the collected dataset of items liked by users is comparable to other datasets with unary historic user feedback that contain positive feedback on items the user probably knows and likes and no feedback on items the user either dislikes or does not know. The algorithmic models trained on MovieLens 1M could therefore be evaluated, i.e. how accurate they were in predicting the liked items of our study

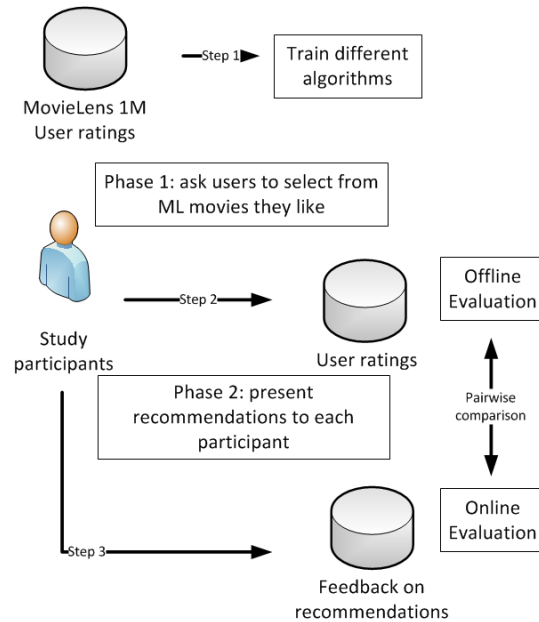


Figure 7.1: Description of the user experimental design.

participants in an offline evaluation scenario. It is worthwhile to mention that such data collection phase continue the user study described in Chapter 6, with the addition of a second group of users who participate in a second time taking the number of them from 107 to 241. In the second phase we invited the same set of study participants to assess how well each item from a set of precomputed movie recommendations matched their preferences and if the recommendation was utile for them (i.e. they would like to watch the movie and did not know about it before).

### 7.3.2 MovieLens Dataset

The dataset used to train the algorithms is the *MovieLens 1M* (ML1M)<sup>1</sup> dataset. This dataset contains 1,000,209 ratings from 6,040 users and 3,706 different movies. Ratings are integer values from an ordinal scale ranging from 1 to 5, where 1 is the worst and 5 is the best feedback. A preprocessing step was necessary to transform this data into unary user feedback representing only *likes* judgments and consisting only of movies with an entry in the Freebase

<sup>1</sup>See <http://www.grouplens.org/>.

data repository. Based on the average rating value for each individual user only 4 and 5 ratings that are greater than the user's average rating score were transformed into unary *like* statements. After removing users and movies with no single remaining positive unary rating value the dataset consisted of 6,038 users and 3,086 movies.

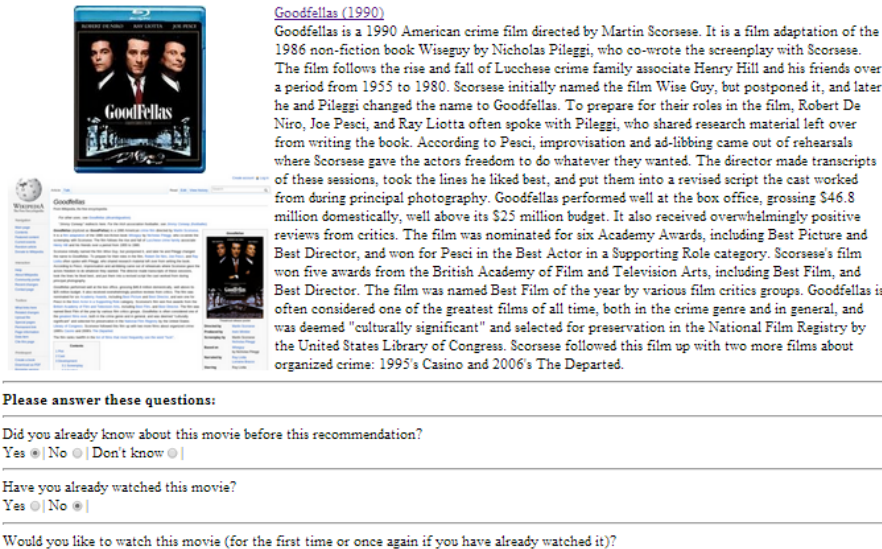
The preprocessed MovieLens dataset was then split into 2 sets: one partition for training different algorithm models (*ML Tr*, MovieLens Training) and one partition for model validation (*ML Val*, MovieLens Validation). Users were not assigned to one or the other set randomly, but following a specific criteria: the *ML Val* needs to follow the same ratings per user distribution as the collected dataset, due to the fact that the number of ratings per user influences accuracy measurements. If the *ML Val* dataset and the collected dataset are similar with respect to this criteria, the number of ratings cannot explain differences in accuracy measurements. It is worthwhile to mention that such splitting was performed between the first and the second phase of the online experiment, after the dataset was collected.

In the first phase we invited movie aficionados from Austria and Italy via our universities' mailing lists to browse our specifically developed movie portal (see Figure 6.2 in the previous chapter) and mark those items that they like. We designed and implemented a Django web application that offers users the possibility to participate at this user study from any location. A pilot test phase preceded the rollout of the web application, where we were able to identify usability flaws and understandability problems of the task. In the first round users could access a portal that contained detailed information about all movies contained in the MovieLens 1M dataset. Users were able to sort movies according to their year of launch, their name or their popularity. English, German and Italian titles of movies were extracted from Freebase in order to ensure correct translations based on the users' language settings, poster images were displayed as a mouse-over action and additional information for each movie was accessible via a link to Freebase. In total 241 users have participated in the first phase that provided on average 137 unary ratings each. We incentivized the disclosure of participants' true preferences by setting up a lottery where winners will receive one arbitrarily selected DVD from their set of liked movies.

Progress Page 1/19

Please decide if the recommendation below matches your interests!

---



[Goodfellas \(1990\)](#)  
 Goodfellas is a 1990 American crime film directed by Martin Scorsese. It is a film adaptation of the 1986 non-fiction book *Wiseguy* by Nicholas Pileggi, who co-wrote the screenplay with Scorsese. The film follows the rise and fall of Lucchese crime family associate Henry Hill and his friends over a period from 1955 to 1980. Scorsese initially named the film *Wise Guy*, but postponed it, and later he and Pileggi changed the name to *Goodfellas*. To prepare for their roles in the film, Robert De Niro, Joe Pesci, and Ray Liotta often spoke with Pileggi, who shared research material left over from writing the book. According to Pesci, improvisation and ad-libbing came out of rehearsals where Scorsese gave the actors freedom to do whatever they wanted. The director made transcripts of these sessions, took the lines he liked best, and put them into a revised script the cast worked from during principal photography. *Goodfellas* performed well at the box office, grossing \$46.8 million domestically, well above its \$25 million budget. It also received overwhelmingly positive reviews from critics. The film was nominated for six Academy Awards, including Best Picture and Best Director, and won for Pesci in the Best Actor in a Supporting Role category. Scorsese's film won five awards from the British Academy of Film and Television Arts, including Best Film, and Best Director. The film was named Best Film of the year by various film critics groups. *Goodfellas* is often considered one of the greatest films of all time, both in the crime genre and in general, and was deemed "culturally significant" and selected for preservation in the National Film Registry by the United States Library of Congress. Scorsese followed this film up with two more films about organized crime: 1995's *Casino* and 2006's *The Departed*.

---

**Please answer these questions:**

---

Did you already know about this movie before this recommendation?  
 Yes  No  Don't know

---

Have you already watched this movie?  
 Yes  No

---

Would you like to watch this movie (for the first time or once again if you have already watched it)?  
 Yes  No  Don't know

Figure 7.2: Screenshot of recommendations evaluation page.

### 7.3.3 Second phase: online evaluation

In the second round we invited users that participated in the first phase to evaluate around twenty recommendations. We were able to invite them because we already asked them to register and leave their address in the first phase. In addition, we avoided duplicate users by the registration procedure.

We asked participants to evaluate around twenty recommendations produced by four different algorithms. Each of our 4 algorithms computed 5 recommendations and we sorted them in a stratified way, where to avoid a bias in the recommendation list towards a specific algorithm. Since two or more algorithms can recommend the same item, the number of recommendations for each user went from a theoretical minimum of 5 to a maximum of 20 items, while the actual average was 17.37. To these lists, we appended two randomly selected movies from the set of liked movies in the first phase in order to ensure that users gave reliable answers. Opposed to the study design of Ekstrand et al. (2014) we did not present recommendation lists of different algorithms in parallel on the same screen for space and methodological reasons. Ekstrand et al. (2014) compared, for instance, satisfaction with two distinct recommendation lists, while we com-

pare predictive accuracy of 4 different algorithms in a within participants study design. Therefore users were asked to evaluate every recommended item separately and to answer a set of questions. For each recommended item a separate screen with information about the movie such as the title, the year of launch, the plot, the poster image extracted from Freebase and a snapshot of the related Wikipedia page (see Fig. 7.2) was displayed. By clicking on the title or on the Wikipedia snapshot the user was able to open a new page with the Freebase and Wikipedia page related to the movie. All the information provided were in the language selected by the user (English, German or Italian). The questionnaire items were the following:

- Did you already know about this movie before this recommendation? (Yes, No, Don't know)
- Have you already watched this movie? (Yes, No)
- Would you like to watch this movie (for the first time or once again if you have already watched it)? (Yes, No, Don't know)

With the first question we measure second order novelty (SON) (Cremonesi et al., 2012), i.e. an item is considered novel if the user has never heard of it. Furthermore, we do not ask for a rating value at this point, because the user had no opportunity to experience or consume the recommended item at this point. However, we determine relevance of recommended items by asking about potential conversion, i.e. if the user considers to watch this movie. By combining answers from both questions we are able to identify utile recommendations, i.e. relevant items that the user did not yet know about. Like in phase one we incentivized a truth telling behavior by setting up a lottery where participants could win a DVD from their individual set of items that they considered to be relevant (i.e. like to watch).

Not all users that participated in the first phase also responded to the invitation to participate in the second phase. In total 122 users returned to the second phase, out of which 100 provided their individual assessment to all recommended items. We selected these 100 users for our study and discarded data from all other users.

### 7.3.4 Algorithms

As already mentioned we compared four different algorithms. Three algorithms compute personalized collaborative filtering results: matrix factorization with



a 80 (MF80) and a 400 (MF400) factor model and an item-based k-nearest neighbor algorithm (KNN-IB). Furthermore, we consider a baseline algorithm that simply recommends the most-popular items (POP).

### Matrix Factorization

In this work we implemented the Matrix Factorization model described in Sindhvani et al. (2010). Let  $X \in \{0, 1\}^{M \times N}$  be a binary matrix, where  $M$  is the number of users and  $N$  is the number of items. The set of nonzeros,  $\mathcal{L} = \{(i, j) : X_{ij} = 1\}$ , denotes likes statement, i.e. user  $i$  likes movie  $j$ .  $X_{ij} = 0$  does not mean that user does not like the movie, but it simply means that the user does not know the movie. We will use  $\mathcal{M} = \{(i, j) : X_{ij} = 0\}$  to denote these unlabeled examples. The  $X$  matrix can be decomposed into  $U = [W_1, \dots, W_M] \in \mathbb{R}^{F \times M}$  and  $V = [H_1, \dots, H_N] \in \mathbb{R}^{F \times N}$ , such that  $X \approx U^T V$ . These matrices contains non-negative real values, which indicate the importance of a latent factor for a user or a movie. The Matrix Factorization model proposed in Sindhvani et al. (2010) solves the factorization problem by solving the following optimization problem:

$$\begin{aligned} \operatorname{argmin}_{U, V, Y_{ij}} J(U, V, \{Y_{ij}\}_{(i,j) \in \mathcal{M}}) = & \sum_{(i,j) \in \mathcal{L}} C_{ij} (X_{ij} - W_i^\top H_j)^2 + \\ & \sum_{(i,j) \in \mathcal{M}} C_{ij} (y_{ij} - w_i^\top h_j)^2 + \lambda \|U\|_F^2 + \gamma \|V\|_F^2 \end{aligned}$$

$J(U, V, \{Y_{ij}\}_{(i,j) \in \mathcal{M}})$  is the objective function whose first two optimization variables are the latent factors,  $U, V$ , while the third set of variables are binary variables, which specify if an unknown rating must be considered as a positive rating, i.e.  $Y_{ij} = 1$ , or must be considered as a negative rating, i.e.  $Y_{ij} = 0$ . The number of  $Y_{ij} = 1$  is constrained with a parameter, which indicates the proportion of unknown ratings that can be considered as positive.  $C_{ij}$  is a cost matrix with real values in the  $[0, 1]$  interval, which expresses the error cost for each rating and it is thought for user-defined error costs. The optimization problem is solved with a simple alternating minimization algorithm in two steps: first  $U, V$  are optimized for fixed setting of the  $Y_{ij}$  variables, then  $Y_{ij}$  are optimized keeping  $U, V$  fixed. In order to optimize  $Y_{ij}$  they are transformed in continuous variables  $P_{ij} \in [0, 1]$ , which represent the probability that an unknown rating is a positive one. The modified formulation of the optimization problem is the

following:

$$\begin{aligned} \operatorname{argmax}_{U, V, P_{ij}} J_T(U, V, \{P_{ij}\}_{(i,j) \in \mathcal{M}}) = & \sum_{(i,j) \in \mathcal{L}} C_{ij} (X_{ij} - W_i^\top H_j)^2 + \\ & \sum_{(i,j) \in \mathcal{M}} C_{ij} (P_{ij} (1 - W_i^\top H_j)^2 + (1 - P_{ij}) (0 - W_i^\top H_j)^2) + \\ & \lambda \|U\|_F^2 + \gamma \|V\|_F^2 - \sum_{(i,j) \in \mathcal{M}} H(P_{ij}) \quad \text{s.t. : } \frac{1}{|\mathcal{M}|} \sum_{(i,j) \in \mathcal{M}} P_{ij} = r \end{aligned}$$

where  $T$  is an optimization parameter which controls the entropy (the  $H$  function) of  $P_{ij}$ .

After that  $U$  and  $V$  were learned, recommendations were computed in the following way:

$$\hat{X}_i = X_i V^\top V$$

where  $X_i$  is a user vector such that  $X_i = \{X_{i,j}\} \forall j = 1, \dots, N$  and  $X_{i,j} = 1$  if user  $i$  liked movie  $j$  or 0 if he did not express a feedback about it.

This algorithm has a great number of parameters, which can influence the quality of the MF. Authors performed sensitivity analysis on  $r$  and  $T$ , which showed that the algorithm is robust with respect to these parameters. For this reason we set  $r = 0.1$  and  $T = 30$ . We executed sensitivity analysis for the regularization parameters  $\gamma$  and  $\lambda$  with a 5-fold cross validation on the ML Tr dataset, and we found that the best value for both of them is 0.1, whatever the number of latent factor is. The  $C$  matrix was not used (i.e.  $C_{ij} = 1 \forall (i, j)$ ), because we did not want to influence the model with external knowledge.

The main parameter that influences accuracy measurements is the number of latent factors  $F$ . We performed several experiments varying such parameter from 1 to 600 with increasing step's sizes and we found that with 80 latent factors the MF achieves the best value of the accuracy measurement considering all items, while with 400 latent factors it achieves the best value on long tail items. Such experiments suggest that a much higher dimensionality of latent factor models seems to be appropriate in order to excel in making non-obvious recommendations. Given that our research questions are based on the accuracy measurement on all items and on long tail items, we included the algorithms with 80 and 400 latent factors in the evaluation study.

### K-Nearest Neighbors

Neighborhood models, also known as memory-based models, represent the most common approaches to Collaborative Filtering (Herlocker et al., 1999). For further details, see Section 2.2.1. Broadly speaking, the idea of the basic user-based CF algorithm is to suggest items which are liked by users with similar tastes. The Item-Based K-Nearest Neighbors model was introduced to ease the computational challenge of User-Based CF for millions of users (Sarwar et al., 2001). The main problem is the need to scan the entire database in order to find neighbors, which it's nearly impossible to do in a real-time environment. Therefore, the basic idea of the Item-Based approach is to compute similarities between items offline, and use them to produce recommendations in real-time. In this case, the similarity between two items is given by the number of users who co-rated these items. The item rating is then computed in the following way:

$$r_{i,j} = \frac{\sum_{j' \in \mathcal{N}_j} \text{sim}(j, j') r_{i,j'}}{\sum_{j' \in \mathcal{N}_j} \text{sim}(j, j')}$$

where  $\mathcal{N}_j$  is the set of the most similar neighbors of item  $j$ .

## 7.4 Discussion on measurement methodology

In recommender systems research it is accepted to employ either one of the two basic evaluation approaches in order to determine the accuracy of a method. According to the machine learning paradigm a recommendation method  $f$  is a function that tries to best possibly predict for given user  $i$  and item  $j$  a hidden rating value  $Y_{i,j}$ , i.e.  $f(i, j) = \hat{Y}_{i,j}$  and the prediction model  $f$  is optimized such that  $\sum_{(i,j)} (\hat{Y}_{i,j} - Y_{i,j})^2$  becomes minimal. Thus, accuracy results are reported with the help of the classic RMSE (root mean squared error) or a similar error metric. Critique of this approach argues that rating values such as star ratings are ordinal scales, where the perceived distance between a 5-star and a 4-star rating value might be different from the distance between a 3-star and a 2-star rating. Furthermore, all errors are equally weighted, i.e. two hypothetical algorithms  $G$  and  $B$  that can correctly predict all hidden rating values except one, where  $G$  predicts the rating for an actual 1-star movie with 3 and  $B$  the rating for an actual 5-star movie also with 3, would achieve the same accuracy score. However, in real life  $B$  would loose user's trust by missing the opportunity to recommend a highly appreciated movie while in case of  $G$  users might not

		Ground Truth	
		Actually Good	Actually Bad
Prediction	Recommended	True Positive (tp)	False Positive (fp)
	Not recommended	False Negative (fn)	True Negative (tn)

Figure 7.3: Confusion matrix

actually care about how well the algorithm does in correctly predicting all shades of bad. In addition, an error-based evaluation does not consider how well the algorithm does with respect to identifying novel items, that the user has not yet rated.

Therefore, the standard evaluation paradigm borrowed from the field of Information Retrieval (IR) appears to better resemble the real life interactions of fielded recommendation systems. The IR methodology dates back to the historic Cranfield experiments (Cleverdon, 1967), where for a set of documents and information needs ground truth is established by exhaustive relevance judgments for each document and each query from domain experts. When applying this approach to the recommender systems domain ground truth is represented by a user times items matrix with binary rating values, where 1s indicate that the user has actually liked the particular item and 0s indicate dislike or no rating. As users tend to give only positive ratings and ignore disliked items, the default assumption in an offline evaluation scenario is to treat unrated items like items with a bad rating. Figure 7.3 depicts a confusion matrix, that is the basis for defining the standard IR measures *precision* and *recall* for a given list of recommended items to a specific user:

$$Precision = \frac{|TruePositives|}{|TruePositives| + |FalsePositives|},$$

$$Recall = \frac{|TruePositives|}{|TruePositives| + |FalseNegatives|},$$

where  $|s|$  denotes the cardinality of a set  $s$ . However, the semantics of the two basic IR measures are somewhat different in offline and online evaluation scenarios:

Table 7.1: Precision

<b>Algorithm</b>	<b>Offline</b>	<b>Online</b>	<b>Offline ML Val</b>
KNN-IB	0.438	0.546	0.602
MF80	0.504	0.598	0.708
MF400	0.454	0.604	0.548
POP	0.34	0.516	0.534

- True Positives might be actually underestimated in an offline scenario, because users could like novel recommended items that they did not know before and therefore no historic rating exists in ground truth. Correspondingly, False Positives might be overestimated in offline scenarios.
- False Negatives might also be underestimated in an offline scenario, because due to the incompleteness of ground truth, many more items might be actually relevant for a specific user, but are not recommended.
- In an online evaluation only True and False Positives can be determined, if users are asked to rate all recommended items. However True Negatives and False Negatives remain unknown.

Based on these assumptions we can hypothesize that precision in an offline evaluation scenario should be lower than precision measured online. In contrast recall cannot be measured in a user centric evaluation, while recall measured on offline data might in reality be lower or higher. Therefore, only precision can be compared in our within user experiments contrasting online and offline evaluation results. Furthermore, we also compute precision for long tail items following the proposition of Cremonesi et al. (2010), where the most popular items in the dataset are treated as they had a bad rating. According to Cremonesi et al. (2010) the most popular items that could aggregate one third of all positive ratings in the dataset are considered as belonging to the short head.

## 7.5 Results

First we test for the hypothesis that offline precision consistently underestimates precision measured in a user-centric evaluation. Table 7.1 and 7.2 contrast figures for precision and precision on the long tail for all methods. All differences between offline and online measurements of precision are statistically significant according to a Wilcoxon signed rank test at the 5% significance level.

Table 7.2: Precision on long tail items

Algorithm	Offline	Online	Offline ML Val
KNN-IB	0.28	0.356	0.219
MF80	0.018	0.054	0.012
MF400	0.36	0.528	0.393
POP	0	0	0

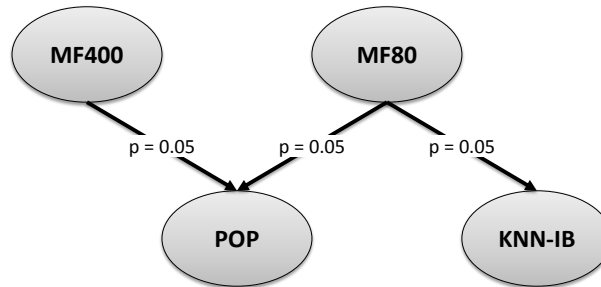


Figure 7.4: Domination graph for precision offline

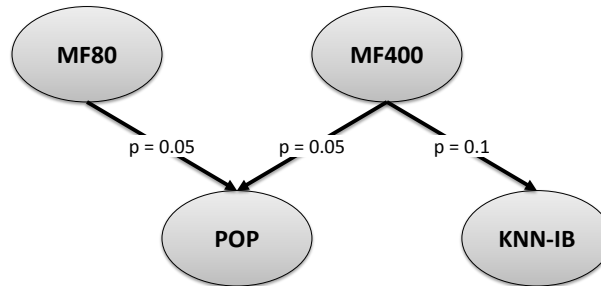


Figure 7.5: Domination graph for precision online

Research question 1 is about the relative ranking of algorithms in offline and online accuracy assessment. In Table 7.1 the precision measurements for all four algorithms and both evaluation settings are given. An additional column gives results for the offline evaluation of algorithms on the withheld MovieLens Validation dataset (ML Val, described in Subsection 7.3.2). Furthermore, Figure 7.4 gives the domination graph based on pairwise comparison of a Wilcoxon signed rank test. Both matrix factorization techniques are significantly better than the popularity baseline and the 80 factor model also clearly outperforms the KNN-IB method. The offline experiments on the ML Val dataset show a similar picture, nevertheless overall accuracy levels are obviously higher, because the algorithms are trained on MovieLens users and not on the data collected

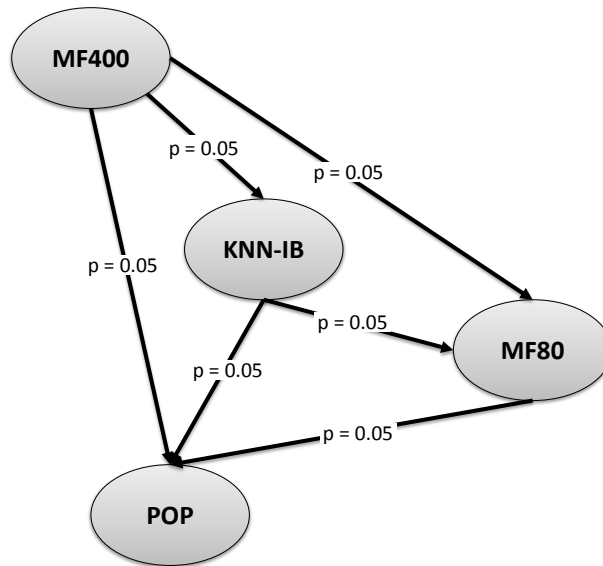


Figure 7.6: Domination graph for precision on long tail items (offline and online)

in our study. However the online measurements clearly contradict this picture. As depicted in Figure 7.5 the 400 factor model outperforms the KNN-IB at a 10% error rate, but the 80 factor model does not. Therefore this study gives empirical evidence that statistically significant differences measured in an offline evaluation scenario cannot be replicated in an online study.

Due to the strong bias of traditional accuracy measurements on popular items, Cremonesi et al. (2010) therefore proposed to exclude the most popular items from accuracy measurements (i.e. count correct predictions of very popular items as False Positives). Research Question 2 therefore asks if Precision on long tail items in offline measurements more reliably predicts the results that can be achieved in the online world. Figure 7.2 gives the actual figures and Figure 7.6 presents the domination graph. The precision measurement of long tail items seems to better discriminate between methods and leads to exactly the same statistically significant ranking of algorithms in both evaluation settings. Obviously it clearly contradicts the traditional precision measurement, the 80 factor model is by far worse in proposing non popular items than the 400 factor model or the KNN-IB model. Again the results on the ML Val dataset show a comparable algorithms' ranking as the offline experiments on the data collected from our study participants.

Table 7.3: Precision on utile recommendations

Algorithm	Online
KNN-IB	0.126
MF80	0.082
MF400	0.116
POP	0.026

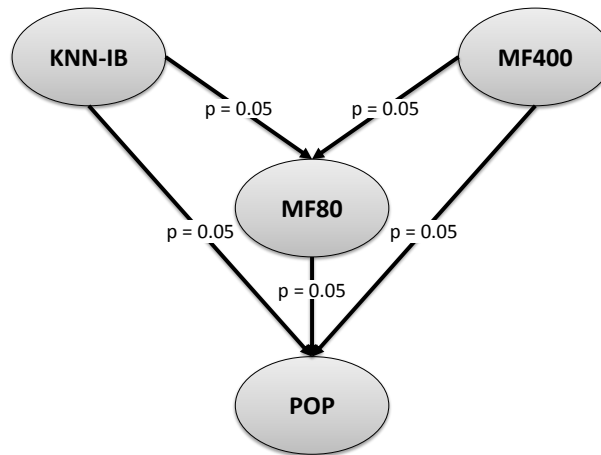


Figure 7.7: Domination graph based on precision for items users liked and did not know (online)

Finally the third research question asks for utile recommendations to users and if accuracy measurements on offline data are able to predict which algorithms will be able to make more utile recommendations to users. We simply define *utile recommendations* as propositions the user likes but did not yet know. In Table 7.3 precision figures for utile recommendations are given, that are obviously much smaller than precision values reported in Tables 7.1 and 7.2. When again testing for statistically significant differences the ranking of algorithms looks completely different from where we started in Figure 7.4. KNN-IB collaborative filtering and the 400 factor model seem to be on par and both clearly outperform the two other methods. Neither traditional precision measurements nor the proposed precision of long tail items measure by Cremonesi et al. (2010) were able to predict this algorithm ranking by offline experiments. Although KNN-IB seemed to recommend more popular items than the 400 factor model the neighborhood method did still very good in identifying items that were novel to our study participants. Consequently, the presented empirical results clearly



challenge the assumption of external validity of common evaluation practices of recommender systems and should therefore stimulate further methodological work into this direction.

## 7.6 Conclusions

The vast majority of the literature on recommender systems apply offline evaluation methodologies to assess the quality of the proposed approaches (Jannach et al., 2012). However, the external validity of such methodologies has been frequently questioned and recent works tried to compare offline and online evaluations. In this work we described a novel comparison approach between an offline evaluation protocol and an online user study. A dataset with “like” statements by 100 users on MovieLens movies has been collected and accuracy statistics for models trained on the MovieLens dataset have been evaluated on the collected dataset. Such statistics have been compared with online statistics collected with a user study, where the same 100 users answered a set of questions on around 20 recommendations proposed to them. This work showed that offline precision underestimates online precision both considering all items and only long tail items. Furthermore, offline precision does not provide the same ranking between algorithms as online precision does, while the offline precision on the long tail does that compared to the online precision on the long tail. Finally, the real utility measured with the online user study ranks algorithm in such a way that is not reproducible with offline evaluation metrics, even on all items and on long tail items. Such results are a further clue that the external validity of commonly used offline evaluation protocols is not always guaranteed.



## Chapter 8

# Conclusions

In this dissertation six contributions have been provided to the recommender systems area. These contributions concern three main aspects of recommender systems, which are *algorithms*, *interfaces* and *evaluations*.

Three contributions tackle side problems of recommender systems, i.e. integrating concepts and time. We showed how textual content associated with items can be leveraged to build concept and item networks, useful to navigate a dataset, to focus on specific concepts and to discover similarities between items. These networks can be exploited in recommender systems to provide a navigation interface, as well as to discover semantic links that can trigger new recommendations. Content can be analyzed even to track hot-topics that fluctuate over time. These topics have to be considered when recommendations are computed, as they can influence short-term interests, but not long-term interests. A methodology to analyze hot-topics has been provided with a case study on a labor law corpus. In some cases even the duration has to be considered: to tackle this problem continuous time Bayesian networks (CTBN) have been proposed. We provided the MapReduce version of two CTBN classifier algorithms, showing the performance speedup with respect to the sequential version.

Another contribution to the algorithmic aspect concerns the analysis of reviews in recommender systems where the user can express his/her opinion about an item, not only with a rating but also with a textual comment. The topic model was applied to understand what users like in items and what items offer with a quality degree. This information is stored in the profiles of users and items that are exploited for different purposes, such as computing recommendations, predicting ratings based on what a user is writing, and providing textual

explanations on multi-criteria rating dimensions, i.e. cleanliness in hotels or quality of food in restaurants. The approach not only achieves recommendation accuracy comparable to collaborative filtering systems, but also employs user models that are only based on review content and therefore offer ways to be made transparent to users or better explain to users why a specific item is recommended.

The contribution to the interface aspect concerns explanations in matrix factorization techniques, which have been shown to be the state of the art in collaborative filtering, but are not easily interpretable. Textual content associated with items was processed with the topic model and exploited to provide latent factor explanations, providing a criteria to understand which latent factors are more related to content and thus are more easily explained. A user study was conducted to evaluate our approach and showed that latent factors more related to textual content are more effective in topic prediction accuracy. Furthermore, on users for whom matrix factorization achieves better accuracy statistics, even topic prediction accuracy improves.

Finally, a contribution to the evaluation aspect has been provided. A user study was conducted to compare offline and online evaluation methods on the same dataset. The user study showed that the external validity of offline evaluation protocols is not always true, and offline precision ranks algorithms in a different way from the ranking given by the user study. On the other hand, offline precision on the long tail seems to be a better indicator to rank algorithms. However, it was impossible to reproduce the real utility ranking measured with the user study with offline statistics.

## 8.1 Future Works

The three contributions on integrating concepts and time in recommender systems represent only a first step towards the inclusion of them into recommender systems. They are ready to be additionally extended and integrated, as well as to be evaluated with offline and online methodologies.

Review analysis can be extended in order to apply more complex methodologies that make use of supervised information, such as ratings and user and item metadata, or sentiment analysis, to provide a richer topic structure. In this way not only prediction accuracy can be increased, but even analytics on multi-criteria dimensions in textual reviews can be improved.

---

Explanations in latent factor models may be improved by designing a procedure to perform parameter selection and an improved matching criteria between topics and latent factors. Topic model extensions, as well as topic model parameter selection, can be studied in order to provide a way that fits better latent factors computed only on the user-item matrix.

Finally, evaluation study results can guide the development of new evaluation protocols that are based on measurements with a verified external validity. The deep analysis of accuracy statistics on the long tail, as well as time-aware evaluation protocols, are two important directions that must be explored.



# Bibliography

- Aciar, S., Zhang, D., Simoff, S., and Debenham, J. (2007). Informed recommender: Basing recommendations on consumer product reviews. *Intelligent Systems, IEEE*, 22(3):39–47.
- ACM, editor (2014). *RecSys '14: Proceedings of the 8th ACM Conference on Recommender Systems*, New York, NY, USA. ACM. 609146.
- Adomavicius, G. and Tuzhilin, A. (2011). Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer.
- Agarwal, D. and Chen, B.-C. (2010). flda: matrix factorization through latent dirichlet allocation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 91–100. ACM.
- Aksoy, L., Bloom, P. N., Lurie, N. H., and Cooil, B. (2006). Should recommendation agents think like people? *Journal of Service Research*, 8(4):297–315.
- Allan, J., Carbonell, J. G., Doddington, G., Yamron, J., and Yang, Y. (1998). Topic detection and tracking pilot study final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218.
- Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern information retrieval*, volume 463. ACM press New York.
- Baltrunas, L., Ludwig, B., and Ricci, F. (2011). Matrix factorization techniques for context aware recommendation. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 301–304. ACM.
- Basak, A., Brinster, I., Ma, X., and Mengshoel, O. J. (2012). Accelerating bayesian network parameter learning using hadoop and mapreduce. In *Pro-*

- ceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pages 101–108. ACM.
- Bell, R., Koren, Y., and Volinsky, C. (2007). Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–104. ACM.
- Benbasat, I. and Wang, W. (2005). Trust in and adoption of online recommendation agents. *Journal of the Association for Information Systems*, 6(3):4.
- Bennett, J. and Lanning, S. (2007). The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35.
- Billsus, D. and Pazzani, M. J. (1999). A personal news agent that talks, learns and explains. In *Proceedings of the third annual conference on Autonomous Agents*, pages 268–275. ACM.
- Billsus, D., Pazzani, M. J., and Chen, J. (2000). A learning agent for wireless news access. In *Proceedings of the 5th international conference on Intelligent user interfaces*, pages 33–36. ACM.
- Blei, D. and Lafferty, J. (2006a). Correlated topic models. *Advances in neural information processing systems*, 18:147.
- Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.
- Blei, D. M., Griffiths, T. L., and Jordan, M. I. (2010). The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)*, 57(2):7.
- Blei, D. M. and Jordan, M. I. (2003). Modeling annotated data. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 127–134. ACM.
- Blei, D. M. and Lafferty, J. D. (2006b). Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM.
- Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.



- Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc.
- Brewer, W. F., Chinn, C. A., and Samarapungavan, A. (1998). Explanation in scientists and children. *Minds and Machines*, 8(1):119–136.
- Bridge, D. and Dunleavy, K. (2014). If you liked herlocker et al.s explanations paper, then you might like this paper too. In *Joint Workshop on Interfaces and Human Decision Making in Recommender Systems*, page 22.
- Bridge, D., Göker, M. H., McGinty, L., and Smyth, B. (2005). Case-based recommender systems. *The Knowledge Engineering Review*, 20(03):315–320.
- Buchanan, B. G. and Shortliffe, E. H. (1984). *Rule-based expert systems*, volume 3. Addison-Wesley Reading, MA.
- Burke, R. (2000). Knowledge-based recommender systems. *Encyclopedia of library and information systems*, 69(Supplement 32):175–186.
- Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370.
- Burke, R. D., Hammond, K. J., and Young, B. C. (1996). Knowledge-based navigation of complex information spaces. In *Proceedings of the national conference on artificial intelligence*, volume 462, page 468.
- Campos, P. G., Díez, F., and Sánchez-Montañés, M. (2011). Towards a more realistic evaluation: testing the ability to predict future tastes of matrix factorization-based recommenders. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 309–312. ACM.
- Carenini, G., Smith, J., and Poole, D. (2003). Towards more conversational and collaborative recommender systems. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 12–18. ACM.
- Chakrabarti, S. (2003). *Mining the Web: Discovering knowledge from hypertext data*. Morgan Kaufmann.
- Chang, J. and Blei, D. M. (2009). Relational topic models for document networks. In *International Conference on Artificial Intelligence and Statistics*, pages 81–88.

- Chang, J., Gerrish, S., Wang, C., Boyd-graber, J. L., and Blei, D. M. (2009). Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296.
- Chen, K.-Y., Luesukprasert, L., Chou, S.-c. T., et al. (2007). Hot topic extraction based on timeline analysis and multidimensional sentence modeling. *IEEE transactions on knowledge and data engineering*, 19(8):1016.
- Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., and Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, volume 60. Citeseer.
- Cleverdon, C. (1967). The cranfield tests on index language devices. *Aslib proceedings*, 19(6):173–194.
- Console, L., Dupré, D. T., and Torasso, P. (1991). On the relationship between abduction and deduction. *Journal of Logic and Computation*, 1(5):661–690.
- Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27.
- Cremonesi, P., Garzotto, F., and Turrin, R. (2012). Investigating the persuasion potential of recommender systems from a quality perspective: An empirical study. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(2):11.
- 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*, pages 39–46. ACM.
- Cuthill, E. and McKee, J. (1969). Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pages 157–172. ACM.
- Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM.
- Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.
- Dean, J. and Ghemawat, S. (2010). Mapreduce: a flexible data processing tool. *Communications of the ACM*, 53(1):72–77.

- Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990). Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.
- Dippelreiter, B., Grün, C., Pöttler, M., Seidel, I., Berger, H., Dittenbach, M., and Pesenhofer, A. (2008). Online tourism communities on the path to web 2.0: an evaluation. *Information technology & tourism*, 10(4):329–353.
- Dixon, E., Enos, E., and Brodmerkle, S. (2013). A/b testing. US Patent App. 14/075,382.
- Ekstrand, M. D., Harper, F. M., Willemsen, M. C., and Konstan, J. A. (2014). User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM conference on Recommender systems (RecSys 14)*. ACM, New York, NY, USA.
- Ekstrand, M. D., Ludwig, M., Konstan, J. A., and Riedl, J. T. (2011). Rethinking the recommender research ecosystem: reproducibility, openness, and lenskit. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 133–140. ACM.
- Felfernig, A. and Burke, R. (2008). Constraint-based recommender systems: technologies and research issues. In *Proceedings of the 10th international conference on Electronic commerce*, page 3. ACM.
- Felfernig, A., Friedrich, G., Jannach, D., and Zanker, M. (2006). An integrated environment for the development of knowledge-based recommender applications. *International Journal of Electronic Commerce*, 11(2):11–34.
- Fogg, B. J. (2002). Persuasive technology: using computers to change what we think and do. *Ubiquity*, 2002(December):5.
- Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine learning*, 29(2-3):131–163.
- Friedrich, G. and Zanker, M. (2011). A taxonomy for generating explanations in recommender systems. *AI Magazine*, 32(3):90–98.
- Fuhr, N. (1992). Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255.

- Garcin, F., Faltings, B., Donatsch, O., Alazzawi, A., Bruttin, C., and Huber, A. (2014). Offline and online evaluation of news recommender systems at swissinfo. ch. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 169–176. ACM.
- Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-6(6):721–741.
- Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.
- Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151.
- Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235.
- Griffiths, T. L., Steyvers, M., Blei, D. M., and Tenenbaum, J. B. (2004). Integrating topics and syntax. In *Advances in neural information processing systems*, pages 537–544.
- Guha, R., Kumar, R., Raghavan, P., and Tomkins, A. (2004). Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*, pages 403–412. ACM.
- Hall, D., Jurafsky, D., and Manning, C. D. (2008). Studying the history of ideas using topic models. In *Proceedings of the conference on empirical methods in natural language processing*, pages 363–371. Association for Computational Linguistics.
- Harel, D. and Koren, Y. (2001). On clustering using random walks. In *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science*, pages 18–41. Springer.
- Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM.

- Herlocker, J. L., Konstan, J. A., and Riedl, J. (2000). Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250. ACM.
- Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53.
- Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.
- Hurley, N. and Rickard, S. (2009). Comparing measures of sparsity. *Information Theory, IEEE Transactions on*, 55(10):4723–4741.
- Jakob, N., Weber, S. H., Müller, M. C., and Gurevych, I. (2009). Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 57–64. ACM.
- Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender systems: an introduction*. Cambridge University Press.
- Jannach, D., Zanker, M., and Fuchs, M. (2014). Leveraging multi-criteria customer feedback for satisfaction analysis and improved recommendations. *Information Technology & Tourism*, 14(2):119–149.
- Jannach, D., Zanker, M., Ge, M., and Gröning, M. (2012). *Recommender systems in computer science and information systems—a landscape of research*. Springer.
- Järvelin, K. and Kekäläinen, J. (2000). Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48. ACM.
- Konstas, I., Stathopoulos, V., and Jose, J. M. (2009). On social networks and collaborative recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 195–202. ACM.

- Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM.
- Koren, Y. (2009). The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81.
- Koren, Y. (2010). Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97.
- Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.
- Kwak, H., Lee, C., Park, H., and Moon, S. (2010). What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM.
- Lamche, B., Adıgüzel, U., and Wörndl, W. (2014). Interactive explanations in mobile shopping recommender systems. In *Joint Workshop on Interfaces and Human Decision Making in Recommender Systems*, page 14.
- Lee, T. Q., Park, Y., and Park, Y.-T. (2008). A time-based approach to effective recommender systems using implicit feedback. *Expert systems with applications*, 34(4):3055–3062.
- Leskovec, J., Huttenlocher, D., and Kleinberg, J. (2010). Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1361–1370. ACM.
- Levi, A., Mokryn, O., Diot, C., and Taft, N. (2012). Finding a needle in a haystack of reviews: cold start context-based hotel recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 115–122. ACM.
- Li, W. and McCallum, A. (2006). Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, pages 577–584. ACM.
- Lin, C. and He, Y. (2009). Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 375–384. ACM.

- Lin, J. (2008). Scalable language processing algorithms for the masses: A case study in computing word co-occurrence matrices with mapreduce. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 419–428. Association for Computational Linguistics.
- Lin, J. and Dyer, C. (2010). Data-intensive text processing with mapreduce. *Synthesis Lectures on Human Language Technologies*, 3(1):1–177.
- Ling, C. X. and Li, C. (1998). Data mining for direct marketing: Problems and solutions. In *KDD*, volume 98, pages 73–79.
- Litvin, S. W., Goldsmith, R. E., and Pan, B. (2008). Electronic word-of-mouth in hospitality and tourism management. *Tourism management*, 29(3):458–468.
- Ma, H., Lyu, M. R., and King, I. (2009). Learning to recommend with trust and distrust relationships. In *Proceedings of the third ACM conference on Recommender systems*, pages 189–196. ACM.
- Ma, H., Yang, H., Lyu, M. R., and King, I. (2008). Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940. ACM.
- Ma, H., Zhou, D., Liu, C., Lyu, M. R., and King, I. (2011). Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM.
- Mahmood, T. and Ricci, F. (2009). Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82. ACM.
- Martin, F. J. (2009). Recsys’ 09 industrial keynote: top 10 lessons learned developing deploying and operating real-world recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 1–2. ACM.
- Massa, P. and Avesani, P. (2007). Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24. ACM.

- McAuley, J. and Leskovec, J. (2013). Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM.
- Mcauliffe, J. D. and Blei, D. M. (2008). Supervised topic models. In *Advances in neural information processing systems*, pages 121–128.
- McCallum, A., Corrada-Emmanuel, A., and Wang, X. (2005). The author-recipient-topic model for topic and role discovery in social networks, with application to enron and academic email. In *Workshop on Link Analysis, Counterterrorism and Security*, pages 33–44.
- McGinty, L. and Smyth, B. (2006). Adaptive selection: An analysis of critiquing and preference-based feedback in conversational recommender systems. *International Journal of Electronic Commerce*, 11(2):35–57.
- McSherry, D. (2003). Similarity and compromise. In *Case-Based Reasoning Research and Development*, pages 291–305. Springer.
- Mcsberry, D. (2005). Retrieval failure and recovery in recommender systems. *Artificial Intelligence Review*, 24(3-4):319–338.
- Metallinos, N. (1991). Persuasion: Theory and research. *Canadian Journal of Communication*, 16(3).
- Mnih, A. and Salakhutdinov, R. (2007). Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264.
- Moshfeghi, Y., Piwowarski, B., and Jose, J. M. (2011). Handling data sparsity in collaborative filtering using emotion and semantic based features. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 625–634. ACM.
- Nass, C. I. and Brave, S. (2005). *Wired for speech: How voice activates and advances the human-computer relationship*. MIT Press Cambridge.
- Nielsen (2014). E-commerce: Evolution or revolution in the fast-moving consumer goods world? Technical report, Nielsen.
- Nodelman, U., Shelton, C. R., and Koller, D. (2002). Continuous time bayesian networks. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 378–387. Morgan Kaufmann Publishers Inc.



- Nodelman, U. D. (2007). *Continuous time Bayesian networks*. PhD thesis, Stanford University.
- Oard, D. W., Kim, J., et al. (1998). Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, pages 81–83. Wollongong.
- Paatero, P. and Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126.
- Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.
- Pareschi, F., Rossetti, M., and Stella, F. (to appear in 2014). Tracking hot topics for the monitoring of open-world processes. In *SIMPDA 2014*. Citeseer.
- Payne, J. W., Bettman, J. R., and Johnson, E. J. (1993). *The adaptive decision maker*. Cambridge University Press.
- Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408.
- Pedhazur, E. J. and Schmelkin, L. P. (2013). *Measurement, design, and analysis: An integrated approach*. Psychology Press.
- Piotte, M. and Chabbert, M. (2009). The pragmatic theory solution to the netflix grand prize. *Netflix prize documentation*.
- Pu, P., Chen, L., and Hu, R. (2011). A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 157–164. ACM.
- Ramirez, E. H., Brena, R., Magatti, D., and Stella, F. (2012). Topic model validation. *Neurocomputing*, 76(1):125–133.
- Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., McNee, S. M., Konstan, J. A., and Riedl, J. (2002). Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 127–134. ACM.

- Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM.
- Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B. (2011). *Recommender Systems Handbook*. Springer.
- Rosen-Zvi, M., Griffiths, T., Steyvers, M., and Smyth, P. (2004). The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494. AUAI Press.
- Rossetti, M., Arcelli, F., Pareschi, R., and Stella, F. (2014). Integrating concepts and knowledge in large content networks. *New generation computing*.
- Rossetti, M., Stella, F., Cao, L., and Zanker, M. (to appear in 2015). Analyzing user reviews in tourism with topic models. In *Information and Communication Technologies in Tourism 2015*. Springer.
- Rossetti, M., Stella, F., and Zanker, M. (2013). Towards explaining latent factors with topic models in collaborative recommender systems. In *Database and Expert Systems Applications (DEXA), 2013 24th International Workshop on*, pages 162–167. IEEE.
- Rossetti, M. and Zanker, M. (working paper for 2015). Assessing the validity of offline recommendation accuracy measurement. *working paper*.
- Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.
- Said, A. and Bellogín, A. (2014). Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM conference on Recommender systems (RecSys 14)*. ACM, New York, NY, USA.
- Salton, G. (1971). *The SMART retrieval system experiments in automatic document processing*. Prentice-Hall, Inc.
- Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

- Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document.
- Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM.
- Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer.
- Schmallegger, D. and Carson, D. (2008). Blogs in tourism: Changing approaches to information exchange. *Journal of vacation marketing*, 14(2):99–110.
- Sen, S., Lam, S. K., Rashid, A. M., Cosley, D., Frankowski, D., Osterhouse, J., Harper, F. M., and Riedl, J. (2006). Tagging, communities, vocabulary, evolution. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 181–190. ACM.
- Shardanand, U. and Maes, P. (1995). Social information filtering: algorithms for automating word of mouth. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co.
- Sindhwani, V., Bucak, S. S., Hu, J., and Mojsilovic, A. (2010). One-class matrix completion with low-density factorizations. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 1055–1060. IEEE.
- Song, S., Moustafa, H., and Afifi, H. (2012). Advanced iptv services personalization through context-aware content recommendation. *Multimedia, IEEE Transactions on*, 14(6):1528–1537.
- Stella, F. and Amer, Y. (2012). Continuous time bayesian network classifiers. *Journal of biomedical informatics*, 45(6):1108–1119.
- Steyvers, M. and Griffiths, T. (2007). Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440.
- Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2006). Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476).
- Thompson, C. A., Göker, M. H., and Langley, P. (2004). A personalized system for conversational recommendations. *J. Artif. Intell. Res.(JAIR)*, 21:393–428.

- Tintarev, N. (2007). Explanations of recommendations. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 203–206. ACM.
- Tintarev, N. and Masthoff, J. (2007). Effective explanations of recommendations: user-centered design. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 153–156. ACM.
- Töscher, A., Jahrer, M., and Bell, R. M. (2009). The bigchaos solution to the netflix grand prize. *Netflix prize documentation*.
- Tsang, E. (1993). *Foundations of constraint satisfaction*, volume 289. Academic press London.
- Tso-Sutter, K. H., Marinho, L. B., and Schmidt-Thieme, L. (2008). Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1995–1999. ACM.
- Vassileva, J. (2008). Toward social learning environments. *Learning Technologies, IEEE Transactions on*, 1(4):199–214.
- Verbert, K., Parra, D., and Brusilovsky, P. (2014). The effect of different set-based visualizations on user exploration of recommendations. In *Joint Workshop on Interfaces and Human Decision Making in Recommender Systems*, page 37.
- Villa, S. and Rossetti, M. (to appear in 2014). Learning continuous time bayesian network classifiers using mapreduce. *Journal of Statistical Software*.
- Villa, S. and Stella, F. (2014). A continuous time bayesian network classifier for intraday fx prediction. *Quantitative Finance*, ahead-of-print(ahead-of-print):1–14.
- Wallach, H. M. (2006). Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984. ACM.
- Wallach, H. M., Murray, I., Salakhutdinov, R., and Mimno, D. (2009). Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1105–1112. ACM.
- Wang, C. and Blei, D. M. (2011). Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM.

- Wang, H., Lu, Y., and Zhai, C. (2010). Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 783–792. ACM.
- Webster, A. and Vassileva, J. (2007). The keepup recommender system. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 173–176. ACM.
- Xiang, Z. and Gretzel, U. (2010). Role of social media in online travel information search. *Tourism management*, 31(2):179–188.
- Ye, Q., Law, R., Gu, B., and Chen, W. (2011). The influence of user-generated content on traveler behavior: An empirical investigation on the effects of e-word-of-mouth to hotel online bookings. *Computers in Human Behavior*, 27(2):634–639.
- Yoo, K.-H., Gretzel, U., and Zanker, M. (2012). *Persuasive Recommender Systems*. Springer.
- Zanker, M., Bricman, M., Gordea, S., Jannach, D., and Jessenitschnig, M. (2006). Persuasive online-selling in quality and taste domains. In *E-Commerce and Web Technologies*, pages 51–60. Springer.
- Zanker, M., Jessenitschnig, M., and Schmid, W. (2010). Preference reasoning with soft constraints in constraint-based recommender systems. *Constraints*, 15(4):574–595.
- Zanker, M. and Schoberegger, M. (2014). An empirical study on the persuasiveness of fact-based explanations for recommender systems. In *Joint Workshop on Interfaces and Human Decision Making in Recommender Systems*, page 33.
- Zehrer, A., Crotts, J. C., and Magnini, V. P. (2011). The perceived usefulness of blog postings: An extension of the expectancy-disconfirmation paradigm. *Tourism Management*, 32(1):106–113.
- Zhen, Y., Li, W.-J., and Yeung, D.-Y. (2009). Tagicofi: tag informed collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems*, pages 69–76. ACM.