Enhanced Vector Space Models for Content-based Recommender Systems

Cataldo Musto
Dipartimento di Informatica
Dottorato in Informatica XXIV ciclo
Università degli Studi di Bari Aldo Moro
Via E. Orabona, 4 - 70125 Bari, Italy
cataldomusto@di.uniba.it

S.S.D.: INF/01

Supervisor: Prof. Giovanni Semeraro

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

April, 2012
Credits

This dissertation was typeset using these open-source programs:

- TeXShop LaTeX Editor
  available at: http://www.uoregon.edu/~koch/texshop/

- MacTeX Distribution
  available at: http://www.tug.org/mactex/

Thesis Supervisor

Prof. Giovanni Semeraro

Chairperson of the Supervisory Committee


Member of the Supervisory Committee


Member of the Supervisory Committee


Submitted April, 2012
Copyright © 2012, Cataldo Musto
Contents

1 Introduction 1
  1.1 Background .............................................................. 1
    1.1.1 Information Overload & Big Data .......................... 2
    1.1.2 Intelligent Information Access .............................. 5
  1.2 Research Motivations ................................................. 7
  1.3 Contributions of the Thesis ....................................... 8
  1.4 Organization of the Thesis ....................................... 10
  1.5 List of Publications ............................................... 11

2 Recommender Systems 17
  2.1 Basics of Recommender Systems .................................. 18
    2.1.1 The Recommendation Problem ............................. 19
    2.1.2 A Recommendation Pipeline ............................... 20
  2.2 Content-based Recommender Systems .......................... 22
    2.2.1 Limitations of Content-based Recommender Systems .... 30
    2.2.2 Examples of Content-based Recommender Systems .......... 31
  2.3 Collaborative Recommender Systems .......................... 34
    2.3.1 Limitations of Collaborative Recommender Systems .... 38
    2.3.2 Examples of Collaborative Recommender Systems .......... 39
  2.4 Demographic, Knowledge-based and
    Community-based Recommender Systems ....................... 42
  2.5 Hybrid Recommender Systems ................................... 43
2.6 Evaluation of Recommender Systems .......................... 45
  2.6.1 Standard Metrics ......................................... 46
  2.6.2 Non-Standard Metrics ................................. 50

3 Enhanced Vector Space Models (eVSM) for CBRS 53
  3.1 Overview of Vector Space Model ............................. 54
    3.1.1 Literature Review .................................... 55
  3.2 A Recommendation Framework based on VSM .................. 57
    3.2.1 Introducing Semantics in VSMs ....................... 58
    3.2.2 Dimensionality Reduction Techniques for VSMs ....... 61
    3.2.3 Modeling Negation in VSMs ............................. 64
    3.2.4 Discussion ........................................... 67
  3.3 Description of eVSM ........................................ 67
    3.3.1 Random Indexing ...................................... 68
    3.3.2 Quantum Negation ..................................... 70
    3.3.3 Building User Profiles ................................ 72
    3.3.4 Generating Recommendation ............................ 74

4 Applications of eVSM 79
  4.1 MovieLens and Yahoo! Webscope: content-based movie recommendation ............................................. 80
    4.1.1 Monolingual Recommendations - MovieLens dataset .... 81
    4.1.2 Monolingual Recommendations - Yahoo! dataset ....... 89
    4.1.3 Multilingual Recommendations .......................... 94
  4.2 Play.me: personalized playlists generator ................... 98
    4.2.1 Description of the Platform ............................ 98
    4.2.2 Play.me@work: a typical use case ...................... 103
    4.2.3 Experimental Evaluation ................................ 104
  4.3 Myusic: content-based music
    recommendations ............................................. 107
    4.3.1 Description of the Platform ............................ 107
    4.3.2 Experimental Evaluation ................................ 115

5 Industrial Applications of eVSM 123
  5.1 Scenario .................................................... 124
  5.2 Data representation ......................................... 126
    5.2.1 Bag-of-Words-based representation ...................... 126
    5.2.2 Tag.me-based representation ........................... 128
  5.3 Description of the tasks .................................... 129
# List of Figures

1.1 The exponential growth of digital information .......................... 4

2.1 An example of Recommender System ................................. 18
2.2 A simplified two-dimensional classification problem ............... 29
2.3 Syskill & Webert user interface .................................. 32
2.4 Item-based Collaborative Filtering .................................. 37
2.5 GroupLens rating and recommendation page ......................... 40
2.6 A typical recommendation from Amazon ......................... 41
2.7 Three ROC curves with different levels of goodness according to their AUC .......................................................... 48

3.1 Similar terms share similar usages .............................. 60
3.2 Building the centroids for relevant and non-relevant documents . 66
3.3 A visual explanation of the Johnson-Linderstrauss lemma .... 71
3.4 Example of language-independent usage of terms .................. 76

4.1 Outcomes of Experiment 1 for RI and W-RI profiles (MovieLens dataset) .......................................................... 83
4.2 Outcomes of Experiment 1 for QN and W-QN profiles (MovieLens dataset) .......................................................... 84
4.3 Outcomes of Experiment 2 for dimension 50 and 100 (MovieLens dataset) .......................................................... 86
4.4 Outcomes of Experiment 2 for dimension 200 and 400 (MovieLens dataset) .......................................................... 87
4.5 Outcomes of Experiment 3 .............................................. 88
4.6 Outcomes of Experiment 1 for RI and W-RI profiles (Yahoo! dataset) .......................................................... 89
4.7 Outcomes of Experiment 1 for QN and W-QN profiles (Yahoo! dataset) .......................................................... 90
4.8 Outcomes of Experiment 2 for dimension 500 and 1000 (Yahoo! dataset) ......................................................... 92
4.9 Outcomes of Experiment 2 for dimension 1500 and 2000 (Yahoo! dataset) ......................................................... 93
4.10 Outcomes of Experiment 3 ................................................. 94
4.11 Play.me architecture .......................................................... 99
4.12 User Preferences from a Facebook profile ................................ 100
4.13 An example of property in DBPedia ........................................ 101
4.14 Coldplay tag cloud from Last.fm ........................................... 102
4.15 Play.me screenshot ............................................................ 104
4.16 Myusic architecture .......................................................... 108
4.17 User Preferences from a Facebook profile ................................ 109
4.18 Links from a Facebook profile .............................................. 110
4.19 Extraction of social data for building Myusic profiles ............... 111
4.20 An example of user profile in Myusic platform ...................... 112
4.21 User profile represented as tag cloud in Myusic platform ........ 113
4.22 An example of recommendation list in Myusic platform .......... 114
4.23 Artist details on Myusic platform .......................................... 115
4.24 Comparison between social media-based profiles and a simple baseline based on popular artists ......................... 117
4.25 Comparison between RI and WRI configuration in Myusic platform ................................................................. 118
4.26 Comparison between RI and QN configuration in Myusic platform ................................................................. 118
4.27 Novelty, Accuracy and Serendipity in the recommendations produced by Myusic ............................................. 120

5.1 Watchmi plug-in ............................................................... 124
5.2 The concept of personal channel ........................................... 125
5.3 A fragment of EPGs XML file ............................................. 127
5.4 An example of BOW-based representation ............................... 128
5.5 Tag.me web interface ........................................................ 129
5.6 A fragment of the Tag.me XML file ....................................... 130
5.7 Tag.me-based representation .............................................. 130
5.8 Length of the descriptions, averaged for all the program types .. 134
5.9 Distribution of the training examples among the 10 partitions .. 135
5.10 Distribution of the training examples among the 17 program types ................................................................. 136
5.11 Comparison between BOW and Tag.me with random indexing on text classification ............................................. 137
5.12 Comparison between BOW and Tag.me with random indexing on text retrieval ................................................. 139
5.13 Comparison between BOW and Tag.me with random indexing on
text retrieval ......................................................... 140
5.14 Comparison between VSM and RI on text classification .......... 141
5.15 Comparison between RI and RI+QN on text retrieval (P@n) ... 144
5.16 Comparison between RI and RI+QN on text retrieval (P@k%) ... 145
List of Tables

4.1 Content-based MovieLens dataset statistics. . . . . . . . . . . . . . 80
4.2 Features used to represent movies . . . . . . . . . . . . . . . . . . . . 95
4.3 Precision @5 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
4.4 Precision @10 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
4.5 Results of Experiments . . . . . . . . . . . . . . . . . . . . . . . . . 106

5.1 Program Types mapping . . . . . . . . . . . . . . . . . . . . . . . . . . 126
5.2 EPGs dataset statistics . . . . . . . . . . . . . . . . . . . . . . . . . . 133
5.3 Average length of the descriptions, divided per program type . . 134
5.4 Comparison between BOW and Tag.me with random indexing on
   text classification . . . . . . . . . . . . . . . . . . . . . . . . . . . 137
5.5 Results of the VSM in the classification task . . . . . . . . . . . . . 141
5.6 Comparison between BOW and random indexing on text classi-
   fication, results divided per program type . . . . . . . . . . . . . . . . 143
5.7 Results of the RI algorithm on text retrieval (P@n) . . . . . . . . 143
5.8 Results of the RI+QN algorithm on text retrieval (P@n) . . . . . 144
5.9 Results of the RI+QN algorithm on text retrieval (P@k%) . . . . 144
5.10 Results of the RI+QN algorithm on text retrieval (P@k%) . . . . 145
Abstract

The recent spread of collaborative platforms and social networks made the authoring of content easier and easier. However, due to this uncontrolled phenomenon, the amount of data continuously grows without a proper control in terms of quality and reliability of produced content. This makes the problem of Information Overload much more felt than the past.

In this scenario, Information Retrieval (IR) and Filtering (IF) tools are able to effectively manage the surplus of data we daily interact with. Recommender Systems (RSs), for example, can help sift this flow of information by providing users with personalized access to textual and multimedia information sources.

The scenario RSs deal with is very natural and common, since people get advice all the time, especially when it is necessary to choose among different alternatives and the knowledge to critically discern them is not enough. This is a well-known problem, usually referred to as Paradox of Choice [Sch05]. Indeed, when the number of possible alternatives is too big with respect to the knowledge that the individuals hold about a problem, it is common to fall in the so-called paradox of Buridan [Res59], namely, the inability to make decisions when too much alternatives are available.

Generally speaking, people use a lot of strategies to solve decision-making processes. Sometimes human recommendations are adequate to effectively face the problem (for example, to find the best restaurants in our own city), but usually they don’t because the information held by our friends is not enough to provide good suggestions in all the cases. In other terms, when knowledge is not enough the individuals need to be assisted in decision-making processes and IF systems like RSs are the tools that can cope the best with these tasks, as already demonstrated by many successful use cases such as Amazon, Netflix, Pandora and so on.

Even if the most popular recommendation algorithms follow the collaborative approach, content-based recommender systems (CBRS) have proved to be effective in real-world scenarios, since they can face with some limitations of col-
laborative algorithms such as scalability and the new item problem. Techniques for CBRS are based on the assumption that user preferences remain stable over time. The relevance of an unseen item for a target user is usually predicted by matching the features stored in a user profile (inferred from the items previously considered as relevant) with those describing the new item.

The goal of this dissertation is to introduce eVSM (enhanced Vector Space Model), a novel content-based recommendation framework that exploits the typical strengths of Vector Space Model (VSM) and extends it by tackling its drawbacks. Specifically, eVSM tries to overcome the limitations of VSM by catching the semantics of the documents, by allowing an incremental construction of the vector space and by managing negative information contained in the documents. These features have been obtained by introducing in VSM a negation operator based on Quantum Logic and a lightweight semantic representation based on distributional models and Random Indexing.

In the experimental sessions the effectiveness of eVSM is evaluated in both offline and online settings, and it emerged that eVSM overcomes several state-of-the-art models in terms of goodness of recommendations and accuracy of the proposed ranking. Specifically, it outperforms in a significant way LSI, the classical VSM and a Bayes text classifier in the task of providing users with recommendations about movies. The general outcomes of the offline evaluations are confirmed in the online ones as well, since eVSM showed its capability of providing good recommendations in terms of novelty and serendipity, as well as gaining user trust during the interaction with the system.
Chapter 1

Introduction

1.1 Background

The desire for knowledge is a natural feeling of mankind, always driven to analyze and decode the world in order to understand the meaning of their own existence. It is not a coincidence that this topic is common to many writers and philosophers of antiquity: in Metaphysics, Aristotle said that "all man naturally desire knowledge". "Knowledge is power" is a famous Bacon’s quote that points out how important this concept was already four centuries ago. Moreover, as stated by Kant, human beings had always looked for knowledge, even at the cost of exceeding their own capabilities.

The scenario is not changed today. We are faced with a sharp dichotomy between the natural knowledge desire and the inability to acquire and manage all the available information. During three centuries the only thing that changed is the physical mean used to spread and transfer knowledge, since today it is available in digital form and is mainly enjoyed through the Web.

Despite this, we cannot think about this as something new. The history shows us that human beings already dealt with this problem and they solved it by looking for measures to effectively handle the surplus of information. Specifically, the challenge for optimizing the management of information has always been played on two levels: the physiological one, related to human brain adaptability, and the technological one, that regards the ability of producing innovations to better support the information management.

It is not by chance that the main developments in the area of communication have occurred when human beings were coped with the inability to effectively manage all the available information. For example, the writing was introduced
by Gutenberg when the oral communication did not allow an efficient knowledge transfer anymore. Similarly, the press was invented when the process of writing became too slow and expensive. Today, the digitalization of information and its distribution through the Internet changed the rules for information management once again.

The introduction of a new medium, as well as the surplus of information that follows, requires that our brain physiologically adapts to it. Recent studies [Ott10] showed that human brain is able to absorb 126 bits of information per second. The naïve strategies we currently use to cope with the staggering amount of data and the feeling of stress that goes with it are perhaps the first steps of an adaptation process that will make the brain better structured, connected and able to effectively process all this data.

Indeed, it is wrong to state that the brain cannot handle so many physiological stimuli [CH11]: as pointed out by Savolainen [Sav07], our senses (such as vision) have a very good capacity of filtering and processing signals, images and messages. For some kind of signals the brain has a remarkable ability to eliminate the noise and receive only the relevant patterns. According to this study, the problem is that nowadays the information is presented in a format for which our brain is not prepared and needs to be assisted. However, also assuming a future improvement of our physiological and mental capacities, we daily interact with 34 billion of bits [Ott10], the equivalent of 393 bits per second. This means that the amount of information we interact with is approximately three times the amount that we are able to absorb. By projecting the problem until 2020, where it is expected [Ott10] that the amount of information will increase by almost 30 times (from the current 1.2 to 35 zettabytes), we come to the conclusion that the problem of the cognitive surplus is very important and some countermeasure is needed.

1.1.1 Information Overload & Big Data

The definition of Information Overload was introduced to describe the feeling of fatigue and confusion that follows the cognitive surplus required to handle the amount of information we have to deal with.

The first mention about this issue dates back to 1852 when in the annual report written by the Secretary of the Smithsonian institute in Washington the attention was drawn on the problem of Information Overload [BR09]. Later, during the 1948 Royal Society’s Influential Scientific Information Conference, the Information Overload began to be felt as a problem [BR09]. In literature [Tof71], Alvin Toffler described a prophetic scenario where the rapid technolog-
ical growth of the society (he called it as *super-industrial society*) caused in the individuals stress and confusion. Currently there is not a single and universally accepted definition of Information Overload. In general, it tends to describe the phenomenon as a state of things where efficiency is jeopardized by the amount of available information [EM04]. More precisely, humans are placed at overload when the information is presented at a rate too fast to be processed by a single person [SF74].

As stated by Ho and Tang [HT01], the Web is a primary cause of information overload and both Web 2.0 and the recent phenomenon of Big Data (see Figure 1.1.1) further worsened the problem. Due to the exponential growth of available digital content, currently we are moving towards the so-called "digital transition". This means that the most of the information is available in digital form instead of hard copy. Furthermore, the spread of mobile devices (there are currently 1.3 billion gigabytes of data traffic on mobile networks only) and the advent of social networking has made physiologically impossible to follow the flow of information, despite Nielsen just showed that users spend about 22% of their navigation time on social networks (and almost 10% only on Facebook, equivalent to the traffic of Google, Wikipedia, YouTube and Amazon together). Even more interesting are the data about the videos uploaded on Youtube: more than 20 hours of video every minute. The amount of videos uploaded to YouTube in two months is equivalent to the amount of TV shows broadcasted by ABC, NBC and CBS from 1948 to present. Furthermore, 30 billion pieces of content are shared on Facebook every month [MCB11]. So, although the general scenario is well-known and already discussed, the current data show us that the problem has a scope that is much more relevant than the past.

As underlined by some research [EM00], it is well-established that work efficiency is closely related to the amount of information that needs to be processed. As long as the level of information is adequate, the efficiency stays acceptable but it rapidly decays when the amount becomes unbearable anymore. Often, Information Overload is seen as a problem that can decrease both productivity and quality of life of the individuals affected by the problem. A poll by Reuters named "Dying for Information" [LR96] identified three types of problems that affect people involved in information overload scenarios: job satisfaction, difficulty of establishing relations and health problems.

Next to those who consider the information overload as a disease [Hal05] that brings with it attention deficits, anxiety, cybercondria and so on, there is

---

1.1 Background

Figure 1.1: The exponential growth of digital information

someone else that does not consider it a real problem, underlining the chance to ignore or minimize its impact by simply introducing some behavioral shrewdness [Smi10]. Finally, a neutral point of view arises from Clay Shirky. He suggests a more critical vision, pointing out that today’s problem is not the abundance of information but rather the absence of appropriate filters that support our physiological brain deficits and help us to select the most important pieces of information. According to Shirky, the key is to filter the information.4

However, as stated by Davis [Dav11], the absence of filters is only one side of the problem of information overload. The development of effective filtering techniques does not solve the problem by itself. To reduce the information is not enough, it is first necessary to reduce the "noise" as well as our eyes do when they perceive the information. Beyond the concept of information abundance, the recent development of the blogosphere introduced the topics of trust, quality and diversity of web content [Cro05]. Nowadays web readers need the ability to select and properly certify the information available on the Internet. This concept, named Information Literacy [Vir03], underlines that the ability to evaluate sources of information is an essential requirement, and tools for Intelligent Information Access need to handle also this side in order to effectively filter the information and aid the users to extricate in this continuous flow of data.

1.1.2 Intelligent Information Access

When we talk about "having access to information" we refer to concrete tasks such as performing searches, filtering results, aggregating similar information, interpreting data and so on. Usually, all these tasks need to handle big collections of semi-structured or even non-structured data. To make the information access more "intelligent", or, in general, more efficient, it is necessary to introduce some techniques to facilitate these activities, in order to decrease the time needed to perform each task and to increase both overall accuracy and user satisfaction.

In general, Intelligent Information Access encompasses a wide group of technologies, ranging over Information Retrieval, Information Extraction, Text Clustering, Information Filtering and so on [KS01].

Information Retrieval (IR) concerns the finding of relevant information from a collection of data (usually unstructured text) [SM84]. Search engines, such as Google and Bing, are typical examples of applications in the area of IR. A formal characterization of an IR model is given by Baeza-Yates and Ribeiro-Neto [BYRN99]. Generally speaking, the goal of a IR systems is to lead the user to those documents that will satisfy the best her own information needs. User information needs are usually represented by means of a query, expressed in a language understood by the system. In the typical workflow of an IR system, the query is submitted to the search engine, whose goal is to understand the meaning of user request and to identify the most relevant pieces of information among those that are stored in the collection. Next, a ranking function orders the documents in a descending relevance criterion and the top entries are finally returned to the user.

In IR research particular interest has been covered by the optimization of ranking functions. Beitzel [BJL+07] pointed out that users usually consult only the results put in the first page or even the first entries of the first page, so it is enormously important to put the most relevant results at the top of the returned list. In 2009 Google introduced in their search algorithms some tip to take into account also personal and contextual information, in order to improve the ranking of the returned results. The exploitation of these information shifted the classical search towards the personalized and contextual search.

However, even in personalized search, the target user needs to explicitly express her own informative need as a query. As stated by Ingwersen and Willett [IW07], to build a query statement representative of searcher information need is a very problematic and challenging task. The exploitation of a limited

vocabulary only based on keywords is of hindrance for users because it forces to use a language too far from the one they used to express their own needs. The problem is further amplified after studies [JSS00] that showed the short length of queries (2.35 terms on average).

Unlike IR, the goal of Information Filtering (IF) is to expose users with the information they want [HSS01] without the need of an explicit query that triggers the whole process. As already pointed out by O’Brien⁶, the development of IF systems is a first step for shifting the paradigm of classical search towards a discovery one, that is to say a scenario where the information is automatically pushed to the users instead of being pulled according to an explicit query. Even if both IR and IF have the goal to optimize the access to unstructured information sources, they have a strictly methodological difference: IF systems are not designed to find relevant information pieces, but rather to filter out the noise from a generic information flow according to some criterion (for example, user preferences).

According to Malone [MGT⁺87], the approaches for Information Filtering can be classified in three different categories: cognitive, economic and social:

- **Cognitive filtering**: based on content analysis.
- **Social filtering**: based on individual judgments of quality communicated through personal relationships.
- **Economic filtering**: based on estimated search cost and benefits.

In the first case the filtering is carried out by simply analyzing the content associated to each informative item. Whether it contains (or, alternatively, it does not contain) specific features, it is filtered out. A typical scenario where this filtering approach is applied is spam detection in e-mail clients. The social filtering complements the cognitive approach by focusing on the characteristics of the users, for example relying on explicit relationships (e.g. an e-mail message received from the supervisor has to be considered as relevant) or by analyzing the users in order to bring out similarity or behavioral patterns that can be exploited to forecast their future behavior (e.g. whether all the users in my group considered a message as relevant, it should be relevant for me, too). Finally, the economic-filtering approach relies on various kinds of cost-benefit assessments and explicit or implicit pricing mechanisms. A simple cost-versus-value heuristic is the length of an e-mail message.

The set of techniques that can be exploited to filter the information flow is very wide, ranging from simple heuristics to complex statistical models. When

a filtering system takes also into account the information about a specific user, being able to deliver a tailored service, it is common to refer to it as personalization systems [MAB00].

A typical example of personalization systems is represented by Recommender Systems [RV97], an emerging research direction in the area of Intelligent Information Access and Information Filtering systems. The term was introduced to define systems able to provide users with suggestions about products and services they may like. The scenario a recommender system deals with is a very natural and social process, because people get advice all the time, especially when it is necessary to choose among different alternatives and the knowledge to critically discern them is not enough. The suggestions can regard various decision-making processes: items to buy, news to read, music to listen to, restaurants to try and so on [RRSK11]. The recommendation task is achieved by exploiting various knowledge sources which store the information collected during past user interactions with the system. Since recommendations are usually calculated by analyzing historical data, they are based on the assumption that user preferences (or, in general, user needs) stay stable and do not change over time. Even if this is a very strong assumption, the applications of recommendation technologies in many domains [LSY03] have already reported success stories regarding the use of these systems in real-word scenarios [PGG+10].

1.2 Research Motivations

In this dissertation, the problem of information overload will be tackled using an approach strictly related to recommender systems. Specifically, the attention will be focused on the so-called content-based recommendation approaches [PB07]. In these models the relevance of an item for a target user is calculated according to the relevance that other "similar" items seemed to have for him in the past. The concept of similarity is based on the analysis of textual content or, in general, on the features that describe the items. The assumption behind content-based recommender systems (CBRS) is very simple: the more a feature occurs in the items considered as relevant, the more is the likelihood that a new item where this feature occurs can be considered as relevant again.

Belkin and Croft in [BC92] pointed out that IF (and specifically cognitive filtering, later evolved in the content-based recommender systems) can be considered as strictly related to IR, since they share the goal of optimizing the access to unstructured data sources but they differ in the aspects that follow:

- **Frequency of use** - IR systems are designed for ad-hoc use while IF
1.3 Contributions of the Thesis

With this thesis we are aiming at advance the state of the art of content-based recommender system. Specifically, the contributions of the thesis can be summarized as follows:

- Analysis of the approaches based on distributional analysis for semantic content representation in CBRS. Most of the approaches in the area of CBRS deal with simple keyword-based representation of systems are designed for long-term users:

- **Representation of information needs** - In IR systems user needs are expressed as queries, in IF systems user needs are modeled by means of user profiles;

- **Database** - IR deals with relatively static databases, while IF deals with more dynamic data;

- **Type of users** - In IR systems users are not known, while in IF the system has a model for each user;

- **Scope of the systems** - In IR systems issues as user modeling and privacy are not considered.

Twenty years later the scenario drastically changed. The only difference that is still standing is the representation of user needs, since IR systems still work in a pull mode and they need an explicit query to trigger the retrieval process. All the other differences are no more in place: first, the use of user profiles is needed also in IR systems in order to produce personalized search results. Furthermore, in order to build and maintain a user profile, IR applications need to take into account privacy issues as well. Finally, the recent trend of real-time search introduced the need of evolving and adapting IR systems to support the retrieval of a continuous flow of changing unstructured data, such as social network streams, RSS feeds and so on.

By summing up, nowadays the overlap between IR and IF is much stronger than the past. So, the goal of the research described here is to investigate whether is it possible to exploit the convergence and the methodological overlap argued by Belkin and Croft to introduce novel recommendation models based on well-known IR techniques, adapted and evolved according to the specific requirements of a recommendation scenario.
content, even if there is no doubt that these representations are not able to catch all the facets of a language. Since the costs-benefits ratio between semantic and syntactical models is still on behalf of the latters, lightweight approaches for introducing semantics in CBRS will be investigated. Specifically, more complex representations of textual content based on distributional models [TP10], that are supposed to catch the semantics of a language without the need of a complete natural language processing (NLP) pipeline, will be analyzed. They will be exploited to infer a richer representation of both items and user profiles. Since most of the approach based on distributional analysis relies on large vector-space representation, techniques for dimensionality reduction that ensure scalability and effectiveness will be introduced. These are primary requirements for systems that deal with continuously changing unstructured data.

- **Analysis of approaches for managing negative user feedbacks in CBRS.** Even if IR system can effectively work without taking into account the information coming from user feedbacks, modeling and catching negative user preferences, as pointed out by Zeng et al. [ZZLZ11], is a key aspect for CBRS since they continuously need to build and maintain an accurate descriptions of user needs. So, several state-of-the-art models for modeling negative user feedbacks will be analyzed and the most promising will be introduced into a novel content-based recommendation model.

- **New algorithms for content-based recommendations in monolingual and multilingual environments.** The Vector Space Model [SWY75] will be adopted as main building block, since it emerged as one of the most effective approaches in the area of IR thanks to its good compromise between expressivity, effectiveness and simplicity. Next, it will be adapted for the exploitation in a recommendation scenario by catching its typical issues, this is to say, the scalability and inability to manage both the semantics of textual content and negative user feedbacks. This model will be named eVSM, acronym of enhanced Vector Space Model, since it is supposed to evolve the classical VSM by overcoming its typical drawbacks, improving its accuracy, and allowing its application for recommendation tasks as well.

- **New approaches for tackling cold-start problem based on social media.** A typical issue of recommender systems is represented by the cold start, that is to say the inability to trigger the recommendation process when the system has no knowledge about user preferences. Several meth-
ods to handle this problem have been proposed, such as [SPUP02]. In this thesis a novel approach based on the exploitation of data extracted from social media sources will be introduced to overcome this problem. The inceptive idea is to gather and crawl the information left by users on social networks and to model them in order to automatically trigger the recommendation process without explicit information about user preferences.

• Implementation and evaluation of system prototypes. The proposed models will be evaluated in both in-vitro and in-vivo scenarios. In the first case classical state-of-the-art datasets and the typical measures based on accuracy and precision will be used, while in the latter it will be developed some ad-hoc web application whose goal will be to let real users evaluate the recommendation produced by eVSMs in several applications domains such as news, music and TV-shows recommendation.

1.4 Organization of the Thesis

The thesis will be organized as follows:

• Chapter 1 - Introduction - gives a contextualization of the work and presents a brief explanation of its motivations. The general concepts are clarified and a description of the contributions of the thesis is given.

• Chapter 2 - Recommender Systems - provides an overview of the literature in the area of recommender systems. All the recommendation approaches are presented. The attention is mainly focused on the collaborative and content-based ones: strengths and weaknesses of each model are pointed out and an analysis of the research trends in the area of RS is provided.

• Chapter 3 - eVSM: enhanced Vector Space Model for Content-based Recommender Systems - is the core of the thesis. First, a description of VSM is given and then an overview of the techniques used to overcome its typical drawbacks is provided. Next, eVSM are described by focusing the attention on both techniques for learning user profiles and models for providing recommendation in mono-lingual and multi-lingual environments.

• Chapter 4 - Applications of eVSM - introduces the applications developed to analyze the effectiveness of eVSMs in offline as well as online
scenarios. For each application a specific experimental session is designed and the results are presented.

• **Chapter 5 - Industrial Applications of eVSM** - focuses the attentions on the experimental evaluations designed for the industrial applications of eVSM.

• **Chapter 6 - Summary and Conclusions** - concludes the thesis by summarizing the goals of the thesis and pointing out the contributions given. Finally, an analysis of future directions for this research is provided.

1.5 List of Publications

The ideas and the results presented in this thesis are part of various peer-reviewed research papers. In this subsection we give a list of the most relevant publications grouped by type and ordered by date.

**International Journals**


**International Volumes**


1.5 List of Publications

International Conferences


National Conferences

International Workshops


National Workshops

- Cataldo Musto, Fedelucio Narducci, Pierpaolo Basile, Pasquale Lops, Marco de Gemmis, Giovanni Semeraro: *Comparing Word Sense Disambiguation*


Chapter 2

Recommender Systems

Recommender systems (RS) deal with a very natural and common process: providing suggestions.

In fact, people get advice all the time, especially when it is necessary to choose among different alternatives and the knowledge to critically discern them is not enough. This is a very typical situation. Sometimes we need suggestions about what movie to watch in a rainy night or which one is the best paper to read about a specific research topic, but the scenario is very common and may be extended to several scopes: music to listen to, books, web pages or news to read, electronic devices to buy, restaurants to try, and so on. The list could be infinite.

People use a lot of strategies to solve decision-making processes [JZFF11]. Sometimes human recommendations are adequate to effectively face the problem (for example, to find the best restaurants in our own city), but usually they do not because the information held by our friends is not enough to provide good suggestions in all the cases. Furthermore, decision-making processes are much more difficult in the Big Data era: the more the number of possible choices (for example, which digital camera to buy in an online store), the more the difficult to evaluate the overwhelming number of alternatives and the more the need for aids that guide us sifting this flow of choices.

This is a well-known problem, usually referred to as Paradox of Choice [Sch05]. Indeed, when the number of possible alternatives is too big with respect to the knowledge that the individuals hold about a problem, to choose become difficult and it is common to fall in the so-called paradox of Buridan [Res59], namely, the inability to make decisions when too much alternatives are available. As stated by Leibniz, in things which are absolutely indifferent, there can be no choice and consequently no option or will, since choice must have
some reason or principle. In other terms, when knowledge is not enough the individuals need to be assisted in decision-making processes, and IF systems like RS are the tools that can best cope with these tasks, as already demonstrated by many successful use cases [LSY03, DDGR07].

2.1 Basics of Recommender Systems

The area of RSs is relatively new, since the first studies about this topic have been proposed around mid-1990s [GNOT92, HSRF95, SM95]. Anyway, the roots of RSs can be traced back to the extensive work in cognitive science [Ric79], approximation theory [Pow81] and Information Retrieval [SB88]. Furthermore, RSs have also a specific connection with consumer choice modeling in marketing [LKM92].

The concept of RS was introduced [RV97] to formally define tools and techniques able to provide personalized information access to large collections of structured and unstructured data, and, specifically, to provide users with advices about items they might be interested in. As stated above, the term items refers to products (such as movies, songs, books, web pages, electronic devices, etc.) or generic services (such as restaurants, travels, jobs or events).

Beyond the inceptive idea, several definitions of what a RS is have been
Recommender Systems

proposed in literature. Olsson [Ols03], for example, defines a RS as a system that helps a user to select a suitable item among a set of selectable items using a knowledge-base that can be hand-coded by experts or learned from recommendations generated by the users. Pemberton [PRP00] introduces a definition more focused on collaborative approaches, since a RS is defined as an example of adaptive filters that uses inferences drawn from users’ known behavior to recommend documents they have not seen. Finally, the most fitting definition is probably provided by Burke [Bur02]: recommender systems have the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options. This definition underlines that RSs are user-oriented tools whose goal is to guide users towards the information they are more interested in.

2.1.1 The Recommendation Problem

A formal definition of the recommendation problem can be formulated as follows [AT05].

Let $U$ be the set of all users and let $I$ be the set of all possible items that can be recommended. Each element of the user space $U$ can be defined as a profile that includes various user characteristics, such as age, gender, income, marital status, etc. In the simplest case, the profile contains only a single element, such as a unique identifier of the user. Similarly, each element of the item space $I$ is defined through a set of features. For example, in a movie recommendation scenario each movie can be represented by means of its genre, its title, the name of the director and so on.

Usually the cardinality of the sets $I$ and $U$ is very large, ranging in hundreds of thousands or even millions of items in some real use case.

Let $f$ be a utility function that measures the usefulness of item $i$ to user $u$, i.e., $f : U \times I \rightarrow R$, where $R$ is a totally ordered set (for example, a set of integer or real numbers within a certain range). The recommendation problem consists in choosing such item $i'_u \in I$ that maximizes user’s utility for each user $u \in U$. More formally:

$$
\forall u \in U, i'_u = \arg \max_{i \in I} f(u, i)
$$

In recommender systems, the utility function is usually represented by means of a rating, which indicates how a particular user liked a specific item, or by means of values ranging between 0 and 1, as in classical statistical models.

The central problem of recommender systems lies in that utility $f$ is usually not defined on the whole $U \times I$ space. In recommender systems where utility
is represented by ratings, $f$ is initially defined only on the items previously rated by the users. For example, in a movie recommendation application, users initially rate some subset of movies they have already seen. Therefore, the recommendation engine should be able to predict the ratings of the non-rated movie/user combinations and issue appropriate recommendations based on these predictions. Once the unknown ratings are estimated, actual recommendations are made by selecting the highest ratings among all the estimated ones. The ratings of the not-yet-rated items can be estimated in many different ways using methods ranging from simple heuristics to complex theories, such as machine learning or approximation theory. The classification of recommender systems is usually done according to their approach to rating estimation.

### 2.1.2 A Recommendation Pipeline

Regardless specific definitions and approaches used to generate recommendations, a classical recommendation pipeline can be easily uttered in the following steps:

1. **Training:** first, the system needs to acquire information about a target user (what she knows, what she likes, the task to be accomplished, demographic or contextual information and so on). This step could be accomplished in an *explicit* or *implicit* way. In the first case the user explicitly expresses her preferences (by means of a numeric scale, for example) on randomly chosen items, while in the latter user preferences are gathered by analyzing her transactional or behavioral data (for example, clicking a link or reading a news article could be considered as a clue of user interest in that item). The analysis of social media-based information sources is also a recent trend for implicit user modeling, as proved by Bu et al. [BTC+10] and Oghina et al. [OBTDR12].

2. **User Modeling:** in general, the concept of *personalization* implies the presence of something describing and identifying the user that interacts with the system. So, the information extracted are usually modeled and stored in a *user profile*. Modeling the user profile is a central step of the pipeline since it is the component that triggers the whole recommendation process. The choice about which information have to be stored and the way the user profile is built, updated and maintained are generally strictly related to the specific filtering model implemented by the system. For example, in a generic content-based recommender architecture [LdGS11] the component for *User Modeling* can be split in a *Content Analyzer*, whose goal is
Recommender Systems

21

to analyze content in order to pop up relevant concepts from unstructured text, and a Profile Learner that stores these concepts into user profiles.

3. Filtering: finally, the information flow is filtered out by exploiting data stored in the user profile. The goal of this step is to rank the items according to a relevance criterion and to provide user with a list of the most relevant items in order to let her express her own feedbacks on the proposed ones. Formally, at the end of the filtering step the system returns a subset of items ranked in a descending relevance order.

The main differences among the different recommendation techniques lie in the way profiles are built and the problem of predicting ratings for unseen items is caught. Common classifications [Bur07] split recommendation models into six classes:

1. Content-based Recommender Systems (CBRS). This class of RSs suggests items similar to those preferred in the past by the user.

2. Collaborative Recommender Systems. This class of RSs suggests items preferred by users with similar needs or preferences.

3. Demographic Recommender Systems. This class of RSs suggests items on the ground of the demographic profile of the user.

4. Knowledge-based Recommender Systems. This class of RSs suggests items whose features meet user needs and preferences according to specific domain knowledge.

5. Community-based Recommender Systems. This type of system recommends items based on the preferences of users friends.

6. Hybrid Recommender Systems. This class of RSs combines two or more recommendation techniques to overcome the typical drawbacks of each approach.

To sum up, in most of the above approaches particular emphasis is put on the concept of similarity. In CBRS the concept of similarity is exploited to identify the items that share common features with those the user already considered as relevant or interesting in the past. In collaborative ones it is used to bring out hidden connections between users that belong to the same community and share similar tastes. In the same way, in demographic and community-based recommender systems the prediction of the degree of interest of a certain item
for a particular user is obtained by means of similarity calculations between demographic or social profiles.

In the next section, a description of the recommendation models and a thorough analysis of their strength and weaknesses is provided. However, the attention will be focused on content-based recommender systems since they represent the core of this dissertation. Collaborative recommender systems are described and discussed in a separate section because they represent the most successful use case in the area of RSs, while other approaches are grouped and summarized in a single section because these models are out of the scope of this thesis.

2.2 Content-based Recommender Systems

Content-based Recommender Systems (CBRS) are based on the assumption that user preferences remain stable over time. Indeed, they suggest items similar to those previously labeled as relevant by the target user.

This class of RSs is strongly based on the analysis and exploitation of textual content since their only prerequisite is that each item to be recommended has to be described by means of textual features. The set of the attributes is strictly domain-dependent: in a book recommendation scenario each item may be described through its author, its genre, the plot and so on, in a movie recommendation one the author could be replaced by the director and the cast, while in a news recommender only the content of the news and the information about the author of the article may be taken into account. The value of the attributes associated to each item can be directly extracted from structured data (for example, the genre of book or the director of a movie) or extrapolated by analyzing unstructured text (for a news article, it is common to exploit only the most significant words in it).

Content-based personalization basically needs two pieces of information: a textual description of the item and a user profile describing user interests in terms of textual features. In a movie recommendation scenario, for example, a user profile may store the preferred genres, directors and so on. Given these information, the recommendation step consists in matching up the characteristics of an unseen item with those describing user interests [TH01]. The result is a relevance judgment that represents the user’s level of interest in that item. Basically, the larger the overlap between the features, the greater the relevance of the new (unseen) item for the user. Obviously, item description and user profile have to share the same vocabulary in order to calculate how much is the overlap between them.
In general, models for CBRSs differ in the way they answer to the following questions:

- How are textual content processed and represented?
- How are user profiles learnt?
- How is the relevance score calculated?

**Content representation.** As stated above, CBRS strongly rely on the analysis of textual content; as a consequence, an effective representation of the information is tremendously important. Sometimes textual content are provided in a structured form (such as movie genre), thus they can be extracted and represented as a simple feature, but sometimes they do not, thus some processing is needed. If we think about a news recommendation scenario, for example, it is impossible to represent the content of the article as it is. Consequently, when CBRS have to represent textual content, Natural Language Processing (NLP) techniques usually come into play. The first attempts to catch these problems were based on the use of simple heuristics, such as the so-called size cutoff. In Syskill & Webert [PB97], for example, each item is represented by means of its 128 most informative words. Similarly, Fab [BS97] adopts a representation based on 100 words occurring in the item description. The optimal number of words to be used was determined experimentally by the Syskill & Webert system for several domains. The evaluation showed that with less than 50 keywords it is likely to exclude important features, while with more than 300 some noise is introduced.

In order to identify the most informative words in a documents it is common to adopt some weighting scheme, such as the classical TF-IDF [SWY75]. The TF-IDF (term frequency-inverse document frequency) is an established technique in the field of IR. The inceptive idea behind it is that the weight of a certain features within a corpus of textual content (documents) is related to two factors: its frequency and its commonality. Specifically, the TF-IDF weight for a keyword $i$ in document $j$ is computed as the product of these two measures:

$$
TF-IDF(i, j) = TF(i, j) \times IDF(i)
$$

The TF (term frequency) calculates how many times the term (feature) $i$ occurs in the document $j$. Since this component is strongly dependent on document length, it is common to couple TF with some normalization scheme. In [Cha02] a relatively simple one is proposed: it relates the occurrences of
2.2 Content-based Recommender Systems

a term to the maximum frequency of the other terms within the document. Formally:

\[ TF(i, j) = \frac{freq(i, j)}{maxOthers(i, j)} \]  

(2.3)

Where \( freq(i, j) \) is the number of occurrences of term \( i \) in document \( j \) and \( maxOthers(i, j) \) is the maximum frequency among all the terms occurring in document \( j \).

In the same way, IDF (inverse document frequency) aims at reducing the weight of very common keywords. The underlying idea is that frequent words are not very helpful to discriminate among documents, thus more weight should be given to words that only appear in a few documents. Let \( N \) be the number of all documents and \( n(i) \) be the number of documents in which keyword \( i \) appears. The inverse document frequency for \( i \) is typically calculated as:

\[ IDF(i) = \log \frac{N}{n(i)} \]  

(2.4)

However, even if TF-IDF is an effective weighting schema, it is totally based on empirical observations regarding text and is not able to effectively deal with the typical issues of natural languages, thus some further processing is usually needed.

First of all, textual content are often noisy and full of poorly informative words such as articles, pronouns, etc. This problem is commonly tackled through techniques typically borrowed by NLP, such as stop words removal, stemming (which aims to replace variants of the same word by their common stem) and POS-tagging (which assigns to each term its correct part-of-speech). The exploitation of Porter algorithm [Por80], for example, is the typical way to introduce a stemmed representation of textual content.

Furthermore, traditional keyword-based representations are unable to capture the semantics of both user interests and item descriptions since they suffer of problems such as polysemy, that is to say the presence of multiple meanings for one word, and synonymy, term used to describe the scenario where multiple words share the same meaning. Semantic analytics is one of the most innovative and interesting branches of the text analytics proposed in literature to solve these problems. The underlying idea is that the use of knowledge bases, such as lexicons or ontologies, can help to annotate items and representing profiles in order to obtain a “semantic” interpretation of user information needs.
Some preliminary attempts of introducing semantics in CBRS content representations by exploiting algorithms for Word Sense Disambiguation (WSD) has been proposed by Semeraro et al. [SDLB07].

To sum up, by following the notation introduced in [AT05] we can formally represent the textual description of item \( i \) as the set \( \text{Content}(i) \), that is to say, the set of features that describes \( i \). Each feature \( f_j \) is usually assigned with a weight \( w_{i,f_j} \) indicating the importance of the feature \( j \) in the description of the item \( i \).

\[
\text{Content}(i) = \{ w_{i,f_1}, w_{i,f_2} \ldots w_{i,f_n} \} \quad (2.5)
\]

Intuitively, \( \text{Content}(i) \) is a set of features filtered and processed through a NLP pipeline and \( w_{i,f_k} \) is the weight associated to the feature \( f_k \) that might be represented by the TF-IDF introduced above.

**Learning User Profiles.** Analogously, the content-based preferences of user \( u \) can be represented as a set named \( \text{ContentBasedProfile}(u) \), defined as follows:

\[
\text{ContentBasedProfile}(u) = \{ w_{u,f_1}, w_{u,f_2} \ldots w_{u,f_n} \} \quad (2.6)
\]

In other terms, in CBRS the user profile consists of a set of features with their corresponding weights. Techniques for modeling user profiles may vary from simple boolean representations (for example, in a movie recommendation scenario each feature may be a movie genre and the value can be set to 1 or 0 whether the user is interested in that genre) to more complex ones such as those based on vector spaces or statistical models. Usually, the choices about which features to store in the user profile and which weighting scheme is the best to be used are strongly coupled with the recommendation model and the scoring function adopted in the CBRS.

Regardless the techniques for profile representation, in order to construct and update a user profile it is necessary to collect as much as possible information about user preferences or tastes. These data are usually gathered during the so-called training phase, where user behavior is analyzed in order to implicitly or explicitly infer her preferences. Typically, it is possible to distinguish between two kinds of information, that is to say explicit and implicit ones: when a system asks the user to explicitly evaluate items, this technique is usually referred to as explicit feedback. There are two main approaches to get explicit relevance feedback: in the simplest case, the user can classify the items as relevant or not relevant by adopting a simple binary rating scale, such as in [BP99].
Alternatively, a discrete numeric scale is usually adopted to judge items, such as in [SM95]. Explicit feedback has the advantage of simplicity, albeit the adoption of numeric scales increases the cognitive load on the user.

The other technique, called implicit feedback, does not require any active user involvement, since user feedback is derived from monitoring and analyzing her activities. Approaches for gathering implicit feedback [OK98] range from simple methods based on relevance scores assigned to specific user actions, such as saving an item, printing it and so on, to more complex ones involving the analysis of free text. In [RIS+94], for example, Resnick et al. propose the analysis of comments to infer user preferences. A recent trend is to infer user preferences by coupling the analysis of free text with sentiment analysis, as in [LCC06], or by exploiting the information coming from social networks [BF10].

To sum up, at the end of the process of profile construction, a representation of user preferences, in terms of features and corresponding weights, is obtained. The way this information is finally exploited to provide users with recommendations depends on the scoring function implemented in the recommendation model.

**Scoring function.** By following the above notation, the utility of item $i$ for user $u$, described as $f(u, i)$, can be computed as a score function that calculates how relevant is item $i$ for user $u$ by matching item description with user profile.

\[
    f(u, i) = score(ContentBasedProfile(u), Content(i))
\]  

The way score function is formulated distinguishes the different content-based recommendation techniques proposed in literature. These can be classified in *heuristic-based* and *model-based approaches*. The formers calculate utility according to heuristic formulas that are mostly inspired by information retrieval methods, such as cosine similarity. The latter predict the relevance relying on models learned from the underlying data through statistical or machine learning-based techniques.

**Heuristic-based approaches.** This class of approaches has its roots in Information Retrieval (IR) [BYRN99]. Indeed, as stated by Belkin and Croft [BC92], IR tasks can be considered as strictly related to Information Filtering (IF) ones since they share the same goal (to introduce techniques to effectively filter the overwhelming information flow) and they differ only a little (IR tools, such as search engines, filter the information flow based on an explicit query while IF ones exploit user profiles). Consequently, the main idea behind
heuristic-based approaches for CBRS is to adopt typical IR techniques to model both items and user profiles and to identify the items that are relevant the most for the user.

A typical approach consists in representing both user profile and items as weighted term vectors, since the above formulas (such as TF-IDF and cosine similarity) fit with a vector space representation where each feature is mapped on a dimension of the space. Given this representation, it is common to reason by analogy and to exploit the user profile as a query. So, by adopting similarity measures it is possible to calculate how similar a point \( (item) \) in that space is with respect to the user profile, with the assumption that the items with the highest cosine similarity are those that best catch user preferences, as it happens in classical search engines.

In literature many attempts to describe the proximity between two vectors have been proposed, such as those introduced by Jaccard [Jac01], Dice [Dic45] or Tversky [Tve77]. In content-based recommender systems, cosine similarity is undoubtedly the most widely used measure to compute the proximity between vectors. Specifically, given two vectors \( d = \{d_1 \ldots d_n\} \) and \( u = \{u_1 \ldots u_n\} \), their cosine similarity \( cosSim(d, u) \) is calculated as follows:

\[
\text{cosSim}(d, u) = \frac{\sum_{i=1}^{n} d_i * u_i}{\sqrt{\sum_{i=1}^{n}(d_i)^2} * \sqrt{\sum_{i=1}^{n}(u_i)^2}}
\]

(2.8)

Thanks to this measure it is possible to rank the items in a descending relevance order according to their similarity scores. Next, the top \( n \) items in this ranking (or those overcoming a certain similarity threshold) can be considered as relevant and recommended to the target user.

A common alternative for similarity-based models is to implement a CBRS as a kNN classifier. By exploiting generic similarity measures the algorithm looks for the \( k \) most similar items among those represented in the vector space and recommends the items whose nearest neighbors (or at least most of them) were already labeled as interesting or relevant by the target user. In [BPC00] the kNN method is used to model short-term interests of the users.

**Model-based approaches.** This class of approaches is much more related to Machine Learning and Artificial Intelligence. In general, they usually build a model of user interests and exploit it to provide users with recommendations. Differently from heuristic-based approaches, where the recommendation task was considered as similar to a retrieval one, in model-based approaches the task is viewed as a form of classification, where each item has to be labeled as relevant or non-relevant for the target user according to a set of manually annotated
training documents. Once the content-based recommendation task has been formulated as a classification problem, various machine learning techniques can be applied such that a CBRS can automatically decide whether a user will be interested in a certain document: the most common model-based approaches fall into the categories of probabilistic methods and linear classifiers.

In the first case CBRS is implemented as a Naïve Bayes text classifier. Naïve Bayes is a probabilistic approach to inductive learning, and belongs to the general class of Bayesian classifiers. These approaches generate a probabilistic model based on previously observed data (in CBRS scenario, user ratings on previously enjoyed items). The model estimates the a posteriori probability, \( P(c|i) \), of item \( i \) belonging to class \( c \), where \( c \) is commonly referred to as the class of the items considered as relevant by the target user. This estimation is based on the a priori probability, \( P(c) \), the probability of observing an item in class \( c \), \( P(i|c) \), the probability of observing the item \( i \) given \( c \), and \( P(i) \), the probability of observing the instance \( i \). Using these probabilities, the Bayes theorem is applied to calculate \( P(c|i) \) as follows:

\[
P(c|i) = \frac{P(c)P(i|c)}{P(i)} \tag{2.9}
\]

To classify the item \( i \), the class with the highest probability is chosen:

\[
c = \arg \max_{c_j} \frac{P(c_j)P(i|c_j)}{P(i)} \tag{2.10}
\]

In a typical CBRS scenario, \( j \) assumes two values since it is set to + and − to define the classes \( c_+ \) and \( c_- \), that is to say, the class of the items considered as relevant for the target users and the one containing the uninteresting items.

In the formula 2.10 \( P(i) \) is generally removed as it is equal for all \( c_j \). The values for \( P(i|c) \) and \( P(c) \) are usually not known, thus are estimated by observing training data. However, estimating \( P(i|c) \) in this way is problematic, as it is very unlikely to see the same document more than once: the observed data is generally not enough to be able to generate good probabilities. The Naïve Bayes classifier overcomes this problem by simplifying the model through the independence assumption: all the features in the observed document \( d \) are conditionally independent of each other given the class. Individual probabilities for the words in a document are estimated one by one rather than the complete document as a whole. Even if the conditional independence assumption is clearly violated in real-world data, empirically evaluations showed [LR94] that the naïve Bayes classifier does a good job in classifying text documents.
Regarding the ways of modeling the documents and their features two distinct models have been proposed: the *multinomial model* and the *Bernoulli* one. The main difference between these models lies in the way the probabilities are estimated from the training data. In the multivariate Bernoulli model, a document is treated as a binary vector that describes whether a certain term is contained in the document. In the multinomial model the number of times a term occurs in a document is also taken into account. In [MN98] it is shown that multinomial model outperforms Bernoulli one, in particular when it deals with longer documents and high dimensional feature spaces.

A typical alternative to probabilistic methods is to implement CBRS through linear classifiers. These learning methods aim to find the coefficients of a linear model to discriminate between relevant and non-relevant documents. Figure 2.2 sketches a simplified two-dimensional classification problem. With only two dimensions the classifier can be represented by a line but the idea can easily be generalized to the multidimensional space in which a two-class classifier then corresponds to a hyperplane that represents the decision boundary.

Support Vector Machines (SVM), described in [Joa98], are probably the most effective implementation of CBRS as linear classifier. The goal of this approach is to identify decision boundaries that maximize the distance (called margin) to the existing datapoints. An evaluation of different techniques for text classifiers is provided in [YL99], where it is shown that SVM-based algorithms perform better than others even if no strict guideline about the technique that best performs in every situation still exists [MRS08].

Finally, some early work investigated the use of other techniques based on
Machine Learning approaches, such as decision trees. [KLS+01] describes an implementation of decision trees for personalizing the set of advertisements appearing on a web page. However, as proved in [PB97], decision trees do not fit recommendation scenarios and usually perform poorly. The main reason for this limited performance lies in the high number of features a recommender system usually has to deal with whereas decision tree learners work better when a relatively small number of features exists.

2.2.1 Limitations of Content-based Recommender Systems

Content-based recommender systems have several limitations, which have been identified in the literature [BYRN99, Bur02, AT05]. The most relevant ones are:

- **Limited content analysis.** Content-based recommendations are constrained by the features that are explicitly associated with the items to be recommended. For example, content-based movie recommendations can only be based on materials written about a movie: actors’ names, plot, genres. The effectiveness of these techniques thus depends on the available descriptive data. Therefore, in order to have a sufficient set of features the content should be either in a form that can be automatically parsed by a computer or in a form in which the features can be manually extracted in an easy way. In many cases, these requirements are very difficult to fulfill. There are some domains where automatic feature extraction is complicated and to assign features by hand is often not practical. For instance, even if some recent attempt underlines the need for further research in this direction [LOL03], it is much harder to apply automatic feature extraction methods to multimedia data such as graphical images, video streams, and audio streams, than it is for text content. A recent trend is to enrich content representation by means of external knowledge sources, such as encyclopedical ones. The Explicit Semantic Analysis (ESA), introduced in [GM06], proposes an indexing technique based on content gathered from Wikipedia articles. An early attempt of coupling CBRS with techniques for knowledge infusion is proposed in [Nar12].

- **Content over-specialization.** CBRS retrieve items that score highly against a specific user profile. Content-based techniques cannot recommend items that are different from anything the user has seen before. Thus, for instance, a person with no experience in ambient music will never receive recommendations about that genre if she never enjoyed something
at least similar to it. To overcome such limitations it may be appropriate to introduce some randomness in the recommendations [SM95]. Alternative approaches, such as that implemented in DailyLearner [BP00], propose to filter out items not only if they are too different from user’s preferences, but also if they are too similar to something the user has seen before. Furthermore, in [ZCM02] a set of five redundancy measures is provided, in order to evaluate whether a document that is deemed to be relevant contains some novel information as well. A general goal therefore is to increase the serendipity of the recommendation lists by including “unexpected” items in which the user might be interested in, as proposed by Iaquinta et al. in [IdGL+08].

- **Portfolio effect.** In certain cases, items should not be recommended if they are too similar to something the user has already seen. To avoid this problem, the user should be presented with a diverse range of options, and not with a homogeneous set of alternatives. For example, it is not necessarily a good idea to recommend a track played by the front-man of a rock band the user already likes, because it is likely that she might be already exposed to that kind of information and she will not feel that recommendation as useful or interesting. In the same way, in a news recommendation scenario, it is not appropriate to recommend too many articles referring to the same story. The automatic detection of novelty and redundancy among recommendations has already been explored and evaluated in the literature [ZCM02].

- **Cold-start** Before a content-based recommender system can really grasp user preferences and provide reliable recommendations, each user has to rate a sufficient number of items. However, the recent explosion of Web 2.0 and social platforms changed the rule for user profiling since in principle it is possible to reuse the information the user already provided (such as comments, posts, tags or data gathered from social networks) and to exploit such information as a starting point to incrementally build and model her profile. In this area, a recent trend is represented by social media-based user profiling [BTC+10, OBTDR12].

### 2.2.2 Examples of Content-based Recommender Systems

Even if the most successful and well-known use cases in the area of recommender systems fall into the category of collaborative approaches, the state of the art of CBRS is as rich as those of collaborative ones.
The most-cited CBRS is probably Syskill & Webert [PB97]. A screenshot of its user interface is provided in Figure 2.3. It falls into the category of web browsing assistants that use past user ratings to predict whether the user will be interested in other pages. NewsWeeder [Lan95] is a Netnews filtering system where each article is described by means of term-frequency vectors. The system is based on explicit user preferences, expressed on a numeric scale, and implements a naïve Bayes algorithm similar to the previously described one.

The scenario that probably best fit with the characteristics of CBRS is news recommendation. In this field several RSs such as NewsRec [Bom04], NewT [SM93], PSUN [SC00], INFOrmer [SR97], NewsDude [BP99], Daily Learner [BP00], and YourNews [ABG*07] are noteworthy. NewsRec is a recommender system for news articles based on SVM as learning technique. NewT is based on explicit positive and negative feedbacks on articles and uses different filtering agents for delivering personalized content. Both NewsDude and YourNews share the inceptive idea of modeling two user profiles, one for representing short-term interests and one for long-term ones. Specifically, NewsDude learns a short-term user model based on TF-IDF and cosine similarity, and a long-term model based on a naïve Bayes classifier by relying on a training set of interesting news articles provided by the user. Similarly, in Daily Learner, a naïve Bayes classifier is exploited for modeling long-term interests, while a kNN text classification algorithm is the core of short-term personalization. PSUN adopts an alternative representation for articles since it analyzes the n-grams that occur in articles labeled as relevant to infer a network of mutually attracting or repelling words, whose degree of attraction is determined by the number of co-occurrences. Each
user has multiple profiles that are combined through a genetic algorithm. INFORmer uses a semantic network for representing both user profiles and articles. A spreading activation technique is used to compare articles and profiles, and a relevance feedback mechanism may be used to adapt system behavior to user’s changing interests.

LIBRA [MR99] is one of the most famous CBRS for book recommendation. It implements a naïve Bayes text categorization algorithm for book recommendation that exploits the product descriptions obtained from the Web pages of the Amazon on-line digital store. In the area of movie CBRS, it is worth to cite INTIMATE [MKP03]. Textual content are gathered from Internet Movie Database (IMDB)\(^1\) while the recommendation engine is based on a text categorization algorithm that uses explicit user ratings released on a qualitative scale (the user is asked to rate a minimum number of movies into six categories: terrible, bad, below average, above average, good and excellent).

The analysis of implicit feedbacks is the focus of Letizia [Lie95], a CBRS implemented as a web-browser extension. It takes as input user’s browsing history and builds a personalized model consisting of keywords related to user’s interests. It relies on implicit feedbacks to infer user preferences. For example, bookmarking a page is interpreted as strong evidence of user’s interest in that page. In a similar way, Personal WebWatcher [Mla99] learns interests of users from the Web pages they visit. It assumes the visited pages as positive examples of user interests and non-visited ones as negative examples.

Finally, SiteIF [MS01] and ITR (ITem Recommender) [DLS07] are CBRS that face with the representation issues previously introduced. The first is a personal agent for a multilingual news web sites. A language-independent content representation is obtained by exploiting MultiWordNet [PBG02] as knowledge source. MultiWordNet is a multilingual lexical database where senses are aligned for all the supported languages. Each news is automatically associated with a list of MultiWordNet synsets by using Word Domain Disambiguation [MS00]. The user profile is built as a semantic network whose nodes represent synsets found in the documents read by the user. During the matching phase, the system receives as input the synset-based representation of a document and the current user model, and it produces as output an estimation of document relevance. Similarly, ITR integrates linguistic knowledge in the process of learning user profiles by exploiting Word Sense Disambiguation. In ITR the typical keyword-based representation (BOW) is replaced by bag-of-synsets (BOS), a richer representation based on WordNet synsets. Thanks to this representation

\(^1\)www.imdb.com
2.3 Collaborative Recommender Systems

In CBRS each user is considered as an atomic entity, since she interacts with the system in order to receive her own suggestions and she plays no role in the process of generating recommendations for other users. Even if this model can work in an effective way, it is certainly far from what actually happens. Indeed, when an advice is needed, it is common to approach other persons that are considered as similar to us or trustful, assuming that like-minded people are a useful source to draw to get suggestions. This inceptive idea is implemented in the class of collaborative recommender systems (CF), where the recommendations are based on the preferences that other people, called nearest neighbors, already expressed.

CF algorithms strongly rely on mathematics since their only input is represented by a matrix where each user is mapped on a row and each item is represented by a column. This matrix, referred to as user/item matrix, is commonly filled in with the ratings provided by users on the items they previously enjoyed. This class of approaches analyzes users activities (for example, their ratings) in order to predict their preferences, with the assumption that users that in the past shared similar tastes will have similar tastes in the future as well. These techniques leverage mathematical data such as ratings and are mainly based on similarity calculations between items or users. Differently from CBRS they do not need to handle or process semi-structured or unstructured data. Thus, they can be used in all domains since the CBRS requisite that each item to be recommended has to be described with a set of textual features is not standing anymore here.

According to Breese et al. [BHK98], algorithms for collaborative recommendations can be grouped into two general classes: heuristic-based and model-based algorithms. As well as for CBRS, heuristic-based algorithms make predictions by operating over the entire collection of ratings while model-based ones exploit the collection of ratings to learn a model which is then used for predictions.

Heuristic-based approaches. The goal of heuristic-based approaches is to analyze the whole ratings space in order to extract the most similar users and to predict the preferences of the target user according to those of her own
neighborhood. The simplest heuristic-based technique is the so-called user-based collaborative filtering. By following the notation introduced in Section 2.2, we can define \( \text{CollaborativeUserProfile}(u) \), user profile for user \( u \), as a row vector of the user/item matrix previously introduced. Formally,

\[
\text{CollaborativeUserProfile}(u) = \{r_{u,1}, r_{u,2} \ldots r_{u,n}\} \tag{2.11}
\]

where \( r_{u,i} \) is the rating provided by user \( u \) on item \( i \). As in CBRS, user ratings can be explicitly or implicitly gathered, by asking user to express her preferences on a qualitative/quantitative scale or by mapping her actions to numerical scores (e.g. purchasing a book can be considered as an evidence of user liking in that item). If user \( u \) did not rate item \( i \), the value \( r_{u,i} \) is set to 0.

The inceptive idea behind this technique is to get \( \text{CollaborativeUserProfile}(u) \) as input and to exploit similarity measures to identify like-minded users whose preferences are similar to those expressed by the active user \( u \). Then, for every item \( i \) that the active user did not enjoy yet, the prediction is computed according to the ratings provided on \( i \) by the neighbors. Formally, the predicted rating of item \( i \) for user \( u \) described as \( f(u, i) \) can be computed as a score function takes as input the \( \text{CollaborativeUserProfile}(u) \) and \( \text{Neighborhood}(u) \) (that is to say, a set of row vectors extracted from the user/item matrix) and returns a numerical prediction of user interest in item \( i \).

\[
f(u, i) = \text{score}(\text{CollaborativeProfile}(u), \text{Neighborhood}(u)) \tag{2.12}
\]

In algorithms for user-based collaborative filtering particular emphasis is put on the formulas for calculating similarity. Beside cosine similarity, already introduced in Formula 2.8, it is typical to extract the neighborhood by exploiting the Pearson’s coefficient, calculated as follows:

\[
sim(u, u') = \frac{\sum_{i=1}^{n}(r_{u,i} - \bar{r}_u) * (r'_{u',i} - \bar{r}'_{u'})}{\sqrt{\sum_{i=1}^{n}(r_{u,i} - \bar{r}_u)^2} * \sqrt{\sum_{i=1}^{n}(r'_{u',i} - \bar{r}'_{u'})^2}} \tag{2.13}
\]

where \( u \) and \( u' \) is a couple of users. However, given a formula for calculating similarity, it is necessary to define also heuristics to define an ideal size of the neighborhood or a threshold above which a user can be labeled as neighbor. An empirical analysis, provided by Herlocker et al. [HKR02], showed that the ideal number strictly depends on the domain, the size of the community and the general users’ agreement. In general, this number can be set between 20 and
50. Values above 50 can introduce some noise in the calculations while lower values can leave out the opinion coming from relevant users.

Once the similarities between the users are determined, it is possible to predict user rating on item $i$ by aggregating neighborhood’s ratings on the same item. In literature many aggregation functions have been exploited: Adomavicius and Tuzhilin [AT05], for example, presented three different alternatives. In the first case (Formula 2.14) the prediction is calculated as the average rating provided by the users in the neighborhood. In the second (Formula 2.15) each rating is weighted with the similarity with the neighbors (in order to weigh more the opinion of the most similar users), while in the third one (Formula 2.16) the rating is further adjusted by taking into account the deviation from each neighbor’s average rating. Formally,

\[ f(u, i) = \frac{1}{N} \sum_{u' \in N} r_{u', i} \]  

(2.14)

\[ f(u, i) = k \frac{1}{N} \sum_{u' \in N} r_{u', i} \times \text{sim}(u, u') \]  

(2.15)

\[ f(u, i) = \bar{r}_u + k \frac{1}{N} \sum_{u' \in N} r_{u', i} \times \text{sim}(u, u') \times (r_{u', i} - \bar{r}_u) \]  

(2.16)

where $r_{u,i}$ is the rating provided by user $u$ on item $i$, $\text{sim}(u, u')$ is the similarity between user $u$ and user $u'$, $\bar{r}_u$ is the average rating of user $u$, and $k$ is a normalization factor.

**Model-based approaches.** Heuristic-based models are very simple and easy to implement. However, in real-world scenarios, they face with a clear bottleneck represented by the neighborhood formation. As the number of users grows, the time needed to compute the similarities grows as well and this is not feasible with the requirements of web applications where the computations have to be performed in real time. In order to effectively tackle this issue, alternative techniques emerged. Model-based approaches for collaborative filtering, for example, build a model based on the set of ratings. In other words, they extract some information from the dataset and use that as a model to make recommendations without exploiting the whole dataset.

The idea behind *item-based collaborative filtering*, used by Linden [LSY03] and implemented in the Amazon online store\(^2\), is to introduce some form of preprocessing on the user/item matrix to speed up similarity calculations. Specifically, this technique builds an *item/item matrix* where the pairwise similarity

\(^2\)www.amazon.com
between items is computed and stored. At run time, the prediction of $r(u, i)$ is obtained by detecting the items most similar to $i$ and by calculating the weighted sum of users's ratings on the items in the neighborhood. The similarity values between items are measured by observing all the users who rated both the items (Figure 2.4). As the number of such items is typically rather small (for example, the NetflixPrize\(^3\) data contains slightly fewer than 5,00,000 users, but only a little over 17,000 movies), the computation of the prediction can be easily accomplished in real time. From the analysis provided by Herlocker et al.\textsuperscript{[HKBR99]}, it emerged that cosine similarity consistently outperforms the Pearson’s coefficient for item-based algorithms. Obviously, the idea of pre-processing and storing pairwise similarities might be applied also to user/user couples. However, as stated by Sarwar \textsuperscript{[SKKR01]}, items similarities are much more stable because the overlap between user ratings is usually small.

Alternatively, the problem of predicting a rating for a user-item pair can be seen as a classification problem \textsuperscript{[BHK98]}. In CF algorithms the problem can be formulated as predicting the most likely rating for a certain item, according to previous user’s ratings. For calculating the probability it is common to use a Bayesian classifier because it is possible to assume that the probability of ratings are conditionally independent. Formally, the problem of predicting that the class (rating) can be formulated as follows,

\(^3\text{www.netflixprize.com}\)
2.3 Collaborative Recommender Systems

\[ P(C = r, r_1, \ldots, r_m) = Pr(C = r) \prod_{i=1}^{m} Pr(r_i|C = c) \]  

(2.17)

where \( r \) is the predicted rating and \( r_1, \ldots, r_m \) are the previous ratings provided by the user. In general, the problem can be seen as a multi-class classification where each class represents a discrete value (for example, between 1 and 5) and the item is assigned to the class with the highest probability.

As proved by Netflix Prize, most of the effort is moving today towards model-based approaches \[ \text{[KB11]} \]. A recent trend is to exploit matrix factorization methods \[ \text{[TPNT09]} \] to derive a set of latent factors that characterizes both users and items. A common technique is to use singular value decomposition (SVD) \[ \text{[DDL}+90] \] as a method to discover the latent factors and reduce the typical sparsity of the user/item matrix as well. With the help of SVD, the original matrix can be collapsed into a smaller-rank approximation in which highly correlated and co-occurring patterns are captured in a single factor. As proved by Koren \[ \text{[KBV09]} \], the effectiveness of these approaches is strictly dependent on the number of singular values to keep. They showed that numbers between 20 and 100 can positively affect the overall predictive accuracy. Finally, Goldberg \[ \text{[GRGP01]} \] investigated also the use of principal component analysis (PCA) in the area of CF.

2.3.1 Limitations of Collaborative Recommender Systems

Collaborative recommenders can effectively tackle some of the limitations of content-based algorithms, such as over-specialization, but they still suffer of important drawbacks.

- **Sparsity.** The effectiveness of a CF algorithm is strictly related to the availability of a critical mass of users and ratings. However, a typical issue of RS is that the number of ratings provided by the users is often very small compared to the number of ratings that need to be predicted so it is important to effectively tackle the sparsity problem in order to provide users with effective recommendations. One way to overcome the problem of rating sparsity is to exploit user profile information and to consider them as additional entries when calculating user similarity over the user/item matrix. In Pazzani \[ \text{[Paz]} \] also gender, age, area code, education, and employment are used as collaborative data to calculate user similarity. In general, the sparsity problem can be handled through dimensionality reduction techniques, such as Singular Value Decomposition (SVD). It
Recommender Systems

is a well-known method for matrix factorization that provides the best lower rank approximation of the original matrix. In [SKKR00] Sarwar et al. investigated the use of SVD in the area of collaborative recommender systems.

- **New user problem.** This problem is related to the aforementioned one since each user has to rate a sufficient number of items before a recommender system can provide her with reliable recommendations. Several techniques have been proposed to address this problem. Most of them are based on hybrid recommendation approaches, by combining collaborative and content-based techniques. An alternative technique was proposed by Rashid et al. in [RAC+02]. It is based on the analysis of the item space in order to find the most informative items a user needs to rate in order to get recommendations. This selection can be performed by exploiting various techniques based on item popularity, item entropy, user personalization, and combinations of the above [YST+04].

- **New item problem.** When a new item is added in a recommender system there is no way to recommend it before more information about it is obtained. This problem is usually addressed by encouraging users to vote the most novel items in order to get users rating and trigger the similarity calculations that feed CF algorithms. However, a typical way to tackle this problem is to combine CF algorithm with CBRS, since content-based approaches do not suffer from this problem.

- **Grey sheep problem.** CF algorithms are based on the analysis of similarities among users. For those whose tastes are unusual with respect to the rest of the population it will be difficult to find other users particularly similar and this will consequently lead to poor recommendations. This problem is also felt for demographic systems, which attempt to categorize users according to personal characteristics. It is important for recommendation algorithm to have the ability to detect gray sheep and to adopt for these users hybrid techniques mainly based on the exploitation of content-based user profiles.

2.3.2 Examples of Collaborative Recommender Systems

The first collaborative filtering systems reported in the literature followed a user-based approach. The Grundy system [Ric79], for example, proposed to use stereotypes to overcome the limited amount of available information for building
2.3 Collaborative Recommender Systems

Figure 2.5: GroupLens rating and recommendation page

user profiles. Goldberg et al. [GNOT92] implemented in the Tapestry system the idea of clustering like-minded users. The first implementations of CF algorithms date back to 1992, thanks to the GroupLens system [KMM+97], whose goal was to recommend Usenet news (see Figure 2.5). In the proposed model, users and news articles are clustered according to existing newsgroups while implicit ratings are computed by gathering the time each user spent reading a news.

Next, noteworthy implementations of CF algorithms are Video Recommender [HSRF95], Ringo [SM95] and the Jester system that recommends jokes [GRGP01]. In Video Recommender the system receives and sends emails to obtain user ratings and to provide video suggestions. The algorithm is based on user-based CF: videos are shown to the users according to descending predicted ratings and classified by video categories. Ringo is a CF system which makes recommendations of music albums and artists. The algorithm is always user-based CF and user profiles are building after a training step where 125 artists are presented. Personality Diagnosis [PHLG00] is platform implementing a CF algorithm that computes the probability that a user shares the same personality type with another user and, in turn, the probability that she will like an unseen item. The probability estimations are derived by applying the aforementioned Bayes approach.
More recently, especially for scenarios where the number of users is much
greater than the number of items, item-based collaborative filtering gained more
and more interests. The most successful use-case of item-based CF is those
implemented in Amazon [LSY03] (see Figure 2.6), whose recommendation engine
follows the description of the abovementioned item-based collaborative filtering
algorithm.

Finally, another well-known example of recommendation algorithm imple-
mented in a real-world use case is represented by Google News, a platform that
crawls and aggregates news articles in order to provide users with personal-
ized news stories. The recommendation model behind Google News exploits a
combination of model-based and memory-based techniques. The model-based
component is based on probabilistic latent semantic indexing (PLSI), proposed
by Hofmann [Hof99], and uses an hashing algorithm called MinHash to analyze
users’ log and group users in different clusters. The memory-based part of the
algorithm is used to dealing with new users by logging her co-visits (the set of
articles she read in a defined time slice) and comparing them with other users’
one. At run time, the overall recommendation score for each item in a defined
set of candidate items is computed by combining the scores obtained by the
three methods (MinHash, PLSI, and co-visits).

Figure 2.6: A typical recommendation from Amazon
2.4 Demographic, Knowledge-based and Community-based Recommender Systems

Demographic Recommender Systems. These systems aim to produce recommendations on the ground of demographic classes, by categorizing users starting from personal attributes. Even if people are reluctant to provide phone numbers and physical addresses, the recent exponential growth of social network and the continuous expansion of Web 2.0 fed again this class of RSs. However, demographic approaches notice less success than others.

An early example of this kind of system was Grundy [Ric79]. The extraction of personal information is made through an interactive dialogue and users responses are matched against a library of user stereotypes in order to provide book recommendations. Another example of a demographic-based recommender is Lifestyle Finder [Kru97]. It attempts to classify a user into a set of 62 clusters and to tailor recommendations to users according to the information about other users in this cluster. In [Paz] is proposed the exploitation of machine learning techniques to obtain a classifier based on demographic data.

Knowledge-based Recommender Systems. These systems generate recommendations by combining information stored in knowledge bases with some inference technique. In general, knowledge-based recommenders are provided with some kind of functional knowledge (e.g. how a particular item meets a particular user need). The user profile can be any knowledge structure that supports this inference. In the simplest case, as in Google, it may simply be the query that the user has formulated. In others, it may be a more detailed representation of the user needs [TQ00]. A particular kind of knowledge-based recommender systems are those implementing case-based reasoning (CBR). In this case the task is solved by retrieving a known solution for a similar problem. In general the process is performed in four steps: retrieval of a similar case, reuse of the retrieved solution, adaption of the solution and re-training. An example of RS implementing CBR is presented in [LR03]. A knowledge-based recommender system avoids some of the drawbacks of other recommendation techniques. It does not have a ramp-up problem since its recommendations do not depend on a base of user ratings. As stated above, it does not have to gather information about a particular user because system judgments are independent of individual tastes.

Community-based Recommender Systems. This class of systems is based on the evidence that people tend to rely more on recommendations provided by trustful friends with respect to those provided by similar (but anony-
mous) individuals [SS01]. Thanks to the recent popularity gained by social network and Web 2.0, Community-based Recommender Systems are earning more and more interest [Gol06]. The idea behind social recommenders is to model information about social relations of users and to exploit them in order to provide recommendations, assuming that the level of interest or our friends in a certain item is a good predictor of our own interest, thus recommendations are based on ratings that user’s friends provided on that item. The research in this area is still in its early phase. Some preliminary attempt [MA04] showed that these recommendations do not overcome those derived from traditional CF approaches, but the interpretation is controversial since in some scenarios social network-based data provided users with better recommendations [GE07].

2.5 Hybrid Recommender Systems

The main idea behind hybrid recommender systems is to combine two or more classes of algorithms in order to achieve some synergy between them, emphasizing their strengths and leveling out their corresponding weaknesses.

For example, a collaborative system and a content-based one might be combined to compensate the new user problem, providing recommendations to users whose profiles are too poor to trigger the collaborative recommendation process. There is no reason why several different techniques of the same type could not be hybridized, for example, as in NewsDude [BP99], two different content-based recommenders such as kNN classifiers and naïve Bayes could work together, and a number of projects have investigated this type of hybrid algorithms. Burke proposed an analytical classification of hybrid systems, listing a number of hybridization methods to combine pairs of recommender algorithms. In [Bur02, Bur07] seven different hybridization techniques are introduced. Specifically:

- **Weighted** - In weighted hybrid recommenders the relevance score of a recommended item is calculated by combining the relevance scores of all of the available recommendation techniques implemented in the system. For example, the simplest combined hybrid would be a linear combination of recommendation scores, as in P-Tango [CGM99]. This approach initially gives equal weight to collaborative and content-based recommenders but the weighting is gradually adjusted according to user feedbacks on predictions. Another strategy consists in treating the output of each recommender as a set of votes combined in a consensus scheme [Paz].
• **Switching** - A switching hybrid uses some criterion to switch between recommendation techniques. This class of recommenders introduces additional complexity into the recommendation pipeline since it requires to determine a switching criterion. In DailyLearner [BP00], for example, the pipeline starts with a content-based recommender but it switches to a collaborative one when the confidence level on the recommendations already produced is not adequate. However, this switching hybrid does not completely avoid the ramp-up problem, since both collaborative and content-based systems have the *new user* problem.

• **Mixed** - In this class, recommendations from several different algorithms are presented at the same time. This approach particularly fits scenarios where it is possible to provide users with a large number of recommendations. This technique is implemented in the PTV system [SC00], whose goal is to assemble a personalized program guide of TV shows: recommendations are generated according to textual features describing each item and preferences of similar users. The mixed hybrid avoids the ‘new item’ start-up problem, since the content-based component can exploit the descriptions of new item even if they have not been rated by anyone, but the *new user* start-up problem remains, since both content and collaborative methods need some data about user preferences.

• **Feature Combination** - This hybridization technique exploits the idea of combining the information coming from different recommendation sources into a single algorithm. For example, content and collaborative techniques might be merged by considering textual features as additional columns of a *user/item* matrix, or, alternatively, by using collaborative data as additional features to feed a content-based algorithm.

• **Cascade** - In the cascade hybrid the recommendations list produced by an algorithm is refined by another one. Usually one recommendation technique is employed to produce a coarse set of candidates and the second algorithm refines and re-ranks it in order to obtain the final list. The cascade hybrid is generally more efficient than a weighted one that applies all of its techniques to all items.

• **Feature Augmentation** - In this class of hybrid RSs the output of a technique is used as input feature for another one. This means that one technique is employed to produce a rating or classification of an item and that information is then incorporated into the processing of the next recommendation technique. For example, the LIBRA system [MR99] makes
content-based recommendations of books based on data found in Amazon.com, using a naïve Bayes text classifier. In the text data used by the system is included related authors and related titles information that Amazon generates using its internal collaborative systems.

- **Meta-level** - The model learned by one recommender is used as input to another. Differently from feature augmentation, where the learned model generates features to input in a second algorithm, in this method is the entire model that becomes the input. For example, the user-specific selection agents of the Fab system [BS97] perform content-based filtering using Rocchio’s method [Roc71] to maintain a term vector model that describes the user’s area of interest. The benefit of the meta-level method, especially for the content/collaborative hybrid, is that the learned model is a compressed representation of a user’s interest, and a collaborative mechanism that follows can operate on this information-dense representation more easily than on raw rating data.

## 2.6 Evaluation of Recommender Systems

How to evaluate the performance of a Recommender Systems? The answer to this question is still an open point since recommender systems have been evaluated in many, often incomparable, ways. Techniques for evaluating recommender systems can be broadly split into three categories: offline experiments, user studies and online evaluations [SG11].

**Offline evaluation** is the most traditional technique, since its methodology comes from Artificial Intelligence and Information Retrieval areas. The goal of this class of evaluation approaches is to estimate how good an algorithm is in predicting user ratings or in producing rankings as close as possible to users’ ideal ones. However, RS performances are strictly related to user experience; the accuracy is not the only feature that a good algorithm has to guarantee. Consequently, methodologies much more focused on the impact of recommender systems on real users emerged. The goal of **user studies**, for example is to analyze the behavior of a set of subject tests interacting with the recommender system. In general, user studies can answer a larger set of questions about the effectiveness of a recommender system (is it possible to analyze, for example, how many recommended news stories are actually clicked) as well as evaluating other features beyond the simple accuracy, such as novelty, serendipity and diversity that are likely to be more difficult to evaluate in offline experiments. On the other side, user studies are more subject of bias, because often users
are not motivated or the users sampling is not sufficiently heterogeneous as to ensure a proper generalization of the results. Finally, online experiments are the most effective way to evaluate a recommender system but they are also the most difficult methodology to adopt since they can be performed only with real recommender systems implementations. Indeed, online experiments are based on analysis performed over a large set of users (commonly unaware that they are involved in an experiments) that interacts with the system. Users are seldom explicitly involved in these kind of experiments since they are self-motivated and only their activity logs are needed. This methodology is often used by search engines, such as Google, in order to evaluate the impact of changes in their algorithms or by online stores, such as Amazon, to state whether a new version of the recommendation algorithm can improve online purchases or user experience.

Given this broad classification, each aspect can be qualitatively and quantitatively inferred through specific metrics. These are split into two classes: standard metrics and non-standard ones. Intuitively, the first are commonly used for offline experiments while the latter are the outcome of online ones, even if in some recent attempt [ZCM02] researchers tried to extract non-standard metrics such as serendipity of recommendations from offline experiments. An extensive and complete review of evaluations techniques is provided by Herlocker et al. in [HKTR04] and, more recently, by Shani in [SG11]. A deep overview of statistical test adopted to validate the results of experimental evaluations is provided in [Dem06] and [OD90].

### 2.6.1 Standard Metrics

Standard metrics have been defined for judging how much the prediction \( \hat{r}_{u,i} \) on item \( i \) deviate from actual rating \( r_{u,i} \) provided by user \( u \) and to evaluate the effectiveness of supporting users to obtain high-quality items. According to these tasks, standard metrics can be classified in the following categories:

**Predictive accuracy metrics.** These metrics determine how close predicted ratings are with respect to true ratings. The most popular metrics are:

- **Mean Absolute Error (MAE).** This metric measures the deviation between prediction and actual rating provided by the user. For each pair \( (\hat{r}_{u,i}, r_{u,i}) \), the metrics calculates the absolute error between \( \hat{r}_{u,i} \) and \( r_{u,i} \). The MAE is computed by first summing these absolute errors of the corresponding \( N \) predictions for all the \( M \) users, and then averaging the sum by the total number of users. The lower the MAE, the more accurate are the
predictions. Formally:

\[
MAE = \frac{1}{MN} \sum_{n=1}^{M} \sum_{n=1}^{N} |\hat{r}_{u,i} - r_{u,i}|
\]  
(2.18)

- **Root Mean Squared Error (RMSE).** This metric follows the same principle of MAE but it squares the error before summing. Consequently, it penalizes large errors since they become much more pronounced than small ones. In literature have been also presented the NRMSE (its normalized counterpart) and average RMSE, better for unbalanced datasets.

\[
RMSE = \sqrt{\frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} |\hat{r}_{u,i} - r_{u,i}|}
\]  
(2.19)

**Classification metrics.** These metrics evaluate how well a recommender system can split the item space between relevant and non-relevant items. Some of them may also take into account the ranking of the items, giving higher scores to the algorithms able to put the most relevant items at the top of the recommendation list. In general these metrics are inherited from Artificial Intelligence and Information Retrieval systems.

- **Precision.** This metrics counts how many items among the recommended ones are actually relevant for the target user. Let TR be the number of true relevant predictions (the number of recommended items that are relevant for the user) and FR as the number of false relevant predictions (the number of recommended items that are non-relevant for the user), the Precision is calculated as follows:

\[
PR = \frac{TR}{TR + FR}
\]  
(2.20)

- **Recall.** This metrics counts how many items among those that are relevant for the target user are actually recommended. Let TR be the number of true relevant predictions (the number of recommended items that are relevant for the user) and FN as the number of false relevant predictions (the number of non-recommended items that are relevant for the user), the Recall is calculated as follows:

\[
RE = \frac{TR}{TR + FN}
\]  
(2.21)
2.6 Evaluation of Recommender Systems

- **F-Measure.** A metric defined as the harmonic mean of the precision and recall metrics. Let $\beta$ be a parameter that determines the relative influence of both precision and recall, the F-Measure is calculated as follows:

$$ F = \frac{(1 + \beta^2) \times PR \times RE}{\beta^2 \times PR + RE} $$

- **Receiver Operating Characteristic (ROC) curve.** This metric, introduced in [Swe98], is used to measure the compromise between presenting the user a high number of relevant items, and recommending her a low number of non-relevant ones. The curve shows the percentage of correctly predicted items with respect to the percentage of wrongly predicted non-relevant one. The number of correct relevant predictions can be increased at the expense of increasing the number of non-relevant predictions (and vice versa). The Area Under the Curve (AUC), showed in Figure 2.7, is one of the most accepted metrics in the Machine Learning community.

**Ranking metrics.** The goal of these metrics is to state whether an algorithm is able to produce a ranking close to the ideal one that a user would produce. These metrics are useful for scenarios where it is expected that the user would enjoy at most the first $n$ entries of the recommended list, so it is important to evaluate the ability of an algorithm of putting the best items at the top of the list.
• **NDPM (Normalized Distance-based Performance Measures).** The goal of Normalized Distance-based Performance Measures, proposed by Yao [Yao95], is to calculate how far is the ranking proposed by the system with respect to user ideal one. More specifically, NDPM is used to measure the distance between the votes given by a single user and those predicted by the system for a set of items. Given a couple of items $t_i$ and $t_j$ extracted from a test set T, the distance between them is calculated through the following schema:

$$\delta_{>u,>s}(t_i, t_j) = \begin{cases} 
2 & \iff (t_i >_u t_j \land t_j >_s t_i) \lor (t_i >_s t_j \land t_j >_u t_i) \\
1 & \iff (t_i >_s t_j \lor t_j >_s t_i) \land t_i \approx_u t_j \\
0 & \iff \text{otherwise} 
\end{cases} \quad (2.23)$$

The value of NDPM on the whole set T is calculated through the following equation:

$$NDPM_{>u,>s}(T) = \frac{\sum_{i \neq j} \delta_{>u,>s}(t_i, t_j)}{2 \cdot n} \quad (2.24)$$

Where $n$ is the number of couple of items. The NDPM values can range between 0 and 1, where 0 indicates the NDPM value of an ideal ranking while 1 is the NDPM value when all the comparison are wrong.

• **NDCG (Normalized Discounted Cumulative Gain).** In certain types of scenarios users may enjoy large portions of the recommendations list. In such cases it is necessary to exploit metrics with a slower decay of the positional discount: Normalized Cumulative Discounted Gain (NDCG) is a measure from Information Retrieval, where positions are discounted logarithmically. Given a list of $N$ items the average Discounted Cumulative Gain (DCG) can be calculated as follows:

$$DCG = \frac{1}{N} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{g_{uij}}{\max(1, \log_b j)} \quad (2.25)$$

where $g_{uij}$ is the "gain" that user $u_i$ has from enjoying item $j$ in her recommendation list and $b$ is the logarithm base, a free parameter of the formula. Given the DCG score and let $DCG^*$ be the ideal score, the normalized version of DCG is calculated as:
\[ NDCG = \frac{DCG}{DCG} \]  

\[ (2.26) \]

2.6.2 Non-Standard Metrics

Standard metrics, although very popular and commonly used in RS community, have a number of limitations. Indeed, they evaluate whether a system is able to reflect user behavior by correctly ranking a set of items in the test set or by predicting if an item she already voted would have been relevant for her or not. If an algorithm predicts user behavior is considered as good, otherwise it does not. This is a very limited vision since these metrics do not take into account user perspective. How novel is a recommendation? How diverse are the recommendations? Are recommendations able to surprise users? Are they actually interesting for her? Standard metrics cannot face with these questions. Consequently, to overcome the previous limitations, a number of additional evaluation metrics have been proposed in the literature.

- **Item Coverage.** This metric calculates the percentage of items for which a recommender system is able to make prediction and it was investigated in [SKB+98]. Systems with lower coverage may be less valuable for users, since they are limited in the decisions they are able to help with.

- **Novelty and Serendipity.** These metrics [SKKR01] measure whether a recommendation is obvious or not. In general, novelty refers to the ability of an algorithm of proposing something new, that is to say, something the user is not aware of. A method for evaluating novel recommendations [SCM08] uses the above assumption that popular items are less likely to be novel. Thus, novelty can be taken into account by using an accuracy metric where the system does not give the same credit for correctly predicting popular items as it does when it correctly predicts non-popular ones.

  On the other side, serendipity also includes a feeling of surprise. A recommendation can be considered as serendipitous whether it is both novel and referring to some information the user is not exposed to. In Murakami et al. [MMO07] serendipity is considered as a form of deviation from "natural" predictions. They propose to consider as more serendipitous recommendations whose score are lower because recommendations with higher scores or higher confidence are likely to be obvious.

- **Diversity.** Diversity is generally defined as the opposite of similarity. In some cases suggesting a set of similar items may not be as useful for the
user, because it may take longer to explore the range of items. A common method for measuring diversity, proposed in [ZMKL05], is to exploit item-item similarity, typically based on the analysis of item content, and to put an item in the recommendation list whether its average (or, alternatively, minimum or maximum) similarity does not overcome a certain threshold.

- **Learning Rate.** This metric approximates how quickly an algorithm can produce good recommendations, and how well the system can help users make more effective decisions according to the data currently available. As stated in [SPUP01], this issue is particularly felt in cold-start situations.

- **Confidence.** Confidence in the recommendations can be defined as the system’s trust in its recommendations or predictions. These metrics measure how certain the recommender system is about whether its recommendations are accurate. In [HKR00] a wide range of different confidence displays are explored, to study which ones are most influential in users making the right decisions.

- **Trust.** The features refers to the level of confidence of users in the recommendation produced by the system or, in general, in the whole recommendation algorithm. The obvious method for evaluating user trust is by asking users whether the system recommendations are reasonable, as in Pu [PC06]. Alternatively, it is possible to assume that trust in the system is correlated with repeated users, as users who trust the system will return to it when performing future tasks.

- **Utility.** Many e-commerce websites employ a recommendation system in order to improve their revenue. In general, it is possible to define various types of utility functions that the recommender tries to optimize. For such recommenders, measuring the utility or the expected utility of the recommendations may be more significant than measuring the accuracy of recommendations. The problem of measuring the utility of the users receiving the recommendations has been investigated by Brazdinas in [BB05].

In [SG11] Shani provides a deeper overview of the abovementioned desired features of a recommender system.
2.6 Evaluation of Recommender Systems
Chapter 3

Enhanced Vector Space Models (eVSM) for CBRS

This chapter describes the underlying ideas behind the design of enhanced Vector Space Model (eVSM), the novel recommendation framework introduced in this dissertation.

eVSM is strictly related to Vector Space Model (VSM), so Section 3.1 focuses on VSM and explains how it can be adapted to be exploited in recommendation tasks. Therefore, after a brief literature overview about the use of VSMs in the area of CBRS, a thorough analysis of VSMs weaknesses and possible solutions is provided.

The main building blocks or eVSM are described in Section 3.3. Particular emphasis is put on the description of Quantum Negation, a novel operator for managing negative feedbacks, and Random Indexing, a technique for dimensionality reduction adopted to guarantee both an incremental reduction of vector space and a lightweight semantic representation based on distributional models. Finally, Sections 3.3.3 and 3.3.4 respectively introduce the approaches for learning user profiles and for generating recommendations in both monolingual and multilingual environments.

Hereafter we might refer to the items to be filtered and to the user profiles as documents. Indeed, in a content-based filtering model the terms are considered synonyms because we assume that items to be filtered are described by means of some textual content. For example, in a movie recommendation scenario we can assume that an item (movie) can be represented by its title, cast, plot and so on.
3.1 Overview of Vector Space Model

The VSM [SWY75] is a well-known model developed in the early 70s for representing textual documents in a vector space. Its first application has been the SMART retrieval system [Sal71], proposed by Salton.

VSM is based on the assumption that given a corpus of textual documents, each one can be represented as a point in a \( n \)-dimensional vector space (where \( n \) is the number of the indexed features). Formally, each document \( d \) can be represented as a vector

\[
d = (w_{f_1}, \ldots, w_{f_n}),
\]

where \( w_{f_i} \) represents the weight of the feature \( f_i \) in the document \( d \). The weight can be assigned by means of different schemes, ranging from the boolean one (assigning 1 to the features that occur in the document, otherwise 0), via simple counting of the occurrences (even normalized) of each feature, to more complex techniques such as the aforementioned TF-IDF [BYRN07] or the Okapi BM25, based on the probabilistic retrieval framework introduced by Robertson and Sparck-Jones [JWR00a, JWR00b].

The typical application of VSM is document retrieval. In this scenario, a user issues a query containing one or more terms according to her own informative need. The query can be modeled as a pseudo-document as follows:

\[
q = (w_{f_1}, \ldots, w_{f_n}).
\]

Each \( w_{f_i} \) is the weight of the feature \( i \) in the query. As well as for the documents, \( w_{f_i} \) is set to 1 whether the term occurs in the query, otherwise is set to 0. Even if more complex techniques for representing and formulating query, such as Query Expansion [CR12], actually exist, it is very common to represent the query in this way.

Given a set of \( m \) documents \( d_1 \ldots d_m \) and a query \( q \), through classical similarity measures such as the aforementioned cosine similarity it is possible to compute how much a document is related or similar with respect to the query. In general, the cosine similarity is computed for all the couples \((d_i, q)\) and the documents \( d_i \) are ranked according to their descending cosine similarity. Finally, the top \( n \) are returned to the target user.

Today VSM is still considered as one of the most effective state-of-the-art approaches in the area of IR thanks to its good compromise between expressivity, effectiveness and simplicity. Indeed, most search engines use VSM to measure
the similarity between a query and a document [MRS08]. However, it suffers from important problems that undoubtedly affect its accuracy and need to be tackled in order to adapt and exploit it in a recommendation scenario.

First, the original formulation of VSM deals with keyword-based content representation. The weight \( w_f \) simply depends on the occurrences of a term within a document (even smoothed according to the popularity of the term within the corpus) and does not take into account neither its position nor its co-occurrences with other terms. This representation is not able to deal with typical problems of syntactical-based representation, such as *polysemy* and *synonymy* and it cannot catch all the language facets, so it is necessary to introduce more complex and richer techniques for modeling documents in vector spaces.

Next, the approach is not incremental. This means that even the addition of a new item to the corpus requires that the whole vector space has to be generated again from scratch. This is a problem typical of real-world scenarios where the items continuously change over time, since the generation of high-dimensional vector spaces is a computationally expensive task. In a recommendation task this drawback is actually really felt. If we think about a news recommendation scenario, for example, the flow of information is continuous and uncontrolled, so it is necessary to introduce techniques able to ensure the scalability requirements typical of real-time computations.

Finally, the VSM as it is cannot manage the information coming from negative feedbacks. This is a well-known drawback that cannot be ignored. Indeed, as proved by many contributions coming from both IR and text categorization areas [KHRM06, Joa98, Roc71], modeling both positive and negative information improves the overall accuracy.

### 3.1.1 Literature Review

The idea of exploiting VSM in order to provide users with recommendations is not totally new. The first attempts in this direction date back to the mid-1990s thanks to Foltz [FD92] and Cohen [CH98]. Cohen’s approach has been evaluated also for technical paper recommendation in [BHCNM01]. Next, Yan and Garcia-Molina [YGM94] adopted a VSM-based filtering model as main building block for an analysis regarding the best data structures to be adopted in an Information Filtering system. Usually, VSM-based models have been coupled with other content-based approaches, as in DailyLearner [BP00], where VSM is used to infer a short-term representation of user preferences or in Fab [BS97], that employs a content-based recommender for building user models based on a vector of term categories and users’ degree of interest in them.
A good overview about the use of IR algorithms in CBRS is provided in [BvdB07], where different techniques and different weighting schemes (such as the TF-IDF and BM25 previously introduced) are compared in the scenario of news recommendation. Next, McCarey [MCK06] investigated the effectiveness of VSM with respect to other techniques such as Latent Semantic Indexing as recommendation algorithm for the librarian domain. In [HXW03], Huang et al. introduced a text filtering system based on VSM. During the training phase, feature selection and pseudo feedback are used to build user profiles while in the filtering phase the information stored in the user profile and the thresholding strategy are adapted according to user feedbacks. The system has been evaluated with the TREC\(^1\) dataset and the results of the experimental evaluation confirmed the effectiveness of the approach. Indeed, as stated by Pazzani [PB07], despite the unquestionable simplicity of the algorithm, in most of the state-of-the-art papers the VSM performs competitively with more complex algorithms, even in recommendation tasks.

Afterwards, research focused on the use of enhanced vector space representations. Soboroff [SN00] proposed the use of Generalized Vector Space Model (GSM) [WZW85] in a recommendation scenario. Similarly, in Mirizzi et al. [MDNR+12] is described a Facebook application for content-based movie recommendation that exploits semantic Vector Space Model (sVSM), a semantic extension of VSM based on Linked Data and DBPedia\(^2\). Also Nouali and Blache [NB04] followed this trend by proposing the idea of modeling both items and user profiles as points in a semantic vector spaces.

Recently, the use of VSM for recommendation tasks has been evaluated in several scenarios. In [MDKD08] the authors describe a product recommender system based on VSM that exploits cosine similarity to find out the closest user profile among those stored in the database. Ghauth [GA09] proposes a novel framework for an e-learning recommender system, based on the idea of recommending learning materials according to the similarity of content items represented in a VSM. The application of CBRS based on VSM in the area of e-learning is also the core of the work proposed by Bighini et al. [BCC03], where an hybrid recommender system that combines content analysis with clustering techniques to provide users with personalized access to learning content is proposed. Next, Patalia [PS09] evaluated VSM for email classification and filtering while Chan [CGT11] exploited VSM for web services recommendation. Finally, [PMBS11] and [KCK12] proposed the use of VSM for recommending news and

\[^{1}\text{http://trec.nist.gov/}\]
\[^{2}\text{http://dbpedia.org}\]
3.2 A Recommendation Framework based on VSM

As already pointed out by Belkin and Croft in [BC92], IF tools (such as CBRS) can be considered as strictly related to IR ones, since they both aim to optimize the access to unstructured data sources and only differ in the way user needs are represented. Indeed, search engines need an explicit query that triggers the retrieval process while in IF tools the whole process is based on the information commonly stored in a user profile.

Consequently, the insight behind this dissertation is to leverage the convergence and the methodological overlap argued by Belkin and Croft to introduce a novel recommendation framework based on well-known IR techniques, adapted and evolved according to the specific requirements of a recommendation scenario. Basically, to adapt VSM for recommendation tasks is simple. Reasoning by analogy, the items to be recommended can be considered (and modeled) as documents, while the query typical of IR systems can be replaced by a vector space representation of user preferences. Next, given this representation, a content-based recommendation model that uses VSM can be defined by exploiting several approaches. Whether the recommendation problem is comparable to a classification one, the previously introduced kNN classifier is a good choice. On the other side, whether the task is considered as similar to a retrieval one, it is possible to use cosine similarity to compute how related (similar) each item is with respect to the user profile, with the insight that the items with the highest cosine similarity can be considered as relevant and recommended to the target users. In this dissertation the latter approach is adopted.

However, before introducing the framework, it is necessary to take into account the outcomes of the previous analysis. The design choices behind the definition of a novel recommendation model have to be addressed to overcome the aforementioned VSM issues. Specifically, eVSM has been designed on the ground of three main building blocks:

1. The model has to be able to catch the semantics of documents;
2. The whole vector space has to be built in an incremental way;
3. The model has to be able to manage the information coming from negative evidences.

Since VSM as it is cannot conciliate all the above requirements, in the next
Section each aspect is thoroughly investigated in order to identify the techniques that can best tackle each issue.

3.2.1 Introducing Semantics in VSMs

To enhance the semantic representation of the documents is a key aspect for those systems, such as CBRS or search engines, whose goal is to simplify and optimize the access to unstructured data sources. Indeed, these platforms are mainly based on the analysis of the features that occur in the documents to be retrieved (or, in a recommendation task, in the items to be recommended).

A wrong understanding of the correct meaning of each feature affects the interpretation of user needs and preferences, and the introduction of some noise coming from non-relevant features is the primary cause behind the generation of a wrong set of recommendations or behind the retrieval of non-relevant documents.

To incorporate semantics into documents, or, in general, into textual content, it is necessary to enrich the representation with information coming from external information sources such as corpus, dictionaries, ontologies, Linked Open Data clouds and open knowledge sources such as Wikipedia. Generally speaking, the approaches for associating the correct word meaning to each features can be broadly split into two categories:

- **Word Sense Disambiguation-based approaches** are based on algorithms that select the correct sense for each feature by discerning several alternatives drawn from a sense inventory.

  Formally, the goal of a Word Sense Disambiguation (WSD) algorithms consists in assigning to a target word $w$ occurring in a document $d$ its correct meaning $m_w$ by exploiting the context in which $w$ occurs. The context is generally defined as the set of words that precede and follow $w$. The set of possible senses is gathered by analyzing a sense inventory, such as WordNet [Mi95] or MultiWordNet [PBG02].

  The use of WSD techniques in both IR and IF areas has been broadly investigated, but so far the results were not always encouraging. In [Voo93], for example, the author extended VSM by employing a WSD algorithm to disambiguate nouns. However, the experiments showed a significant degradation with respect to a standard retrieval system. Similarly, in [RS95] the authors built a knowledge-based IR system that uses WordNet but the accuracy of the system was lower if compared to a classical retrieval system.
based on TF-IDF and syntactic representation. The only evidence of effective IR system integrating semantics is described in [FLGD87], where the authors performed a manual disambiguation of 100 documents. However, even if these word sense-based vectors outperformed the syntactical representation, the idea of a manual disambiguation is not applicable in real world environments. An attempt of combining WSD with user profiling in the area of CBRS has been proposed in [SDLB07], where a WSD algorithm is used to shift the typical bag-of-words representation towards one based on WordNet synsets. The effectiveness of the approach is confirmed by the experimental evaluation since these semantic WordNet-based user profiles significantly improved the overall accuracy of the recommender.

The use of ontologies as mean for introducing semantics in IR system is investigated in [DKW02], where the authors annotated documents with concepts extracted from a domain ontology. Quickstep [58] is a system for the recommendation of on-line academic research papers. The system adopts a research paper topic ontology based on the computer science classifications made by the DMOZ open directory project\(^3\) (27 classes used). Semantic annotation of papers consists in associating them with class names within the research paper topic ontology, by using a kNN classifier. Interest profiles are computed by correlating previously browsed research papers with their classification. News@hand [18] is a system that adopts an ontology-based representation of item features and user preferences to recommend news. Item descriptions are vectors of TF-IDF scores in the space of concepts defined in the ontologies. User profiles are represented in the same space, except that a score measures the intensity of the user interest for a specific concept. The matching is performed as a cosine-based vector similarity. Furthermore, Middleton proposed in [MSD04] an ontological approach for user profiling in CBRS in the scenario of paper recommendation. The encouraging experimental results boosted this research direction, as proved by Pais’ review article [PdCDL08] and other recent works [CBFTH11, CLS06, JKM10].

- **Discriminative approaches** are based on algorithms that exploit un-annotated corpora in order to discriminate and distinguish the different meaning of each feature.

Discriminative approaches are based on a simple insight: as humans infer the meaning of a word by analyzing the context each word is used in,
discriminative algorithms analyze the usage of a word in a large corpus of textual documents to get information about its correct meaning. In general, these techniques follow the distributional hypothesis, which states that words with similar distributions (that is to say, similar usages) share a similar meaning. For example it is possible to state that beer and wine refer to similar concepts since they both mostly co-occur with the same terms (such as drink or glass) (See Figure 3.1).

![Figure 3.1: Similar terms share similar usages](image)

Operationally, these algorithms adopt a VSM-based representation based on a term-context (instead of a term-document) matrix, where each word is associated to the context that should identify the word meaning. In other terms, the position that a term assumes in the vector space depends on the contexts it has been used and the relationships relying with other words into the space.

As stated by Lowe [Low01], it is common to refer to these representations as semantic spaces (or Word Spaces). For example, in Latent Semantic Analysis [LD97], the document is considered as a context and the original space is projected over a low-dimensional one by a truncated SVD decomposition.

However, despite the growing attention, these techniques have not been widely applied in both IR and IF research areas, excluding some preliminary attempt proposed by Schuetze [SP95]. The lack of a proper investigation about the use of distributional models (DM) in the area of CBRS, joint with the advantages of the lightweight semantic representation they can provide is the main reason behind the choice of adopting DM as technique for representing both documents and user profiles in our framework. The analysis about the use of distributional approaches in the area of CBRS continues in the next section, since in eVSM a technique
called Random Indexing is adopted to infer and represent items and user profiles in a semantic vector space.

### 3.2.2 Dimensionality Reduction Techniques for VSMs

More than fifty years ago Bellman [Bel61] coined the term *curse of dimensionality* to refer to the problem, typical of machine learning, of handling high-dimensional spaces. Indeed, the effectiveness of classical algorithms and techniques drops down as the number of dimensions (*features*) to manage grows up and the data become sparse. In VSMs this happens when the number of items to be represented gets bigger and the number of features gets bigger, as well.

The simplest way to reduce the dimension of a vector space is to exploit feature selection, that is, the technique of selecting a subset of relevant features for building robust learning models. Approaches for effective feature selection range from mathematical models to simple heuristics. Even if a detailed overview of techniques for feature selection is out of the scopes of this dissertation, it is worth to cite the analysis provided by Forman in [FGE03], where different metrics for feature selection, ranging from Document Frequency (DF) to Information Gain (IG) and Odds Radio, are compared. In general, as stated by Yang [YP97], the effectiveness of a specific technique is strictly related to the nature of data used in the task and to the task itself. In the area of CBRS most of the attempts presented in literature agree on the utility of feature selection, but they mainly focus on simple heuristics. As introduced in the previous chapter, Pazzani [PB97] adopted in Syskill & Webert a rough feature selection based on the 128 most informative terms. Next, Chakrabarti [Cha02] suggests to implement feature selection by simply filtering out common words and use only domain-specific or task-specific features. Moreover, the application of methods for natural language processing can be seen as a form of feature selection as well. *Stopwords removal*, for example, is a simple heuristics-based method whose goal is to filter out the space from irrelevant features according to a list of noisy terms. Similarly, the application of a stemming algorithm, such as Porter’s [Por80] one, reduces the number of features because many terms are merged into a single feature represented by its own root.

More complex approaches for dimensionality reduction are based on matrix factorization. A good overview of these techniques is provided by Fodor in [Fod02]. In a recommender systems scenario the goal of matrix factorization is to analyze the ratings provided by the users and derive from the emerged patterns a set of latent factors that characterize both items and users behavior. The
3.2 A Recommendation Framework based on VSM

use of these techniques has been encouraged by the Netflix Prize competition\(^4\) where these approaches have proved to be particularly helpful to improve the predictive accuracy of collaborative filtering (CF) algorithms [KBV09].

Singular value decomposition (SVD) [GK65] is one of the most common approach adopted in the area of recommender systems to discover the latent patterns hidden in mathematical structures, such as the user-item matrix typical of CF algorithms. Specifically, SVD is a matrix factorization technique that factors an \(m \times n\) matrix \(R\) into three matrices:

\[
R = U \times S \times V^T,
\]

where \(U\) and \(V\) are two orthogonal matrices of size \(m \times r\) and \(n \times r\) and \(S\) is a diagonal matrix of size \(r \times r\) whose diagonal entries are the singular values of matrix \(R\). All the entries of matrix \(S\) are positive and stored in decreasing magnitude order. The rank of matrix \(R = U \times S \times V\) is \(r\). The matrices obtained by performing SVD are particularly useful since SVD provides the best rank-\(r\) approximations of the original matrix \(R\).

The idea of using SVD in recommender systems area has been applied since the early 90s. In [SKKR00] the authors analyze how the application of SVD can affect the effectiveness of recommender systems, and many interesting insights emerged from the experimental evaluation. The SVD-based approach performed consistently worse than traditional collaborative filtering in e-commerce scenario, while in a movie recommendation one produced results better than a traditional CF algorithm. In general, as stated by Koren [KBV09], the impact of dimensionality reduction on recommendation algorithms is strictly related to the amount of data kept in the SVD.

However, a well-known drawback of SVD is represented by the complexity of the approach. Even if the factorization might be computed offline, it is not simple to state how frequently the matrix has to be generated again. In order to tackle this issue, Sarwar et al. proposed an incremental version of SVD [SKKR02].

Next, more specific techniques for matrix factorization emerged. Hofmann [Hof99], for example, proposed probabilistic Latent Semantic Analysis (pLSA) to infer user communities and interest patterns by analyzing user ratings. A different approach to dimensionality reduction was proposed by Goldberg et al. [GRGP01] in Eigentaste. They applied principal component analysis (PCA) in a joke recommendation scenario. The empirical evaluation showed that in some

\(^4\)www.netflixprize.com
settings Eigentaste’s predictive accuracy is comparable to a kNN algorithm. Furthermore, in \[\text{LWC}^+\text{06}\] is investigated the use of algorithms for non-negative matrix factorization (NMF) \[\text{LS00}\] in the area of CF.

In general, the aforementioned approaches have been evaluated in literature with respect to collaborative algorithms. For content-based recommendation it is common to reduce the dimension of the space by exploiting Latent Semantic Indexing (LSI) \[\text{DDL}^+\text{90}\], a method that has been extensively used in the field for inferring and representing the meaning of the terms through statistical computations applied to a large corpus of text. The algorithm for LSI works in two steps: first, the corpus is represented into a matrix in which each row stands for a word and each column stands for a text passage (or other contexts, such as paragraphs, sections or a single word). Next, the aforementioned SVD is applied in order to split the original matrix into two matrices of reduced dimensionality that respectively represent the original rows (terms) and the original columns (passages or contexts) in terms of latent orthogonal factors. It is generally agreed that through LSI it is possible to tackle the typical problem of synonymy \[\text{LD97}\] since many terms sharing a similar meaning may be merged into a single factor. Moreover, as pointed out by Berry \[\text{Ber92}\], the reduced orthogonal dimensions resulting from SVD are less noisy than the original data and capture the latent associations between terms and documents. The use of LSI is also investigated by Symeonidis in \[\text{Sym07}\], where a feature-reduced user model for RSs is proposed. In this work the author builds a feature profile for the users while LSI is exploited to build a pseudo-feature user concept able to reveal his real preferences. The experimental evaluation showed that the approach significantly outperforms existing CF, CBRS and hybrid algorithms. Furthermore, the use of LSI in the area of CBRS has been already investigated \[\text{MCK06, FD92}\] and it emerged a better accuracy with respect to other CBRS, regardless the specific domain it has been applied in.

However, LSI suffers from scalability problems inherited from the use of SVD as technique for dimensionality reduction. Consequently, the research has been oriented towards more scalable and incremental techniques such as those based on Random Projections (RP). By following this approach, the original matrix \( A \) is transformed into a reduced matrix \( A^* \) as follows,

\[
A^* = A \times R
\] (3.4)

where the row vectors that compose the matrix \( R \) are built in a pseudo-random way and are realizations of independent and identically distributed zero-mean vectors, scaled to have unit length. More details about the structure of
row vectors in the random matrix are provided next, because in our framework a dimensionality reduction method based on RP is adopted.

The method, originally proposed in the context of clustering text documents [KKL+00], showed results comparable to those obtained with PCA, by taking only fraction of the time [Kas98]. The development of the random projection was originated by the studies about near-orthogonality by Hecht-Nilsen [HN94]. Furthermore, in Johnson and Lindenstrauss' lemma [JL84] is demonstrated that whether the random matrix R is built by following specific constraints, the distances remain proportional also in the reduced vector space, so it is still possible to perform similarity calculations between points represented in the reduced space based on Random Projection with a minimum loss of significance. This important outcome has been experimentally confirmed [Kas98, Mag02].

Even if the use of RP is still less widespread with respect to SVD, it has already been investigated in literature. In [BM01], Bingham et al. showed the effectiveness of the approach with image and text data. Specifically, they compared RP to SVD and PCA, and the experiments showed that RP can provide both good accuracy and low computational costs. However, these results were not confirmed by Fradkin et al. [FM03] since in their experimental setting PCA outperformed RP, even if the computational advantages behind RP still make this model suitable for complex tasks. A first attempt towards the use of RP in Information Retrieval is described by Basile [BCS], while other experiments about the effectiveness of RP in Machine Learning-based tasks are proposed in [IM98] and [Das00]. However, the only attempt of exploiting RP in the area of recommender system is shown in [CSPZ10] and [SCJ11], where the authors applied it for CF algorithms. Finally, in [LG03] the authors provided a comparison between RP and LSI. The experimental evaluation showed that LSI, as expected, outperforms RP. However, the enormous gain in efficiency still makes RP feasible as dimensionality reduction algorithm for CBRS, especially for real-world scenarios.

The lack of a proper literature about the exploitation of RP in the area of CBRS, joint with the clear computational advantages that follow the use of RP, mainly motivated choice of adopting this class of approaches for dimensionality reduction to represent items and user profiles in the eVSM framework.

### 3.2.3 Modeling Negation in VSMs

Generally speaking, modeling user feedbacks is tremendously important for user-oriented platforms such as search engines or recommender systems. As proved by many contributions coming from both IF and IR areas, techniques for rele-
vance feedback can improve the overall accuracy of the algorithms [SB90, All96]. However, most of the literature focuses on the exploitation of the information coming from positive user feedbacks, even if many studies confirmed that also negative evidences can improve the effectiveness of IR [YHJ03] and IF systems [ZZLZ11].

The concepts behind negative relevance feedback are discussed by Dunlop [Dun97]. In this work emerged the idea of representing negation by subtracting an unwanted vector from a query, even if nothing about how much to subtract is stated. This insight is implemented in the Rocchio algorithm, that is one of the most common formalization of a relevance feedback technique for vector space models. In [Roc71] the problem is defined as those of finding an optimal query that maximizes the difference between the average vector of relevant documents and that of non-relevant ones (See Figure 3.2). The process of formulating such an optimal query is seen in Rocchio as an iterative process, where each loop moves the query further from the non-relevant documents and closer to the relevant ones. This is done through a re-weighting of the query according to the results returned by the query at the previous iteration. Formally,

\[
Q_{i+1} = \alpha \times Q_i + \beta \left( \frac{1}{|D^+|} \sum_{d^+ \in D^+} d^+ \right) - \gamma \left( \frac{1}{|D^-|} \sum_{d^- \in D^-} d^- \right),
\]

(3.5)

where \(Q_i\) is the query at step \(i\), and \(D^+\) and \(D^-\) are respectively the set of relevant and non relevant documents, that are summed in order to build the aforementioned average vector. Finally, \(\alpha\), \(\beta\), and \(\gamma\) are parameters use to tune the movement of the query in the vector space. An extensive study about tuning parameter of Rocchio algorithm for text classification has been performed by Moschitti [Mos03]. Even if there is no theoretical basis behind this formula, empirical evaluations showed that the retrieval performance can be improved, especially after the first iteration [PB07]. A more complex method for negative relevance feedback is proposed in [LAWX09], where Li et al. introduce an algorithm that analyzes a textual corpus in order to identify general concepts that are not to be considered in the relevance feedback procedure, by assuming that only very specific concepts have to be taken into account to refine the query and move it towards the relevant documents. The approach seems to be effective since in the experimental settings, based on the Reuters dataset\(^5\), it improved the overall accuracy with respect to classical algorithms for RF.

However, negative relevance feedback is not the only technique that can be exploited to represent negation in vector spaces. In IR area, for example, is very

\(^5\)http://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection
common to model it as a form of post-retrieval filtering. The underlying idea [SM84] behind this approach is to filter out the result set from the documents containing unwanted terms. Formally, given a query $a \text{ NOT } b$ this form of negation is obtained by filtering out the documents returned by the query $a$ from those where the term $b$, put in the negative part, appears. This technique suffers of a clear drawback, since very important documents where the term $b$ appears once are not returned to the user. Consequently, it is common to implement negation in a less rigid way, by introducing similarity measures that simply penalize documents (instead of disqualifying them) when unwanted terms appear.

Furthermore, in [WP03], Widdows and Peters proposed a scheme for negation in vector spaces based on logic connectives defined in Quantum Mechanics [BVN36]. This form of negation is one of the outcomes of Widdows’ analysis about the convergence between Information Retrieval and Natural Language Processing areas with the principles of Quantum Mechanics, by following Van Rijsbergen’s experience [van04] in this field. The insight behind this operator is that the negation can be considered as a form of orthogonality between vectors since two non-related concepts should have no features in common. Under a theoretical point of view, in this scenario the vector $a \text{ NOT } b$ represents the projection of the vector $a$ on the subspace orthogonal to those generated by the vector $b$. The experimental evaluation described in [Wid03] showed that
this novel form of negation is effective in removing unwanted terms and their synonyms and related terms as well.

The strong Quantum Logic-based theoretical basis behind this form of negation, joint with some general criticism about the use of Rocchio-based approaches for modeling negative feedbacks [Kow97] and with the lack of a proper investigation about the effectiveness of this operator in the area of CBRS are the motivations that triggered the choice of adopting Quantum Negation as method for representing negative evidences in our framework. In the next section a thorough description of the operator and its application in a CBRS scenario is given.

### 3.2.4 Discussion

To sum up, none of the state-of-the-art models is able to overcome at the same time all the VSM weaknesses. Regarding the semantics, approaches based on WSD can improve the semantic representation but there is not a general agreement about their effectiveness in the area of CBRS. On the other side, techniques such as LSI is can ensure a better semantic modeling but they are actually not scalable. In general, scalable distributional models (DM) can fulfill the need for semantic modeling with efficiency as well but there is a lack of a proper literature on the use of DM in the area of CBRS. However, all these approaches cannot represent negative user feedbacks, so the final outcome of this overview is that a complete framework for content-based recommendations has to merge the different techniques that can best handle each of the aforementioned issues. In the next section the design choices between eVSM are properly explained and justified.

### 3.3 Description of eVSM

The enhanced Vector Space Model (eVSM) is a novel recommendation framework, based on VSM, which has been adapted and extended and in order to exploit the effectiveness of the original VSM in a recommendation scenario as well.

First, document representation is enhanced by exploiting distributional approaches. The insight is to use this class of models as main building block for representing both items and user profiles in a vector space thanks to the lightweight semantic modeling they can provide. Specifically, Random Indexing, is adopted as incremental technique based on Random Projection that guarantees the scalability required by CBRS. Next, the framework is enriched through a negation operator based on Quantum Logic that let model negative
user preferences. The combination of distributional models with random indexing and negation operator based on quantum logic represent the core of the eVSM framework.

3.3 Description of eVSM

3.3.1 Random Indexing

Distributional Models got their name from distributional semantics and describe a set of techniques originally introduced in computational linguistics and cognitive sciences. These approaches are based on the assumption that the semantics of a term can be inferred by analyzing its use in large corpus of textual data. Specifically, these techniques rely on the distributional hypothesis, which states that "Words that occur in the same contexts tend to have similar meanings".

By following the famous Wittgenstein’s sentence ("Meaning is its use") it is possible, as already demonstrated by Rubenstein and Goodenough in the mid-1960s [RG65], to infer the meaning of a term (such as leash) by analyzing the meaning of the other terms it co-occurs with (dog, animal, etc.). Similarly, the correlation between different terms (e.g., leash and muzzle) can be inferred by analyzing how similar are the contexts in which they are used.

The use of this methodology provides a clear advantage, since a model to represent terms and documents in semantic vector spaces, the aforementioned Word Spaces [Low01], can be built in a totally unsupervised way according to the use of the terms in a corpus of textual data. However, since these approaches usually rely on large vector space representations, it is necessary to couple them with dimensionality reduction techniques able to reduce the computational costs required to build and update the model.

In eVSM terms and documents (or, by generalizing, items and user profiles) are represented by exploiting Random Indexing (RI), an incremental technique for creating small-scale Word Spaces that merges the advantages of distributional models with the efficiency of dimensionality reduction based on Random Projection. The technique has been proposed by Sahlgren [Sah05] and exploits Patti Kanerva’s works [Kan88, KKH00] on sparse distributed representations.

As Latent Semantic Indexing (LSI), RI represents terms and documents as points in a semantic vector space that is built according to distributional hypothesis. However, differently from LSI, RI uses Random Projection as technique for dimensionality reduction instead of SVD.

The goal of RI is to approximate the n-dimensional term-document matrix typical of VSM with a more compact and flexible k-dimensional representation, where $k \ll n$. According to distributional hypothesis, this representation is based on the concept of context, so each term is no more represented according
to the documents it occurs in, as in classical VSM, but according to the contexts it occurs in. This choice gives two clear advantages: first, the dimension $k$ of the vector space is a simple parameter, so it is not fixed and can be simply adapted to the requirements of the specific application domain. Intuitively, the larger the vector space, the higher the precision in representing word similarities as well as the need for computational resources to represent and update the model.

Next, the definition of what the context is can be adapted to the scenario as well. Given a word, we could think at the context as a piece of text, variable in size, which surrounds that word. The context of a term could be a sliding window of a couple of terms that surround it on the left and on the right, a sentence, a paragraph, or the whole document. In this dissertation, for example, we exploited the simplest formulation we can provide, since the context of a term is defined as the whole document it occurs in.

This novel $k$-dimensional representation is obtained by following a simple and incremental strategy:

1. A $k$-dimensional context vector is assigned to each context (document, in this setting).

2. The vector space representation of a term (denoted by $\vec{t}$) is obtained by summing the context vectors of all the contexts it co-occurs with.

3. The vector space representation of a document (denoted by $\vec{d}$) is obtained by summing the vector space representation of all the terms that occur in it.

At the end of the step 2 a WordSpace is built, while at the end of step 3 a DocSpace is generated. In a WordSpace it is possible to compute similarities between different terms while in a DocSpace this is done for documents. However, since both WordSpace and DocSpace share the same $k$-dimensional representation, it is possible to project an element of WordSpace (that is to say, a term) into a textscDocSpace in order to find the documents that are related the most with the term. Similarly, it is possible to project an element of DocSpace into WordSpace and to use cosine similarity to find the terms that better describe a certain document (as in collaborative tagging systems). In eVSM both items and user profiles are elements of a DocSpace, so similarity calculations are performed in that space.

As stated above, RI is strictly related to Random Projection, which put its theoretical basis in the aforementioned Hecht-Nielsen’s studies about near-orthogonality [HN94]. Consequently, the generation of the context vectors is
not totally random. On the contrary, specific constraints have to be followed. Specifically, each context vector:

- Contains only values in \{-1, 0, 1\}.
- Contains value distributed in a random way, but the number of non-zero elements has to be much smaller. Specifically, a very common choice is to use Gaussian distribution for the elements of the context vectors. However, Achlioptas [Ach01] has shown that much simpler distributions (zero mean distributions with unit variance) can also be used.

To sum up, given a set of documents, by following this approach we can build a low-dimensional approximation of the vector space that is supposed to give a better semantic modeling of the documents since each term is no longer represented in an atomic way, as in the classical keyword-based methods, but its position in the space might depend on the terms it co-occurs with. Furthermore, the approach is totally incremental. When a new document (item) comes into play, the algorithm randomly generates a context vector for it and updates the vector space representation of each term that occurs in the document. The technique can scale well because the calculation of the vector space representation of this new document does not require to generate again the whole vector, but it is simply obtained by summing the context vectors of the terms that occurs in it.

Finally, as stated above, the main advantage behind RI is that in this low-dimensional space, as proved by the Johnson-Lindestrauss lemma [JL84], whether the pairwise orthogonality between vectors is verified, the distance between points remains proportional, so it is possible to perform calculations and compute similarity between items represented in the vector space with a minimum loss of accuracy balanced by the enormous gain in efficiency. As shown in Figure 3.3, in the reduced vector space the point that is the nearest to \(X\) is always \(Z\) as in the original vector space, even if the numerical value of their pairwise similarity will be obviously different.

### 3.3.2 Quantum Negation

Thanks to RI it is possible to build low-dimensional **WordSpaces** and **DocSpaces** that maintain the original expressivity of the model, even if the content are represented with a considerable reduction in terms of features. However, this novel representation inherits a classical issue of VSMs since the information coming from negative evidences (such as relevance feedbacks and user preferr-
Enhanced Vector Space Models (eVSM) for CBRS

Figure 3.3: A visual explanation of the Johnson-Linderstrauss lemma

ences) is not taken into account. In CBRS this problem is actually felt when user profiles have to be built.

Indeed, when user profiles are built this is an important aspect because negative user preferences have to be modeled as well as positive ones. As shown by Morita et al. [MS94], as the number of positive-only feedbacks grows up their discriminative power drops down and the quality of user profiles drops down as well. Consequently, in order to tackle this drawback the aforementioned quantum negation (QN) operator is introduced in eVSM. It was preferred to other forms for representing negation, such as Rocchio formula, because of its more solid theoretical foundations based on Quantum Logic.

The underlying idea is very simple: two words (or, in general, two concepts or even two documents) can be considered as mutually irrelevant if they have no features in common. In a vector space representation this means that their scalar product is 0, that is, they are pairwise orthogonal. By analogy, in a CBRS scenario we can split a user profile into two separated components, a positive one, where the information about what a user likes is stored, and a negative one, where we represent what is irrelevant for her. These two vectors have to be mutually orthogonal as well.

Consequently, according to the principles of Quantum Logic, since the negation is seen as a form orthogonality between vectors, an expression such a NOT b represents the projection of the vector a on the subspace orthogonal to those generated by the vector b. In a CBRS scenario, if a and b are respectively the
positive and negative profile vectors, the negation operator will be used to identify the subspace that will contain the items with as more as possible features coming from a and as little as possible ones coming from b. Or, alternatively, as close as possible to the positive preference vector and as far as possible to negative one.

However, the idea of combining QN with a representation based on Random Indexing is not totally new, since Widdows already investigated [WC10] the exploitation of quantum negation into distributional models. He developed also an open source package, called Semantic Vectors\(^6\), where the model is implemented. The strong theoretical basis and the preliminary experimental evidences [WF08] that confirmed the goodness of the approach drew inspiration for eVSM design and largely motivated the choice of adopting a representation based on both Random Indexing and Quantum Negation for modeling user profiles in a recommendation framework based on VSM.

3.3.3 Building User Profiles

Given a DocSpace built according to the algorithm introduced in Section 3.3.1, it is necessary to define an approach to infer user profiles as well. In this thesis four different profiling models are introduced, all based on Random Indexing as technique for semantically represent user preferences and Quantum Negation to model negative evidences. The main difference among the approaches lies in the way the information about both positive and negative user preferences is combined in order to represent the user profile in a DocSpace.

Random Indexing-based Profiles

This approach is based on the assumption that the information coming from the items a user liked in the past can be a reliable source of information to build accurate user profiles. Therefore, let \(d_1 \ldots d_n \in D\) be a set of already rated items, and \(r(u, d_i) (i = 1 \ldots n)\) the rating given by the user \(u\) to the item \(d_i\). We can describe the set of relevant items for user \(u\), denoted by \(I_u\), as follows:

\[
I_u = \{d \in D | r(u, d) \geq \beta\}
\]  

Thus, given a threshold \(\beta\), the profile of a user consists of the set of the terms occurring in the documents she liked in the past. As stated above, Random Indexing is exploited to build the user profile in an incremental way, that is to

\(^6\)http://code.google.com/p/semanticvectors/
say by simply summing all the document vectors for each document in $I_u$. Let $|I_u|$ be the cardinality of the set $I_u$ and let $\vec{d}_i$ be the vector space representation of the document $d_i$, the user profile $\vec{p}_u$ can be defined as follows:

$$\vec{p}_u = \sum_{i=1}^{ |I_u| } \vec{d}_i$$

(3.7)

That is undoubtedly the simplest Random Indexing-based filtering model that could be defined. In the experimental evaluation we will refer to this model as RI.

**Weighted Random Indexing-based Profiles**

The main drawback of the RI method is that the user profile $\vec{p}_u$ is built without taking into account the ratings provided by the target user for the items she liked. In other terms, it is independent from the ratings provided by the target user (provided that they are above or below the threshold $\beta$). The second model, called Weighted Random Indexing-based (W-RI), enriches the previous one by simply associating to each document vector, before combining it, a weight equal to the rating provided by the user for it. More formally:

$$\vec{p}_u = \sum_{i=1}^{ |I_u| } \vec{d}_i \ast r(u, d_i)$$

(3.8)

In this way the model will increase the weight of the items liked by the user. Obviously the score $r(u, d_i)$ can be also normalized with the highest rating that can be assigned to an item.

**Quantum Negation-based Profiles**

The main idea behind Quantum Negation-based model (QN) is to exploit the negation operator to represent in the user profile both positive and negative preferences, as in the classical text classification approaches (e.g. Naïve Bayes, Support Vector Machines and so on). We can think at this model as an extension of the previously described RI model. Unlike RI, in which a single user profile $\vec{p}_u$ is built, in QN profiling model two user profile vectors, one for positive preferences and one for negative ones, are inferred. The set of relevant items $I^+_u$ and the positive user profile vector $\vec{p}^+_u$ are identical to the set of relevant items $I_u$ and the user profile $\vec{p}_u$ in RI, while the set of non-relevant items, denoted by $I^-_u$, is defined as follows:
3.3 Description of eVSM

\[ I_u^- = \{ d \in D | r(u, d_i) < \beta \} \]  \hspace{1cm} (3.9)

The negative user profile vector, denoted by \( \vec{p}_{-u} \), is built by summing the vector space representations of the items in \( I_u^- \). Formally:

\[ \vec{p}_{-u} = \sum_{i=1}^{ |I_u^-| } \vec{d}_i \]  \hspace{1cm} (3.10)

Thus, given the profile vectors \( \vec{p}_{+u} \) and \( \vec{p}_{-u} \) we can use QN-based model to instantiate the vector \( \vec{p}_{+u} \) \textit{NOT} \( \vec{p}_{-u} \), that is exploited to find the items represented in the vector space that contain as much as possible features that occur in the documents in \( I_u^+ \) and as little as possible features from \( I_u^- \).

Weighted Quantum Negation-based Profiles

As RI, the QN model has its weighted counterpart, called \textbf{W-QN}. This model shares the same idea of the W-RI model and the same weighting schema described in 3.3.3, with the unique difference that in the negative profile \( I_u^- \) the items with a lower rate are given higher weights in order to exclude as much as possible the features disliked by the target user. More formally, the set \( I_u^+ \) and \( I_u^- \) are built by following the same formula introduced in the previous section, while the vectors \( \vec{p}_{+u} \) and \( \vec{p}_{-u} \) are inferred in this way:

\[ \vec{p}_{+u} = \sum_{i=1}^{ |I_u^+| } \vec{d}_i \ast r(u, d_i) \]  \hspace{1cm} (3.11)

\[ \vec{p}_{-u} = \sum_{i=1}^{ |I_u^-| } \vec{d}_i \ast (MAX - r(u, d_i)) \]  \hspace{1cm} (3.12)

where \( MAX \) is the highest rating that can be assigned to an item (document).

3.3.4 Generating Recommendation

The recommendation framework proposed in this dissertation tries to prove that the exploitation of the classical IR-based techniques and similarity measures can be useful for filtering items represented as points in an \textit{enhanced vector space}. Furthermore, one of the strongest point of the eVSM lies in that it can be
Enhanced Vector Space Models (eVSM) for CBRS

exploited, without any adaptation, to provide users with recommendations in both monolingual and multilingual environments.

In general we refer to monolingual scenarios when both items to be recommended and user profiles are represented through features coming from the same language. On the other side, a multilingual environment is those where the language that describes profiles and recommendations might be different. For example, a user reads online news articles written by an Italian newspaper (so her user profile will contain Italian features) and the algorithm is capable of providing recommendation with relevant news articles written in different languages as well.

The problem of providing users with multilingual recommendations is actually not trivial, since one of the typical issues of CBRS is that recommendations are language-dependent. An English user, for example, frequently interacts with information written in English, so her profile of interests mainly contains English features. In order to receive suggestions about items whose textual descriptions are provided in a different language, she must explicitly express her preferences on items in that specific language, as well. This means that the information already stored in the user profile cannot be exploited to provide suggestions for items whose description is available in other languages, although they share some common characteristics (e.g. an Italian and an English documentary might refer to the same topic but their plots could be written in two different languages).

Distributional models (DM) can simply overcome this issue since it is possible to state that they are inherently multilanguage. Indeed, items (and, consequently, user profiles) representation is inferred by analyzing their usage and, generally speaking, it is not wrong to assume that word usage is almost language independent (especially for those languages that belong to the same class, such as Latin or European ones), since we can state that a concept, expressed in different way depending on the language, is used and co-occurs with the same set of high-level concepts. For example, as shown in Figure 3.4, it is possible to state that the concept of beer co-occurs with verbs referring to the act of drinking and nouns referring to something the drink is put inside, and this is true regardless the specific language. The specific languages only influence the way these high-level concepts are expressed, so in Italian the concept of birra will co-occur with the verbs bere and bicchiere, while in English the term beer will co-occur with drink and glass. According to this (strong) assumption, in the next section is described an approach for exploiting this language-independent representation to provide users with multilanguage recommendations.
3.3 Description of eVSM

Monolingual Recommendations
The steps for generating monolingual recommendations are straightforward. The main idea behind eVSM is to build a vector space (specifically, a DocSpace) for each user, where both user profile and items to be filtered are represented through the techniques described in the previous section.

Given this representation, similarity measures between vectors (such as the classical cosine similarity) can be used in order to get the set of the most relevant items for the target user, by assuming that the points in the space with the highest cosine similarity can be considered as relevant for her.

As in classical IR systems, given a set of items $i_1 \ldots i_n$ and a user profile $u$, the cosine similarity is computed for all the couples $(i_j, u)$ and the documents $i_j$ are ranked according to their descending cosine similarity. Finally, the top $n$ are returned to the target user. Alternatively, thresholding strategies can also be adopted to recommend only the items whose similarity score overcomes a certain value.

Multilingual Recommendations
The approach for generating multilingual recommendations is slightly different: for each item the textual content are collected in two language (for example, in English and Italian) in order to build a multilingual space. The main difference between a multilingual space and a monolingual one is that in this space each item $i$ has two fields $i_{L1}$ and $i_{L2}$, which store the same content in two different languages (for example, in a movie recommendation scenario, the plot in English and Italian). It is important to underline that not necessarily the content of $i_{L2}$ is the perfect translation of $i_{L1}$. As stated above, the power of distributional approaches lies in that two terms, in different languages, are similar because they share the same context.
Next, to generate multilingual recommendations we need to iterate twice the RI algorithm, in order generate four spaces: two WordSpace $WS_{L1}$ and $WS_{L2}$ and two DocumentSpace $DS_{L1}$ and $DS_{L2}$. These spaces are built as follows:

1. a context vector is assigned to each context (e.g., a document) as described in $RI$ algorithm. We call this space $RB$ (random base);

2. the semantic vector for a term in $WS_{L1}$ is computed as the sum of the context vectors in $RB$ for the movies (plots) which contain that term in the field $i_{L1}$;

3. the semantic vector for a movie (plot) in $DS_{L1}$ is computed as the sum of the semantic vectors for the terms in $WS_{L1}$ which occur in that movie (plot) in the field $i_{L1}$;

4. the semantic vector for a term in $WS_{L2}$ is computed as the sum of the context vectors in $RB$ for the movies (plots) which contain that term in the field $i_{L2}$;

5. the semantic vector for a movie (plot) in $DS_{L2}$ is computed as the sum of the semantic vectors for the terms in $WS_{L2}$ which occur in that movie (plot).

Since all the spaces share the same random base $RB$, it is possible to exploit this uniform representation to compare elements that belong to different spaces. Specifically, we can infer a user profile for $L1$, project it into the DocSpace for $L2$ and compute how much it is similar to items in $DS_{L2}$.

As for monolingual recommendations, given a set of items described in $L1$ $i_1^{L2} \ldots i_n^{L2}$ and a user profile built in $L2$ $u_{L1}$ the cosine similarity is computed for all the couples $(i_j^{L2}, u_{L1})$ and the documents $i_j$ are ranked again according to their descending cosine similarity.

In the next section the effectiveness and the accuracy of this approach is investigated by comparing it with other state-of-the-art approaches for providing users with cross-lingua recommendations.
3.3 Description of eVSM
Chapter 4

Applications of eVSM

This chapter summarizes the results of the evaluations performed on eVSM in both offline and online experiments. Specifically, the content-based recommendation framework is evaluated in different settings in order to validate its effectiveness with respect to other state-of-the-art approaches as well as with real users.

In Section 4.1 eVSM deals with the task of providing user with recommendations about movies. The accuracy of the approach, calculated by exploiting the well-known MovieLens and Yahoo! Webscope dataset, is compared to other techniques such as VSM, LSI and a Bayes text classifier. This evaluation also analyzes how the overall accuracy of the algorithm is influenced by the parameters of the models, such as context vectors size and the technique adopted for building user profiles in both multilingual and monolingual scenario. Next, Section 4.2 introduces Play.me, a platform that generates personalized musical playlists according to social media information extracted from Facebook. In this case eVSM is adopted as enriching technique to add new artists related to those the user already likes. In the experimental evaluation eVSM is compared to another enriching approach based on the exploitation of Linked Data. Section 4.3 describes Myusic, a platform for content-based music recommendation that adopts eVSM as CBRS. Myusic has been implemented to validate the effectiveness of the approach in a real use case. Specifically, a user study involving 50 users was designed, with the goal to evaluate the goodness of the recommendations in terms of novelty, diversity and serendipity.
4.1 Movielens and Yahoo! Webscope: content-based movie recommendation

In this experimental session the effectiveness of RI and QN models, as well as of their weighted variants W-RI and W-QN, is evaluated with respect to other state-of-the-art models in the task of providing users with recommendations about movies worth to be watched. In the first experiment the accuracy of the algorithms and the goodness of the rankings is evaluated with respect to different datasets in a monolingual scenario, while in the latter the settings were adapted to a multilingual one.

In the monolingual setting both a subset of the 100k MovieLens dataset\(^1\) and the Webscope dataset\(^2\) provided by Yahoo! were exploited, while for the multilingual one the English version of the Movielens dataset was enriched by crawling Italian content.

**Movielens dataset** contains 40,717 ratings provided by 613 different users on 520 movies. Since the original dataset did not contain any information about the content of the movies, the content were crawled from both the English and Italian version of Wikipedia. In particular the crawler gathered the *Title* of the movie, the name of the *Director*, the *Starring* and the *Plot*. For the monolingual experiments only the English content were considered while for monolingual ones the Italian content were taken into account as well. It is worth to note that English and Italian movie descriptions are not a mere translation from one language to another: in Wikipedia different contributors have provided the English and Italian movie plots, thus different sets of terms have been used. For both approaches the text in each slot has been tokenized, stemmed and the stopwords have been removed.

In Table 4.1 some statistics about the dataset is provided: the original term-document matrix contained more than 25k rows (*features*) and 520 columns (*items*) on average. These data show that the dataset taken into account is largely unbalanced since more than 80% ratings is positive.

**Yahoo! Webscope dataset** is composed by 221,364 ratings provided by 7,642 different users on 106,959 movies. The content available for each item

\(^1\)[http://www.grouplens.org/node/73]
\(^2\)[http://webscope.sandbox.yahoo.com/]

<table>
<thead>
<tr>
<th>Items</th>
<th>520</th>
<th>Ratings</th>
<th>40,717</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratings (avg/user)</td>
<td>66.44</td>
<td>Positive ratings</td>
<td>83.8%</td>
</tr>
</tbody>
</table>
is: *Title, Genre, Synopsis, Director, Running Time, Distributor* and other less
relevant information. However, many fields are empty for several movies. The
ratings are already split in a training and test set, but only 2,309 users have
ratings in either sets. We select all the movies for which the synopsis are avail-
able and for which ratings are expressed. We also select the user that have both
a training and a test set (some users have only training examples). After this
selection, the dataset contains 84,050 ratings provided by 2,309 different users
on 6,943 movies. The text in the plot and in the synopsis is tokenized, stemmed,
and stopwords are eliminated.

**Design of the experiments.** In both evaluations user profiles were
learned by analyzing the ratings stored in the datasets. Each rating was ex-
pressed as a numerical vote on a 5-point Likert scale, ranging from 1=strongly
dislike to 5=strongly like. In both monolingual and multilingual experiment all
the ratings above 2 were considered as positive.

The sessions were organized through a 5-fold cross validation [Koh95]: for
each fold and for each user a vector space for the user profile and for the items to
be filtered is built. By exploiting a simple cosine similarity measure we ranked
the items, assuming the nearest ones as the most relevant. The metric used
to evaluate the effectiveness of the approaches was the *Average Precision@n*,
where *n* was set to 1, 3, 5 and 10. It was preferred because, differently from
the classical Precision@n, it takes into account also the position of the correctly
classified items. In order to validate the performance of the different profiling
strategies we used the *paired t-test* [OD90].

In both experiments the attention was focused on the following four ques-
tions:

- **Experiment 1:** does the dimension of the context vectors affect the
  predictive accuracy of the recommendation models?
- **Experiment 2:** do the weighting scheme and the negation operator im-
  prove the predictive accuracy of the recommendation models?
- **Experiment 3:** how does the framework perform with respect to other
  content-based recommendation approaches?

### 4.1.1 Monolingual Recommendations - Movielens dataset

**Experiment 1**

The effectiveness of the profiling models implemented in the recommendation
framework was evaluated with different sizes of context vectors, by setting the
size parameter to 50, 100, 200 and 400. By considering that the size of the
original DocSpace was almost 25k (that is to say, the number of features in
the dataset), we reduced the size of more than 99%. Intuitively, with smaller
contexts vector there is a bigger loss of information due to the compression of
the vector space, balanced by the gain in efficiency in building and updating
the model.

The outcomes of the experiment are sketched in Figure 4.1 and 4.2. From
a rough analysis of the precision it immediately emerges that the dimension
of context vectors does not affect the precision of the algorithm since it does
not seem that a specific configuration systematically overcomes the other ones
and the differences among the different configurations are really small. Albeit
it might be expected that larger context vectors improve the accuracy of the
recommendations, the experimental evidence shows that this is not true. On the
contrary, even with size=50 the predictive accuracy of eVSM remains acceptable
so it is possible to state that with a stronger application of the dimensionality
reduction technique, the framework is able to produce accurate recommendations as well. A paired t-test performed on the experimental data confirmed
this inceptive outcome and showed that the gap among the configurations is
not always significant and, above all, does not follow a definite trend.

Experiment 2

Figure 4.3 depicts the results of Experiment 2. The impact of the weighting
scheme emerges from a comparison of RI to W-RI and QN to W-QN while the
effectiveness of the negation operator can be inferred by comparing RI to QN
and W-RI to W-QN.

For both experiments, the outcomes are clear. In all settings the average
precision of weighted configurations overcame those of their unweighted coun-
terparts. However, the differences are not noteworthy. Regardless the size of
the context vectors the gain in precision is around 0.2% for all the configurations,
ranging from 0.05% to a peak of 0.52% by comparing QN to W-QN with
size=100 on AV-P@1. It is worth to note that in all comparisons the gaps got
their peak in AV-P@1 and AV-P@3. This underlines that weighted profiles are
able to better catch user preferences since the most relevant items for the tar-
get user are put at the top of the recommendation lists. However, regardless
the size of the vector space, the differences are not statistically significant and
noteworthy.

Furthermore, the results confirm the effectiveness of the negation operator
based on Quantum Logic as well. In this case the gap between the configurations
Figure 4.1: Outcomes of Experiment 1 for RI and W-RI profiles (MovieLens dataset)
4.1 Movielens and Yahoo! Webscope: content-based movie recommendation

Figure 4.2: Outcomes of Experiment 1 for QN and W-QN profiles (Movielens dataset)
that adopt negation operator and those only based on positive preferences is bigger. With size=100, the differences between W-RI and W-QN range between 0.7% in AV-P@3) and 1.1% in AV-P@1. However, for AV-P@5 and AV-P@10 the differences stay under 0.15%. With size=400 the gap is more stable since it always maintains between 0.4% and 0.5% in most of the comparisons. A paired t-test showed that the differences are significant \( p<0.05 \) for P@1 and P@3 with size=100 by comparing RI to QN and W-RI to W-QN, while for size=50 and size=400 only the difference between W-RI and W-SV in P@1 is significant. Between the other configurations did not emerge any difference.

In general, by analyzing the charts in Figure 4.3 and 4.4 at a glance, it emerges that the W-QN configurations gained the best results in all metrics. Regardless the size of the vector space, the results show that the combined use of negation operator coupled with a weighing scheme for better modeling user preferences can substantially improve the predictive accuracy of the whole model, thus confirming the assumptions and the insights behind the design of the eVSM framework. A paired t-test confirmed this inceptive outcome since a comparison between RI and W-QN models showed a significant difference between the profiling techniques for both AV-P@1 and AV-P@3 (except for size=50) with \( p<0.05 \). For AV-P@5 and AV-P@10 (except for size=400, where the test is significant for \( p<0.05 \)), even if the results show a clear trend the gap did not appear statistically relevant.

**Experiment 3**

In Experiment 3 the predictive accuracy of the best configuration of eVSM (that is to say, W-QN) is compared to other state-of-the-art algorithms based on VSM in order to validate the framework with respect to other approaches already proposed in literature. Specifically, eVSM is first compared to LSI and VSM. In the first case the goal is to evaluate the effectiveness of eVSM with respect to other well-known techniques for semantic modeling and dimensionality reduction, while in the latter the goal is to quantify the impact of the combined use of negation and distributional models on the classical vector space model. Finally, as third baseline a recommendation model based on a Bayes text categorization algorithm, one of the most common approaches for implementing a CBRS, is adopted. The Bayesian model exploited in the comparison is described thoroughly in [LdGS+09]. The results of this comparisons are shown in Figure 4.5. For Experiment 3 only the results with \( \text{size}=100 \) and \( \text{size}=400 \) were provided. Obviously the results of the baselines do not change since they are not dependent on the size of the vector space.
4.1 Movielens and Yahoo! Webscope: content-based movie recommendation

**Figure 4.3:** Outcomes of Experiment 2 for dimension 50 and 100 (Movielens dataset)
Figure 4.4: Outcomes of Experiment 2 for dimension 200 and 400 (Movielens dataset)
The main outcome of this final experiment is that eVSM overcomes all the baselines in both configurations. The differences are outstanding specifically for AV-P@1 and AV-P@3 where the gap among eVSM and the baseline is around 1%. These results further confirm that eVSM puts the most relevant items at the top of the recommendation lists. Unlike previous experiments, the difference remains stable also for AV-P@5. The gap among the algorithms progressively drops down in AV-P@10, but this can be justified in the nature of the dataset. As shown in Table 4.1, the dataset considered for this evaluation is very unbalanced. More than 80% ratings are positive, so as the recommendation lists get bigger the likelihood of putting a relevant item in the list gets bigger, too, and this undoubtedly produces an attenuation in the differences between the different algorithms. In a scenario where most of the possible recommendations are a priori relevant, the ability of an algorithm of providing a better ranking of the recommendations is actually desired, and eVSM showed in all the experiments its effectiveness in carrying out this task. Finally, a paired t-test confirmed the above outcomes since the differences are significant for P@1 (size=100 and 400), P@3 (size=100 and size=400) and P@10 (size=400), so it is possible to state that the eVSM framework outperformed the other approaches in a significant way.

**Figure 4.5: Outcomes of Experiment 3**
4.1.2 Monolingual Recommendations - Yahoo! dataset

Experiment 1

According to the different scenario, the size was set to 500, 1000, 1500 and 2000. The outcomes of the experiment are shown in Figure 4.6 and 4.7. As emerged in the previous experimental setting, it seems that the dimension of the context vectors does not influence the predictive accuracy of the algorithm. The precision does not follow a clear trend, and, above all, it does not emerge a correlation between the dimension of the vector space and the accuracy of the algorithm, since the accuracy sometimes reachs its peak by setting the size parameter to 500 or 1000.

Figure 4.6: Outcomes of Experiment 1 for RI and W-RI profiles (Yahoo! dataset)

Generally speaking, the gap among the different configuration is little. It always stays under 0.3 points % and in some settings, such as W-RI profiles with P@3, there is even no gain in making the vector space larger. Thus,
also in this scenario, it is possible to state that a stronger application of the
dimensionality reduction technique still makes the framework able to produce
accurate recommendations. This inceptive outcome is confirmed by a paired
\textit{t-test} performed on the experimental data.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{fig47.png}
\caption{Outcomes of Experiment 1 for QN and W-QN profiles (Yahoo! dataset)}
\end{figure}

\textbf{Experiment 2}

The results of Experiment 2 are shown in Figure 4.3. As already explained for
the experiment performed on the Movielens dataset, the impact of the weighting
scheme emerges from a comparison of RI to W-RI and QN to W-QN while the
effectiveness of the negation operator can be inferred by comparing RI to QN
and W-RI to W-QN.

By simply analyzing the charts, it emerges some clear outcome. First of
all, RI profiling model overcomes its weighed counterpart in all the compar-
Comparisons. Usually the gap is really little, with a peak of 0.27 points % in P@1, but W-RI profiles never perform better than RI. This means that the simple weighting schema is not effective for modeling user preferences. However, the gap is not statistically significant because of the small differences among the different configurations. Similarly, in most of the comparisons QN profiles got the best results with respect to W-QN, with a peak of 0.5 points. However, in some comparison W-QN profiles overcome QN ones (for example in P@5, except for size=500). This result is probably due to the nature of the dataset: it is largely unbalanced (almost 90% of positive ratings), so when some negative feedback is collected, this can positively affect the whole accuracy of the model. However, the differences between the configurations are not significant as well, so it is possible to state that in this experimental setting the adoption of the weighting scheme does not influences eVSM performances.

Furthermore, the results confirm the effectiveness of the negation operator based on Quantum Logic with this dataset as well. In this case the gap between the configurations is particularly significant for P@1, where the gain in precision is around 1 point % in all the comparisons, with a peak of 1.4 points % with size=500 by comparing W-RI to W-QN. For P@3 and P@5 the gap ranges between 0.5 and 1 %, but the profiles that include negative user feedbacks always perform better with respect to the models that do not model negative evidences. The effectiveness of the negation operator is also confirmed by a paired t-test performed on the experimental data: the difference between the different configurations is always significant with the exception of the comparison between W-RI and W-QN in P@3, regardless the size of the vector space.

Finally, by analyzing the charts in Figure 4.8 and 4.9, it emerges that the QN configurations gained the best results in all metrics. Differently from the previous experimental scenario, the unweighted profiles got the best results.

**Experiment 3**

In Experiment 3 eVSM is evaluated with respect to other state-of-the-art algorithms for text classification. Specifically, the configuration that showed to perform best in the previous experiment (that is to say, QN, with size=2000) is compared with Support Vector Machines (SVM) and a Bayes text classifier. The results of this comparisons are shown in Figure 4.10.

The main outcome of this final experiment is that eVSM overcomes all the baselines. As already showed for the Movielen dataset, eVSM is able to outperform the other models in a significant way. In this evaluation the gap brushes 2 points % (eVSM vs. SVM, in P@3). This result is confirmed also for P@1
4.1 Movielens and Yahoo! Webscope: content-based movie recommendation

Figure 4.8: Outcomes of Experiment 2 for dimension 500 and 1000 (Yahoo! dataset)
Figure 4.9: Outcomes of Experiment 2 for dimension 1500 and 2000 (Yahoo! dataset)
and P@5, where eVSM always performs better than both Bayes and SVM. The difference is always stable and never drops down 1 point %. All the differences between the algorithm are statistically significant, so it is possible to close this second experiment by confirming the conclusions the first experiment came to: eVSM outperforms state-of-the-art models in the task of providing users with recommendations about movies.

![Yahoo Dataset - Comparison](image)

**Figure 4.10: Outcomes of Experiment 3**

### 4.1.3 Multilingual Recommendations

In this experimental evaluation we would like to test:

- whether user profiles learned in a specific language could be effectively exploited for recommending items in a different one.

- whether the accuracy of a cross-language recommender system is comparable to that of a monolingual one.

- whether a specific approach gets a significant improvement w.r.t. the other ones, becoming preferable in some specific recommendation scenario.

The experiments were carried out in a movie recommendation scenario in which the languages adopted in the evaluation phase were English and Italian. In this experimental setting W-RI and W-SV profiling models have been compared to a multilingual CBRS implemented by combining a Bayesian classifier with a Word Sense Disambiguation algorithm based on MultiWordNet able to
produce a cross-lingual representation for both items and user profiles. This combined approach is thoroughly described in [LMN+10]. A comparison between the Italian and the English versions of the dataset is provided in Table 4.2. The term synset refers to the representation adopted in the CBRS that exploits WSD.

Specifically, four different settings have been compared.

- **Exp#1 – ENG-ITA**: profiles learned on movies with English description and recommendations provided on movies with Italian description;
- **Exp#2 – ITA-ENG**: profiles learned on movies with Italian description and recommendations produced on movies with English description.
- **Exp#3 – ENG-ENG**: profiles learned on movies with English description and recommendations produced on movies with English description;
- **Exp#4 – ITA-ITA**: profiles learned on movies with Italian description and recommendations produced on movies with Italian description.

The experiment was executed for each user in the dataset. The ratings of each specific user and the content of the rated movies have been used for learning the user profile and measuring its predictive accuracy, using the aforementioned measures. Each experiment consisted of:

1. selecting ratings of the user and the description (English or Italian) of the movies rated by that user;
2. splitting the selected data into a training set $T_r$ and a test set $T_s$;
3. using $T_r$ for learning the corresponding user profile by exploiting the:
   - English movie descriptions (Exp#1) or Italian movie descriptions (Exp#2);
4. evaluating the predictive accuracy of the induced profile on $Ts$, using the aforementioned measures, by exploiting the:

- Italian movie descriptions (Exp#1) or English movie descriptions (Exp#2);

The methodology adopted for obtaining $Tr$ and $Ts$ was the aforementioned 5-fold cross validation.

**Discussion of Results**

Results of the experiments are reported in Table 4.3 and 4.4, averaged over all the users.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>W-SV</th>
<th>W-RI</th>
<th>Bayes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP#1 – ENG-ITA</td>
<td>84,65</td>
<td>84,65</td>
<td>85,61</td>
</tr>
<tr>
<td>EXP#2 – ITA-ENG</td>
<td>84,85</td>
<td>84,63</td>
<td>85,20</td>
</tr>
<tr>
<td>EXP#3 – ENG-ENG</td>
<td>85,23</td>
<td>85,29</td>
<td>85,23</td>
</tr>
<tr>
<td>EXP#4 – ITA-ITA</td>
<td>85,27</td>
<td>84,84</td>
<td>85,71</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Experiment</th>
<th>W-SV</th>
<th>W-RI</th>
<th>Bayes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP#1 – ENG-ITA</td>
<td>84,73</td>
<td>84,43</td>
<td>84,60</td>
</tr>
<tr>
<td>EXP#2 – ITA-ENG</td>
<td>84,77</td>
<td>84,54</td>
<td>84,56</td>
</tr>
<tr>
<td>EXP#3 – ENG-ENG</td>
<td>85,10</td>
<td>84,86</td>
<td>84,89</td>
</tr>
<tr>
<td>EXP#4 – ITA-ITA</td>
<td>85,11</td>
<td>84,86</td>
<td>84,93</td>
</tr>
</tbody>
</table>

By summing up, in this experimental session two deeply different approaches have been compared. The first one, based on a classical Bayes classifier, exploits external linguistic knowledge and relies on the assumption that a representation based on MultiWordNet synsets can be an effective bridge to represent user preferences expressed in different languages. The second one, based on eVSM, does not require any linguistic pre-processing and is totally based on distributional hypothesis. It assumes that the similar distribution of the terms, even in different languages, makes the preferences independent from the language and a simple projection of the user profile built in one language into the space built in another one is sufficient to provide the user with cross-language recommendations. In general, the main outcome of the experimental session is that the strategy implemented for providing cross-language recommendations is quite
Applications of eVSM

There is no significant difference by comparing the accuracy of the models previously presented. More specifically, user profiles learned using examples in a specific language can be effectively exploited for recommending items in a different language, and the accuracy of the approach is comparable to those in which both learning and recommendation phase are performed on the same language. Specifically, the approach based on the Bayesian classifier gained the best results in the Precision@5. This means that this model has a greater ability to rank the best items at the top of the recommendations list. Furthermore, it is worth to note that the comparison of the results of the Exp#1 with the results of the Exp#3 shows that the cross-lingua recommendations based on the profiles learned in English improve the Precision with respect to the monolingual one. This is due to the better accuracy of the WSD process for English content, since the disambiguation process can introduce some noise when the synset-based representation is built.

The best result was obtained for all of the approaches by running Exp#3 and Exp#4, that is a classical mono-language recommender system. It is worth to note that the result of Exp#4 related to movies whose description is in Italian is quite satisfactory. The result of the second experiment, in which Italian movie descriptions are used for learning profiles that are then exploited for recommending English movies is also satisfactory.

However, the model based on the Bayesian classifier and MultiWordNet might seem too elaborate, because of the several operations needed to represent documents through MultiWordNet synsets. On the other side, the absence of a linguistic pre-processing is one of the strongest point of the approaches based on the distributional model and the results gained by the W-SV and W-RI models in the Precision@10 further underlined the effectiveness of these techniques. Indeed, in all of the experiments, the Precision of the W-SV model is higher with respect to the W-RI model and the Bayesian one. The first result confirms the outcomes of the monolingual session which show the importance of modeling negative user preferences and the goodness of the negation operator based on Quantum logic.

In conclusion, both models gained good results. Even if in most of the experiments the cross-lingua recommendation approaches got worse results with respect to the mono-lingual ones, the differences in the predictive accuracy are not statistically significant. In general the Bayesian approach fits better scenarios where the number of items to be represented is not too high, and this can justify the application of the pre-processing steps required for building a representation based on MultiWordNet synsets, while the distributional mod-
els, thanks to their simplicity and effectiveness, fit better in scenarios where real-time recommendations that ensure a good accuracy need to be provided.

4.2 Play.me: personalized playlists generator

Play.me is a tool that exploits social media sources for generating personalized music playlists. The platform is based on the inceptive idea that the information extracted from social networks such as Facebook and Last.fm is very meaningful and can be exploited for personalization tasks. The information about user preferences in music, for example, can be easily gathered from these social networks (e.g. by analyzing posted links or attended events) and it can be used to model a profile of interests.

The use of social media is also a very cheap and effective way to overcome the classical problem of cold start in recommender systems. Furthermore, these social media-based playlists are enriched with new artists related to those the user already likes. Specifically, two different enrichment techniques are compared: the former relies on the knowledge stored on DBpedia, the structured version of Wikipedia, while the latter is based on similarity calculations between textual descriptions of the artists, performed on a semantic DocSPACE built through Random Indexing and Distributional Models. The final playlist based on favorite artists is then ranked according to a relevance criterion and is finally presented to the user that can listen to the songs and express her feedbacks.

4.2.1 Description of the Platform

The general architecture of Play.me is sketched in Figure 4.11. The playlist generation is performed in five different steps, each one handled by a different component. The process is triggered by the user, who invokes the Playlist Generator module. The set of her favorite singers is built by mapping the preferences gathered from her Facebook profile with a set of artists extracted from Last.fm. Given this preliminary set, the Playlist Enricher adds new artists to those previously extracted from Facebook. Next, for each favorite artist the most popular tracks are extracted and ranked. The final playlist is then built and shown to the target user who can express her feedback on the proposed tracks. More details about the components and the whole generation process are provided in the next sections.
Data Crawling

Play.me needs a set of artists to feed the playlist generation. The Crawler module queries Last.fm through its public APIs in order to build a corpus of artists from which the playlists are built. This process is performed in a batch way, so it is regularly scheduled in order to maintain the data always updated. For each artist in Last.fm, the name, a picture, the title of the most popular tracks, their play count and a set of tags describing that artist are crawled. All the extracted data are finally stored on a local database.

Preference Extraction

A common weakness of personalization systems is that suggestions can be provided only when explicit user preferences are already gathered. It is typical to refer to this problem as cold start. Thanks to Web 2.0 applications, and specifically with the advent of social networks, the need for explicit preferences is not standing anymore because users leave on these platforms a lot of information about what they like. Facebook profiles, for example (see Figure 4.12), contain explicit information about the artists a user likes. These data can be easily used to model user profiles by avoiding the typical training procedures and overcoming the cold start. The goal of the Extractor module is to connect to Facebook profiles, to extract user preferences and to map them with the data gathered from Last.fm in order to build a preliminary set of artists the user likes.
Even if Facebook profiles also contain a lot of implicit data that can be analyzed to infer user preferences in music, such as links, attended events, content of posts and so on, the first version of Play.me only relies on explicit data.

**Playlist Enrichment**

As pointed out by Ricci in [RRSK11], thanks to personalization techniques (such as recommender systems) it is possible to shift the classical information seeking paradigm towards a discovery one. In this scenario, for example, given a target user the system helps to discover new artists she might like. This step is carried out by the *Playlist Enricher* module, whose goal is to add new artists to the set of those the user already likes according to the preferences explicitly expressed in the Facebook profile.

Formally, given a user $u$, we define the set of favorite artists extracted from the Facebook profile as $F_u = \{f_1, f_2, ..., f_n\}$. For each $f_i$, the *Playlist Enricher* builds a set $E_{f_i} = \{e_1, e_2, ..., e_m\}$ that contains the $m$ most similar artists among those stored in the corpus. Finally, the final set of artists produced is defined as:

$$E_u = \bigcup_{i=1}^{n} E_{f_i}, \quad (4.1)$$

In this work two techniques for enriching the playlists with new artists are implemented and evaluated. The first one is based on the exploitation of Linked Data, while the latter uses cosine similarity in vector spaces.

**Enrichment based on Linked Data.**

The first approach for enriching the playlist with new artists relies on the exploitation of DBPedia [KBAL09]. DBpedia\(^3\) is a project whose goal is to represent in a structured form (by means of RDF triples) the information stored in Wikipedia.

\(^3\)http://dbpedia.org
Nowadays DBpedia represents the nucleus of the so-called Web of Data, since the information is made freely available online and it is possible to submit queries and to get complex information thanks to the structured nature of the data and the relationships that rely among them. In DBpedia, each Wikipedia entry is mapped to a DBpedia concept. Each concept is assigned with a unique Uniform Resource Identifier (URI), e.g. http://dbpedia.org/resource/Coldplay, and each concept can be imagined as a node in a graph. Node pairs are connected by means of relations, called properties in DBpedia. As showed in Figure 4.13, thanks to DBpedia it is possible to represent complex information in a structured form. For example, the following triple encodes the fact that Coldplay is a music band whose genre is Alternative Rock.

The approach is based on the assumption that each artist can be mapped to a DBpedia node. The inceptive idea is that the similarity between two artists can be calculated according to the number of properties they share (e.g. two Italian bands playing rock music are probably more similar than an Italian and an English band that play different genres). Consequently, after a brief lookup of properties that could be useful for computing similarity, the properties dbpedia-owl:genre (describing the genre of the artist) and dcterms:subject (that provides information about the musical category of the artist) were adopted.

Operationally, this step was performed through a powerful query language for Linked Data, namely SPARQL (SPARQL Protocol and RDF Query Language), an RDF query language, similar to a query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format. By simply querying a SPARQL endpoint containing information stored in DBpedia the set of artists related to those already liked by a user can be extracted. For example, the set of artists related to Coldplay can be extracted by using the following query:

```
select distinct ?artist where {
};
```
Since DBPedia returns an unordered set, the returned artists are ranked according to their play count in Last.fm. The first $m$ artists returned by the SPARQL endpoint are considered as relevant (and related) and added to the set of the favorite artists.

**Enrichment based on Random Indexing.** As stated in Section 4.2.1, each artist in Last.fm is described through a set of tags as in classical collaborative tagging systems [GH06]. Each tag provides information about the genre played by the artist, such as *post-rock*, or about typical features of her songs, such as *melancholic* (see Figure 4.14). Given a set of tags $T = \{t_1, t_2, \ldots, t_k\}$, each artist is described through a set $A = \{n_{t_1}, n_{t_2}, \ldots, n_{t_k}\}$, where $n_{t_i}$ is the number of times the tag was used to annotate that artist in the Last.fm community. By following the classical Vector Space Model (VSM) [SWY75], each artist can be represented as a point in a $k$-dimensional vector space.

The inceptive idea behind this enrichment approach is that, by comparing tags describing the artists, it is easy to understand how similar/related they are. Besides, it is possible to assume that artists related to those the user already likes can be considered as relevant for her, as well.

Thus, given a VSM-based representation, Random Indexing was exploited to build a semantic DocSpace and *cosine similarity* was used to calculate how much two artists are similar or related.

In this scenario, for each artist in $F_u$, the cosine similarity between the artist and all the other ones stored in the database was calculated. The $m$ artists with the highest cosine similarity are added by the **Playlist Enricher** module to the list of favourite ones.

**Playlist Generation**

The **Playlist Generation** module is the core of Play.me. Given a user $u$, its goal is to build a playlist $P_u$ that is supposed to be of interest for her.
Given a user $u$ and a set of artists $A_u = F_u \cup E_u$ as the union of the favorite artists with those added by the Playlist Enricher module, it is possible to define a set $S_a = \{s_1, s_2...s_z\}$ that contains the songs played by an artist $a \in A_u$. So, the set of the candidate tracks for user $u$ can be defined as follows:

$$P_u = \bigcup_{a \in A_u} S_a$$ (4.2)

**Playlist Ranking**

The tracks in $P_u$ need to be ranked, so it is necessary to introduce a scoring function to sort the tracks in a descending relevance order. For each track $s_i$ in $P_u$ the score is calculated as follows:

$$score(s_i) = play(s_i) \times source(s_i)$$ (4.3)

where $play(s_i)$ is the normalized play count of the song $s_i$ in the Last.fm community, while $source(s_i)$ is a weight assigned to the source of the song, that is set to $\alpha$ for tracks played by an artist $a \in F_u$ (in the Facebook profile), and $1 - \alpha$ for tracks played by an artist $a \in E_u$ (produced by the enrichment process).

Thus, the scoring function puts at the top of the recommendation list the most popular songs, but it takes into account also the source the songs come from. If the user wants a playlist based only on her favorite artists, without any enrichment, $\alpha$ is set to 1. On the other side, $\alpha$ set to 0 will exclude the songs coming from artists already known. In general, this parameter allows the user to obtain playlists based on new artists rather than ones containing only already liked artists.

Finally, the playlist generated by the system is returned to the user, that can express her feedback on the proposed tracks. User feedbacks are used to re-rank the tracks. A positive feedback on a track will increase its score $score(s_i)$ by 20%, while a negative feedback will decrease it in the same way.

### 4.2.2 Play.me@work: a typical use case

A working implementation of Play.me is available online\(^4\). After a quick registration, the system requires the access to the Facebook profile of the user in order to extract her preferences in music. As in Figure 4.12, it is possible to assume that the user likes *Coldplay* and *U2*, so the Extractor module will add them in a list of favorite artists. The Playlist Enricher module, according to the

\(^4\)http://193.204.187.223:8080/sssc
enrichment strategy adopted, will enrich that set. By exploiting the VSM and the cosine similarity, it will extract *Radiohead* and *Travis* as artists similar to *Coldplay* and *MGMT* and *The Strokes* as artists similar to *U2*. Next, given this set of six singers, the *Playlist Generator* will query Last.fm to get their most popular songs. Finally, the songs will be ranked setting 0.7 as weight for Facebook artists and 0.3 for those extracted by the *Playlist Enricher* module. The final playlist is returned to the user, that can listen to the tracks and express a feedback.

A screenshot of the user interface is provided in Figure 4.15. In the higher part of the interface there is a slideshow of the songs that pops out when the mouse is moved over there. The user can move among the tracks in the playlist and can start listening to one by simply clicking on it. In the lower part there is the title of the song, the name of the artist and it is possible to express a binary feedback.

### 4.2.3 Experimental Evaluation

The goal of the experimental evaluation is to identify the enrichment technique able to generate the most relevant playlists. An experiment involving 30 users
under 35, heterogeneously distributed by sex, education and musical knowledge (according to the availability sampling strategy) was carried out. The final crawl of Last.fm was performed at the end of November, 2011, and data about 228,878 artists were extracted. For each artist were got also the top-5 tracks, so the final Play.me dataset contained information about more than 1 million tracks.

Each user explicitly granted the access to her Facebook profile to extract data about favorite artists. At the end of the Extraction step 325 artists were extracted, so each user had about 10 favorite artists on average. For the enrichment technique based on DBpedia the SPARQL endpoint located at http://dbpedia.org was queried. It contained 3.64 million nodes with more than 100,000 related to musical artists or albums. For the cosine similarity-based technique the Last.fm APIs were used to extract the most popular tags associated to each artist. The less significant and meaningful tags (such as seenlive, cool, and so on) were considered as noisy and filtered out.

In order to identify the best enrichment technique, users were asked to use the application for three weeks. In the first two weeks the system was set with a different enrichment technique, while in the last one a simple baseline based on the most popular artists on Last.fm was used. Playlists were enriched with tracks played by the most popular artists regardless user preferences. For each round of the experimental evaluation the users granted the access to their Facebook preferences. Given the playlist generated by the system, users were asked to explicitly express their feedback on the proposed tracks. For each track it was possible to express a binary feedback or to simply ignore the suggestion (See Figure 4.15). Users were not aware of the enrichment technique adopted in that round. Regarding the parameters, the value of $\alpha$ was set to 0.5, so playlists equally contained tracks played by favorite artists and by new inferred ones. The parameter $m$ indicates the number of similar artists extracted for each favorite one, and user behavior was compared to three different configurations, with $m$ set to 1, 2 and 3. This means that each artist extracted from Facebook was enriched with respectively 1, 2 or 3 new artists. To get the final results, the precision of the system was calculated as the ratio between the number of positive feedbacks and the total number of suggested tracks. The overall results are reported in the Table 4.5.

It is worth to notice that the two enrichment strategies are able to outperform the baseline. Regardless the enrichment technique and the number of the suggested artists, the number of positive feedbacks always overcomes the baseline based on the popularity-based criterion. This means that data extracted
from social networks actually reflect user preferences and the intuition of modeling users according to the information gathered from social sources is valid. The enrichment technique that gained the best performance is that based on the cosine similarity. It overcomes the DBpedia-based strategy of roughly 10 points % with $m=1$ and $m=2$ and 6 points % with $m=3$, and the baseline between 11 and 18 points %. These results were someway expected since this technique is based on solid algebraic calculations. It is known that through this measure it is possible to effectively reflects user preferences by calculating the similarity between two items represented in a vector space.

Even though this technique gained the best results, a deeper analysis can provide different outcomes. Indeed, with $m=3$ the gap between DBpedia and cosine similarity decreases of almost 5 points. This means that with higher values of $m$ a pure content-based representation introduces much more noise than DBpedia, whose effectiveness remains constant (only 2 points % of difference among the different configurations). In other terms, with low similarity values, it is likely that the cosine-based enrichment suggests artists that are not actually interesting for the target user. The introduction of some thresholding strategy might be useful to avoid this problem. The good results obtained by the baseline based on popularity can be justified by the low diversity of the users involved in the evaluation. Since most of them like very common artists such as U2, Coldplay and so on, a simple popularity-based approach is able to provide accurate results. It is likely that by involving users with a more specific musical knowledge and uncommon preferences the results obtained by the baseline might get worse.

Despite its results, the DBpedia-based enrichment technique might represent a valuable alternative to avoid the typical drawbacks of pure content-based representations. Indeed, it might be helpful for providing explanations about the produced recommendations by analyzing the relations among the favorite and suggested artists. Moreover, that enrichment strategy is inherently multilingual, differently from those based on the Vector Space Model and similarity based on string matching.
Finally, DBpedia might also help to face the other typical problem of content-based filtering approaches, such as the *overspecialization*. Indeed, suggested artists might be relevant but too similar to those the user likes. This means that suggestions might be accurate but obvious, thus not useful. Hence, a future direction is to further investigate on the use of DBPedia and Linked Data in order to exploit the wealth of data and relations to obtain more serendipitous (unexpected) results. This allows to focus on some other important aspects besides accuracy for obtaining valuable music recommendations, as devised in a very recent work [SHS12].

### 4.3 Myusic: content-based music recommendations

Myusic is a platform for content-based music recommendation. The application has been developed as case study to evaluate the effectiveness of eVSM in a real world scenario. The underlying idea behind the design of Myusic is that the exploitation of data extracted from social network can be a useful source to feed a content-based recommendation algorithm. In this case the information coming from Facebook profiles has been extracted in order to model user preferences in music according to the links posted by a target user and on the Facebook pages she likes.

Given this Facebook-based profile, Myusic can suggest relevant artist by adopting the aforementioned profiling and recommendation techniques implemented in the eVSM framework. In this section a general overview of the architecture of Myusic is provided. Next, a typical use case of the platform is described and finally the results of a user study involving 50 persons that evaluated the tool in terms of accuracy, novelty and diversity of the produced recommendations are given.

#### 4.3.1 Description of the Platform

The general architecture of Myusic is sketched in Figure 4.16. A different component handles each of the steps that are performed to generate the recommendations. First, the platform needs a set of artists to recommend, so the *Crawler* queries Last.fm in order to gather a set of artists to feed the recommender system. Next, a preliminary user profile is built by mapping the preferences gathered from her Facebook profile with those stored in the local artists set through the *Extractor* module.
Given a set of favorite artists returned by the Extractor, the Profiler comes into play. Its goal is to build a profile of interests according to the profiling models implemented in the eVSM framework (that is to say, RI, W-RI, QN or W-QN). User profile is built upon user preferences extracted from Facebook as well as ratings and feedbacks she provides by interacting with the platform. Finally, the Recommender exploits the information stored in the user profile in order to calculate the similarity between the vector representing the user profile and that representing all the available artists, as explained when the recommendation step implemented in eVSM has been described.

The final recommendation list is built by ranking the artists according to their descending cosine similarity with respect to the user profile.

**Data Crawling**

First, Myusic needs a set of artists to feed the playlist generation. The goal of the Crawler module is to query Last.fm in order to get a list of artists from which the recommendation list is built. As stated in the previous section regarding the Play.me application, this process is performed in a batch way. For each artist in Last.fm, the name, a picture, the title of the most popular tracks, their play count and a set of tags that describe that artist are crawled. All the crawled data are finally stored on a local database.
Applications of eVSM

109

Figure 4.17: User Preferences from a Facebook profile

Preference Extraction

The design of the platform is based on the inceptive idea that the information extracted from social media is an useful source to draw to get data about user preferences and to overcome the typical cold start problem. Facebook profiles, for example (see Figure 4.17), contain explicit information about what artists a user likes.

Similarly, by analyzing the links posted by the target user it is possible to infer implicit preferences about her tastes (see Figure 4.17). So, the Extractor module connects to Facebook, extracts user preferences, and maps them with the data gathered from Last.fm in order to build a preliminary set of artists the user likes. These information are locally stored and modeled in her own profile in order to let her receive recommendations also during the first interaction with the system, thus avoiding the cold-start.
As shown in Figure 4.19, the user performs the login through the Facebook Connect\(^5\) platform in order to start the extraction of data from her Facebook profile. Next, the user interacts with a feedback page where she can split the extracted artists between those she actually likes and those she does not. This is done to avoid the noise coming from data that are not real (for example, a link about a song the user hates and she ironically posted).

**Building User Profiles**

Given a set of favorite artists, the Profiler has to build a profile of interest of the target user. The process is performed in two steps. First, a weight is given to each artist in the list.

The weight of a specific artist is defined according to a simple heuristic: the fact the a user posted a link about a specific artist can be considered as a *light* evidence of her preference, while the fact the she explicitly clicked "like" on her Facebook page can be considered as a *strong* one. Consequently, a weight \(t\) is assigned to the artists whose name appear in the list of the links posted by the user while a weight \(t + 1\) is assigned to those occurring in the list of

\(^5\)https://developers.facebook.com/
Figure 4.19: Extraction of social data for building Myusic profiles
Facebook pages the user likes. Whether an artist occurs in both lists (that is to say, the user likes it and posted a song, as well) a weight \( t + 2 \) is assigned. For example, on a 5-points discrete scale a score 3 is assigned to the artists coming from posted links, 4 is assigned to those coming from Facebook pages and 5 is assigned whether they appear in both. An example of user profile is provided in Figure 4.20.

Next, according to the profiling models introduced in Section 3.3.3 the vector representing the profile of the target user can be instantiated. For example, by adopting the W-RI profiling model the user profile is built through the following formula:
Applications of eVSM

Figure 4.21: User profile represented as tag cloud in Myusic platform

\[ \vec{p}_u = \sum_{i=1}^{\vert I_u \vert} \vec{d}_i * r(u, a_i) \] (4.4)

where \( I_u \) is the set of her favorite artists, \( r(u, a_i) \) is the weight assigned to the artist \( a_i \) and \( \vec{a}_i \) is the vector space representation of the artist. Since each artist is described through a set of tags \( t_1 \ldots t_n \) extracted from Last.fm, the vector space representation is a weighted vector \( \vec{a}_i = (w_{t_1}, \ldots, w_{t_n}) \) where \( w_{t_i} \) is the weight of the tag \( t_i \) (for example its popularity in the Last.fm platform).

Formally, a user profile in Myusic can be represented as the vector that merges the (weighted) vector space representations of all the artists she likes. As shown in Figure 4.21, Myusic profiles can be simply represented by means of the classical tags clouds. The larger the tag, the bigger the user interest in that specific music genre.

Recommendations Generation

Finally, given a vector space representation of both artists and user profile, through the similarity measures implemented in the eVSM framework it is possible to produce as output a ranked list of suggested artists.

This is done by calculating the cosine similarity for all the possible couples \((\vec{p}_u, \vec{a})\) where \( \vec{p}_u \) is the vector space representation of user \( u \) while \( \vec{a} \) is the vector describing the artist \( a \). In Figure 4.22 an example of recommendation list is provided. The target user can click on each artist to check the details, listen to the most popular songs (gathered from YouTube) and express a feedback on her (see Figure 4.23) in order to start again the recommendation process.

Obviously, the extraction of preferences from Facebook gives only information about what artists a user likes, but nothing is inferred about what she does.
4.3 Myusic: content-based music recommendations

Figure 4.22: An example of recommendation list in Myusic platform
not like. Since the information about negative user preferences is as important as those coming from positive feedbacks, the platform let users express feedbacks on the recommendations. As shown in the previous picture, a user can label a recommendation with a negative feedback. This information is stored in her negative user profile and exploited whether the QN or W-QN settings are adopted as profiling technique.

4.3.2 Experimental Evaluation

The goal of the experimental evaluation was to validate the outcomes of the offline evaluation (discussed in Section 4.1) on eVSM framework in an online setting and in a different domain as well. Specifically, a user study involving 50 users under 30, heterogeneously distributed by sex, education and musical knowledge (according to the availability sampling strategy) was performed. They interacted for two months.

A crawl of Last.fm was performed at the end of November, 2011 and data about 228,878 artists were extracted. Each user explicitly granted the access to her Facebook profile to extract data about favourite artists. At the end of the Extraction step, a set of 980 different artists the 50 users like were extracted from Facebook pages. Generally speaking, 1,720 feedbacks were collected. 1,495 of them came from Facebook profiles, while 225 were explicitly provided by the users (for example, expressing a feedback on their recommendations). The collected feedbacks were highly unbalanced since only 116 (6.71%) on 1,720 were
Last.fm APIs were exploited to extract the most popular tags associated to each artist. The less expressive and meaningful ones (such as *seen live*, *cool*, and so on) were considered as noisy and filtered out.

The design of the user study was oriented to answer to the following questions:

- **Experiment 1**: Does the cold-start problem can be mitigated by modeling user profiles which integrate information coming from social media?
- **Experiment 2**: Does the users actually perceive the utility of adopting weighting schemes and negation when user profiles are represented?
- **Experiment 3**: How does the platform perform in terms of novelty, serendipity and diversity of the proposed recommendations?

**Experiment 1**

The goal of the first experiment was to state whether the use of social media is an useful strategy to tackle the cold-start problem and provide users with good recommendations even during their first interaction with the system. In order to validate this hypothesis users were asked to login and to extract their data from their own Facebook page. Next, a user profile was built according to a profiling model randomly chosen and a preliminary set of recommendations was proposed to the target user.

In order to evaluate the effectiveness of the Extractor we compared the recommendation list generated through eVSM to a baseline represented by a list produced by simply ranking the most popular artists. Next, we asked users to tell which list they preferred between the one based on the most popular artists and that based on their user profiles built on the ground of the data extracted from Facebook. Obviously, they were not aware about which list was the baseline and which one was built through eVSM. A plot that summarizes users’ answers is provided in Figure 4.24.

From the charts it is straightforward to note that users actually prefer social media-based recommendations, since 74% of them preferred that strategy with respect to a simple heuristic based on popularity of the artists stored in database. However, even if the results gained by this profiling technique were outstanding, it is necessary to understand why 26% of the users simply preferred the most popular artists. Probably, there is a correlation between users’ knowledge in music and the list they choose. It is likely that users with very generic tastes prefer a list of popular singers. Similarly, it is likely that users with a poor
knowledge in music might prefer a list of well-known singers with respect to a list where most of the artists, even if related to their tastes, were unknown. A larger evaluation with users, split according to their musical knowledge, may be helpful to understand the dynamics behind users' choices.

**Experiment 2**

One of the major outcomes of the experiments discussed in Section 5.4 was that both the Weighting Scheme and the Negation Operator were able to improve the Precision of eVSM in an offline evaluation. Consequently, the goal of the second experiment was to confirm that outcome in an online setting as well. Users were asked to validate this hypothesis by choosing the configuration that better reflected their preferences between one adopting the weighting schema and one that did not, and between a configuration that exploited negation with another one where the user profile was built only according to positive feedbacks. This experiment was performed in two steps.

In the first part users were asked, as in Experiment 1, to login and to extract their data from their own Facebook page. Next, two user profiles were built by following respectively the RI and the W-RI profiling models. Finally, recommendations were generated from both profiles and users were asked to choose the configuration they preferred. As well as in Experiment 1, they were not aware about which one were the recommendations generated through weighted profile and those produced through its unweighted counterpart. Results of this experiments are provided in Figure 4.25.

Unlike what was expected, users did not perceive as useful the introduction of a weighting scheme designed to give greater significance to the artists the user likes the most. On the contrary, the RI profiling model was the preferred
one for 70% of the users involved in the experiment. Similarly, in the second part of the experiment the RI profiling model was compared to the QN one, in order to evaluate the impact on user perception about the importance modeling negative preferences, or, in general, negative feedbacks. Results are shown in Figure 4.26.

Also in this case the results were conflicting with respect to the initial hypothesis, since 65% of the users preferred the recommendations generated through the profiling technique that does not model negative user preferences.

Even if the results of Experiment 2 did not confirmed the outcomes of the offline evaluation of eVSM they are actually interesting. First, they confirmed the usefulness of combining offline experiments with user studies thanks to the different outcomes they can provide. Indeed, in user-centered applications such as content-based recommender systems, user perception and user feedbacks play
a central role and these factors need to be taken into account. In general, further investigation is needed because most of these results may be due to a specific bias of the designed experiment. Regarding the weighting scheme, for example, the offline experiments already showed that for most of the comparisons the gap among the different configurations was not statistically significant, thus the online evaluation just emphasized the absence of a real usefulness.

Furthermore, different experimental settings may bring out the usefulness of the negation operator. As stated above, the extraction of data from Facebook pages crawls information about what a specific user likes, so very few negative feedback were collected (less than 7%). Consequently, the negative part of the user profile was very poor and this might justify the results. “It is likely that collecting more negative feedbacks would be enough to confirm the usefulness of negative information.”

Experiment 3

Finally, in Experiment 3 users were asked to express their preference on the recommendations produced through the RI profiling model (since it emerged as the best one from the previous experiment) in terms of novelty, accuracy and diversity. The results of this experiment are sketched in Figure 4.27.

In general, the results are encouraging since most of the users expressed a positive opinion about the system. Specifically, Myusic has a good impact on final users in terms of trust, since the opinion of 92% of the users ranges from Good to Very Good. This is probably due to the good accuracy of the recommendations produced by the system. More than 80% of the users considered as accurate or very accurate the suggestions of the system. Similarly, the results in terms of diversity were positive as well since more than 60% labeled the level of diversity among the recommendations as Very Good. In this session Myusic has demonstrated to be also a useful tool to discover novel artists: indeed, only 14% of the users did not discover any artist during the evaluation session. The remaining 86% discovered at least one artist, with a peak of 32% for those that discovered one or two new singers. The only aspect that needs improvements regards the novelty of recommendations since 34% of the users labeled as not novel the suggestions produced by the system. This value was someway expected since it is a typical issue of CBRS and obviously eVSM inherits it.

However, even if these results encourage carrying on this research, they have to be considered as preliminary since this part of the evaluation needs to be extended by comparing user opinions about eVSM with that about other state of the art models such as LSI, VSM or collaborative models as showed in Section
Figure 4.27: Novelty, Accuracy and Serendipity in the recommendations produced by Myusic
5.4 for the offline experiment.
4.3 Myusic: content-based music recommendations
This chapter summarizes the outcomes of the internship in the Human Interaction & Experiences (HI&E) group of the Philips Research Center in Eindhoven, the Netherlands.

During the internship period the eVSM has been evaluated in the context of the research carried out by APRICO Solutions, a software company and part of Philips Electronics. It develops video recommenders and it specifically focuses on the personalization of TV guides.

The electronic program guide (EPG) data used in this research are provided by Axel Springer, a strategic partner of APRICO Solutions. Powered by APRICO Solutions’ recommendation technology, Axel Springer has developed a personal TV guide called Watchmi, currently available in three forms: online, as plug-in for Microsoft Windows Media Center and embedded in the Eviado One HD-Receiver for satellite and cable TV. Watchmi seamlessly integrates TV and Internet content, learning from the user interaction and recommending shows and videos that match the user’s preferences. Figure 5.1 shows a screenshot of the Watchmi plug-in for the Microsoft Windows Media Center.

---

1 www.aprico.tv
2 www.axelspringer.de
3 www.watchmi.tv
4 www.watchmi.tv/en/watchmi-search
5.1 Scenario

One of the concepts developed at APRICO Solutions is the concept of personal channels. A user can create a personal channel by selecting a TV show or an Internet video asset as seed. Based on the seed attributes, similar TV shows and Internet videos are automatically selected and aggregated into a playlist that can be viewed as a linear channel next to the traditional broadcast TV channels. The basic architecture of a personal channel is shown in Figure 5.2. Each channel has a boolean filter that preselects TV shows and Internet videos based on the characteristics of the video seed used to create the channel. The shows that pass the filter are prioritized by a recommender that learns from the interaction of the user with the channel and through explicit ratings. Note that in this concept, users are not explicitly modeled, but their multiple interests and preferences are captured by the multiple personal channels, each having a dedicated recommender.

Each TV show is represented by a number of features such as the title of the show, a short description of it and so on. In addition, each TV show is labeled with a program type, which is selected from a fixed set of 17 program types including 'sport', 'movie', and so on. The requirements that arise from that scenario (a continuous flow of information with new shows proposed every day and items described by means of textual content) perfectly fit with the features of content-based filtering techniques or text classification algorithms. Further-
more, to justify the necessity of a scalable solution, it is possible to observe that for a set of approximately 50 broadcast channels in the German language, there are approximately 2,000 TV shows broadcast each day. Additionally, there can be considerably more video-on-demand assets accessible whenever desired. Furthermore, YouTube is reported to grow at a rate of 48 hours of video assets per minute and during 2010 more than 13 million hours of video have been uploaded.

In this scenario the proposed framework has been evaluated in two separated tasks:

- **Classification task.** Given a set of TV shows the algorithm has to label each one with the program type it belongs to. So, given a user profile represented in terms of program types, the system can automatically build her personal channel by selecting the TV shows labeled with the program types she likes.

- **Retrieval task.** Given a set of TV shows, the user expresses her interest for a specific program type and the algorithm has to retrieve the TV shows that are supposed to be related to it.

In both tasks eVSM behavior has been evaluated with respect to different techniques for user profiling (with Quantum Negation and without it) and different parameters, such as the size $k$ of the context vectors.
5.2 Data representation

As stated above, the electronic program guide (EPG) data used in this evaluation are provided by Axel Springer. Specifically, a set of XML EPG feeds got from 47 German-language broadcast channels has been used. Each XML file contained a set of metadata useful for the representation of the items, such as the title of the show, its synopsis and the set of the persons that appear in it. In Figure 5.3 a short fragment of an XML file is shown. For each TV show an id of both the show and the channel (attributes bid and chid), a title (node tit) a short and a long synopsis (nodes lonsyn and shosyn) and a set of data about the persons that appear in the show (node mprsn) are given. Furthermore, the XML file also contains some non-textual metadata (such as the start and end time of the show, the duration or some data about the broadcaster). Finally, for each TV-show a unique program type is given as well (node mprgt).

All the 133,579 TV shows that compose the dataset have been broadcast by German-language channels from April, 2009 to April, 2011. In Table 5.2 the complete mapping between program type ids and program type names is provided.

5.2.1 Bag-of-Words-based representation

First, each TV show has been represented through the classical BOWs (bag-of-words), that is to say, by simply listing the words that appear in the text and counting the occurrences for each one. Given an item \(i\) and \(n\) words occurring in its description \(f_1...f_n\), the BOW representation of the item \(i\) is given by:

\[
bow_i = \{(f_1, w_1)\ldots(f_n, w_n)\}
\] (5.1)
Figure 5.3: A fragment of EPGs XML file
Where \( w_k \) represents the weight for the word \( f_k \). As already discussed, it could be represented as the number of the occurrences of \( f \) in the text, its normalized value or through more complex weighting schemes.

Next, the set of TV-shows has been processed by filtering out the textual content with a set of 996 stopwords for German language. In Figure 5.4 an example of the BOW-based representation is provided. From the original XML file only the information inside the nodes \( tit \) (title of the show) and \( losys \) (long synopsis) is extracted. Then the set of the keywords (in German) obtained from the XML has been filtered out from the stopwords and finally, according to the specific weighting technique to be used, to each word has been assigned a weight.

**Figure 5.4: An example of BOW-based representation**

### 5.2.2 Tag.me-based representation

During the internship the classical BOW representation has been compared to a novel form of representation that exploits open knowledge sources such as Wikipedia. *Tag.me*\(^5\) is an online tool developed by the University of Pisa (Italy) whose goal is to produce a Wikipedia-based representation of text fragments. Given a text, *Tag.me* returns a set of Wikipedia concepts (pages) that occur in the text, assigning to each one a relevance score: the higher the relevance score, the higher the likelihood that the concept occurring in the text actually refers to the Wikipedia concept. The service is available (only for content in English and Italian) via web interface (Figure 5.5) and through a REST Web Service.

In this experiment *Tag.me* has been used to produce TV-Show representation based on Wikipedia concepts instead of simple features. Specifically, the output of the BOW-based representation has been translated in English\(^6\) through the Google Translate APIs and then used to invoke the Tag.me Web Service. Next, the XML file produced by *Tag.me* (Figure 5.6) has been processed by filtering out all the Wikipedia concepts with a relevance score (\( \rho \) node) under 0.1. The

---

\(^5\)http://tagme.di.unipi.it/

\(^6\)As underlined above, Tag.me can be used only with texts in Italian or English
relevance score has been set through a rough heuristic, without experimental evidences.

In Figure 5.7 the final output of the Tag.me-based processing is shown. Thanks to this representation each TV-show is described by means of a set of high-level concepts, making the representation less noisy and more meaningful.

5.3 Description of the tasks

As stated above, the eVSM framework has been evaluated in both classification and retrieval tasks. In this section is described how RI, QN and VSM have been tailored to handle them.

5.3.1 VSM for TV-Shows Classification

The VSM has been adopted as baseline to evaluate the effectiveness of eVSM. In this setting a vector space has been built and each TV show (document) has been represented as a vector of weighted features as follows:

1. For each TV show a set of words is collected, by using the synopsis and
Figure 5.6: A fragment of the Tag.me XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<text xml:lang="en">
<text>The Alps are spectacular scenery, wilderness, recreation areas for tourists, adventure playground for extreme sports and experimentation for researchers.
</text>
<annotations>
<annotation>
<spot pos="41" len="4">alps</spot>
<title>alps</title>
<id>881</id>
<rno>0,30999</rno>
</annotation>
<annotation>
<spot pos="101" len="14">extreme sports</spot>
<title>extreme sports</title>
<id>9933</id>
<rno>0,19777</rno>
</annotation>
</annotations>
</text>
```

Figure 5.7: Tag.me-based representation

alps extremesport playground wilderness
recreation tourism knowledge experiment research
the title of the show.

2. The set of the words is filtered out through a fixed set of 996 stopwords for German language.

3. Given a document, the weight of each word is calculated through the TF/IDF score.

Next, given the vector space representation of each TV show, the vector space representation of each program type was obtained as the sum of the vector space representations of the TV shows that belong to that program type. Formally, given a set of TV shows $s_1...s_n$, a set of program types $t_1...t_m$ and a function $pt(s_i)$ that returns the program type for the show $s_i$, the set $S(t_i)$ can be defined as the set of the TV-shows that belong to the program type $t_i$.

$$S(t_i) = \{ s_k \ni \forall k, pt(s_k) = t_i \}$$  \hspace{1cm} (5.2)

Given the set $S(t_i)$, with a cardinality of $k$, the vector space representation of the program type $t_i$ is simply defined as:

$$\vec{t_i} = \sum_{j=1}^{k} s_j$$ \hspace{1cm} (5.3)

Where each $s_k \in S(t_i)$. Finally, given the vector space representation of both TV shows and program types, the cosine similarity was exploited as similarity measure. Given a TV show to be classified the program types were ranked according to their descending cosine similarity and the unclassified TV show was assigned to the program type whose cosine similarity got the highest value.

### 5.3.2 eVSM for TV-Shows Classification

Given a vector space built by following the approach described in 5.3.1, Random Indexing is applied to reduce the dimension of the vector space. The algorithm was evaluated by comparing its behavior with respect to three different parameters:

1. Dimension of the context vectors.

2. Minimum number of occurrences of the features.

3. Number of training cycles.
The first parameter refers to the length of the vectors represented in the vector space. Since in the complete DocSpace each vector has a dimension of almost 306k, by setting the dimension to 1000 the size of the vector space is reduced of approximately 99.5%. Through the second parameter the data were filtered out from all the terms whose number of occurrences does not overcome a fixed threshold, while the number of training cycles regards the opportunity to iterate the process described in Section 3.3.1 by exploiting as input context vectors the vectors obtained at the end of the first training cycle. All these aspects will be discussed thoroughly in the design of the experimental evaluation.

So, given the reduced vector space representation of the TV shows, the representation of the program type is obtained by following the same steps described in the previous section. First, a (reduced) program type vector is built upon the (reduced) TV shows vectors that belong to that category then cosine similarity was exploited as similarity measure to rank the program types. Finally the unclassified TV show was assigned to the program type with the highest cosine similarity.

5.3.3 eVSM for TV-Shows Retrieval

Regarding the retrieval task a vector space is built by following the approach described in 5.3.1, as well. Next, the random indexing algorithm is applied with the parameters introduced in Section 5.3.2. The only difference between the classification task and the retrieval one is that, given the TV shows and the vector space representation of the program types, here the input is represented by the program type instead of the TV show.

So, for each program type, the cosine similarity is used to get the set of the n TV shows that are supposed to belong to that category. In the experimental evaluation the behavior of the algorithm is tested with different values of n. This aspect will be discussed in the next session.

For the retrieval task the random indexing algorithm has been extended by means of the quantum negation (QN) operator previously introduced, in order to model also the negative evidences about program types. Specifically, by following the definitions provided in section 5.3.1, given the vector \( \vec{t}_i \) it is possible to build a vector \( \neg \vec{t}_i \) in this way:

\[
\neg \vec{t}_i = \sum_{j=1}^{z} s_j
\]

(5.4)
Where \( s_k \notin S(t_i) \). In other terms, \( \neg \vec{t_i} \) represents the vector that models the information coming from the TV shows that do not belong to \( t_i \). The combined use of \( \vec{t_i} \) and \( \neg \vec{t_i} \) let retrieve the TV shows whose description contains as much as possible features coming from \( \vec{t_i} \) and as little as possible features coming from \( \neg \vec{t_i} \).

### 5.4 Experimental Evaluation

The goal of the experimental evaluation was to measure the effectiveness of both eVSM and VSM in term of predictive accuracy and goodness of the proposed ranking in both retrieval and classification tasks. The experimental session has been carried out on a dataset composed by 133,579 TV shows broadcast from a set of 47 channels in German language. Table 5.4 summarizes some statistics about the dataset.

The complete dataset contains 306,006 features, with 42.11 features per TV show (averaged over all the dataset). The Tag.me representation has proved to be a very strong form of feature selection since with this representation the number of the features is reduced by 75.6%, from approximately 306k to approximately 75k. The average number of features per item decreased from 42.11 to 9.21, with a reduction of 79.8%. In Table 5.4 are shown the same data, split by program type.

In Figure 5.8 are shown the data, by normalizing the values for the average number of features. From the chart emerges that the use of the Tag.me-based representation has a considerable impact on the number of the features that describe both TV-Shows and program type, but it does not affect the ratio among program types since the ones with longer descriptions remain unchanged also with the new representation, excluding minor exceptions.

#### 5.4.1 Experimental Design

Two different experimental sessions have been designed. In each one we tried to give an answer to a different question by comparing the predictive accuracy of the models in both classification and retrieval tasks.
### Table 5.3: Average length of the descriptions, divided per program type

<table>
<thead>
<tr>
<th>Program type</th>
<th>BOW</th>
<th>Tag.me</th>
<th>Program type</th>
<th>BOW</th>
<th>Tag.me</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miscellaneous</td>
<td>52.74</td>
<td>12.92</td>
<td>Report</td>
<td>52.70</td>
<td>12.47</td>
</tr>
<tr>
<td>Movies</td>
<td>37.26</td>
<td>8.61</td>
<td>Magazine</td>
<td>48.30</td>
<td>11.26</td>
</tr>
<tr>
<td>Short Movies</td>
<td>38.98</td>
<td>8.28</td>
<td>News</td>
<td>36.51</td>
<td>9.35</td>
</tr>
<tr>
<td>TV Series</td>
<td>35.85</td>
<td>6.49</td>
<td>Weather</td>
<td>57.33</td>
<td>8.33</td>
</tr>
<tr>
<td>Sport</td>
<td>47.05</td>
<td>10.98</td>
<td>Videoclip</td>
<td>20.13</td>
<td>4.00</td>
</tr>
<tr>
<td>Show</td>
<td>46.77</td>
<td>9.67</td>
<td>Preview</td>
<td>27.00</td>
<td>12.00</td>
</tr>
<tr>
<td>Documentary</td>
<td>43.68</td>
<td>10.59</td>
<td>Advertising</td>
<td>7.52</td>
<td>4.21</td>
</tr>
<tr>
<td>Events</td>
<td>51.02</td>
<td>13.38</td>
<td>Music</td>
<td>46.24</td>
<td>11.96</td>
</tr>
<tr>
<td>Reportage</td>
<td>50.30</td>
<td>11.62</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 5.8: Length of the descriptions, averaged for all the program types
1. **Session 1**: which one is the best representation technique between the classical *BOW* and the novel *Tag.me-based* representation?

2. **Session 2**: which one is the model that provide the best predictive accuracy?

The experimental evaluation has been carried out through the classical *k-fold cross validation*, with $k=10$. The complete dataset has been split into 10 partitions and for each run 9 partitions were used for training the models and the last one for testing them. Figure 5.9 shows that the number of training examples for each program type is equally distributed among all partitions. This ensures a better stability of the results that emerged from the experimental session.

![Figure 5.9: Distribution of the training examples among the 10 partitions](image)

In Figure 5.10 the distribution of the TV shows among all the categories is provided. It follows that the dataset is very unbalanced towards some program types such as TV Series (id 4), Movies (id 2) and Documentary (id 8) in comparison to other program types with a very little number of instances, such as Weather (id 13).

For the *classification* task the Accuracy ($Ac$) was used to evaluate the effectiveness of the proposed models. It can be calculated as

$$ Ac = \frac{\text{correctlyClassified}}{\text{totalClassified}} \quad (5.5) $$

For the *retrieval* task the Precision@n ($Pr@n$) and the Precision@k% ($Pr@k\%$) were used, with $n = \{5, 10, 50, 100, 200, 500\}$ and $k = \{5, 10, 25, 50, 70, 100\}$. They can be calculated as

$$ Pr@n = \frac{TP}{\min(n, \#Ts)} \quad (5.6) $$
5.4 Experimental Evaluation

Figure 5.10: Distribution of the training examples among the 17 program types

and

\[
Pr@k% = \frac{TP}{\#Ts \times k%},
\]

(5.7)

where \#Ts is the cardinality of the test set for a specific program type. The metrics were calculated separately for each program type and for each of the 10 runs of the evaluation. Finally, the partial values were averaged in order to get the final results.

Random Indexing algorithm was run by comparing its behavior with different values for the parameters introduced in Section 5.3.2. Specifically, after a session of parameter tuning the following values were used:

1. Dimension of the vectors: 500 and 700 for the classification task, 500, 1000, 1500 and 2000 for the retrieval one
2. Minimum occurrences: 2 for the classification task, 1 and 3 for the retrieval one
3. Training cycles: 1, 2 for the classification task, 1 for the retrieval one

5.4.2 Discussion of the results

In this section the outcomes of the experimental evaluation are discussed. It is split into two subsections, one for each experimental session that has been carried out. In each subsection the experimental data are plotted and an interpretation of the results is given. The results have been also validated by means of Friedman’s and Wilcoxon’s statistical tests.
Table 5.4: Comparison between BOW and Tag.me with random indexing on text classification

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Occurrences</td>
</tr>
<tr>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td>700</td>
<td>2</td>
</tr>
<tr>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td>700</td>
<td>2</td>
</tr>
</tbody>
</table>

Session 1: data representation. In order to define the best way to represent the data in the TV shows recommendation scenario, the behavior of RI was compared to the approaches previously introduced for data representation.

Classification task. In Table 5.4 are summarized the results for the classification task of the random indexing algorithm with both BOW and Tag.me-based representations. The same data are plotted in Figure 5.11 as well.

It clearly emerges that the BOW representation gains the best results with all the combination of the parameters. The difference between BOW and Tag.me ranges between 5 and 7 points % with one training cycle up to almost 10 points % with two training cycles. A Wilcoxon test confirmed the inceptive outcome since the gap between the different configurations is statistically significant.

Retrieval task. In Figure 5.12 and 5.13 are plotted the results of the
RI and RIQ algorithms in the retrieval task with data representation based on both Tag.me and BOW. For the sake of simplicity, the results are plotted only for the parameter dimension set to 1000 and 2000. Even for this task the classical representation based on BOW gained the best results in term of $P@n$. While the plot in Figure 5.13 shows a very regular trend, with a gap ranging from almost 25% for $P@5$ and $P@10$ to almost 10% for the $P@100$ and $P@200$, always in favor of the BOW-based representation, from Figure 5.12 it could emerge a slightly different interpretation: here the BOW representation got the best results for the $P@5$, $P@10$, $P@50$ but with the RI algorithm the trend decreases faster and the Tag.me-based representation overcomes the BOW representation for $P@200$ and $P@500$. This could be due to the different nature of these representation techniques. Tag.me provides a less noisy representation so there is a lower likelihood the tail of the list of retrieved TV-show contains some show that does not belong a specific program type. On the other side the BOW representation provide most significant results because with these techniques the features that are more relevant emerged and this leads to a better ability in ranking the TV shows.

**DISCUSSION:** The final outcome of this first session is that the representation based on BOW is still more effective in comparison to the one based on Tag.me. This one, even if it provides a more meaningful representation based on high-level Wikipedia concepts, has proved to be not reliable for these kind of tasks. Indeed, the reduction in terms of features (almost 78%, from 42 to 9, on average) is too strong and a so small set of features makes difficult to learn a solid model for classification and retrieval. Although the results are not satisfactory, it is desirable to repeat the experiments again with different thresholds for relevance value (the $\rho$ node of the XML returned by Tag.me). With a lower threshold value the Tag.me algorithm could return more Wikipedia concepts and this could help to increase the predictive accuracy of this representation technique.

**Session 2: predictive accuracy.** The goal of the second experimental session was to define which model was capable to provide the best predictive accuracy in this scenario. The VSM has been used as baseline for the experiment and the results have been compared to random indexing (and quantum negation) in both classification and retrieval tasks. For all the experiments shown in this session only the BOW representation was adopted since it gained the best results in the first session.

**Classification task.** In Table 5.5 the results of the VSM in the classification task are provided. The predictive accuracy was satisfactory for most of the program types since 7 out of 17 got results over 70%, although the precision for
Figure 5.12: Comparison between BOW and Tag.me with random indexing on text retrieval
Figure 5.13: Comparison between BOW and Tag.me with random indexing on text retrieval
Table 5.5: Results of the VSM in the classification task

<table>
<thead>
<tr>
<th>Program type</th>
<th>VSM</th>
<th>Program Type</th>
<th>VSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miscellaneous</td>
<td>11.2%</td>
<td>Report</td>
<td>15.1%</td>
</tr>
<tr>
<td>Movies</td>
<td>76.4%</td>
<td>Magazine</td>
<td>63.7%</td>
</tr>
<tr>
<td>Short Movies</td>
<td>35.0%</td>
<td>News</td>
<td>54.1%</td>
</tr>
<tr>
<td>TV Series</td>
<td>74.3%</td>
<td>Weather</td>
<td>100.0%</td>
</tr>
<tr>
<td>Sport</td>
<td>89.6%</td>
<td>Videoclip</td>
<td>78.9%</td>
</tr>
<tr>
<td>Show</td>
<td>65.3%</td>
<td>Preview</td>
<td>0.0%</td>
</tr>
<tr>
<td>Documentary</td>
<td>62.7%</td>
<td>Advertising</td>
<td>93.6%</td>
</tr>
<tr>
<td>Events</td>
<td>63.1%</td>
<td>Music</td>
<td>81.4%</td>
</tr>
<tr>
<td>Reportage</td>
<td>57.0%</td>
<td>(Average)</td>
<td>68.7%</td>
</tr>
</tbody>
</table>

some program types (Miscellaneous, id 1 and Magazine, id 11) was really low. The average value gained by VSM in the experimental evaluation is 68.7%.

By merging the average results gained by the VSM with the data already shown in Table 5.4, we can provide a first comparison between VSM and RI. The data are plotted in Figure 5.14.

Figure 5.14: Comparison between VSM and RI on text classification

From the plot it follows that the application of the dimensionality reduction technique produces significantly worse results since the VSM gets results almost 15 points % better in comparison to RI. These results were someway expected since the loss of information with a compression of 99.5% in term of dimensions of the vector space (from 306k to 500) is too strong and it obviously produces a worsening in the predictive accuracy. Furthermore, it emerged that the precision of the model is directly related to the dimension of the vector space since the
results with \textit{size=700} are better in comparison to the results with \textit{size=500} and to the application of the second training cycle, that increases the precision of between 7 and 12 points \%. These outcomes were confirmed also by a Wilcoxon test \((p<0.01)\), except for the comparison between the configurations \textit{700-2-1} and \textit{700-2-2}.

Some interesting outcome also came from the results of the RI algorithm on the classification task by splitting the results per program type. By analyzing the results put in Table 5.6 it follows that even if the overall accuracy is lower, the RI algorithm got results at least comparable in comparison to those of the VSM for most of the classes. It is possible to state that RI provides most accurate classification for 10 out of 17 classes in comparison to the baseline, especially for classes with a smaller number of instances in the training set. This is an interesting outcome since it proves that even a so compact representation of the model could be able to give results that are almost good as VSM ones. Furthermore, a Friedman test showed that the difference between the different configuration and the baseline is not significant in terms of macro-average, so it is possible to conclude that RI can perform as well as VSM in the classification task.

Retrieval task. Tables 5.7 and 5.8 summarize the results obtained by RI and RI+QN algorithms on the text retrieval task with the P@n. It seems that for this task the parameters do not affect the retrieval accuracy of the model since for all the metrics the differences between each run of the algorithm are below 0.2\%. In general, by increasing the dimension of the vector space the predictive accuracy slightly grows up. However, the gained accuracy does not justify the bigger amount of resources needed to generate a larger vector space. On the other side, the main outcome of this experimental session is that the use of the Quantum Negation is able to effectively manage the information coming from negative evidence since the RI+QN algorithm gained an improvement in P@n of almost 20 points \% in all the metrics. In Figure 5.15 are plotted the results of this comparison between RI and RI+QN and it shows the big difference in predictive accuracy between them. A Wilcoxon test \((p<0.01)\) confirmed that the application of the negation operator can produce significant improvements in the precision.

In Tables 5.9 and 5.10 we put the results of the same algorithms for the P@k\%. Even for this evaluation similar outcomes were got since it does not emerge a significant difference between the different settings of the parameters for both algorithm, while the difference between RI and RI+QN also with this metric ranges between 16 and 20 points \%. 
Table 5.6: Comparison between BOW and random indexing on text classification, results divided per program type

<table>
<thead>
<tr>
<th>Program Type</th>
<th>500-2-1</th>
<th>500-2-2</th>
<th>700-2-1</th>
<th>700-2-2</th>
<th>VSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.37</td>
<td>0.44</td>
<td>0.35</td>
<td>0.40</td>
<td>0.11</td>
</tr>
<tr>
<td>2</td>
<td>0.35</td>
<td>0.49</td>
<td>0.40</td>
<td>0.51</td>
<td>0.76</td>
</tr>
<tr>
<td>3</td>
<td><strong>0.95</strong></td>
<td><strong>0.88</strong></td>
<td><strong>0.95</strong></td>
<td><strong>0.89</strong></td>
<td>0.35</td>
</tr>
<tr>
<td>4</td>
<td>0.47</td>
<td>0.72</td>
<td>0.58</td>
<td>0.72</td>
<td>0.74</td>
</tr>
<tr>
<td>5</td>
<td><strong>0.90</strong></td>
<td>0.87</td>
<td><strong>0.91</strong></td>
<td>0.87</td>
<td>0.90</td>
</tr>
<tr>
<td>6</td>
<td>0.48</td>
<td>0.48</td>
<td>0.48</td>
<td>0.52</td>
<td>0.65</td>
</tr>
<tr>
<td>7</td>
<td><strong>0.72</strong></td>
<td><strong>0.65</strong></td>
<td><strong>0.74</strong></td>
<td><strong>0.66</strong></td>
<td>0.63</td>
</tr>
<tr>
<td>8</td>
<td>0.23</td>
<td>0.25</td>
<td>0.24</td>
<td>0.23</td>
<td>0.63</td>
</tr>
<tr>
<td>9</td>
<td>0.41</td>
<td>0.33</td>
<td>0.43</td>
<td>0.33</td>
<td>0.57</td>
</tr>
<tr>
<td>10</td>
<td><strong>0.32</strong></td>
<td><strong>0.28</strong></td>
<td><strong>0.30</strong></td>
<td><strong>0.33</strong></td>
<td>0.15</td>
</tr>
<tr>
<td>11</td>
<td>0.39</td>
<td>0.37</td>
<td>0.36</td>
<td>0.36</td>
<td>0.64</td>
</tr>
<tr>
<td>12</td>
<td><strong>0.84</strong></td>
<td><strong>0.79</strong></td>
<td><strong>0.83</strong></td>
<td><strong>0.78</strong></td>
<td>0.54</td>
</tr>
<tr>
<td>13</td>
<td><strong>1.00</strong></td>
<td><strong>1.00</strong></td>
<td><strong>1.00</strong></td>
<td><strong>1.00</strong></td>
<td>0.74</td>
</tr>
<tr>
<td>14</td>
<td><strong>0.94</strong></td>
<td>0.74</td>
<td><strong>0.92</strong></td>
<td>0.74</td>
<td>0.79</td>
</tr>
<tr>
<td>15</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>16</td>
<td><strong>0.99</strong></td>
<td><strong>0.94</strong></td>
<td><strong>0.99</strong></td>
<td><strong>0.94</strong></td>
<td>0.94</td>
</tr>
<tr>
<td>17</td>
<td><strong>0.83</strong></td>
<td>0.79</td>
<td><strong>0.81</strong></td>
<td>0.77</td>
<td>0.81</td>
</tr>
<tr>
<td>macro-avg</td>
<td>0.636</td>
<td>0.625</td>
<td><strong>0.642</strong></td>
<td>0.627</td>
<td>0.638</td>
</tr>
</tbody>
</table>

Table 5.7: Results of the RI algorithm on text retrieval (P@n)

<table>
<thead>
<tr>
<th>Approach</th>
<th>Dim.</th>
<th>Occ.</th>
<th>P@5</th>
<th>P@10</th>
<th>P@50</th>
<th>P@100</th>
<th>P@200</th>
<th>P@500</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI</td>
<td>500</td>
<td>3</td>
<td>0.638</td>
<td>0.586</td>
<td>0.447</td>
<td>0.423</td>
<td>0.399</td>
<td>0.362</td>
</tr>
<tr>
<td>RI</td>
<td>500</td>
<td>1</td>
<td>0.641</td>
<td>0.593</td>
<td>0.444</td>
<td>0.408</td>
<td>0.379</td>
<td>0.346</td>
</tr>
<tr>
<td>RI</td>
<td>1000</td>
<td>3</td>
<td>0.639</td>
<td><strong>0.596</strong></td>
<td>0.450</td>
<td>0.424</td>
<td>0.400</td>
<td><strong>0.364</strong></td>
</tr>
<tr>
<td>RI</td>
<td>1000</td>
<td>1</td>
<td>0.659</td>
<td><strong>0.596</strong></td>
<td>0.447</td>
<td>0.413</td>
<td>0.386</td>
<td>0.350</td>
</tr>
<tr>
<td>RI</td>
<td>1500</td>
<td>3</td>
<td>0.639</td>
<td>0.595</td>
<td><strong>0.452</strong></td>
<td><strong>0.426</strong></td>
<td><strong>0.401</strong></td>
<td>0.363</td>
</tr>
<tr>
<td>RI</td>
<td>1500</td>
<td>1</td>
<td><strong>0.663</strong></td>
<td>0.594</td>
<td>0.440</td>
<td>0.410</td>
<td>0.381</td>
<td>0.345</td>
</tr>
<tr>
<td>RI</td>
<td>2000</td>
<td>3</td>
<td>0.638</td>
<td>0.594</td>
<td>0.446</td>
<td>0.422</td>
<td>0.395</td>
<td>0.358</td>
</tr>
<tr>
<td>RI</td>
<td>2000</td>
<td>1</td>
<td>0.654</td>
<td>0.594</td>
<td>0.443</td>
<td>0.408</td>
<td>0.381</td>
<td>0.344</td>
</tr>
</tbody>
</table>
5.4 Experimental Evaluation

![Figure 5.15: Comparison between RI and RI+QN on text retrieval (P@n)](image)

### Table 5.8: Results of the RI+QN algorithm on text retrieval (P@n)

<table>
<thead>
<tr>
<th>Approach</th>
<th>Dim.</th>
<th>Occ.</th>
<th>P@5</th>
<th>P@10</th>
<th>P@50</th>
<th>P@100</th>
<th>P@200</th>
<th>P@500</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI+QN</td>
<td>500</td>
<td>3</td>
<td>0.812</td>
<td>0.755</td>
<td>0.647</td>
<td>0.616</td>
<td>0.592</td>
<td>0.559</td>
</tr>
<tr>
<td>RI+QN</td>
<td>500</td>
<td>1</td>
<td>0.816</td>
<td>0.762</td>
<td>0.640</td>
<td>0.613</td>
<td>0.584</td>
<td>0.554</td>
</tr>
<tr>
<td>RI+QN</td>
<td>1000</td>
<td>3</td>
<td>0.829</td>
<td>0.760</td>
<td>0.654</td>
<td>0.627</td>
<td>0.606</td>
<td>0.576</td>
</tr>
<tr>
<td>RI+QN</td>
<td>1000</td>
<td>1</td>
<td>0.824</td>
<td>0.773</td>
<td>0.650</td>
<td>0.623</td>
<td>0.601</td>
<td>0.570</td>
</tr>
<tr>
<td>RI+QN</td>
<td>1500</td>
<td>3</td>
<td>0.824</td>
<td>0.766</td>
<td>0.659</td>
<td>0.635</td>
<td>0.611</td>
<td>0.581</td>
</tr>
<tr>
<td>RI+QN</td>
<td>1500</td>
<td>1</td>
<td>0.822</td>
<td>0.767</td>
<td>0.653</td>
<td>0.629</td>
<td>0.604</td>
<td>0.573</td>
</tr>
<tr>
<td>RI+QN</td>
<td>2000</td>
<td>3</td>
<td>0.826</td>
<td>0.763</td>
<td>0.663</td>
<td>0.637</td>
<td>0.614</td>
<td>0.583</td>
</tr>
<tr>
<td>RI+QN</td>
<td>2000</td>
<td>1</td>
<td>0.826</td>
<td>0.769</td>
<td>0.654</td>
<td>0.629</td>
<td>0.607</td>
<td>0.576</td>
</tr>
</tbody>
</table>

### Table 5.9: Results of the RI+QN algorithm on text retrieval (P@k\%) 

<table>
<thead>
<tr>
<th>Approach</th>
<th>Dim.</th>
<th>Occ.</th>
<th>P@5%</th>
<th>P@10%</th>
<th>P@25%</th>
<th>P@50%</th>
<th>P@75%</th>
<th>P@100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI</td>
<td>500</td>
<td>3</td>
<td>0.564</td>
<td>0.525</td>
<td>0.459</td>
<td>0.408</td>
<td>0.378</td>
<td>0.355</td>
</tr>
<tr>
<td>RI</td>
<td>500</td>
<td>1</td>
<td>0.575</td>
<td>0.523</td>
<td>0.462</td>
<td>0.412</td>
<td>0.383</td>
<td>0.355</td>
</tr>
<tr>
<td>RI</td>
<td>1000</td>
<td>3</td>
<td>0.563</td>
<td>0.516</td>
<td>0.460</td>
<td>0.409</td>
<td>0.381</td>
<td>0.355</td>
</tr>
<tr>
<td>RI</td>
<td>1000</td>
<td>1</td>
<td>0.581</td>
<td>0.531</td>
<td>0.466</td>
<td>0.415</td>
<td>0.384</td>
<td>0.356</td>
</tr>
<tr>
<td>RI</td>
<td>1500</td>
<td>3</td>
<td>0.567</td>
<td>0.521</td>
<td>0.458</td>
<td>0.407</td>
<td>0.381</td>
<td>0.355</td>
</tr>
<tr>
<td>RI</td>
<td>1500</td>
<td>1</td>
<td>0.574</td>
<td>0.526</td>
<td>0.465</td>
<td>0.412</td>
<td>0.385</td>
<td>0.354</td>
</tr>
<tr>
<td>RI</td>
<td>2000</td>
<td>3</td>
<td>0.553</td>
<td>0.514</td>
<td>0.457</td>
<td>0.406</td>
<td>0.381</td>
<td>0.352</td>
</tr>
<tr>
<td>RI</td>
<td>2000</td>
<td>1</td>
<td>0.573</td>
<td>0.526</td>
<td>0.460</td>
<td>0.410</td>
<td>0.384</td>
<td>0.353</td>
</tr>
</tbody>
</table>
Figure 5.16: Comparison between RI and RI+QN on text retrieval (P@k%)

Table 5.10: Results of the RI+QN algorithm on text retrieval (P@k%)

<table>
<thead>
<tr>
<th>Approach</th>
<th>Dim.</th>
<th>Occ.</th>
<th>P@5%</th>
<th>P@10%</th>
<th>P@25%</th>
<th>P@50%</th>
<th>P@75%</th>
<th>P@100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI+QN</td>
<td>500</td>
<td>3</td>
<td>0.837</td>
<td>0.790</td>
<td>0.713</td>
<td>0.638</td>
<td>0.585</td>
<td>0.534</td>
</tr>
<tr>
<td>RI+QN</td>
<td>500</td>
<td>1</td>
<td>0.842</td>
<td>0.791</td>
<td>0.709</td>
<td>0.632</td>
<td>0.578</td>
<td>0.528</td>
</tr>
<tr>
<td>RI+QN</td>
<td>1000</td>
<td>3</td>
<td>0.844</td>
<td>0.803</td>
<td>0.725</td>
<td>0.651</td>
<td>0.597</td>
<td>0.548</td>
</tr>
<tr>
<td>RI+QN</td>
<td>1000</td>
<td>1</td>
<td>0.850</td>
<td>0.802</td>
<td>0.722</td>
<td>0.648</td>
<td>0.591</td>
<td>0.543</td>
</tr>
<tr>
<td>RI+QN</td>
<td>1500</td>
<td>3</td>
<td>0.855</td>
<td>0.810</td>
<td>0.734</td>
<td>0.657</td>
<td>0.603</td>
<td>0.552</td>
</tr>
<tr>
<td>RI+QN</td>
<td>1500</td>
<td>1</td>
<td>0.855</td>
<td>0.806</td>
<td>0.732</td>
<td>0.653</td>
<td>0.599</td>
<td>0.548</td>
</tr>
<tr>
<td>RI+QN</td>
<td>2000</td>
<td>3</td>
<td>0.860</td>
<td>0.808</td>
<td>0.734</td>
<td>0.657</td>
<td>0.603</td>
<td>0.554</td>
</tr>
<tr>
<td>RI+QN</td>
<td>2000</td>
<td>1</td>
<td>0.851</td>
<td>0.810</td>
<td>0.732</td>
<td>0.656</td>
<td>0.600</td>
<td>0.551</td>
</tr>
</tbody>
</table>
DISCUSSION: The final outcome of this second session is that Random Indexing provided good results for the classification task, with an overall Precision that is lower with respect to the baseline but with a good improvement in most of the single program types and a comparable accuracy in terms of macro-average. In general, it is possible to state that RI can perform as well as VSM in the text classification task. On the other side, for the retrieval one, it emerges that the Quantum Negation is able to significantly improve the goodness of the ranking in comparison to the simple Random Indexing algorithm. The QN operator seems to be the only parameter able to influence the results since neither the dimension of the vector space or the minimum number of occurrences of the terms lead to some difference in the P@n and P@k% for both algorithms.
Chapter 6

Summary and Conclusions

This chapter represents the ideal closure of the research path undertook in this thesis. It gives a brief recap about the main topics the thesis focuses on and summarizes the major contributions provided by this dissertation. Certainly, the investigation cannot be considered as complete since many open points are still standing, so the last section sketches some possible research directions worth to be investigated in the future.

6.1 Recap

In this thesis we analyzed how to exploit the convergence and the methodological overlap standing between IR and IF to introduce a novel recommendation framework based on IR techniques.

According to Belkin and Croft’s insight, this research investigated how the classical Vector Space Model (VSM) could be adapted to the requirements and the dynamics of a recommendation scenario. Specifically, by analyzing VSM three main weaknesses emerged, that is to say, the inability to represent semantics, the limits of a model for representing user preferences only based on positive feedbacks and, generally speaking, scalability issues. In order to effectively tackle these drawbacks, a deep state-of-the-art analysis was performed: distributional models and random indexing emerged as the most promising techniques able to provide VSM with a representation that conciliates both semantics and scalability requirements. Differently from LSI, for example, these models approximate the original vector space through an incremental approach defined on the ground of the random projection theory. Next, in order to model negative user preferences as well, a negation operator based on quantum mechanics was introduced. It was proposed by Widdows, exploits well-known
6.2 Contributions

How does this dissertation contribute to advance the state of the art in the area of CBRS? Which contributions does it provide?

Generally speaking, the contributions of this thesis can be summarized by following the research lines sketched in Section 1, when the research motivations have been introduced.

- **Introduction of techniques for content representation based on distributional models.** These techniques have been introduced to over-
come the limits of the classical keyword-based representation models. The main advantages that follow the use of this approaches is that they can enrich text representation with a lightweight semantics based on term co-occurrences in the same contexts, without the need of a complete natural language processing (NLP) pipeline. Specifically, random indexing emerged as the most promising technique for coupling semantic representation with scalability requirements that arise from the need of algorithms for dimensionality reduction of the vector space. The exploitation as random indexing as main building block for representing both items and user profiles has proved to be a promising and previously uninvestigated research direction.

- **Introduction of techniques for representing negative user feedbacks based on quantum logic.** The problem of catching negative user feedbacks is a central issue for CBRS since for building accurate user profiles negative evidences are as important as positive ones. This dissertation proposed a novel operator based on quantum logic that can effectively model negative preferences. This operator inherits from quantum logic the vision of negation as a form of orthogonality between vectors, so it let define user profiles as points in a vector space where a positive and a negative component are combined. Even if the use of this operator in the area of CBRS has never been investigated, the experimental results showed its effectiveness, especially in scenarios, such as Philips EPG personalization, where a lot of negative evidences are collected.

- **A novel profiling and recommendation framework for monolingual and multilingual environments.** The eVSM framework introduced in this dissertation adopts the VSM as main building block and extends it with the aforementioned random indexing and quantum negation. The framework is based on the assumption that user preferences can be approximated through the concept of proximity in vector spaces, so the combined use of vector space representation with similarity calculations can be exploited to provide users with suggestions in a generic domain (assuming that each item can be described with means of a well-defined set of features). This vision has proved to be effective since the experimental evaluation showed how eVSM can outperform other approaches for CBRS. One of the main advantage of the framework is that the representation based on distributional models is inherently multilanguage, so the model can be simply adapted in order to exploit it for cross-language
recommendations as well.

- **New approaches for tackling cold-start problem based on social media.** Cold start is a typical issue of recommender systems. Even if several methods to cope with this problem have been proposed, the exploitation of social media emerges since it is very effective and straightforward way to overcome it. The inceptive idea is to gather and crawl the information left by users on social networks and to model them in order to automatically trigger the recommendation process without explicit information about user preferences. In this thesis an approach based on the analysis of the information coming from Facebook and Twitter pages has been introduced. Beyond its simplicity, the technique proved to be actually effective to model a preliminary profile of user interests in a content-based music recommendation scenario.

### 6.3 Future Work

Even if the overall results can be considered as positive, many aspect of this dissertation can be certainly improved and several promising future research direction may be sketched:

- **Modeling Contextual Information.** CBRS are user-centered tools whose goal is to catch users preferences and to provide them with the information that best match their needs. However, it’s too simplifying to consider the user as disconnected from the context he is in. Indeed, there are many context-related facets that influence the whole recommendation process: weather conditions influence the choice about a place where spend the weekend in. User mood influences the choice about the music to listen to. Latest news may influence the suggestions about articles or books to read. A movie suggestion for Valentine’s or Christmas Day must take into account whether the person is engaged or has a children. All these aspect can be considered as contextual factors, and a recommender system needs to model and to exploit them in order to provide recommendations much more accurate and valuable. So, a possible future direction of this research regards the integration of contextual data in the eVSM model. Even if it is not trivial to model contextual data in user profiles or to include them into a vector space representation, this direction is worth to be investigated since it could positively affect the effectiveness of the framework.
• **Integration of Linked Data.** In keyword-based representations each term can be modeled in several ways, ranging from simple boolean models to co-occurrences counting, passing through techniques for weighting how important a term is within a certain corpus. However, all these representation share the fact that they are totally *plain*. This means that even in their most complex form, there is no relationship between the terms modeled in a user profile or in a document. The evidence about user interest for a basketball team is a clue about her preference about basketball or sport in general, but it’s not straightforward to model that in keyword-based approaches since each term is represented as a simple feature, without any connection with other entities. Even if some attempt in the direction of ontological user profiling tried to tackle this issue, the recent spread of Linked Open Data boosted the research toward this direction. The exploitation of Linked Data in the area of CBRS may provide several advantages: for example, it could slightly reduce the problem of over-specialization of CBRS and it could provide better explanations about the proposed recommendations. Moreover, Linked Data-based representation are inherently multilanguage, so by introducing algorithms able to map textual features to Linked Open Data (LOD) nodes it is possible to model cross-language user profiles with no costs. Finally, the use of Linked Data could make easier the interoperability of user profiles among different applications. To put eVSM in the LOD cloud is surely one of the possible directions to investigate in the future, in order to make the framework more *standard* in terms of languages and formalisms adopted to represent information.

• **Introduction of Social Media and Open Knowledge Sources for CBRS.** Preference acquisition has always been labeled as the *bottleneck* of the personalization pipeline. Consequently, a lot of research has been addressed toward techniques for modeling user preferences without the need of intrusive training procedures. To harvest social media is a recent trend in the area of CBRS: it can merge the un-intrusiveness of implicit user modeling with the accuracy of explicit techniques since the information left by users is freely provided and usually reflects *real* user preferences. In this dissertation some naïve attempt about the exploitation of social media for personalization has been proposed, but the effort towards this is worth to be much more since it is a really promising direction and many techniques to model social media in user profiles can be proposed. A possible future research line may be related to the exploitation of social
knowledge, such as that contained in Wikipedia. These sources represent the new frontier for those systems, such as CBRS, that need content-based information to feed and trigger the personalization process. For future work, it is desirable to combine open knowledge sources with social media in order to enrich the poor feature-based representation based on social data (such as Facebook pages, where no content except from the title or a brief description is provided) with rich textual content coming from trustful sources such as Wikipedia.
Bibliography


[PB07] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang


[PRP00] Duncan Pemberton, Tom Rodden, and Rob Procter. Groupmark: A WWW recommender system combining collaborative and information filtering. In *Proceedings of the 6th ERCIM Workshop*


[SM95] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth". In Irvin R.


