TECHNIQUES AND ALGORITHMS FOR SOCIAL MEDIA

Platform for searcH of Audiovisual Resources across Online Spaces

PHAROS

Project IST-45035

Deliverable D2.1.1 WP2.1

Deliverable
Version 1.0 – 30 September, 2007
Document ref: *pharos.D211.L3S.WP2.1.V1.0*

Programme Name:	IST
Project Number:	45035
Project Title:	PHAROS
Partners:	Coordinator: ENG (IT) Contractors: France Telecom (FR), Fast Search & Transfer (NO), L3S Research Center (DE), Fraunhofer IDMT (DE), Ecole Polytechnique Federale de Lausanne (CH), Knowledge Media Institute Open University (UK), Fundació Barcelona Media Universitat Pompeu Fabra (ES), Technical Research Centre of Finland VTT (FI), Circom Regional (FR), Metaware (IT), Web Models (IT), SAIL LABS Technology (AT)
Document Number:	D2.1.1
Work-Package:	WP2.1
Deliverable Type:	Report
Contractual Date of Delivery:	30/09/2007
Actual Date of Delivery:	10/10/2007 (projected)
Title of Document:	Techniques and Algorithms for Social Media
Author(s):	L3S (Bhaskar Mehta, Avare Stewart, Chen Ling, Raluca Paiu, Claudiu Firan, Kerstin Bischoff, Sergey Chernov), VTT (Vainikainen Sari), FT (Cecile Bothorel). FBM (Oscar Celma)
Editor:	Bhaskar Mehta (L3S)

Approval of this report:	Executive Board
History:	
Keyword List:	social media, recommender systems, tagging, personalization, blogs
Availability:	This report is limited to PHAROS consortium distribution.

Disclaimer

This document contains confidential information in form of description of the PHAROS project findings, work and products and its use is strictly regulated by the PHAROS Consortium Agreement and by Contract no. FP6-45035. Neither the PHAROS Consortium nor any of its officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein. Without derogating from the generality of the foregoing neither the PHAROS Consortium nor any of its officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage, personal injury or death, caused by or arising from any information, advice or inaccuracy or omission herein. This document has been produced with the assistance of the European Union. The contents of this document are the sole responsibility of PHAROS consortium and can in no way be taken to reflect the views of the European Union.

PHAROS is a project partially funded by the European Union.

Table of contents

1 Introduction **1**
- 1.1 What is Social Media? 1
- 1.2 Social Media in PHAROS 2

2 State of the Art **4**
- 2.1 Collaborative Tagging 4
  - 2.1.1 Motivations for tagging 4
  - 2.1.2 Characteristics of tags - strengths and problems 5
  - 2.1.3 Tagging habits and time sensitivity 6
  - 2.1.4 Clustering tags and building ontologies from Folksonomies 6
  - 2.1.5 Approaches to support tagging - Tag suggestions 7
  - 2.1.6 Enhanced information access via tags / browsing by tags 7
  - 2.1.7 Exploring tags for (multimedia) information retrieval 8
- 2.2 Opinion mining 8
  - 2.2.1 NLP shallow parsing 9
  - 2.2.2 Machine Learning Techniques 11
  - 2.2.3 Combination of NLP and Machine Learning 12
- 2.3 Blog Analysis 13
  - 2.3.1 Overview of Blogs and public forums 13
  - 2.3.2 Building user profiles from Blogs 14
- 2.4 Recommendation Algorithms 16
  - 2.4.1 Collaborative filtering 18
- 2.5 Recommendation and Latent Semantic Analysis 20
- 2.6 Conclusions 22

3 Algorithms For Creating User Profiles **23**
- 3.1 Creating A User Profile By Analyzing User's Tags 23
  - 3.1.1 Analysis Of User Tags 24
  - 3.1.2 Tagging Habits 25
  - 3.1.3 Time Sensitivity 26
  - 3.1.4 Tag Usage Frequency 26
  - 3.1.5 Tag Co-occurrence Usage Frequency 27
  - 3.1.6 Lexical Analysis 29
  - 3.1.7 Algorithm and Evaluation Result 31

3.1.8 Optional features 36
3.1.9 Suggestions for handling tags within Pharos 38
3.1.10 Conclusions and future work 38
3.2 Information Diffusion In Blogosphere 39
3.2.1 Problem Definition 41
3.2.2 Discovery of Information Diffusion Paths 44
3.2.3 Performance Study 49
3.2.4 Related Work 52
3.2.5 Conclusions and Future Work 53
3.3 Opinion analysis in user-created textual contents 53
3.3.1 Review analysis using NLP approach 54
3.3.2 Opinion analysis using machine learning techniques approach 64
3.3.3 Conclusion and perspectives 66

4 Algorithms For Personalization and Recommendation 68
4.1 Using User Generated Metadata For Music Recommendation 68
4.1.1 Related Work 68
4.1.2 Tag-Based Profiles vs. Track-Based Profiles 70
4.1.3 Track-based Profiles 71
4.1.4 Tag-based Profiles 72
4.1.5 Music Recommendations 74
4.1.6 Track-based Recommendations 75
4.1.7 Tag-Based Recommendations 75
4.1.8 Tag-Based Search 76
4.1.9 Evaluation & Results 78
4.2 Personalized Ranking using LSI 80
4.2.1 Singular Value Decomposition (SVD) 80
4.2.2 Latent Semantic Analysis 81
4.2.3 Personalized VLSI 82

5 Conclusions & Future Work 85

Executive Summary

This deliverable titled *Techniques And Algorithms For Social Media* aims at describing the advances made in the first 9 months of the PHAROS project in the area of Social Media. The report has been authored by various partners participating in WP 2.1, and has been edited by L3S Research Center.

The report focuses on the following Social Media: Blogs, Collaborative Tagging systems and Recommender Systems. In addition, techniques for trust in Metadata are also described briefly. Specifically, Chapter 1 gives a brief introduction of this document, and describes some important background information about the role of Social Media in the PHAROS platform. Chapter 2 reports current state-of-the-art algorithms for Social Media. Chapter 3 discusses the details of new algorithms and techniques developed by the PHAROS consortium for building user profiles. Recommendation and Personalization algorithms are discussed in Chapter 4. Finally, conclusions and future work are discussed in Chapter 5.

This document provides a first draft of how to proceed in the implementation of social media modules in the PHAROS platform. Although algorithms have been proposed in this report, more work has to be done to engineer them and to integrate them into the overall platform together with other modules. Following this report, there will be a internal report where architectural details are discussed in more details.

1 Introduction

The PHAROS project aims to build the next generation Audiovisual search engine. This is indeed an idea whose time has come[1]; we now possess enough computational power and storage resources to perform computationally intensive processing of music and video data. This processing can go beyond facetted and metadata search into the more advanced content and context oriented search domains. The vision of PHAROS is to successfully face the challenge of scaling new technology to millions of video.

One important driving force in the growth of the Internet is the participation from end users. Broadband connections been added at an ever growing pace in the recent past; in addition, the consumer has been using the extra bandwidth for entertainment avenues like music, video and gaming. Recently, a new trend has been noticed where end users are not just passive observers but content creators as well; this has been referred to at *Web 2.0*. The phrase *Web 2.0* refers to a perceived second generation of web-based communities and hosted services such as social-networking sites, wikis and folksonomies which aim to facilitate collaboration and sharing between users. The pace at which this new phenomenon is growing has surprised many, and has lead to new business models using readily-available intuitive modular elements [1]. It has been suggest that the addition of user generated content has increased the economic value of the Web, potentially surpassing the impact of the dotcom boom. The main success of Web 2.0 has been ushering in the age of *Social Media* as a low cost and highly inclusive rival to traditional media.

1.1 What is Social Media?

Social media is a term referring to online technologies and practices that people use to share content, opinions, insights, experiences, perspectives, and media themselves. It can take various forms: text, images, video, or metadata. Traditional forms include bulletin boards and online forums; newer forms include blogs, bookmark sharing, social networks, wikis and online voting and reviewing.

There are obvious similarities with traditional media, however there are also a number of characteristics which make social media fundamentally different from previous forms of media. Social media is unbounded, unlike TV or radio broadcasts (time limits) or newspapers and magazines (page bounded). Social Media is often authored by relatively unknown and non professional people; it can be edited by people other than the authors themselves, and often make use of multiple media at the same time. Often, the perception of others to social media is available in real time, as opposed to time bound and restricted feedback in traditional media

[1] *No one can resist an idea whose time has come.* – Victor Hugo

(*e.g.* letters to the editor).

On the flip side, social media can be very specific/narrow in appeal, and potentially unreliable. The quality of social media is usually below traditional media, though public editing has proved very successful in projects like *Wikipedia*. The low startup time and investment leads to unscrupulous *fly-by-night* operators who may potentially be motivated by private/commercial interests. A commercial downside of social media is the degrading value of traditional media and the free usage of proprietary media without appropriate royalties being paid. Privacy protection of readers and publishers is also a concern, especially among parents of young (children) authors.

Inspite of all its drawbacks, the impact of social media is a positive one, and there are clear benefits for service providers: companies and organizations can efficiently get feedback from consumers and provide personalized services by observing their online interaction with social media providers. Various people provide crucial information about themselves openly (email, preferences, opinions etc) which were previously much more difficult to perceive and measure. This information can be used to provide *personalized* service and tailor service offering to trends observed in social media. Researchers have found a high degree of correlation between blog mentions and books sales [2] highlighting that online chatter maybe helpful in predicting the popularity of traditional media.

1.2 Social Media in PHAROS

PHAROS is placed in a good position with respect to the new Web: there is a lot of momentum in users annotating and tagging audiovisual sources. However, there is a gap between the availability of this information and efficient exploitation for the purposes of improved access to desired content. Further, personalization has been known to suffer from bootstrapping problem, where by the experience for a new user can be unsatisfactory. In addition, there are other avenues of user information where user express their strong and personal opinion; this is the world of blogging, which is also a web 2.0 phenomenon. Other public spaces like Social networks and online forums are also rich sources of information about people and their preferences. The vision of PHAROS would be well served buy creating technology to bridge the aforementioned gap.

The aim of this document is to describe the current technologies dealing with Social Media. In addition, we report new algorithms and techniques developed by the PHAROS consortium exploiting Social media for the purposes of search. The goal of the algorithms listed int his document is to *extract knowledge about users from available data on the web.* This data can be collected from public sources like blogs, review sites, collaborative projects like Wikipedia, Del.icio.us etc. In addition, descriptions and metadata of content indexed by the PHAROS search engine can also be extended using similar approaches, and used to increase relevance of results to specific users.

Since the project is in its early stages, some of the work presented is likely to evolve to more

PHAROS specific technology and software components. Here we discuss primarily the scientific and research aspects of new social media technology. We will also release an updated version of this document which Architectural implications of Social Media Technology will be discussed in Detail.

2 State of the Art

Social Media has been a hot topic for research in the last few years and continues to attract attention and signifacnt research funding around the world. In order to create innovative algorithms and techniques, we first investigate the state-of-the-art in the relevant areas of Social Media. We look at the following in details: Collaborative Tagging, Opinion and Sentiment analysis, Blog Analysis and Recommendation Algorithms.

2.1 Collaborative Tagging

Within the Web 2.0 context and its increased user participation, collaborative tagging has become very popular. In various platforms, users assign freely selectable words – keywords or category labels – to shared content e.g. bookmarks (*del.icio.us, connotea.org*), photos (*Flickr!*), videos (*YouTube*) or songs (*Last.fm*) to describe, organize and easily share these resources. Similarly, bloggers may tag their blog posts or even allow their readers to do so. A review on the most important tagging systems is given in [3]. Some scientific work has been done to research tagging motivations, patterns as well as to support the tagging process and to explore how to exploit tagging to improve information retrieval in general. Research is also underway on how eLearning [4] and Knowledge and Expertise Management in Organizations [5] can benefit from such user generated metadata - or according to [6] *just more content* provided for the resources.

2.1.1 Motivations for tagging

A systematic analysis of tagging systems shows that motivations for tagging are quite manifold and so are the kind of tags used. Marlow et al. [7] identify organizational motivations e.g. for tags that identify what (or who) something is about (*biology*) or what it is (*book*), for tags refining existing categories, or for tags serving self-reference (*myStuff*) or task organization (*toRead*). Other reasons are opinion expression, the attraction of attention, and self-presentation [7, 8]. The predominant motivation seems to vary depending on the kind of application used: tagging rights, tagging support, aggregation model, object type, source of material, resource connectivity and social connectivity probably influence why certain tags are (not) used [7]. Thus, according to [9], in systems with free-for-all opinion expression, self-presentation, activism and performance tags become more and more popular/ frequent while in self-tagging systems like del.icio.us, users tag almost exclusively for their own benefit of en-

hanced information organization [8]. For del.icio.us tags, the following summary can be made out of the identified tag categories: topic, type of referenced resource, proper name (person, company, product, event, and location), subjective tags (adjectives, ratings), self reference, toDo tags and time. In [10], the following tag categories were found for Flickr! tags (describing photos): place, time, event, name, action and camera.

2.1.2 Characteristics of tags - strengths and problems

Tags are an alternative means for information organization revealing as well certain information about user interests and personality. While more traditional approaches rely on controlled vocabularies, hierarchical and exclusive taxonomies, thesauri or even ontologies – all usually compiled *top-down* by experts to assure metadata quality –, *folksonomies* (folk + taxonomy) are usually flat, uncontrolled and dynamic. Via different tags, they allow for multiple categorization of an information item and thus provide multiple entry points and facilitate serendipitous encounters. Similarly, some (prototype) desktop applications [11, 12] aim at complementing or even replacing the strict folder structure on the PC by tags as organizational metaphor. However, tags do not allow for structured (location based) information access via subclasses / folders and they do not represent relationships between tags explicitly [13, 8]. Acknowledging this organizational problem of scaling a flat system, bundles were recently introduced in del.ico.us to allow grouping of tags [14].

The success of tags and tagging is much based on the opportunity to use them any way that the user finds useful and to choose any words he or she wants. The *bottom-up* or grass root approach of tagging moreover faces some well-known problems: people use different words to describe the same thing, or a word has several different meanings (polysemy). People may also describe things at various levels of detail - an expert in a subject will use more detailed and specific words, whereas others use more general words. Also different forms of the same word (singulars, plurals, typos) [8] are to be found. Challenging as well in analyzing and utilizing tags is describing things and concepts that would need two or more words. Here different applications set different restrictions and give different opportunities. In del.icio.us a tag must be one word, and if a user wants to describe a concept needing more than one word, he needs to engage in a *workaround*. In some applications the creation of tags consisting of several words is made easier. In searching or browsing tags, these problems lead to incomplete result sets or to many undesirable results.

Some researchers have investigated how much this unstructured or possibly very idiosyncratic[1] labeling dominates the structure of tagging systems. Since sharing data relies on at least some common ground, it seems that stable patterns evolve after some time. Golder & Huberman [8] found that most pages in del.cio.us reach their peak (in terms of being added as a bookmark and tagged) after 10 days. Moreover, a tags's frequency remains nearly fixed after it has been used for about 100 bookmarks, This means that relative tag proportions become stable. In [15] and [16] folksonomies stabilized into a power law distribution - given

[1] A tag is used often by only one user

sufficient active users [15]; these distributions are typical of complex systems (like human language) and indicate that the use of a shared vocabulary converges eventually, i.e. *semantics emerge*. Golder & Huberman [8] give two possible explanations: imitation of other user's behavior and shared knowledge. However, a *long-tail* of idiosyncratic and low-frequency tags remains. According to [15], these may only be used personally to *tweak* results and could be ignored for general applications exploiting folksonomies.

Analyzing the network properties of folksonomies (del.icio.us, bibsonomy) [17] showed that the characteristic path lengths between users, resources and tags (modelled in a tripartite hypergraph) are relatively short and the network is highly connected. This *small world* property of folksonomies enables *serendipitous encounters* of interesting users and contents. Again, a stable pattern for tag-co-occurrences was found - a *fat tailed distribution* as a typical indicator of the underlying complex dynamics of human interaction. Deviations from this curve may probably be attributed to spamming activity.

2.1.3 Tagging habits and time sensitivity

In addition to differences in vocabularies, there are also differences between people in how they tag. Some are very diligent and use many tags, some only few. Also, a single user's tagging habits evolve during time as they become more used to tagging and understand which ways of tagging serve their personal interests best. Users have different habits to use tags concurrently. The general habit is to tag from general tags to more detailed tags [8]; some users use a lot of synonyms or different spelling variants (plural, singular, acronyms) of a word or concept. As already mentioned, users also have different ways of combining words. There are tags in which each word is separated by a certain character, by a hyphen, by an underscore, by a slash or by a space, two separate tags can always be used together to indicate specific meaning.

2.1.4 Clustering tags and building ontologies from Folksonomies

Since top-down-constructed ontologies are often very static/slowly-evolving, capturing the emergent semantics of folksonomies in terms of inducing ontologies from tagging systems seems promising. Mika [18], for example, exploits co-occurrence and properties of tags to induce clusters and hierarchical orderings – super-concepts and sub-concepts – of tags. By building an *actor-concept* graph from users and the tags that they assigned, different sub communities can be modeled so that concepts appear in the context of the community that they belong to. The author shows how to expand this idea for web pages in general. Similarly by using a subsumption-based model, [14] induces faceted ontologies from Flickr! data, i.e. *non-exclusive* ontologies that allow for multiple category membership.

2.1.5 Approaches to support tagging - Tag suggestions

In bottom-up tagging systems, the lack of shared/controlled vocabulary may result in a *long tail* of seldom used or unique tags. Since many users may only use few tags to describe a resource, approaches were proposed to automatically find appropriate tags for resources. TagAssist [19] is such a system that suggests suitable tags for unlabeled/new blog posts based on similar labeled posts. In contrast to the tag suggestions found in systems like del.icio,us, here not only popularity/frequency of a tag is considered. Besides normalization and compression (or stemming) of tags, the system exploits different heuristics and information retrieval measures to select the best candidate tags. A similar technique was presented in [20]. An approach for personalized suggestions is given by [21]. Tags previously assigned by the users are recommended for new web pages based on the similarity a website has with the pages already tagged. In addition to automatically generating tags by finding similar tagged content/context, [22] proposes the introduction of a reputation score for users to combat *tag spam*. In suggesting tags from collective user authorities, a goodness measure (adjusted by a reward-penalty algorithm) takes the criteria for a good tagging systems into account to spot high quality tags (high coverage of multiple facets, high popularity, uniformity, of a certain type/function).

2.1.6 Enhanced information access via tags / browsing by tags

Since tag clouds as aggregated information about tag-usage are an important way to access tagged content, some papers are concerned with building more effective, clearly arranged and properly displayed clouds. For example, the authors in [23] propose algorithms to resolve the problem of inadequate font size display due to browser features. In [24], tags were used to create a user profile and the corresponding visualizations for personal(ized) information access. Usually tag clouds are built as alphabetically ordered lists of tags where the font size represents the popularity (i.e. the prior usage frequency). In contrast, this approach also considers relationships between tags in terms of tag co-occurrence as well as the dynamics of tagging - the time sensitivity of tag importance. The co-occurrence was considered important, because tag combinations provide more information about the user for two reasons; first, more than one word may be needed to describe a concept and there may be different viewpoints or aspects to a topic. By putting more emphasis on recent tags, it is possible to find out about new and recent interests. The authors also experimented with different visualization techniques for user profiles. However, this issue is still not sufficiently resolved. Similarly investigating the visualization of tag evolution (in Flickr!), Dubinko et al. [25] developed algorithms to find the most interesting tags to be displayed in Flash [2] animations.

[2] *Marcomedia Flash* is a popular UI technology supported by many Web Browsers

2.1.7 Exploring tags for (multimedia) information retrieval

In [26], tag clouds are used in combination with content-based image retrieval techniques to augment navigation possibilities (via tag browsing or image query by example) and to ground the semantics provided by the tags into the image data. In [27], the authors similarly show how low-level features like color and texture may help to overcome synonymy or homonymy/polysemy, while social tags can be exploited to bridge the semantic gap to a certain extent. In music retrieval, tags can be used as an alternative or additional means to find songs. For example, in [28] Last.fm songs are not only recommended based on the tracklists (song and artist) of similar users, but also by considering (descriptive) tags. This way some semantics may be included even without content analysis of songs. Based on the idea that tags in bookmarking systems usually provide good summaries of web pages and that they indicate the popularity of a page, [29] investigated the use of tags for improving web search. Their *SocialSimilarityRank*, which measures the association between tags, and SocialPageRank, which account for the popularity among taggers in terms of a frequency ranking, lead to significant better web retrieval. Also in the style of PageRank[30], [16] suggest Adapted PageRank and a FolkRank to improve efficient searching via personalized and topic-specific ranking within the tag space. FolkRank, for example, leads to a personalized topic-specific ranking. This can be used to recommend interesting users, resources and related tags to increase the chance of 'serendiptious encounters'. As well it can be used to find latent communities of interest because the top tags retrieved by FolkRank usually fall within one topic. Marchetti et al. [31] address the problems of free tagging systems by suggesting semantic assertions to tags. In SemKey, the user is supposed to choose a one of three relations between a concept tag and a resource: *hasAsTopic*, *hasAsKind*, *myOpinionIs*. This way advanced searching is enabled. In addition, they implemented a Wikipedia based sense disambiguation module. The TAGMAS approach [32] on the other hand tries to ease the management and search of fragmented personal bookmarks and tagged data stored in different platforms on the web. They present an architecture to integrate and federated-ly search heterogeneous data (model) into a *FolkDesktop*.

2.2 Opinion mining

Opinion mining in trademark product reviews is an important field where a lot of research have been done. Dave et al. [33] present a method for automatically classifying reviews according to the polarity of the expressed opinions, i.e. the tool labels reviews either positively or negatively. They index *opinion words* and establish a scale of rating according to intensity of words, word intensity is established by using machine learning techniques. Finally, to classify a new review, they build an index reflecting the polarity of each sentence by counting identified words.
In an article by Morinaga et al. [34], the authors explain how they verify the reputation of targeted products by analyzing customers' opinions. They start by seeking Web pages *talk-*

ing about a product, e.g. a television; then they look for sentences which express opinions in these websites, and finally they determine if the opinions are negative or positive. They determine it by locating in reviews opinion words which were indexed previously in an *opinion dictionary* .
Other articles present work closely related to the previous one e.g. Turney [35], which classifies reviews in two categories: *recommended* and *not recommended*, or Wilson et al. [36] which categorize sentences according to polarity and strength of opinion, or Nasukawa et Yi [37] which seek opinions on precise subjects in documents.

Two principal methods can thus be used to extract opinion from reviews; there include NLP (Natural Language Processing) or Statistical machine learning techniques. Below we discuss both NLP and statistical methods.

2.2.1 NLP shallow parsing

Liu et al. [38] describe a system which compares competitive products by using product reviews left by the Internet users. The system, named *Opinion Observer*, finds features such as pictures, battery, zoom size, etc. in order to explain the sentiment about digital cameras. They designed a supervised pattern discovery method to automatically identify product features from pros and cons in reviews. A language pattern contains a sequence of words and can be instantiated in many ways: e.g. *<verb> included <noun> [feature] <verb> is <adjective> stingy*. From the multiple instantiations, they extract association rules of the type: *<N1> <N2>* $\rightarrow$ *[feature]* to keep the relevant statistical pattern, and generate language patterns from those rules: *<N1> [feature] <N2>*. They analyze the reviews with those patterns and compare the opinion on each of these characteristics. A component decides the orientation of the extracted feature according to the adjectives extracted near the features. Finally, they classify sentences as negative or positive by determining the dominant orientation of the opinion words of the sentence. The result of the comparison between two products is given in the form of diagram with features on X-coordinate and opinions polarity on Y-coordinate.

Opinion Observer is an example of a complete system based on the fine-grained analysis of sentences and a process counting *Sentiment signs* (words, expressions, patterns). Like many others, they need a Sentiment Dictionary with as many words or expressions as possible yielding opinions. To build such a dictionary, different techniques are possible but they have all the same first steps: manually creating a set of words and expressions carrying the sentiment named; this set is called as the *seed set*, and is useful to find other words and expressions yielding opinions; a third step classify words and expressions according to their semantic orientation (positive, negative, but seldom neutral). In order to increase the initial word set, Turney [35] proposes the following method: to find semantic orientation of not classified words or expressions, they count the frequency of these words or expressions beside a word or expression already classified and he defines the semantic orientation of words or expressions

studied according to their neighbors. Each time an adverb or an adjective is encountered, they extract pair of consecutive words:

- Adjective with noun,
- Adverb with adjective when they are not followed by a noun,
- Adjective with adjective when they are not followed by a noun,
- Noun with adjective when they are not followed by a noun,
- Adverb with verb.

The second extracted word allows confirmation of the polarity of the first adjective or adverb by giving an outline of the context of the sentence. This method, counting co-occurrences with words semantically oriented and manually selected, is also used by Yu and Hatzivassiloglou [39] in order to determine which words are semantically oriented and the direction and strength of their orientation. They have also tried this method by including pairs of adjectives into the seed set.

A second method consists of using linguistic co-locations of words/word-groups with a similar significance. To determine words sharing the same significance, Pereira et al. [40] and Lin [41] propose two methods to search words with the same significance (synonyms) where the meaning is unknown.

Another method to increase the seed set (described by Hatzivassiloglou and McKeown [42]) consists of using conjunctions between a word which semantic orientation is known and a non-classified word. For example, if there is the conjunction "*and*" between two adjectives, we can consider that the terms have a close signification. On the contrary, if there is the conjunction *but* between two adjectives, we can suppose that the two words have a different semantic orientation.

One additional method to find semantic orientation of words uses *WordNet* (Miller et al. [43]). In order to determine semantic orientation of a new word, Hu and Liu [44] use sets of synonyms and antonyms present in WordNet to predict semantic orientation of adjectives. In WordNet, words are organized in a tree (see Fig. 2.2.1). To determine polarity of a word, they traverse the trees of synonyms and antonyms of this word and if they find a seed word in the synonyms, they allocate the same class, but if they find seed word in the antonyms, they allocate the opposite class. If they do not find any seed word, they remake the analysis with synonyms and antonyms, and so on until finding a seed word.

This method has a potential error because words can have different meaning according to the context and thus they can have synonyms not signifying the same meaning. For example, the word *like* is a synonym *love* but in the sentence "*It is like that*", it have not the same meaning. This method finds a positive opinion in this sentence whereas such an opinion is actually not expressed. However, using the same method after having linguistically processed the corpus before (i.e. grammatical analysis) could be more effective. For the previous example, if the

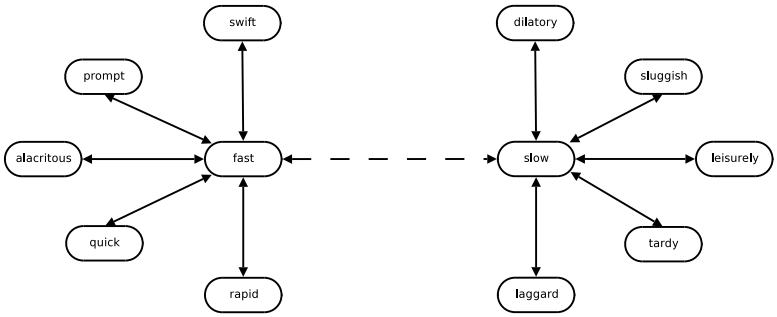

Figure 2.1: Tree of synonyms and antonyms in WordNet (full arrow = synonyms, dotted arrow = antonyms)

seed word is *like/VERB*, we would not find opinion in the sentence *It is like that*.

To measure more precisely the strength of opinion expressed in a sentence, a possible method is to extract adverbs which are associated with adjectives. Indeed, Benamara et al. [45] propose a classification of adverbs into five categories : adverbs of affirmation, adverbs of doubt, adverbs of weak intensity, adverbs of strong intensity and adverbs which have a role of minimizer. A system of attributing points according to the category of the adverb allows to calculate strengths to adverb-adjective combinations. We have used this classification for our opinion adverbs.

To discover a polarity (negative or positive) in a sentence, we can count the number of terms with positive semantic orientation and negative semantic orientation; if there are more positive terms, the sentence is declared positive, else it is declared negative, unless there are as many positive as negative terms, in which case either sentence is declared neutral (Yu and Hatzivassiloglou [46]). Additionally, the last term carrying opinion which might be used to determine sentence polarity (Hu and Liu [44]). Otherwise, we can extract opinion one by one associated with the feature to which relates the opinion (Wilson et al. [36], Hu and Liu [47]).

2.2.2 Machine Learning Techniques

Machine learning techniques have also been used for opinion mining. To quantify opinion from product reviews, Pang et al. [48] show that machine learning techniques perform better than counting methods used previously, with an accuracy of 83% in the polarity classification of reviews. These systems aim to classify reviews into classes, e.g. *positive/negative*, including the *neutral* class; alternatively, this can be considered a regression problem of predicting rates from 0 to 5. Wilson et al. [36], according to the section below, prepare their data with an NLP annotating tool describing the strength of opinion words, but to classify the reviews, they use

machine learning techniques instead of simply counting the salient words. They test three machine learning algorithms well-known within NLP field: BoosTexter (Shapire and Singer [49]), Ripper (Cohen [50]) and SVMlight (Joachims [51]) on a corpus annotated by Wilson and Wiebe [52]. They achieve significant improvements compared to previous research. Pang et al. [48], who work on sentiment classification in movie reviews, use two other algorithms in addition to Support Vector Machine algorithm: Naive Bayes classification and maximum entropy classification.

Machine learning techniques can be used also to build the Sentiment Dictionary. Riloff and Wiebe [53] is an example of such a usage. They focus on the bootstrapping process that learns linguistically rich extraction patterns for subjective expressions. They show that it is better to learn expressions because they are linguistically richer than individual words or fixed phrases.

2.2.3 Combination of NLP and Machine Learning

Combining NLP shallow parsing and machine learning techniques appears to be the most effective solution to extract knowledge in connection with opinions in any corpus; this idea was first presented by Turney and Littman [54]. They use an algorithm for unsupervised learning of semantic orientation which creates association between seven positive words (good, nice, excellent, positive, fortunate, correct and superior), seven negative words (bad, nasty, poor, negative, unfortunate, wrong and inferior) and others words of a corpus. Nigam and Hurst [55] have developed an automated system for detecting opinion about a topic of interest. They use NLP techniques to detect expression yielding opinion and to put marks on the text near each word (grammatical feature and $+$ or $-$ for positive or negative opinion). Then they use machine learning techniques to classify documents according their opinion.
Hatzivassiloglou and McKeown [42] also use both NLP and Machine learning to exploit the semantic orientation of adjectives. They use NLP techniques to extract adjectives from corpus and they apply a clustering algorithm to classify adjective yielding opinion according their polarity.

Although full comprehension of natural language text remains poor, the state-of-the-art in Natural Language Processing techniques enriched by Machine Learning (e.g. for Sentiment Dictionary construction), offer improved results in classifying the polarity of a textual content expressing sentiment. For the majority of the papers, results reach an accuracy of 80% or more in opinion prediction.

2.3 Blog Analysis

According to *Wikipedia*[3], blogs have been defined as a website where entries are written in chronological order and commonly displayed in reverse chronological order. In essence, blogs are a form of social media which is characterized by a self publishing nature. Proliferation of Blogs is fueled in part by the fact that they are relatively easy to create and they allow users to freely express themselves in many ways, artistically or even politically. In addition to self expression, blogs have added new complexities and structures of social networks on top of the existing web. Finally, from a sociological perspective, blogs in their totality represent the *wisdom of the masses* that often cannot be explained or captured with traditional media. For these reason, they are considered an interesting artefact of social media and will be examined in this section. First, we discuss the state-of-the-art in blog analysis and then present ways in which user profiles can be built from blogs. Finally we conclude with some directions and open challenges related to blog sphere.

2.3.1 Overview of Blogs and public forums

A blog is a web page usually published using a blogging software, which annotates text with timestamps. A typical anatomy of a blog/weblog includes:

Blog entry/post : indivisible unit of the blog page

Title : Descriptive words used to preface the post

Timestamp : time and date at which the post was written Author person who wrote the post

Permalink : identifier for each post on the blog page

Hyperlinks : links to external or internal web site or blogs

RSS : link to receive syndication of the blog site

Comment : reader / audience feedback on the content of the post

Annotation : such tags or classification with provide metadata describing the topic of the post

Reader endorsement : such as dig or vote representing a readers preference for the content

Trackback : hyperlink to post writing previously in time used as a basis for seeding the current post

Given their anatomy, blogs essentially induce networked structures on top of the existing web and social media. In an effort to capture and study this social phenomenon, research on several aspect of blogs is being conducted; these include: *sentiment analysis, visualization, link/structural, analysis, trend analysis, content analysis, groupware, authorial analysis , tagging, communities* , and to a limited extent - *spam & diffusion*. Most recently, *micro-blogging* is growing and represents a form of lightweight blogging; such systems include Twitter[4]. Since

[3] en.wikipedia.org/wiki/Blog

[4] www.twitter.com

our emphasis is on personalization, we investigate the state-of-the-art with respect to user profiling using blog data.

2.3.2 Building user profiles from Blogs

In this section. we discuss: two types of information which can be acquired in order to build user profiles from blogs: the first type discussed is *explicit information* based on self promotion tagging, and endorsement. The second is *implicit information* based on co-occurrence associations and blogger emotion. We present the strengths and weaknesses of each approach and conclude with current challenges and open issues.

Explicit Information

Explicit information is knowledge which is supplied directly by users, for example by selecting relevant topics, rating content or by completing a questionnaire. Typical examples in social media include registration processes such as StudiVZ[5], where the user is asked to select their discipline from a list of available choices, or the registration to a syndication service, where the user is asked to supply a set of interesting topics in order to receive news syndications. Explicit information can be gathered from blogs based on self promotion tagging, and endorsement.

Endorsement : In endorsement based information, blog users express a liking for the content of others' blogs. There are several ways in which this is done: the first is a *blog roll* where users explicitly link blogs which they have read and suggest others to read as well. A profile can be build from a blog roll and users with semantically similar or statistically similar blogs can be used to support matching of like-minded users. Another form of endorsement is *eprops* and rating. As depicted in the Fig. 2.2. users who read a blog can either give stars(ratings) or *eprops* to express a preference for the content of the post. The granularity for these two mechanisms are different. In the five-star system, the granularity is *multi-valued* – on a scale of 1 to 5, whereas an eprop represents a *unary* granularity. Unary granularity represents the fact that if a user endorses the post with an eprop, we know the content is preferred; on the other hand, if the blog reader does not offer an eprop, we can not infer a preference for the content. In Fig. 2.3. we can see users who supplied an eprop. In such systems, users provide direct feedback about what blog post they liked.

Promotion in Blogs : In self promotion, blog writers associate tags with articles they wrote and promote their blog and information about themselves via a *blog aggregator*, such as Bumpzee[6]. With Bumpzee, users can promote information about themselves by creating

[5] www.studivz.net/
[6] www.bumpzee.com

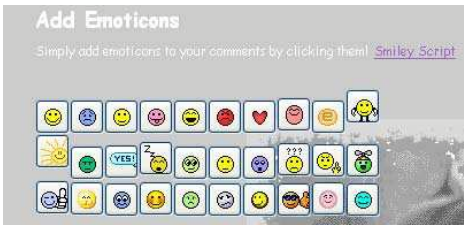

Figure 2.2: eprops Endorsement

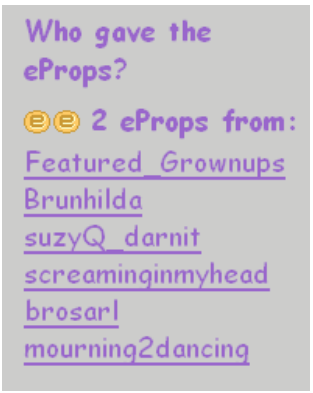

Figure 2.3: Example of "eprops"

communities, which people can join. This effectively allows common blogs to be aggregated within the community. A key aspect is that users can specify which posts to include or exclude based on the tags in the blog. A blog user can direct a system such as Bumpzee to only add posts with the *inline skating* tag to the *Inline Skating* community. In this manner, users explicitly select relevant topics describing the content of their post and make such information available on a larger scale by promoting their blog content with a blog aggregator.

Blog-buzz.com – Similar to Digg[7], but for blog posts. Blog-Buzz is a blogger driven site which allows bloggers to share, discover and promote blog posts that they find interesting. This site essentially makes the playing field level for bloggers since the competition to get *Digg*-endorsed is high for bloggers as compared with other types of websites. Blog-buzz users submit blog posts in a variety of categories and users *Buzz* blogs which they support as interesting. The more *buzzes* a blog has, the more it has a chance of increasing visibility and getting it to the main Blog-buzz page.

Social bookmarking : Yet another form of endorsement is the social bookmarking. Social bookmarking is a system or network in which users store lists of Internet resources and blogs that they find interesting. These lists can be accessible to the public by users of a specific network or website. Other users with similar interests can view the link to the post by topic, category, tags, or even randomly. Typical examples include *Del.icio.us*, *Netvouz*, or *Blue Dot*. In Blue Dot, for example, links to a page are made part of a Blue Dot user profile.

In all these promotion systems, profile information is made available about the users based the endorsement of others. Typically for a given user, all users who made the endorsement can be seen – thereby creating a community of like-minded users based on endorsement. The advantages of explicit information is that it offers a precise representation of the user, However, the data can be subject to either: 1) deliberate misrepresentation, or 2) user bias because they essentially represent a user's self-evaluated perception of a particular item. Furthermore, it

[7] www.digg.com

requires explicit action on the part of the user to record their preferences.

Mood : The reported mood of bloggers can be used to build profiles. Mood typically represents a finer granularity of a user – taking into account the current location and other ambient factors such as music which they are currently listening to. Additionally, users can associate emoticons with their posts to express addition mood or emotions not easily or automatically detectable from the writing

Implicit Information

Implicit methods estimate and infer knowledge about the user by observing and tracking user behavior and interaction with an information system. The actions of the user must be interpreted to impute a vote or preference. In contrast to explicit methods, implicit methods of collecting data attempt to reduce the burden to the user by implicit tracking, tied typically into the user/system interface. In this section, we discuss implicit information based on emotion and co-occurrence association.

Sentiment : Profiling users based on blog sentiment is gaining in popularity. In this approach, the content of a blog post is used to determine if the writer or commenter holds a positive, negative of neutral opinion regarding the post. When combined with topic analysis of the content, a profile of a user and commenters can be built by representing the sentiment of users for a given topic.

Association Co-occurrence : Based on the area of bibliographic citations, profiles of users can be built from the bloggers whom they cite either using *clipmarks* or *trackbacks*. One example of clipmarking is Clipmarks.com ,which essentially is a form of block quoting. This allows a blog user to clip just the chosen bits of a post and insert them into their own blog as quotes. Trackbacks allow user to include a link to another post in the content of their blog. Associations allow us to build implicit profiles of the user by using the content, tag and profile information of the *cited* bloggers as a representation for the citing blogger.

2.4 Recommendation Algorithms

Recommendations are a part of everyday life; when in doubt, we usually rely on external knowledge and judgments of others to make informed decisions about a particular action. There are many factors which may influence the decision making process of a person – this is a complex task and the subject of much research. Automatic recommender systems Ideally, a recommendation system should be able to track as many factors as possible. In the context

of social media, recommender systems have an important role to play; for consumers, recommendation of content with is likely to be well received based on past user experience can help in discovering interesting but previously unknown sources of information (e.g. blog posts from new bloggers) : for content providers, user feedback can be easily interpreted which can be used to better identify target users and potential advertising opportunities.

Recommender systems have been built for entertainment content domains, such as movies, music, book recommendation etc [56]. Generally speaking, the reason people could be interested in using a recommender system is that they have so many items to choose from – in a limited period of time, that they cannot evaluate all the possible options. A recommender system should be able to filter thru all this information and bring forward the most relevant iformation to the user. Intuitively, the recommendation problem can be split into two subproblems; the first one is a *prediction problem*, and is about the estimation of ratings for a set of items that a user has not seen. The second problem is to recommend a ranked list of N items – assuming that the system can predict ratings for yet unrated items. The most relevant problem is the first one i.e. prediction. Once the system can estimate items into a totally ordered set, the recommendation problem is as simple as listing the first N items with the highest estimated value.

There are many approaches to solve the recommendation problem as stated above. One approach is that the system provides informed guesses, based on ratings that other users have provided, as well the current user's ratings. This approximation is called *collaborative filtering*. Another approach is that the system collects information describing the items and then, based on the user's preferences, the system is able to predict which items the user will like. This approach is known as *content-based filtering*, as it does not rely on other users' ratings but on the description of the items. Another approach is demographic filtering, that stereotypes the kind of users that like a certain item. Finally, there is also a method that combines some of the previous approaches; it is fittingly called the *hybrid approach*. However, the common aspect of all these methods is that a *prediction problem* needs to be solved.

The prediction problem can be formalized as follows [57]: Let $U = \{u_1, u_2, \dots u_m\}$ be the set of all users, and let $I = \{i_1, i_2, \dots i_n\}$ be the set of all possible items that can be recommended.

Each user u_i has a list of items I_{u_i}. This list represents the items that the user has expressed his interests[8]. Note that $I_{u_i} \subseteq I$, and it is possible that I_{u_i} be empty[9], $I_{u_i} = \emptyset$. Then, the function, $P_{a,j}$ is the predicted likeliness of item i_j for the active user u_a, such as $i_j \notin I_{u_i}$.

The prediction problem is reduced to creating a list of N items, $I_r \subset I$, that the user will like the most (i.e the ones with higher $P_{a,j}$ value). The recommended list cannot contain items from the already specified user ratings, i.e. $I_r \cap I_{u_i} = \emptyset$.

The space I of possible items can be very large, and similarly, the user space U, can also be enormous. In recommender systems, the output of a prediction function is usually numerical e.g. $r \in= 1..D$. User ratings are triplets (u, i, r) where r is the value assigned – explicit or

[8] Note that these information can also be implicitly assumed from previously describe methods like opinion analysis.

[9] Specially when the user creates an account to a recommender system

implicitly – by the user u to a particular item i.

2.4.1 Collaborative filtering

The main idea behind collaborative filtering (CF) is that the user gives feedback to the system, so the system can provide informed guesses, based on ratings that other users have provided. The more feedback the user gives, the easier it is for the system to return decisive recommendations. The first system that implemented the collaborative filtering method, in 1992, was the *Tapestry Projec*t at Xerox PARC [58]. The project coined the term *collaborative filtering* . Other early systems were: a music recommender named Ringo ([59], [60]), and GroupLens, a system for rating USENET articles [61]. A compilation of other systems from this early time period can be found in [62].

CF methods work by building a matrix of users' preferences (or ratings) for items. Each row represents a user profile, whereas the columns are items. The value $R_{u,i}$ is the rating of the user u_i for the item i_j.

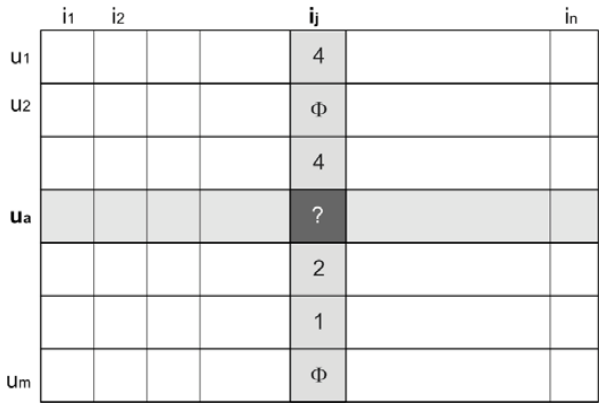

Figure 2.4: Collaborative Filtering: the task of prediction is equivalent to filling missing entries of the user-item matrix

Figure 2.4 depicts the matrix of user-item ratings. The predicted rating value of item i_j, for the active user u_a, $P_{a,j}$, can be computed as the mean of the ratings' values of users similar to u_a. This technique allows to solve the user profile–item matching problem. Equation 2.1 shows the predicted rating score of item i_j, for user u_a. $\bar{R}_a$ is the average rating of user u_a.

$$P_{a,j} = \bar{R}_a + \sum_{u \in Neighbours(u_a)} sim(u_a, u)(R_{u,j} - \bar{R}_u) \tag{2.1}$$

This approach is known as User-based or Memory–based Collaborative filtering algorithms. Yet, to predict $P_{a,j}$, the algorithm needs to know beforehand the set of users similar (i.e neigh-

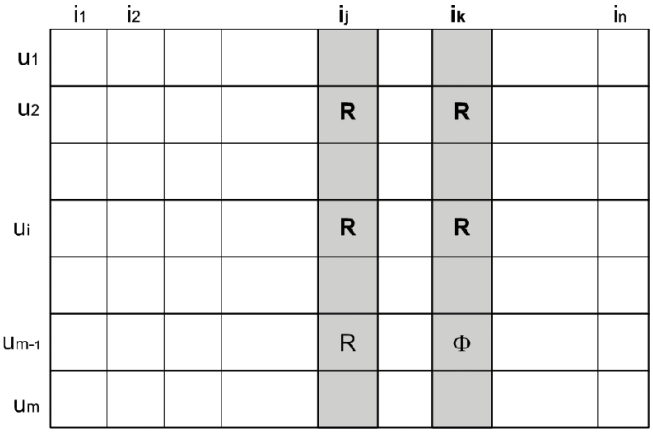

Figure 2.5: Item-Item Collaborative Filtering

bors) to the active user, u_a, and how similar they are ($sim(u_a, u)$). This is analogous to solve the user profile matching problem. The most common approaches to find the neighbors of the active user are cosine similarity (see equation 2.2), *K–Nearest Neighbours* and clustering based on stereotypes [63].

Item-item Based Collaborative Filtering

Ihe item-based method is based on item similarity rather than user–user similarity. This method looks into the set of items that a user has rated, and computes the similarity among the target item to decide whether it is worth to recommend it to the user or not.

Figure 2.5 depicts the co–rated items from different users. In this case, it shows the similarity between items i_j and i_k. Note that only users u_2 and u_i are taken into account, but u_{m-1} is not because it has not rated both items. The first step is to obtain the similarity between the two items i_j and i_k. This can be achieved by using cosine similarity, correlation–based similarity, or adjusted cosine similarity methods [57]. Let the set of users who rated i and j be denoted by U, and $R_{u,i}$ denotes the rating of user u on item i. Equation 2.2 shows the definition of the cosine similarity:

$$sim(i,j) = cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|i\| * \|j\|} = \frac{\sum_{u\in U} R_{u,i}R_{u,j}}{\sqrt{\sum_{u\in U} R_{u,i}^2}\sqrt{\sum_{u\in U} R_{u,j}^2}} \quad (2.2)$$

However, for the item-based similarity, the cosine similarity does not take into account the differences in rating scale between different users. The adjusted cosine similarity (equation 2.3) makes use of user average rating from each co–rated pair, and copes with the limitation of cosine similarity. $\bar{R}_u$ is the average rating of the u–th user:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_u)^2}} \tag{2.3}$$

Correlation–based similarity commonly uses the $Pearson - r$ correlation coefficient. The correlation between two variables reflects the degree to which the variables are related. Equation 2.4 defines the correlation similarity. Note that this measurement is done only over the items commonly rated by two users. $\bar{R}_i$ is the average rating of the i–th item:

$$sim(i,j) = \frac{Cov(i,j)}{\sigma_i \sigma_j} = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_i)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_j)^2}} \tag{2.4}$$

Once the similarity among items has been computed, the next step is to predict to the target user a value for the active item. A common way is to capture how the user rates items similar to the active item. This can be done by computing the sum of the user's ratings – only for the items similar to the active item – weighted by the item similarity.

2.5 Recommendation and Latent Semantic Analysis

In age of information overload, it has become important to find the most relevant information for an information need. Once the subset of relevant information has been found, the problem of presenting these results to the user still exists. Approaches like PageRank and the success of Google have proved that *Ranking* is an important part of any information filtering solution. In this section, we explore the use of Latent semantic indexing for ranking data. The essential assumption is that data is multi-dimensional, i.e. each record has several variables (e.g. one variable for every word observed in the entire text corpus), and this data is succinctly represented as a matrix. Generalizations have been made to collaborative data (user votes, user tags) as well. We envision a scenario where user preferences over a number of seemingly dis-similar items are available; the benefit of using LSI is that implicit user preferences can be discovered

Latent Semantic Analysis (LSA) [64] was a hot topic for during last decade, and continues to incite interest [65]. It involves performing Singular Value Decomposition (SVD) on a matrix representing text documents in a Vector Space Model. The benefits of LSA were discovering polysemy and hynonemy, while limiting the dimensionality of text indices to a fraction of the original index size. In one of the pioneering works on SVD,[10] [66] the authors compared it to a Semi-Discrete Decomposition (SDD). This algorithm compresses the original matrix by order of magnitude, but the decomposition itself becomes two orders of magnitude slower than SVD.

One of the first ideas for improving LSA's speed was outlined in [67]. The random projection

[10] As we are interested in SVD mainly as in the main LSA mechanism, we will use terms LSA and SVD as synonyms

was suggested as a pre-processing step before actual SVD computation. Such a projection should preserve the matrix mostly rank unchanged, but can reduce its size making the following SVD much faster. Later, in another work [68], the random projection was actually tested on the image and text data. Experimental results show that the quality is close to that of LSI, while the computation costs are many times smaller. It was suggested that LSA can be used not only as a pre-processing step but also a noise-reduction method on its own if the computational costs are crucial.

Following the same intuition, a linear time version of approximate SVD was proposed in [69, 70] and later extended in [71]. It is based on a randomized algorithm, which picks a number of columns from original matrix w.r.t. specific probability distribution. The resulting sample matrix is scaled and used for approximation of SVD. In subsequent work [72, 73], researchers tested this method on image data and it performed at least order of magnitude faster than a normal SVD, with an acceptable approximation error. Some low-level details on an experimental prototype can be also found in [74].

One of the most relevant experimental evaluations has been done in [75]. This work is also using the idea from [69] of SVD approximated from a sample, but testing it on a TREC collection of Associated Press news articles. About $80,000$ documents were indexed using different sampling strategies, about 10% of the matrix has been sampled with around 0.13% of non-zero values. The most striking result is that none of the SVD versions performed consistently better than the usual Vector Space Model (VSM). This effect is explained as a casual property of the provided queries. Among various versions of SVD used, the vaiation using weighted sampling showed the best improvement in the experiments, while on average it was outperformed by uniform sampling. Interestingly, the approximation error is not transformed linearly into the precision degradation and uniform sampling strategy on average outperform exact SVD. It would be interesting to perform futher investigation on this problem.

One of the few large-scale applications for LSI is presented in [76]. The paper addresses a problem of query routing and document retrieval in a large peer-to-peer network. Yet again, the idea of reducing the input matrix for SVD is explored, but this time a specific domain knowledge is used. In a query routing task the documents from the similar peers can be aggregated into large documents and the total number of columns in the final matrix is reduced. In addition, the entries with low weights are discarded. The SVD on a resulting matrix is several orders of magnitude more efficient. Among other observations authors note that proper normalization of terms and documents almost double recall, besides, LSI serves as a natural document clustering. Unfortunately, the retrieval quality of LSI is still inferior in comparison to VSM-based Okapi model.

Another work [77] presents an efficient method to compute SVD of a product of two matrices. It improves the speed of SVD computation directly on a product matrix and shows an easy way to extend this method to product of multiple matrices. Authors note that the algorithm is based on Jacobi-like methods, which have moderately higher complexity than QR methods. This algorithm was successfully applied in [78] for the parallel SVD computation in a distributed network. The latter paper provides some interesting insights into the implementation-level details. A detailed distributed architecture for the SVD computation on a large cluster of ma-

chines has been presented in [79]. The final goal is to speed up the computation by moving parts of it on a hardware level.

In one of the recent papers [80], a new problem statement was presented for the SVD application to the classification tasks. The main idea of the method is to change representation of the data samples from vectors to small matrices. Therefore, instead of a single large matrix one should perform low-rank approximation on a large set of small matrices. The classification quality of new algorithm is better than of original SVD while speed is about 10 times higher.

A good overview of the SVD-based classifiers from Netflix[11] competition is presented in [81]. The paper contains interesting comparison of different methods with respect to their accuracy and speed convergence. Hints on parameters which can speed up the convergence are also provided.

A major motivation for our work is provided by the recent paper [82]. Variable LSI is a new modification of SVD approximation which minimizes approximation error w.r.t. specific query distribution. Experiments show that the same approximation quality can be achieved using ten times less underlying concepts. This might be an interesting direction to explore, modifying original matrix with the user-specific term distributions. We explore this idea in more detail in Chapter 4.

2.6 Conclusions

In this chapter, we have described a body of previous work relating to Social Media and techniques for analyzing it. While several other works can be described here, we have limited ourselves to the most significant and relevant ones. In the next chapter, we will look at new algorithms for exploiting social media.

[11] www.netflixprize.com

3 Algorithms For Creating User Profiles

A user profile can be defined as knowledge about the user, obtained either explicitly or implicitly, that is used by a system to improve the interaction between the system and the user. Attempts to model users has been long sought and are important, in order to evaluate the relevance of data items according to the user profile and attempt to deliver relevant, personalized information. Moreover, when addressing the issue of information overload, the application of user profiles is a crucial and established practice.

In the previous chapter, we have described a body of work relevant to Social Media. With an depth knowledge of the sate-of-the-art, the PHAROS consortium has performed innovation research improving on the *status quo*; we present below novel approaches and techniques to exploit social media for creating user profiles. Most of the presented work has additional experimental support and has been published in academic conferences in the recent month. We start with Collaborative Tagging (multiple approaches), followed by Blog diffusion and finally sentiment analysis from user generated content.

3.1 Creating A User Profile By Analyzing User's Tags

Users collect and manage various kinds of resources on the Internet and many applications rely on tags for supporting the management of resources. As can be seen from the analysis reported in the Chapter 2, tags do not only tell something about the resources but they also tell about the person who made the tags: the person is likely to store resources that he or she finds interesting so they reflect the person's interest and are therefore a good source for an interest profile. What is good about tags is that they benefit the user directly, which is an incentive to make them, thereby producing data for the profile without any extra work. Our vision is that the user indicates which services and userids can be used by an application in order to harvest the user's tags, and these are then processed into a personal profile. The profile is available to the user and the user is given an opportunity to modify the profile. The profile can be used for ranking search results, for recommending content, supporting browsing and linking to other users with similar interests.

Tags and tagging can be analyzed from different viewpoints. Our purpose here is to use tags as indicators of personal interests and as a source for creating an interest profile. Tags create networks between people if and when people tag the same resources and/or use same tags. Also, tags create information about people's views on the actual resource and how interesting and relevant they find them: however, in this study, we do not address these issues.

3.1.1 Analysis Of User Tags

Like already mentioned, tags are utilized in different types of applications. del.icio.us made tagging popular; there tags are used to manage and share bookmarks. The application lets users store any bookmarks they want, and this way it makes a good source of information relating to a person's interests. We gathered tagging data of 1035 users from del.icio.us to test the creation of user profile from user's tags. The dataset consists of the 100 latest tagged resources and their tags as well as the list of all the tags that the users in the dataset have created and how many times they have been used. The varied nature of resources bookmarked and tagged at del.icio.us makes it a good source for analyzing user interests. The methods for analyzing tags can be utilized in tags created in other services as well (e.g. Flickr!).

Characterizing tags There are different alternatives to analyze tags; single tags, tag combinations in connection to a single resource and tag grouping based on their meaning (topic, opinion, and resource type). Resources that are referenced can be analyzed in combination with rating information and opinions. The tags of other users and other metadata can be used as additional information source regarding the linked resource. In the state-of-the-art (see Sec. 2.1), several research papers that report studies about the type of tags people use were presented. Different types of tags can be used for different purposes when analyzing a user's interests or describing resources. Topics (like *travel, semanticweb, cat, cars*) can be used for analyzing user's interests as well as characteristics of tagged resource. Type of referenced resources (like *blog, wiki, video, music*) can be utilized as information of user's preferred resource types. Proper names (like person, company, product, event, location (Nokia, BBC, Oslo) can be used as an indication of user's interests as well. Additional information may be obtained from the web to reason a user's interests further. For example, from Nokia company profile, we can harvest information about their business areas; mobile devices, mobile networks which can also indicate the interests of a user who has used the tag Nokia. Subjective tags (like interesting, great, +++) as well as toDo tags (like toRead, toPrint) can be used together with other tags to confirm the user's interest areas. Self reference type of tags (like Mystuff, mywork) can be used by analyzing the linked resources, which give indication of the user's own activities. These initial tag categories can be condensed into three main ones that are the most relevant from a user profile perspective. These categories are user's interest, opinion and type of resource. The category of user's interest combines topic, proper name and self-reference categories together. Opinion is a combination of subjective tags and toDo tags categories.

Assumptions in our approach: Our initial intention was to utilize these three categories in creating a user profile. However, the first analysis of our tagging data showed that Opinion alone does not bring much new information into creating a user profile because the opinion is related to the resource. The user is probably interested in a topic of the resource although s/he is not postively opinionated about particular tagged resource of that topic. Opinion tags are more relevant when analyzing the tagged resource. Another initial idea was also to give

more weight to the tags that were used together with toDo types of tags, because they express user's intended future activities. However, not all users use this type of tags and for those who did, the influence to the total tag usage frequency was marginal. So, at this stage, we have concentrated on the user's interest category. Tags describing the type of resource can be utilized to analyze user preferred sources (see chapter user preferred sites for more details) We believe that categorizing tags helps in analyzing the masses of very heterogeneous tags. We will continue working with this aspect in task 2.1.2, not only for creating a user profile out of tags but also for utilizing user created metadata for describing resources.

3.1.2 Tagging Habits

Tagging behaviour and practices vary a lot between people, because tagging is made mostly for personal use, not for common good. There are several characteristics that can be used for describing and comparing users' tagging habits. These characteristics are the number of different tags of each user, the total number of used tags, the number of tagged resources as well as the number of tags assigned to a resource in average. Also, statistics about the lexical analysis of each user's tags may reveal something about a user's tagging habits: how many of the user's tags are nouns, adjectives etc. and how much there are recognized words compared to unrecognized words. Users have different habits to use tags concurrently. The general habit is to tag from general tags to more detailed tags /1/; some users use a lot of synonyms or different spelling variants (plural, singular, acronyms) of a word or concept. As already mentioned, users also have different ways of combining words. There are tags in which each word is separated by a certain character, by a hyphen, by an underscore, by a slash or by a space, two separate tags can always be used together to indicate specific meaning. Below there are some statistics about the tagging habits of del.icio.us users in our dataset (1035 users, the latest 100 tagged resources / user):

- 1,55% of the users have over 1000 tags, 2,2% had over 900 tags, 14,2% had over 500 tags, 50% had over 178 tags and 5,1% had less then 10 tags. The users which have very few tags are not used for analyses.
- The average number of tags that users assign to a resource (bookmark) was 3.1 ± 2.4 tags/resource, the maximum was 20,5 tags/resource.
- The average number of total number of tags of a user was 273 ± 234 tags/user and the maximum was 2046 tags/ user.

As a first step in analyzing tagging data, we have compared the results of two users who have different tagging habits. One of them is a user who has approximately the average number of total tags as well as an average number of tags per resource. The other one is a heavy user with plenty of tags and tag co-occurrences. The heavy user also makes a lot of different spelling variants of the same words. Results of these experiments showed that the same parameters used for tagging data of both users do not produce satisfactory results.

Conclusion and an assumption in our approach: Users' tagging habits must be taken into account in the parameters to be used in the tag analysis. This influences the selection of the most relevant tag usage frequencies. The impacted method for determining a threshold for selected user's tags is described in Section 3.1.4.

3.1.3 Time Sensitivity

In our del.icio.us dataset, the number of tagged resources is limited to the 100 latest tagged resources and their associated tags. The dataset also includes the list of all the tags the user has created and how many times each tag has been used. We can assume that each user's latest tags describe their current interest. The comparison of the recent tag usage to the total tag usage gives some hints about the long term interests as well as increasing and decreasing interests of the user.

Conclusion and an assumption in our approach: The list of tags describing users' interest is influenced both by the recent and total tag usage, but the most recently used tags have more influence on the profile. At the moment, this has been taken into account as optional feature. The comparison between an initial profile that has been created based on user's most recent tags and a profile based on user's total tags is made. The tags that are not listed in the initial profile, but are listed in the profile made based on all tags of the user are potential candidates to be included in a user's profile. These tags could be shown to the user and the user could then decide whether he or she wants to include them into his or her interest profile as well. Our use scenario is that the user can indicate several services that can be used to harvest user's tags. As mentioned earlier, the data availability depends on the APIs of the service. In most services not all of the user's data is available. In case of del.icio.us we have the data about the recent tags of the user as well as some information about all tags of the user. Because this might not be the case in every service we wanted to keep mixing of recent and all tagging data very simple at this stage of development. The approach will be reconsidered when we have analyzed the available tagging data of different services.

3.1.4 Tag Usage Frequency

Tag usage frequency indicates the user's interest of the topic. We can assume that, it is unlikely that rarely used tags relate to the core interests of the user. The absolute tag usage frequency is the total amount tags used by a user. Relative tag usage frequency is the percentage of a tag usage relative to the total frequencies of all tags. Relative frequencies describe the probability of the occurrence of a tag, which this information can be utilized in statistics analysis for creating a statistical distribution of the used tags.

Conclusion and an assumption in our approach: The interests of the user are ranked with the help of tag usage frequencies. The more a tag is used, the more interested the user is in

the topic. The original tag usage frequency list of a user is modified based on lexical and co-occurrence analysis of user's tags. Optionally users are given an opportunity to modify their profile. This can be based on a compared result of analyzed profiles based on user's recent and total tags and the most frequently used unrecognized tags of a user as well as freely selectable tags of a user. There are several alternatives in selecting the criteria for ranking tags into a profile. The tags can be delimited by criteria like

- select tags that have been used at least x times,
- select top x tags,
  - Where x is influenced by tagging habits of the user. (If user has a lot of tags x is bigger than with user who has less tags.)
- based on a distribution

Our first idea was just to use top x tags as criteria to select tags for a user profile. Manual inspection of user's tags showed that selecting approximately the 25 most used tags would describe user's top interests. One limitation of this approach is that it selects top tags independent of the fact how many times a tag is used, also leaving out tags that have been used as much as the last selected tag. Because of users' different tagging habits, using this as criteria gives quite varying results, selecting less important tags of those users who have fewer tags and two few tags for those who have a lot of tags. So instead of selecting top x tags for a profile, our approach is to use a relative cumulative tag usage frequency distribution to determine criteria which tags are included into profile. Relative cumulative tag usage frequency is summarized cumulatively from relative tag usage frequencies when the tags are ordered based on their frequency (see Table 3.1). Relative cumulative tag frequency is used for determining the threshold (x) that is used for selecting tags for a user profile. The tags that the user has used at least x times is used as criteria for selecting tags for a user profile. 30 % of the user's most used tags were used as the cumulative relative tag usage frequency for determining the threshold. By selecting approximately 30 % of the most used tags, we are selecting only the tags belonging to highly or medium used category of tags. If only highly used tags are wanted to be a part of a profile, a percentage should be lower. Also the result of lexical analysis influences for a selected percentage, because of a high number of unrecognised tags. The percentage can be customised. The minimum threshold for all users is that tags are used at least 2 times. So even if the threshold (calculated based on relative cumulative frequency) would indicate to select all the tags of the user, only the tags that have been used more than once are selected. The tags that have been used only once do not indicate a the high interest for the topic. Threshold (x) = tags that a user has used at least x times. The method of selecting threshold for a user is visualized in Fig. 3.1.

3.1.5 Tag Co-occurrence Usage Frequency

In analyzing the tag co-occurrence, absolute and relative tag frequencies are calculated the same way as in the tag usage frequency (see Sec. 3.1.4). The absolute tag co-occurrence

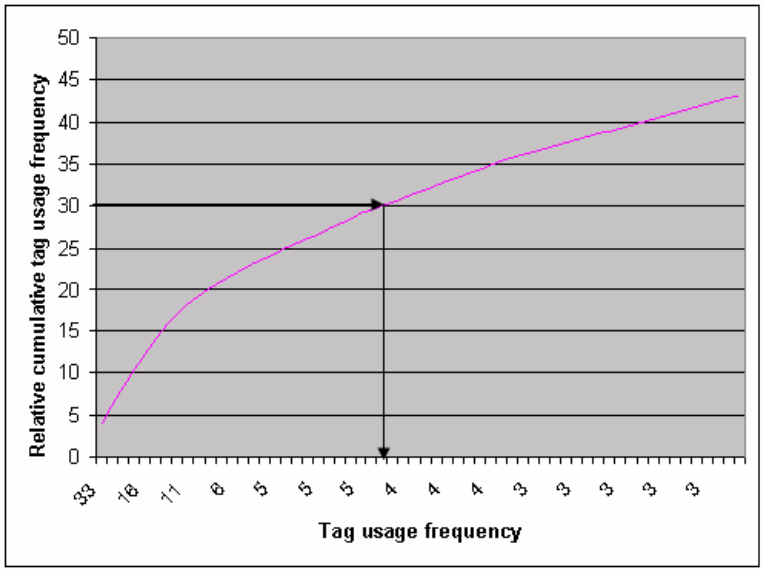

Figure 3.1: An example of how cumulative tag usage frequency is used for determining a threshold for selecting a user's tags for a profile. 30 % is used as criteria corresponding to threshold 4. Tags that have been used at least 4 times are selected for a user's profile

usage frequency is the total amount of co-occurrences. The relative tag co-occurrence usage frequency is the percentage frequency of the co-occurring set of tags relative to the sum of frequencies of all co-occurrences. The most used tags are often general tags (like web), not describing the user's viewpoint to the topic. That is one reason why tag co-occurrences are interesting when reasoning users' interests. By taking both the single tag usage frequencies and the co-occurrence usage frequencies into account, users' interests can be better evaluated. Relative tag and tag co-occurrence usage frequencies which describe the probability of the occurrences can be used in statistical analysis in order to determine the relevancy of the co-occurrences. This is useful in evaluating the relevance of co-occurring tag sets which have the same co-occurrence frequency. This is not a problem with users who have few tags per resource, but may become more relevant with users who have a great number of tags per resource. Another parameter relating to the co-occurrences of tags is how often tags are used, together compared to the total usage of the tag as percentage. If this percentage is 100%, the tag is always used with the determined tag. This approach needs a restriction that the tag is used at least x times (e.g. >2), because if the tag is used only once, its percentage with its co-tags is 100 %. The restriction could be varied based on the tagging habits of the user. This method could be utilised for analysing unrecognised words and for example to minimise the influence of spelling variants of the same tag. When two separate words have always been used together, it indicates that these words have a specific meaning together and in some cases it might be possible to treat these words as *one word*.

Conclusion and an assumption in our approach: The co-occurrences of tags indicate in which aspects the user is interested in a certain topic. The tags selected for a profile are complemented with the information of their most co-occurring tags. Co-occurrences can be restricted by including only the co-occurrences that have been used at least *x* times together. One drawback of our dataset is that tag co-occurrence data for all of a user's tags are not available. So a comparison between recent tag co-occurrences and all tag co-occurrences of a user can not been made. Our use case scenario is that users can give their user-name to different social media services and the tagging data can be gathered through APIs and combined to the profile. The problem is that official APIs of different services allows us to gather different kind of data and not always all data we are interested in. Despite of these restrictions the algorithm should provide a best approximation of user's profile although the available dataset may not be complete. Our experiments have been made only with a dataset from del.icio.us, but later on it would be interesting to test also tagging data of other datasets e.g. Flickr!. Differences in datasets makes automatic creation of user's profile based on user's tags even more challenging.

3.1.6 Lexical Analysis

The lexical class of the tag can give some insight into the importance of the tag. The most important part in determining user's interests would be the tags describing the topic. The topics are mostly nouns. Adjectives on the other hand express an opinion of the user and are relevant when analyzing the tagged resource, but do not have so much relevance in creating the user profile because users usually bookmark resources that are of interest to them. For doing the lexical analysis we used the MIT Java Wordnet Interface to categorize the used tags. This library includes a stemmer, which had to be used before analyzing lexical classification of the word to get any reasonable results. Lexical analysis is a language dependent. We have analyzed English tags and used above mentioned tools for analysis. For different languages tools like stemmers and dictionaries a version developed for the particular language need to be used. Also, the way of creating tags and compound tags might vary between languages. As tags are often formed by word combinations, such as *socialmedia*, a lot of the tags were not recognized as real words and could therefore not be mapped completely or at all. All of the tags were also not in English and were therefore excluded from further analysis.

Another problem with the lexical analysis was that many words have more than one potential meaning in different categories (noun, adjective, and verb). In our del.icio.us case, it can be assumed that people mostly use nouns.

Conclusion and an assumption in our approach: For analyzing user interests our focus is on tags that have recognized lexical class. Lexical analysis is used for cleaning user's tags. A method for analyzing the tags (like *blog, blogs, blogging*) that have the same word root was developed. The result of the analysis can be utilized to create additional connections between tags enhancing search for different spelling variants of a tag. We assume that all of these tags describe the same interest, although from tagged resource point of view they may have a

different connotation. Because the number of unrecognized words is quite high, selecting only tags that have recognized lexical class restricts the tags of the users quite a lot, especially with users who have few tags. An additional analysis is needed for the unrecognized words. This includes the analysis of word combinations and different ways of writing the same words (like social_media, socialmedia, socialMedia). Lexical analysis could be used also for determining synonyms of the words, or words that belong to same *category* based on their meaning. A deeper semantic analysis of the tags would help in ranking the search results based on users' interests. One problem is that although we make recognition of word combinations better, not all new compound words like business model or social networking are found in Wordnet. One approach could be to use DBpedia (`http://dbpedia.org/` or Yago to check whether a description of a term can be found in these databases and also utilise them to get more information about the semantic meaning of a tag.

User-preferred sites

The dataset can be used also for analysing the user-preferred sites. This is expressed as the URL of a site and the number of bookmarked resources per site (at least >1). This gives us information about the user's preferred sources for certain subject areas. By analysing the tags that belong to category "type of resource" (like music, book, blog, video, movie, song) we can harvest user's preferred sites for a certain type of resource like for music or books. Co-occurring tags can reveal what kind of music or books the user is interested in when looking for material in this site. For example, the user might be interested to search books relating to his/her profession from one site and books for entertainment from another site.

Conclusion and an assumption in our approach: At this stage we have concentrated on harvesting user's interests rather than user-preferred sites, but this is one interesting case where tag categories can be utilized. In Pharos showcases, it was mentioned that users can express their preferred sources. This is one opportunity to increase this type of information automatically. Algorithm 1 describes the main and optional steps in our profile creation algorithm.

The optional steps need to be considered whether they are taken to the profile creation or not. They are functionalities that effects also for the user interfaces, how the profile is visualized to user and how user can make modifications.

Related research

Tagging is a relatively new phenomenon, but there is some related research to ours.

In [24], tags were used to create a user profile. The approach was built on two tag features: tag co-occurrence and time sensitivity, and only resources that have at least two tags are included in the analysis. The co-occurrence was considered important, because tag combinations

provide more information about the user for two reasons; first, more than one word may be needed to describe a concept and there may be different viewpoints or aspects to a topic. By putting more emphasis on recent tags, it is possible to find out about new and recent interests. Characterization of users' tags and analysis of tagging systems are researched in papers [3, 7, 22]. We have used these studies are as a basis for seeing what kind of characteristics there are in tags, and how they can be categories relating to creating a user profile. These papers are relevant also when the support for tagging within the Pharos system is considered. In [10], research is reported where *Flickr!* tags are converted into RDF descriptions. The idea is to extract the semantic meaning of tags used for describing a given photo and build a pertinent RDF annotation for this photo. The developed method is based on linguistic rules and natural language treatment as well as on integrating some human knowledge to be able to provide semantic description for pictures from tags. Our approach differs from the earlier work by utilizing both single tags and tag combinations and by connecting lexical analysis to the tags. This way we are able to better pinpoint each user's the interest areas and to create a profile that is easy for the user to evaluate. Our algorithm also takes into account the tagging habits in order to adapt to the very varying tagging habits.

3.1.7 Algorithm and Evaluation Result

The result is an algorithm based on the above-mentioned analysis to create a personal profile. The user profile includes user's interests defined as collection of tags and their co-occurring tags with corresponding information about the tag and tag co-occurrence usage frequencies. The formal definition of a profile of a user U:

Profile $U = < Tag_i, F_i, RF_i, < CoT_j, CF_j >>$ where

- Tag_i = user's tag
- F_i = absolute tag usage frequency = number of times user (U) has used Tag_i
- RF_i = relative tag usage frequency % = $100 \times F_i$ /Total frequency of all the user's tags
- CoT_j = co-occurred tag of Tag_i
- CF_j = absolute tag usage frequency for co-occurred tag CoT_j

The result could be expressed also as a personal ontology. There are several interesting ontologies relating to tags that could be utilised to some extent. The SCOT-ontology [1] describes the result of user's tagging activity containing same kind of information that we have. SKOS [2] provides a model for expressing folksonomies and makes it possible to expand search terms and the meaning of the tags describing the user's interest (e.g preLabel, altLabel; spoken language, misspelling, synonyms, association, related, broader, narrower, isSubjectOf). In Fig. 3.2, there is an example of how these ontologies could be used as part of the user profile. This could be linked as part of the FOAF profile of a user. Output format need to be defined as part of the user model and remains a part of future work.

[1] http://scot-project.org/?p=5
[2] http://www.w3.org/2004/02/skos/

Algorithm 1 Create User profile based on Tags

1: Count tag usage frequencies (absolute and relative frequency as well as relative cumulative tag usage frequency) for recognized tags, for tags that are identified belonging to some lexical class.
2: Use relative cumulative tag usage frequency (30 %) for determining the threshold (x) that is used for selecting tags for a user profile. The tags that the user has used at least x times is used as criteria for selecting tags for a user profile. (The method is described in chapter "Tag usage frequency").
3: Select tags that have been used at least x times for a profile and list them along with information about the tag, possible word roots for the tag, tag usage frequency and relative tag usage frequency.
4: Count the tag co-occurrence frequencies for selected tags. List the selected tags along with the co-occurred tags, frequencies and corresponding tag information for the co-occurring tags. Only list co-occurred tags that have a recognized lexical class and that have been used at least two times together.
5: Handling of the most used unrecognized tags {Optional steps}
6: Count tag usage frequencies (absolute and relative frequency as well as relative cumulative tag frequency) for unrecognized tags. Repeat steps 2 and 3. -> The most used unrecognized tags will be shown to user. The user can modify and include them in the profile if they are relevant.
7: Repeat steps 1 to 3 for total tags of the user.
8: Compare result set with one created in step 3 and select only tags that are not found in that previous result set. -> Tags will be shown to user and user can include them in the profile if they are still core interest of a user.
9: Analyze the user preferred sites

Output: Return User Profile

Tag	Tag usage frequency	Relative tag usage frequency %	Cumulative relative tag usage frequency %
businessModels	7	2.8926	2.8926
MySpace	6	2.4793	5.3719
advertising	6	2.4793	7.8512
socialnetworking	6	2.4793	10.3305
wikipedia	6	2.4793	12.8098
2check	5	2.0661	14.8759
web2.0	5	2.0661	16.942
semanticWeb	4	1.6529	18.5949
search	4	1.6529	20.2478

Google	4	1.6529	21.9007
++	4	1.6529	23.5536
ep2007	4	1.6529	25.2065
screenscraping	4	1.6529	26.8594
business	3	1.2397	28.0991
socialsoftware	3	1.2397	29.3388
Yahoo	3	1.2397	30.5785
research	3	1.2397	31.8182
Japan	3	1.2397	33.0579
jp	3	1.2397	34.2976
communities	3	1.2397	35.5373
openSource	3	1.2397	36.777
Flickr	3	1.2397	38.0167
mashups	3	1.2397	39.2564
SecondLife	3	1.2397	40.4961

Table 3.1: An example of the most used tags selected for a user profile based on the original tag cloud without further analysis. Bolded tags are the tags that have a recognized lexical class.

The approach is tested with tagging data of users, which have different tagging habits. We have mainly focused on users who have average or high number of tags per resources. Comparisons between the profile created by our algorithm and basic tag frequency list are made. In Table 3.1 and 3.2, one can see an example of the tags selected for a profile for a user whose tagging habits represents an average user. The total number of the user's tags is 242 and the average number of tags per resource is 2.42. Table 3.1 represents the tags selected for a user profile based on tag usage frequencies without further lexical analysis. As we can see the most used tags include tags like "++" and "2check" that has no relevance for a user profile, but there are also relevant tags like "businessModels" and "socialnetworking" that were unrecognized. With extra analysis these tags could be analysed and included for a profile. In Table 3.2, one can be see the tags selected for a profile based on the tag frequency as well as lexical analysis of tags. Although the number of unrecognized tags is quite high, we restricted the creation of a user profile only for tags that have a recognized lexical class. The reason is that we wanted to create a cleaner and a better sense for a tag cloud and to avoid importing all nonsense tags into the user profile. Although the profile does not express all the interests of a user and it might drop some relevant (unrecognized) tags from a profile it gives hints about user's interest without that user need to do an extra work for expressing it. What we do need is an additional analysis for unrecognized words to get more results. This includes handling of compound words better, support for handling synonyms, and semantic meaning of a word.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xml:base="http://scot-project.org/scot/ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:skos="http://www.w3.org/2004/02/skos/core#"
xmlns:sioc="http://rdfs.org/sioc/ns#"
xmlns:scot="http://scot-project.org/scot/ns#">

<scot:Tagcloud rdf:about="http://del.icio.us/moth">
    <dc:title>An example of SCOT Ontology, which is enhanced with skos-properties</dc:title>
    <scot:totalTags>242</scot:totalTags>
    <scot:hasTag>
        <scot:Tag rdf:about="http://del.icio.us/moth/advertising">
            <scot:name>advertising</scot:name>
            <skos:prefLabel xml:lang="en">advertising</skos:prefLabel>
            <skos:altLabel xml:lang="en">advertise</skos:prefLabel>
            <skos:related rdf:resource="http://del.icio.us/moth/marketing"/>
            <scot:ownAFrequency>6</scot:ownAFrequency>
            <scot:ownRFrequency>2.4793</scot:ownRFrequency>
            <scot:cooccurWith rdf:nodeID="Cooccurrence_0"/>
        </scot:Tag>
    </scot:hasTag>
    <scot:hasTag>
        <scot:Tag rdf:about="http://del.icio.us/moth/search">
            <scot:name>search</scot:name>
            <skos:prefLabel xml:lang="en">search</skos:prefLabel>
            <skos:altLabel xml:lang="en">searching</skos:prefLabel>
            <scot:ownAFrequency>4</scot:ownAFrequency>
            <scot:ownRFrequency>1.6529</scot:ownRFrequency>
        </scot:Tag>
    </scot:hasTag>

    <scot:hasTag>
        <scot:Tag rdf:about="http://del.icio.us/moth/marketing">
            <scot:name>marketing</scot:name>
            <skos:prefLabel xml:lang="en">marketing</skos:prefLabel>
            <skos:altLabel xml:lang="en">market</skos:prefLabel>
            <skos:related rdf:resource="http://del.icio.us/moth/advertising"/>
            <scot:ownAFrequency>2</scot:ownAFrequency>
            <scot:ownRFrequency>0.8264</scot:ownRFrequency>
        </scot:Tag>
    </scot:hasTag>

</scot:Tagcloud>

<scot:Cooccurrence rdf:nodeID="Cooccurrence_0">
    <scot:cooccurAFrequency>2</scot:cooccurAFrequency>
    <scot:cooccurTag rdf:resource="http://del.icio.us/moth/advertising"/>
    <scot:cooccurTag rdf:resource="http://del.icio.us/moth/marketing"/>
</scot:Cooccurrence>

</rdf:RDF>
```

Figure 3.2: An example of SCOT ontology, which is enhanced with skos-properties

Semantics would help us to cluster the tags that are related to each other.

	Tag usage frequency	Relative tag usage frequency %	Cumulative relative tag usage frequency %	Co-tags	Co-tags (no lexical restriction)
advertising	6	2.4793	2.4793	marketing	marketing
search	4	1.6529	4.1322		
Google	4	1.6529	5.7851	Yahoo	Yahoo, personalisation
business	3	1.2397	7.0248		web2.0

research	3	1.2397	8.2645		
communities	3	1.2397	9.5042		wiki
Japan	3	1.2397	10.7439		
Yahoo	3	1.2397	11.9836	Google	Google, personalisation
tagging	2	0.8264	12.81		socialnetworking, ++
participatory	2	0.8264	13.6364		
technology	2	0.8264	14.4628		
people	2	0.8264	15.2892		
identity	2	0.8264	16.1156		
analysis	2	0.8264	16.942		
reputation	2	0.8264	17.7684		
marketing	2	0.8264	18.5948	advertising	advertising
travel	2	0.8264	19.4212		
Ethiopia	2	0.8264	20.2476		
trends	2	0.8264	21.074		
newspapers	2	0.8264	28.5116		

Table 3.2: An example of the most used tags selected for a user profile based on only tags that has recognized lexical class. Only co-tags that has a recognized lexical class and which has been used at least two times together are selected for a profile. As a comparison the result of co-tags with no lexical restriction, which has been used at least two times together are shown on the table. Tags in **Bold** typeface are the tags that were selected also for the profile on Table 3.1.

Tags are used to express a user interest in a user profile and they can be used to support searching and browsing content in Pharos. This was another reason to use only recognized tags for a profile. Because the tags will be utilized in other service than they were originally created the recognized lexical words will support information retrieval and keyword based search better. For selected tags also co-occurred tags that have recognised lexical class and which have been used at least two times together are included for a profile. Co-occurrences indicates a user's viewpoint to the topic described as a tag. Co-occurred tags could be utilised to widen the search terms to better find the content of user's interest and also to rank search results based on this information. For our first example user (table 2), there are not so many co-occurrences that have been used more than once. As a comparison, on table 3 one can see a part of the user profile for a user who has a lot of co-occurred tags. These co-occurred tags reveal quite well user's viewpoint to the topic that has been selected for a profile. We can utilise also information about the tags that has same word root for enhancing search terms and creating relations between tags. For our example user, words like search, searching and blog, blogs have same word root.

Tag	Tag usage frequency	Relative tag usage frequency	Relative cumulative tag usage frequency	Co tags (recognised lexical class)
Design	33	39,759	39,759	web, usability, research, tools, mobile, social, accessibility, html, development, tutorial, interface, marketing, communication, CSS, designer, business, learning, search, guidelines, community, elderly, standards, user, science, innovation, color, technology, analysis, advertising, coding, emotion, browser, email, information, documentation, UX, ethnography, architecture, website, furniture, tips, collaboration, markup, patterns, programming
Web	20	24,096	63,855	design, social, community, tutorial, usability, html, research, CSS, technology, mobile, collaboration, communication, business, internet, relationships, writing, tips, advertising, programming, software, content, standards, tagging, blog, interface, tools, analysis, culture, documentation, guidelines, coding, SMS, marketing, information, development, patterns, text, work
mobile	17	20,482	84,337	design, research, SMS, social, user, twitter, web, community, elderly, guide, guidelines, widget, ethnography, usability, cellphone
usability	16	19,277	103,614	design, UX, web, IA, research, accessibility, tool, user, tools, architecture, information, elderly, search, interface, marketing, communication, tips, mobile, advertising

Table 3.3: A part of the user's interest profile, based on all tags of the user. Bolded tags are the tags that were not listed on a user profile made based on recent tags of the user.

3.1.8 Optional features

Our initial idea is that the result of a user profile is shown to the user who can evaluate and modify it. This gives us various opportunities to let the user verify and modify the interest profile. We can improve the recognition of unrecognized tags, but not solve all of them. Because of that, one opportunity is that besides the selected tags for a profile also the most used

unrecognized tags are shown to the user. In this case, users can include them into the profile if they want to, but also correct misspelled words. For the above analysis, the recent tags of a user were used. In our dataset we also have a list of all the tags that the user has created and how many times each of them has been used. The plan was to use both recent and total tag usage for creation of profile. The same analysis as in table 2 is made for all users tags, except for the co-occurrence analysis. The co-occurrence data is not available for all user's tags, due to the limitation of the del.icio.us APIs. The list of selected tags from the user's recent and total tags is compared. The tags that are not listed in the initial profile, but are listed in a profile made based on all tags of the user are potential candidates to be included in a user's profile (see Table 4). These tags could be shown to a user and a user can decide whether she or he wants to include them into their profile of interest as well. The user is the only one who could indicate whether these tags are still a current interest of the user. In case of our example user, the tags like blog, mobile, metadata, copyright, internet, project, searching, educational, Africa would be suggested to be included in the user's profile. More evaluations need to be done with actual users.

Tag	Tag usage frequency	Relative tag usage frequency %	Cumulative relative tag usage frequency %
blog	56	3.0468	3.0468
Google	32	1.741	4.7878
participatory	31	1.6866	6.4744
advertising	28	1.5234	7.9978
newspapers	23	1.2514	9.2492
research	18	0.9793	10.2285
mobile	17	0.9249	11.1534
tagging	17	0.9249	12.0783
news	16	0.8705	12.9488
video	15	0.8161	13.7649
business	15	0.8161	14.581
metadata	14	0.7617	15.3427
identity	14	0.7617	16.1044
collaborative	13	0.7073	16.8117
conferences	13	0.7073	17.519
copyright	12	0.6529	18.1719
internet	12	0.6529	18.8248
communities	12	0.6529	19.4777
Yahoo	11	0.5985	20.0762
project	11	0.5985	20.6747
social	11	0.5985	21.2732
innovation	10	0.5441	21.8173
technology	10	0.5441	22.3614

searching	10	0.5441	22.9055
educational	10	0.5441	23.4496

Table 3.4: A part of the user's interest profile, where the user has high number of tags pr resource. **Bold** tags are the tags that were not listed on a user profile made based on recent tags of the user.

3.1.9 Suggestions for handling tags within Pharos

If the Pharos system supports tagging, following features should be considered for implementation in order to promote creating tags that can be easily utilized.

- support for creating tags consisting of more than one word (compound tag)
- support for tag categories (opinion, task organizing, topic) would help in the analysis of users' tags
- support for re-using one's tags (helps in minimizing typos and different variations of the same word)
- suggestions for tags

If there is tagging support within Pharos, the algorithm could be utilized directly to analyze these tags, but even then, a possibility to import tags from other systems could be useful.

3.1.10 Conclusions and future work

In our work, we created a user profile by analyzing user created tags in del.icio.us. It confirmed us that user's tags reflect the person's interest and are a good source for interest profile data. This approach would help users to express their interests without extra work.

Because of the nature of freely choosable words and different tagging habits among user, users' tag clouds are very heterogeneous. Tag clouds might be *messy* for other people than the user personally. We addressed this issue by using lexical analysis together with tag usage frequencies to clean user's tags and for selecting only the most used tags for a profile. Relative cumulative tag usage frequency was used to select threshold for tags which were included in a user's profile. This approach modifies threshold for different users by taking into account different tagging habits of users. We focused on English tags, lexical analysis need to be customized for different languages.

The user profile includes user's interests defined as collection of tags and their co-occurring tags with corresponding information about the tag and tag co-occurrence usage frequencies. Only the tags that have a known lexical class were selected for a profile.

Despite the use of lexical analysis, there were still many unrecognized tags and the analysis of unrecognized tags needs to be improved. This includes analysis of compound words as well as better analysis for synonyms and semantic meaning of a word.

Because the lexical analysis drops out tags that are not recognized, optionally users could be shown the most used unrecognized tags, so that a user can import them into the profile as well if they want to. More evaluation needs to be done with actual users to evaluate whether the created profile is acceptable for them and what needs to be improved. Anyway it is important that the user is given an opportunity to modify the profile.

The approach for taking account the recent and all tagging data of a user will be reconsidered when we have analyzed the available tagging data from other services as well. Another aspect relating to time sensitivity of tags is how the user profile is updated afterwards. A user's interest profile will be influenced by added tags in external services, as well as tags used in PHAROS itself. That needs to be addressed in future work.

Although the tags in del.icio.us do not necessarily relate straight to multimedia like music or movies, they describe user's interest areas. Most users are interested about available multimedia content relating to their interest areas as well, not only relating to their music or movie preferences. It would be interesting to harvest the tags of the user from different services, because depending on the scope of the service different interest areas of the user can be detected to complement the created profile. In future work we will test our algorithm with another dataset, for example tagging data from Flickr! or Last.fm. The methods for combining the results of different services for creating one combined user profile will be created.

For the PHAROS platform application, ser profile can be used for supporting browsing of content as well as for ranking search results based on user's interests. One opportunity is to show a user the search results found in PHAROS as well as content that user has tagged in other services with a same tag that was used as keyword for searching. Another way to utilize user's tags in PHAROSs is to give a user an opportunity to create RSS feeds based on user's interest expressed as tags in a user profile.

Besides the tags, information in user created playlists or content posted to different groups by a user would be a good source to enhance information in a user profile as well as metadata of content itself. This will be one of our future interest areas as well.

3.2 Information Diffusion In Blogosphere

The Internet has emerged as a vital medium for marketing and advertising by providing global exposure of information to large audiences worldwide at very low cost. According to Interactive Advertising Bureau (IAB) and Price Waterhouse Coopers (PWC), Internet advertising revenues for 2006 are estimated at 16.8 billion, a 34 percent increase over the previous revenue record of 12.5 billion in 2005 [83]. Many forms of online advertising and marketing have been developed $-$ keyword-targeted search engine advertising, permission email, floating animated page takeovers, interactive on-page rich media ads, to name a few [84]. Further,

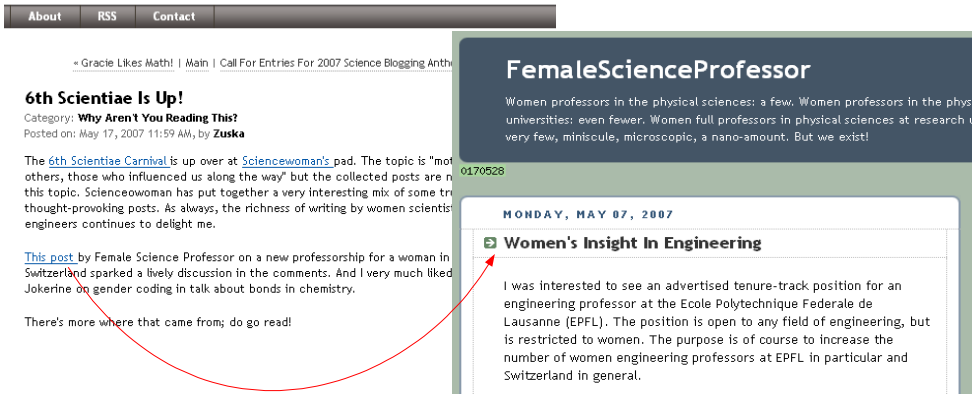

Figure 3.3: Example of a hyperlink relationship between blogs.

recent research has found a significant correlation between blog mentions and book sales [2], thus demonstrating that blogosphere is an interesting source for predictive models. In this context of trend prediction and trend analysis, it is important to study how information flows in blogosphere.

In this section, we focus on the problem of discovering *information diffusion paths* from blogosphere to capture the patterns of information flow through blogs. Based on the discovered knowledge, advertisers can be enlightened to select an optimal set of blogs as their target audience or to estimate the extent to which individuals will be influenced by a campaign.

How information is propagated in networks is an important problem and has been investigated in many areas. In the field of sociology, the study of the *diffusion of innovation* focuses on examining the role of *word of mouth* in spreading innovations [85]. In recommender systems, user access patterns are leveraged to model information flow and generate effective personalized recommendation [86]. Trust propagation is an active research issue which models the propagation of trust scores and predict trust and distrust between users based on trust propagation models [87]. In this paper, we aim to discover information propagation in blogspace in terms of blog sequences, called *information diffusion paths*. Each path indicates that a sequence of blogs frequently propagating information sequentially[3].

One way of observing information propagation between blogs is based on hyperlinks, e.g., when a post (from one blog) provides a hyperlink to another post (from another blog). As shown in Figure 3.3, the hyperlink relationship between two blogs indicates that the information from the right blog flowed to the left one. Such hyperlink-based blog networks have been studied in [88] to capture the activity bursts within web communities. However, a blogger who read a previous post from another blog may write a post on a similar topic without citing the previous one. Hence, by discovering information diffusion paths from the blog network formed by explicit hyperlinks, one may not be able to discover all information diffusion paths. In our

[3] In this paper, the notions of "information flow", "information diffusion" and "information propagation" are used alternatively.

Table 3.5: Topic blog sequence database.

Topic	*Blog Sequence*
1	$\langle(b_1, 03/05/07), (b_2, 04/05/07), (b_3, 06/05/07)\rangle$
2	$\langle(b_2, 09/05/07), (b_3, 12/05/07)\rangle$
3	$\langle(b_1, 15/05/07), (b_2, 17/05/07)\rangle$
4	$\langle(b_3, 12/05/07), (b_2, 18/05/07), (b_4, 21/05/07)\rangle$
5	$\langle(b_1, 19/05/07), (b_2, 25/05/07), (b_4, 27/05/07)\rangle$

work, we instead discover information propagation patterns by analyzing the content of posts and tracking the topics among blogs.

3.2.1 Problem Definition

Some preliminary definitions are necessary before formally defining the problem of mining information diffusion paths from blogosphere, which we will give in the following paragraphs.

A blog community collected in a given time period $[t_s, t_e]$ is a set of n blogs $\Omega = \{b_1, b_2, \ldots, b_n\}$. Each blog $b = (p_1, p_2, \ldots, p_m)$ contains a set of published posts, where each post is associated with a publishing time point, $T(p_i)$, such that $t_s \leq T(p_i) \leq t_e$.

Since we aim to discover how information flows from blog to blog by analyzing content instead of hyperlinks of posts, we make a *closed world assumption*, similar to [85], that in a given blog community, all posts on a topic except the first one are the result of communication within the community[4]. Given a particular topic, we can retrieve a sequence of posts from different blogs, $\langle p_1, p_2, \ldots, p_k\rangle$, talking about the same topic sequentially, e.g., $\forall i \in [1, k), T(p_i) \leq T(p_{i+1})$. Let $B(p_i)$ be the blog to which the post p_i belongs. Given a post sequence on some topic, we can get a corresponding blog sequence $\langle B(p_1), B(p_2), \ldots, B(p_k)\rangle$. Given the closed world assumption described above, such a sequence indicates that a piece of information flowed from $B(p_1)$ through $B(p_k)$. Formally, given some particular topic, the corresponding *topic blog sequence* is defined as follows:

Definition 1 $[Topic\ blog\ sequence]$ *Given a particular topic* c*, a topic blog sequence* $Q(c) = \langle(b_1, t_1), (b_2, t_2), \ldots, (b_k, t_k)\rangle$*, is a list of blog-time pairs such that each blog* b_i *publishes a post on the topic* c *at time* t_i*. Moreover,* $\forall i \in [1, k), t_i \leq t_{i+1}$.

[4]This assumption can be weakened by increasing the thresholds of *support* and *strength* defined in Section 3.

Given a blog community collected in a certain time period, suppose there exist k topics in the set of all posts of the blog community, we can then get k topic blog sequences as above. Thus, the blog community can be modeled as a *blog sequence database* as follows.

Definition 2 $[Blog\ sequence\ database]$ *Given a blog community collected in time period* $[t_s, t_e]$*, which contains a set of blogs* $\Omega = \{b_1, b_2, \ldots, b_n\}$ *and a set of posts on* k *topics* $\Gamma = \{c_1, c_2, \ldots, c_k\}$*, it can be modeled as a blog sequence database* D *in the form of* $(i, Q(c_i))$*, where* i $(1 \leq i \leq k)$ *is the identity of a topic and* $Q(c_i) = \langle (b_1, t_1), (b_2, t_2), \ldots, (b_m, t_m) \rangle$ $(c_i \in \Gamma, b_i \in \Omega, t_s \leq t_i \leq t_e)$ *is a topic blog sequence.*

Table 3.5 shows an example blog sequence database. The first entry indicates that blogs b_1, b_2 and b_3 have published posts on the topic c_1 on May 3rd, May 4th, and May 6th in 2007[5].

A blog sequence $S = \langle b_1, b_2, \ldots, b_m \rangle$ is an ordered list of blogs. Different from a topic blog sequence, each element of a blog sequence is an individual blog instead of a blog-time pair. We say a topic blog sequence $Q(c_i) = \langle (b_1^i, t_1^i), (b_2^i, t_2^i), \ldots, (b_m^i, t_m^i) \rangle$ *supports* a blog sequence $S = \langle b_1^j, b_2^j, \ldots, b_n^j \rangle$, denoted as $Q(c_i) \sqsupseteq S$, if $\exists\ v\ (1 \leq v \leq m - n + 1)$ such that $b_1^j = b_v^i, b_2^j = b_{v+1}^i, \ldots, b_n^j = b_{v+(n-1)}^i$. For example, the blog sequence $S = \langle b_1, b_2 \rangle$ is supported by three topic blog sequences, $Q(c_1), Q(c_3)$ and $Q(c_5)$, in Table 3.5.

Given a blog sequence database, if a blog sequence is frequently supported by topic blog sequences, this indicates that information is usually propagated through this sequence of blogs. Hence, we define the *support* measure of a blog sequence to reflect how frequently it is supported by a blog sequence database.

Definition 3 $[Support]$ *Given a blog sequence database* D *and a blog sequence* S*, the support of* S *with respect to* D*, denoted as* $Supp_D$*, is the fraction of topic blog sequences in* D *which support* S*.*

$$Supp_D(S) = \frac{|\{Q(c_i) | S \sqsubseteq Q(c_i)\ \&\ Q(c_i) \in D\}|}{|D|}$$

where $|D|$ *is the total number of topic blog sequences in the database.*

The support measure takes on values from 0 through 1. The more topic blog sequences supporting a blog sequence, the higher the support value of the blog sequence. For example, let $S = \langle b_1, b_2 \rangle$. The support of S with respect to the database in Table 3.5 is 0.6 because it is supported by three topic blog sequences.

The support of a blog sequence reflects how frequently the information flows through the sequence. In order to evaluate how quickly the information flows, we define a *strength* measure for a blog sequence. Note that, a blog sequence might be supported by multiple topic blog sequences, where each has different time intervals between successive blogs. For example, the blog sequence $\langle b_1, b_2 \rangle$ is supported by $Q(c_1), Q(c_3)$ and $Q(c_5)$ in Table 3.5. In $Q(c_1)$, it takes one day to propagate the information from b_1 to b_2. However, it takes

[5] In this paper, we use time points with a granularity of one day for simplicity. Any other time granularity can be used similarly.

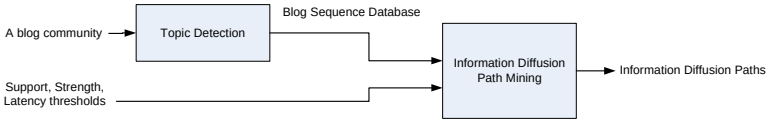

Figure 3.4: The framework of information diffusion path mining.

2 days in $Q(c_3)$ and 6 days in $Q(c_5)$. We define the *latency* of a topic blog sequence $Q = \langle(b_1, t_1), \ldots, (b_m, t_m)\rangle$, denoted as $Lat(Q)$, as the total time intervals between successive blogs $\sum_{1 \leq i < m}(t_{i+1} - t_i)$. We defined the *matched topic blog subsequence* of a topic blog sequence, $Q(c_i) = \langle(b_1^i, t_1^i), (b_2^i, t_2^i), \ldots, (b_m^i, t_m^i)\rangle$, with respect to a blog sequence, $S = \langle b_1^j, b_2^j, \ldots, b_n^j\rangle$, as $M(Q, S) = \langle(b_v^i, t_v^i), (b_{v+1}^i, t_{v+1}^i), \ldots, (b_{v+(n-1)}^i, t_{v+(n-1)}^i)\rangle$, if $\exists\, v\ (1 \leq v \leq m-n+1)$ such that $b_1^j = b_v^i, b_2^j = b_{v+1}^i, \ldots, b_n^j = b_{v+(n-1)}^i$. For example, given $Q(c_1)$ in Table 3.5 and the blog sequence $S = \langle b_1, b_2\rangle$, the matched topic blog subsequence $M(Q(c_i), S) = \langle(b1, 03/05/07), (b2, 04/05/07)\rangle$. Then, the strength of a blog sequence can be defined as follows.

Definition 4 $[Strength]$ *Given a blog sequence database D, some latency threshold δ, and a blog sequence $S = \langle b_1, b_2, \ldots, b_m\rangle$, the strength of the sequence, denoted as $Streg_{D,\delta}(S)$, is*

$$Streg_{D,\delta}(S) = \frac{\{Q | Lat(M(Q, S))) \leq \delta\ , S \sqsubseteq Q, Q \in D\}}{\{Q | S \sqsubseteq Q, Q \in D\}}$$

That is, the strength of a blog sequence is the fraction of supporting topic blog sequences which have the latency of their matched topic blog subsequences less than or equal to the given latency threshold. For example, let the latency threshold $\delta = 3$. The strength of $S = \langle b_1, b_2\rangle$ with respect to the database in Table 3.5 is $2/3$ because it is supported by three topic blog sequences (i.e., $Q(c_1)$, $Q(c_3)$, and $Q(c_5)$) and two of them (i.e., $Q(c_1)$ and $Q(c_3)$) have the latency of matched topic blog subsequences less than the given threshold. The strength value of a blog sequence ranges from 0 to 1. If all supporting topic blog sequences have their latency of matched topic blog subsequences no greater than the specified threshold δ, the strength of the blog sequence is 1. If none of the matched topic blog subsequences has the latency less than or equal to the threshold δ, the strength of the blog sequence is 0.

Given a blog sequence database, we are interested in blog sequences which are not only supported frequently by the database but also supported frequently by topic blog sequences which propagate information quickly. Thus, we define an *Information Diffusion Path* (IDP) as a blog sequence satisfying constraints specified as follows.

Definition 5 [*Information Diffusion Path*] *Given a blog sequence database D, a support threshold α, a latency threshold δ and a strength threshold β, a blog sequence S is an Information Diffusion Path (IDP) if* 1) $Supp_D(S) \geq \alpha$ *and* 2) $Streg_{D,\delta}(S) \geq \beta$.

For example, given the support threshold $\alpha = 0.4$, the latency threshold $\delta = 3$ and the strength

threshold $\beta = 0.6$, the blog sequence $S = \langle b_1 b_2 \rangle$ is an IDP since $Supp_D(S) = 0.6 > \alpha$ and $Streg_{D,\delta}(S) \approx 0.67 > \beta$.

Then, the problem of *information diffusion path mining* can be formally stated as follows. Given a blog sequence database D, a support threshold α, a latency threshold δ and a strength threshold β, the problem of information diffusion path mining is to discover the set $\{S | Supp_D(S) \geq \alpha \ \& \ Streg_{D,\delta}(S) \geq \beta\}$. In the rest of this paper, we omit the subscripts D and δ if they are clear from the context.

Let us point out that there are two fundamental differences between information diffusion path mining and classical frequent sequential pattern mining.

- Information diffusion path mining is more strict in defining the support relationship between sequences. In classical frequent sequential pattern mining, a sequence $\langle b_1, b_2, b_3 \rangle$ supports $\langle b_1, b_3 \rangle$ too. In information diffusion path mining, the sequence supports $\langle b_1, b_2 \rangle$ and $\langle b_2, b_3 \rangle$ only.
- In classical frequent sequential pattern mining, there is only one constraint on the support value of sequences. In contrast, we have two constraints in information diffusion path mining, i.e. the support value and the strength value of blog sequences.

Due to the differences, existing algorithms for frequent sequential pattern mining cannot be used directly to mine information diffusion paths. We thus developed a new algorithm for information diffusion path mining, which is described in the next section.

3.2.2 Discovery of Information Diffusion Paths

The framework we proposed for mining information diffusion paths from blogosphere is shown in Figure 3.4. Given a blog community collected in a given time period and a set of thresholds as the input, we need to identify a set of topics from the collection of posts in the blog community first. Then, the blog community is transformed to be a blog sequence database. Next, we mine information diffusion paths from the database with respect to given thresholds. The output is the set of desired information diffusion paths satisfying the thresholds. In the following, we will explain the two main components of the framework, *topic detection* and *information diffusion path mining*, in turn.

Topic Detection

As discussed, we track the information flowing from blog to blog based on the topics of posts. Given a blog community with a collection of posts, we need to detect the topics from the posts. There exist a number of algorithms in detecting topics from a text corpus, such as LSA [89] and pLSA [90]. We employ Latent Dirichlet Allocation (LDA) [91], which is a probabilistic generative model, to detect topics from post collections. LDA is not prone to overfitting and can be generalized easily to new documents. The number of parameters of the LDA model

does not grow with the size of the corpus. Briefly, given k topics (k can be determined by cross-validation), the probability of the ith word in a given post is formulated as

$$P(w_i) = \sum_{j=1}^{k} P(w_i|z_i = j)P(z_i = j)$$

where z_i is a latent variable indicating the topic from which the ith word is drawn. $P(w_i|z_i = j)$ is the probability of the ith word given the jth topic. $P(z_i = j)$ is the probability of choosing the jth topic for the post. The parameters θ_j^w, the probability of using word w in topic j, θ_j^p, the probability of topic j in post p, can be estimated with Gibbs Sampling [86]. Interested readers are referred to [91] for the details of LDA.

After detecting k topics from the post collection, each post is assigned k values which are the probabilities that the post belongs to the topics $c_1, c_2, \ldots, c_k$. We group each post to the cluster for which it is assigned the highest probability. Given a cluster, we retrieve the blog identities and time points of each post to generate a topic blog sequence. Based on this, a blog community can be represented as a blog sequence database.

IDP Mining

We now describe how to mine information diffusion paths from a blog sequence database with respect to user specified thresholds. Particularly, we designed a new data mining approach called *IDP-Miner* based on *FP-Growth* [92] and *FS-Miner* [93], which are algorithms for frequent itemset mining and frequent sequence mining respectively. The reason why we develop *IDP-Miner* based on the two algorithms is that, similar to frequent itemset mining and frequent subsequence mining, the *downward closure* property [92] holds for the support measure defined for blog sequences.

Basically, both *FP-Growth* and *FS-Miner* construct a special tree structure to register the compact information of the database for frequent pattern mining. However, since IDPs are defined differently from classical frequent itemsets or sequences, the data structures constructed by the two algorithms do not preserve sufficient information for IDP mining. Hence, we designed a new data structure, *IDP-Tree*, which not only registers the sequential information but also the temporal information of blog sequences.

IDP-Tree

An IDP-Tree is composed of two components: a tree structure and a header table.

- A tree structure with a special root node R and a set of blog sequence prefix subtrees as children of the root. Each node in the tree structure has a ***node-name*** b_i that represents a blog from the blog sequence database. Each edge $\langle b_i, b_j \rangle$ in the tree structure corresponds to matched topic blog subsequences of topic blog sequences with respect

to a blog sequence $S = \langle b_i, b_j \rangle$. Each edge is associated with two fields ***edge-strength*** and ***edge-link***. *Edge-strength* is a vector of groups of latencies of the matched topic blog subsequences represented by the edge. Note that, the number of total elements in the vector corresponds to the number of topic blog sequences that support the blog sequence $S = \langle b_i, b_j \rangle$ in the particular tree path. *Edge-link* is a pointer which points to the next occurrence of the edge $\langle b_i, b_j \rangle$.

- A header table maintains three fields: ***edge-name*** $\langle b_i, b_j \rangle$ stores the name of an edge in the tree structure, ***edge- count*** stores the number of topic blog sequences supporting the blog sequence represented by this edge, and ***edge-head*** is a pointer which points to the first occurrence of the edge in the tree.

For example, an example IDP-Tree is shown in Figure 3.5. The algorithm of constructing an IDP-Tree is described in Figure 3.6. We illustrate the algorithm by giving an example of constructing an IDP-Tree for the database in Table 3.5. Suppose the support threshold α be 0.4. A blog sequence $\langle b_i, b_j \rangle$ is a potential IDP if it is supported by no less than $\alpha \times |D|$=$0.4 \times 5 = 2$ topic blog sequences. Hence, we scan the database for the first time to find the set of $\langle b_i, b_j \rangle$ such that each is supported by no less than 2 topic blog sequences. For example, $\langle b_1, b_2 \rangle$, $\langle b_2, b_3 \rangle$, and $\langle b_2, b_4 \rangle$ are found because they are supported by 3, 2 and 2 topic blog sequences respectively. Then, we create an entry for each $\langle b_i, b_j \rangle$ in the header table, which is shown in the left part of Figure 3.5.

Then, we scan the database for the second time. For each topic blog sequence, we check whether it supports any blog sequences in the header table. If yes, we insert the matched topic blog subsequences into the tree structure. For example, consider the first topic blog sequence in Table 3.5. Since it supports $\langle b_1, b_2 \rangle$ and $\langle b_2, b_3 \rangle$, the matched topic blog subsequences are inserted as follows. When inserting $\langle (b_1, 03/05/07), (b_2, 04/05/07) \rangle$, since there exists no child node b_1 of the root R, we create the node b_1. Similarly, we create the node b_2 as a child of b_1. Next, for the edge $\langle b_1, b_2 \rangle$, we associate a vector of edge-strength with the first element of the first group as the latency of the matched topic blog subsequence, which is 1. Then, we link the edge-head of $\langle b_1, b_2 \rangle$ in the header table to this edge. The matched topic blog subsequence $\langle (b_2, 04/05/07), (b_3, 06/05/07) \rangle$ is inserted under the current node. The other topic blog sequences can be inserted similarly. The constructed IDP-Tree is shown in Figure 3.5.

Note that, the edge-strength of the edge $\langle b_1, b_2 \rangle$ in Figure 3.5 is a vector containing three groups of latencies. The first two groups contain respective latencies of the current matched topic blog subsequence, when it is followed by the two child nodes in some topic blog sequences. The last group contains the latency of the current matched topic blog subsequence when it is not followed by any other matched topic blog subsequence in some topic blog sequence (e.g., $Q(c_3)$). Hence, each edge $\langle b_i, b_j \rangle$, where b_j has k child nodes, is associated with an edge-strength which contains either k or $(k + 1)$ groups of latencies.

The major difference between IDP-Tree and FP-Tree [92] is as follows: FP-Tree is a tree structure constructed for frequent itemset mining. Hence, the header table maintains information about frequent individual items (blogs). Differently, we record information about frequent blog sequences $\langle b_i, b_j \rangle$ in the header table. The reason is that if there exists no frequent blog se-

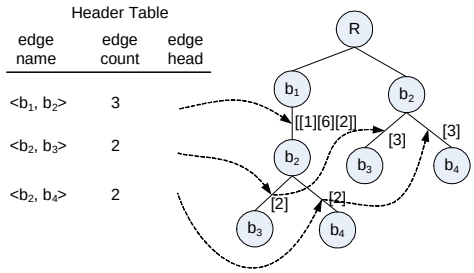

Figure 3.5: An example IDP-Tree.

Input: A blog sequence database D, support threshold a,
Output: An IDP-Tree

Discription:
1. Scan D to find S = <b_i, b_j> s.t. Supp(S) >= a
2. For each S, create an entry in the header table
3. Create a root node R
4, For each blog sequence Q_i in D
call InsertTree(R, Q_i)
4. Return IDP-Tree

Function: InsertTree(N, <b_1, b_2, …, b_m>)
1. if <b_1, b_2> exists in the header table
2. if N has a child node C s.t. C.node-name = b_1
append a latency to N-C.edge-strength
3. else
create a node C with C.node-name = b_1
create a vector as N-C.edge-strength
link N-C to the edge-head in the header table
4. if <b_2, …, b_m> is not empty
call InsertTree(C, <b_2, …, b_m>)

Figure 3.6: IDP-Tree construction algorithm.

quence $\langle b_i, b_j \rangle$, we do not need to construct the data structure any more even if there exist frequent individual blogs.

IDP-Tree is mainly different from FP-Tree [92] and FS-Tree [93] on the following aspect: FP-tree/FS-Tree only registers the number of times an item (blog) or an edge (blog sequence) is supported. However, we maintain not only the number of times a blog sequence is supported, but also the latency of supporting topic blog sequences so that the strength value of a blog sequence can be obtained from the data structure as well.

IDP-Miner

After constructing the IDP-tree, we can mine IDPs with respect to a set of given thresholds from the structure directly without referring to the blog sequence database anymore. The algorithm of mining IDPs from the IDP-Tree is shown in Figure 3.8. We illustrate the algorithm

by showing an example mining procedure from the IDP-Tree in Figure 3.5. Basically, there are three steps involved in the mining procedure.

Extracting Related Paths. Given an entry $\langle b_i, b_j \rangle$ in the header table, all occurrences of the edge in the tree can be extracted by following its edge-head in the header table and corresponding edge-links in the tree. For each edge connected by edge-links, we extract the path from the root to the current edge. For example, consider the last entry in the header table in Figure 3.5, $\langle b_2, b_4 \rangle$. By following the corresponding edge-head and edge-links, we can extract two paths: $(\langle b_1, b_2 \rangle : [[1][6][2]], \langle b_2, b_4 \rangle : [2])$ and $(\langle b_2, b_4 \rangle : [3])$. (The numbers after the colons are the associated edge-strengths.) However, not all the information in the extracted paths are useful for mining IDP involving $\langle b_2, b_4 \rangle$. For example, the edge-strength of $\langle b_1, b_2 \rangle$ in the first path contains three groups of latencies which correspond to the latencies when it is followed by $\langle b_2, b_3 \rangle$, $\langle b_2, b_4 \rangle$ and empty respectively. Since b_4 is the second child of b_2 in the path, we extract the second group of edge-strength of $\langle b_1, b_2 \rangle$. Then, the first path turns out to be $(\langle b_1, b_2 \rangle : [2], \langle b_2, b_4 \rangle : [2])$. It can be observed from the path that $\langle b_1, b_2 \rangle$ occurs before $\langle b_2, b_4 \rangle$ in one topic blog sequence because now the size of each edge-strength in the path is one. And the values of the edge-strengths indicate that it takes 6 days for the information to be propagated from b_1 to b_2 and 2 days from b_2 to b_4.

Constructing Conditional IDP-Tree. After extracting related paths for a blog sequence $\langle b_i, b_j \rangle$, we construct a conditional IDP-Tree to mine IDPs involving $\langle b_i, b_j \rangle$. Basically, this can be done by inserting the extracted the paths into a tree structure in a backward manner. We create necessary nodes and edges and share them when possible. For example, Figure 3.7 shows the conditional IDP-Tree constructed for $\langle b_2, b_4 \rangle$. The edge-strength of each edge is created similarly.

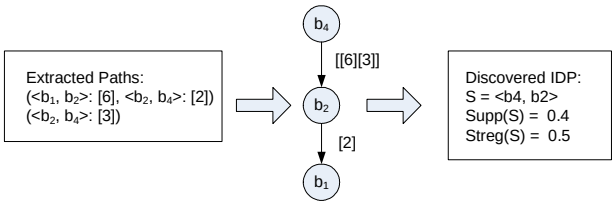

Figure 3.7: Conditional IDP-Tree of $\langle b_2, b_4 \rangle$.

Discovering IDP. Given a conditional IDP-Tree of a blog sequence $\langle b_i, b_j \rangle$, IDPs involving the blog sequence can be discovered by performing a depth-first traversal of the conditional IDP-Tree. While traversing the tree, we compute the support and strength of derived blog sequences. For example, we firstly derive the blog sequence $S_1 = \langle b_2, b_4 \rangle$ by traversing the tree (Note that, the edge $\langle b_4, b_2 \rangle$ represents the blog sequence $\langle b_2, b_4 \rangle$ since we construct the conditional IDP-tree by inserting related paths in the backward manner). The support of S_1 can be computed from the size of edge-strength associated with the edge $\langle b_4, b_2 \rangle$. Hence, $Supp(S_1) = 2/3 \approx 0.67$. When computing the strength of the blog sequence, we need to convert the edge-strength to a bitmap. For each element in the vector of edge-strength, if it is no greater than δ, the bit is set as 1. Otherwise, the bit is zero. For example, suppose

the threshold δ is 3, the bitmap corresponds to the edge-strength of $\langle b_4, b_2 \rangle$ is $\langle 01 \rangle$. Then, $Streg(S_1) = 1/2 = 0.5$. Suppose the strength threshold β is 0.5, $S_1 = langleb_2, b_4\rangle$ is an IDP.

Then, we further traverse the tree to derive the blog sequence $S_2 = \langle b_1, b_2, b_4 \rangle$. Since the size of the edge-strength of $\langle b_2, b_1 \rangle$ is 1, $Supp(S_2) = 1/5 < \alpha$. Hence, it is not an IDP. Since all nodes in the conditional IDP-tree is traversed, the mining procedure for this conditional IDP-tree stops. Note that, if the support of S_2 satisfies the threshold, we need to compute the strength of S_2. In order to do that, we should accumulate corresponding elements of edge-strengths in a path to obtain the correct latency of a matched topic blog subsequence. For example, we add the first element of edge-strength of edge $\langle b_4, b_2 \rangle$ with the edge-strength of edge $\langle b_2, b_1 \rangle$ to get the latency 8. Then, the corresponding bitmap is $\langle 0 \rangle$ and the strength of S_2 is zero.

Input: IDP-Tree, support threshold 0, strength threshold 0,
latency threshold 0
Output: a set of IDPs I

Description:
1. Set an empty set I
2. For each blog sequence S_i in header table
3. Extract all paths reachable from S_i edge-head
4. Extract useful edge-strength values for all paths
5. Construct conditional IDP-Tree from extracted paths
6. Perform depth-first traversal on the conditional IDP-Tree to mine IDPs
7. I = I 0 discovered IDPs
8. Return I

Figure 3.8: The algorithm of IDP-Miner.

3.2.3 Performance Study

In this section, we conduct experiments to evaluate the performance of the developed *IDP-Miner* algorithm in Section 3.2.2 and the effectiveness of information diffusion paths in Section 3.2.2. We implemented our approach in the Java. All experiments were run in a Windows environment using a PC with a 2.00GHz Pentium processor with 2.00GB of RAM. In our experiments, data collected from the DailyKos blogspace (http://www.dailykos.com) was used. The data set consisted of $3,563$ time-stamped blog posts authored by a total of 300 authors. All blog posts are published in March, 2006.

Performance of IDP-Miner

We first conduct experiments to evaluate the efficiency of *IDP-Miner*. We respectively generate 300, 400 and 500 topic clusters from the set of posts. Then, three corresponding blog sequence databases with different numbers of topic blog sequences can be obtained. We mine *IDPs* from the three database by varying the support threshold. The experimental results are

shown in Figure 3.9. Firstly, it can be observed that when the support threshold increases, *IDP-Miner* is more efficient in discovering the set of *IDPs*. The reason of this observation is that when the support threshold is getting greater, fewer blog sequences satisfy the threshold. Hence, it takes less time to discover *IDPs*. Secondly, we observed from the figure that *IDP-Miner* works faster on the database with more topic blog sequences. The reason is that given the fixed number of blog posts, the more clusters are generated, the shorter each topic blog sequence is. Consequently, fewer recursions will be run by *IDP-Miner* and more efficiency can be gained. Thirdly, we noticed that when the support threshold is set around 0.6%, the running time on databases with 300 and 400 topic blog sequences degrades sharply. After inspecting the generated *IDPs*, we realized that this situation is caused by significantly reduced number of *IDPs* that satisfy this threshold. We did not show the efficiency of *IDP-Miner* with respect to the variation of the strength threshold and the latency threshold because *IDP-Miner* does not narrow the search space based on the two thresholds. Hence, the efficiency of *IDP-Miner* does not vary obviously along with the variation of the two thresholds.

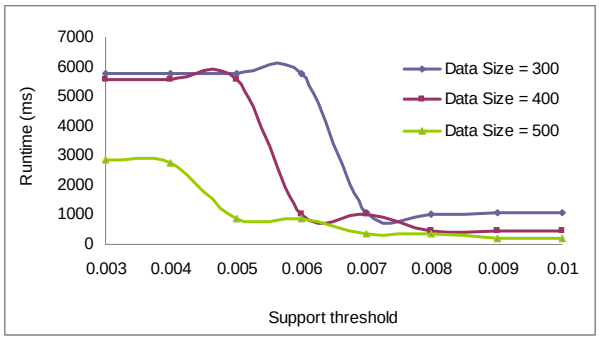

Figure 3.9: Efficiency of IDP-Miner.

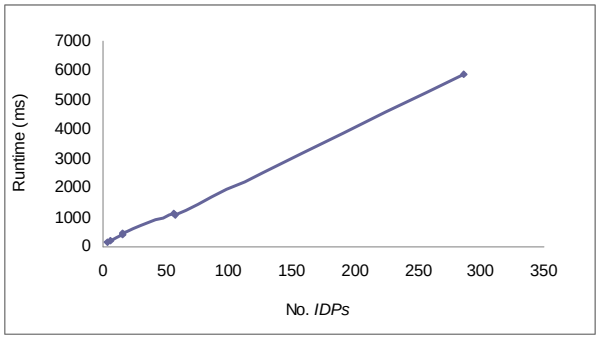

Figure 3.10: Scalability of IDP-Miner I.

We then conduct experiments to evaluate the scalability of *IDP-Miner*. Figure 3.10 shows the scalability of *IDP-Miner* with respect to the number of *IDPs*. We observed that *IDP-Miner*

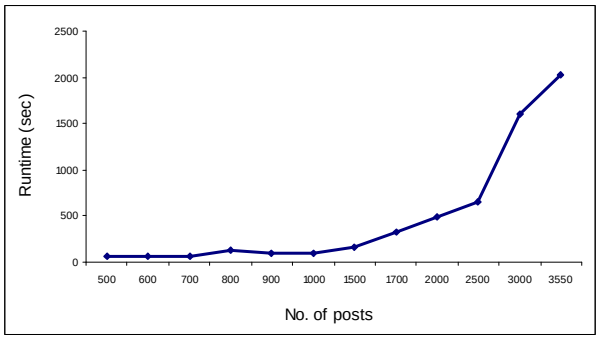

Figure 3.11: Scalability of IDP-Miner II.

scales well along with the increase of the number of discovered *IDPs*. We further examine the scalability of *IDP-Miner* with respect to the number of blog posts. The results are shown in Figure 3.11. Again, *IDP-Miner* exhibits quite good scale-up feature.

Performance of IDP

We also conduct experiments to study the effectiveness of *IDPs* mined from blogosphere. We generate a blog sequence database from the dataset with 500 topic blog sequences. Then, we randomly select 80% of the topic blog sequences to mine *IDPs*. The remaining 20% topic blog sequences are used to test the accuracy of discovered *IDPs*. We use the metric *hit ratio* to compute how many blog sequences in form of $\langle b_i, b_j \rangle$ in the test dataset are predicted correctly by discovered *IDPs*. Note that, each discovered *IDP* may be a sequence of blog sequences with more than two blogs. We separate each *IDP* into blog sequences in form of $\langle b_i, b_j \rangle$. Then, a correct prediction means a blog sequence $\langle b_i, b_j \rangle$ occurred in both $IDPs$ and test dataset. The experimental results are shown in Figure 3.12. We observed that *IDPs* mined from the content of blogs have good accuracy with high hit ratios. The more number of *IDPs* are discovered, the higher hit ratio can be achieved.

We intended to compare the accuracy of *IDPs* mined from the content of blog posts against *IDPs* mined from blog sequences formed from explicit hyperlink among blog posts. However, we observed very limited hyperlink relationship between blog posts. We then resorted to the sequential relationship between commenters on each post. For example, each post's commenter were ordering in ascending order according to the commenting time. It yielded 2,777 sequences of commenters. However, we discovered that there are only 30 blog sequences in form of $\langle b_i, b_j \rangle$ which occur twice in the set of $2,777$ sequences. No blog sequence that occurs more than twice, which means the dataset is too sparse to be used as a competitor. Hence, we believe it is more appropriate to discover information diffusion patterns in blogosphere based on content analysis.

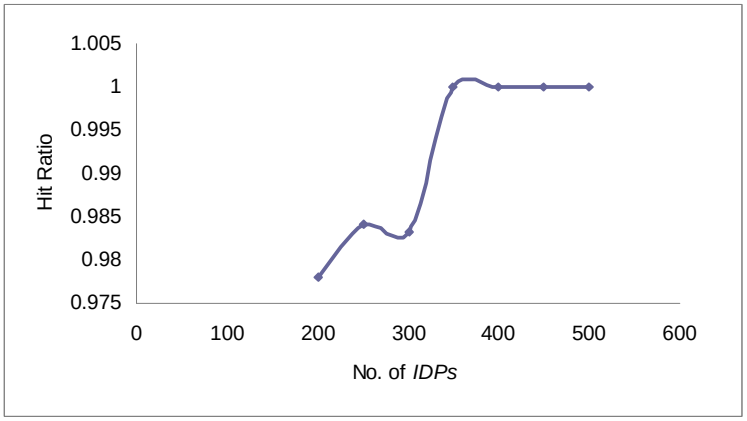

Figure 3.12: Hit ratio of IDPs.

3.2.4 Related Work

In this section, we briefly review related work from the two areas of information propagation and frequent sequential pattern mining.

As mentioned earlier, information propagation is well researched in a lot of areas such as social network, personal recommendation and trust scoring etc. In social network, the spread of a piece of information is viewed as the propagation of innovation. Two fundamental models are proposed in the literature to model the process by which nodes adopt innovations: threshold model [94] and cascade model [95]. Song et al. [86] leveraged the access patterns of users to model the information flow and generate effective personnel recommendation. Basically, they considered the temporal order of information adoptions made by users and computed a Markov chain based model. Then, the probability of information flow between two users are calculated by assuming a uniform distribution on the probability of each possible path between two users. Although weblog community is getting more and more popular, it is surprising that only a limited work on discovering information propagation was done in the blogosphere. One of the most similar work to ours is done in [85]. They also analyzed the content of posts and model the collection of weblogs at two levels. One is a macroscopic characterization of topic propagation which models the topics of posts as long-running *chatter* topics with multiple *spike* topics. The other is a microscopic characterization of propagation from individual to individual by using the theory of infectious diseases. Thus, the task of the second level is partially similar to the objective our work. However, they did not take into account the time spent on propagating information from blog to blog. Kumar et al. [88] analyzed the evolving link structure of blogspace to capture the bursts of activity within blog communities. Their work is thus totally different from ours.

Frequent sequential pattern mining is one of the major problems of association rule mining. There exists a number of algorithms for frequent subsequence mining; the interested reader

can refer to [96] for a detailed survey on association rule mining.

3.2.5 Conclusions and Future Work

In this section, we proposed to discover information diffusion paths from blogosphere by analyzing the content of blog posts. We modeled the task as a problem of frequent pattern mining by representing a blog community collected in a certain time period as a blog sequence database. Then, we define interestingness measures for blog sequences and define Information Diffusion Paths (IDP) base on the measures. Particularly, an *IDP* is defined as a sequence of blogs that frequently discuss similar topics sequentially and frequently discuss similar topics around similar time points. We proposed a framework for discovering information diffusion paths from blogoshpere and designed a new algorithm for information diffusion path mining. We also discussed the potential applications of information diffusion paths in online advertising. Our preliminary experiments showed promising results that the developed *IDP-Miner* algorithm is efficient in discovering *IDPs* and discovered *IDPs* are accurate in predicting the future information flow in the blog community.

Our ongoing work includes that discovering information propagation from blogosphere by analyzing both the content and the hyperlinks in a blog community, considering that blogs connected by hyperlinks more likely adopt information from each other. We are also interested in applying our work in other applications such as blog recommendation.

3.3 Opinion analysis in user-created textual contents

The Internet is more than ever an information source of high importance. With Web 2.0 and an increasing number of community-based sites, online users share more and more their opinions and preferences on all and sundry. This new and reachable information offers an enormous opportunity for services eager to better satisfy their users. By learning users' expectations and needs, services have the opportunity to adapt as well as possible their services or products, especially in the Personalization and Recommendation context.

Web sites like the foreseen PHAROS platform need detailed user profiles to recommend good multimedia contents to their customers; however, in practice, people do not declare very precise interests. Therefore, it is difficult for an automatic recommender to make accurate predictions, especially for new users[6]. When user profiles and the content descriptions are too vague to be matched, recommendation quality is poor and thus satisfying a broad user-base is a real challenge. To complement *explicit* knowledge provided by the users themselves, systems observe users and record interaction data (e.g. logs, queries, clicks, or any feedback on the contents consumption) in order to build *implicit* profiles. Such implicit knowledge is used by statistical or machine learning techniques, such as Collaborative Filtering. But even if these

[6] This is called the *cold start* problem.

techniques become more and more robust to the *cold start problem*, we are convinced that understanding the broader usage patterns on the Internet and applying this knowledge to a particular set of users within a particular website (e.g. PHAROS platform) will help in providing a better matching between users and items.

What is proposed here is to capture users' preferences in user-created content such as movies reviews, blogs or natural language annotations about multimedia contents. We focus on a method which analyzes the natural language contents provided by a community site gathering movie fans (source *www.flixster.com*); from their reviews, we extract vocabulary dedicated to express and share sentiment. With such a dictionary and our home-made tools, we aim to build *opinion* profiles describing both people and communities. The challenge will be to use this knowledge in the particular context of Pharos platform: we propose to reuse our opinion prediction system on Pharos users blogs/textual content to add new rated items to their profiles.

For the content acquisition, we focused ourselves on the Flixster website. *www.Flixster.com* is a community site offering Web 2.0 functionalities. Movies fans find news, trivias, movie descriptions, and also many places where they can share their opinions about films and actors. It gathers currently approximately $23,000,000$ users, and $33,000$ movie cards that any user can rate and comment. Each registered user has a personal homepage, where he can describe his preferences and create lists of favorite films and actors. Users can also create a social network with other registered users and communicate with them.

Here are three screenshots of *www.flixster.com*. The homepage (see Fig. 3.13), an example of movie page (see Fig. 3.14) and an example of user page (see Fig. 3.15).

To acquire data, we first have used *wget*. We have then obtained a seed of 50,000 user homepages. By using users' identity on these pages, we have crawled reviews and social network of approximately 10,000 users which gave us 360,000 reviews, 9,209 movie cards and 4,983 actor cards to analyze. The reader may refer to 3.16 to get a global understanding of the whole interconnected data.

In the current study, we focus on the reviews. We point here that each review is also rated by the author according to a 5 stars scale $0, ..., 5$. These ratings will be used to evaluate the predictions made automatically by our tool. We will see that predicting such precise rates are non trivial, and is a challenge for humans as well (see [97]). From those rates, we reclassify reviews into 2 classes: *negative* and *positive*.

3.3.1 Review analysis using NLP approach

The final aim of our presented analysis is to identify the opinion expressed in the textual part of a review. Our approach consists in comparing the two techniques on our corpus and design a method to predict opinion (positive or negative).

Figure 3.13: Flixster homepage (source *www.flixster.com*)

The NLP techniques are based on a dictionary issued from a manual selection of words revealing opinions. We have tested two dictionaries: one is issued from a very general lexicon, General Inquirer (containing 4207 words). The other dictionary (150 words) has been elaborated from a frequency analysis of our own corpus: we have kept words appearing often and attribute manually an opinion class; we have added an English words set already classified by Stone et al. [98] and Kelly and Stone [99].

To analyze textual comments, we use a NLP tool named TiLT. TiLT (*Traitements Linguistiques de Textes*) is a text analyzer developed within France Telecom R&D in the framework of NLP research.

Textual corpus The textual corpus represents data to be analyzed. For us, textual corpus is a set of movie reviews. Let us note that the English used is not the one we learn at school... The reviews are very similar to forum messages. They present common characteristics such

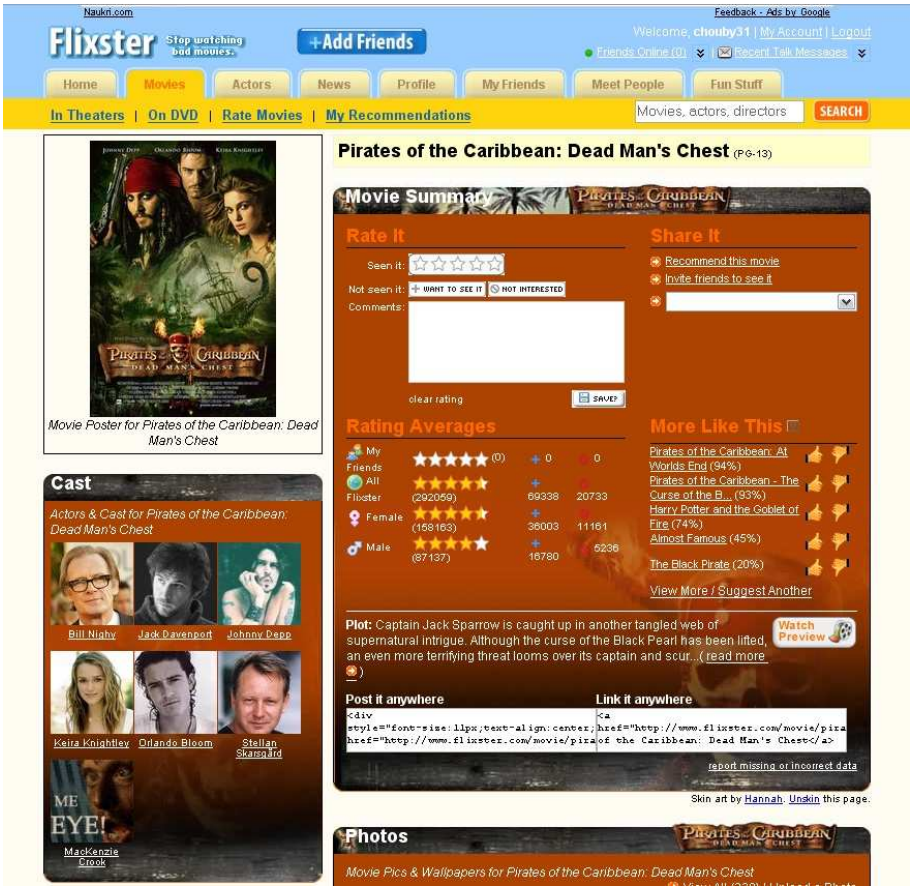

Figure 3.14: Page of movie (source *www.flixster.com*)

as:

- Accumulation of the punctuation: “”!!! ”
- Use of smiley: “ :-) ”
- Particular writing SMS type: “ ur ” , “ gr8 ” (“ your ” , “ great ”)
- words stretching: “ veryyyy coooool ”

One of the positives using TiLT is to help in resolving typos, understanding abbreviations, smileys, and so on.

Figure 3.15: Page of user (source *www.flixster.com*)

The text analyser TiLT

This tool articulates itself around a set of modules implementing the advanced techniques of natural language processing (see figure **??**). It has various functionalities which range from language recognition (which determines choice of lexical, grammar etc.), then continues with text cutting in increasingly fine elements, with lexical analysis and parsing to finish with semantic analysis. As of now, eight languages are covered at different levels: French, English, German, Spanish, Arabic, Polish, Portuguese and Chinese.

Rule Segmentation TiLT requires the following data during the segamenation: sentences are cut out into segments according to segmentation rules which were indicated (which can be points, commas, brackets . . .). Each segment is constituted of one type and one character string. The type allows to direct posterior treatments carried on this segment. Here is an

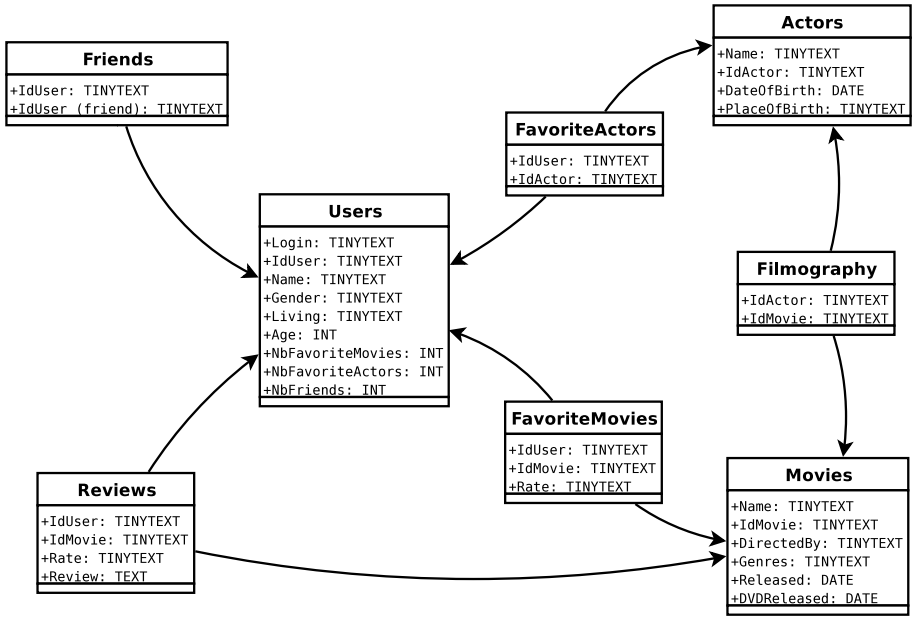

Figure 3.16: Data base of Flixster data

example of results of this step:

“ *The company increased its figure by 1%* ”
(WORD. “ The ”)(SPACE. “ ”)(WORD. “ company ”)(SPACE. “ ”)(WORD. “ increased ”) (SPACE. “ ”)(WORD. “ its ”)(SPACE. “ ”)(WORD. “ figure ”)(SPACE. “ ”)(WORD. “ by ”)(SPACE. “ ”)(NUMBER. “ 1 ”)(PERCENT. “ % ”)

Each type of words or symbols (like “ % ”) are found using segmentation data, which are obviously modifiable. This step result is independent of all grammatical concept or sentence meaning.

Monolingual lexicon There exists one lexicon by language. Each lexicon is composed of several data files indexing simple words and locutions in their canonical form. They comprise mainly morphosyntaxic information as identifiers of meaning which allow to do the bond between the lexical entry contents on the lexicon and semantic data of thesaurus or as grammatical features.

Grammatical features A grammatical feature is a grammatical and linguistic indication allowing to describe in details words of a language. Each word can have several grammatical features that is why features generally depend on the context of the sentence. TiLT allows to allot new grammatical features to words from lexicon.The analysis produces we can for

example add opinion polarity to words carrying opinion.

Minimal analysis This step consists to apply to each identified segment a certain number of actions, according to its type, in order to associate to it lexical interpretations which correspond to it. With the precedent example, TiLT does not go in the lexicon strings which are typified NUMBER or PERCENT, it uses to lexicon only for strings typified WORD. If a word is unknown by the lexicon, different correction methods can be applied to it. In addition, a morphological analysis mechanism can be also called to supplement word analyze or word correction. It is important to notice that the use of correction methods on unknown word can associate to a single segment multiple lexical forms.

The corrections applied for the project are:

- Phonetic correction
- Typographical correction
- Morphological correction
- MorphoPredictive correction

If one correction type gives a solution, further corrections are not applied.

Lemmatisation Lemmatisation is the process of determining the lemma for a given word. The words of a language use several forms according to their grammatical gender (masculine or feminine), their number (plural or singular), their grammatical person (you, he, she ...), their grammatical mood (subjunctive, indicative ...) thus giving rise to several forms for a same word. Lemmatisation of words is done in the following way:

- For a verb: the lemma is the bare infinitive
- For other words: the lemma is the word in the singular and masculine

For example, the verb " to walk" may appear as " walk" , " walked" , " walks" , " walking" . The base form, " walk" , that one might look up in a dictionary. The sentence " *Depp's films are soooo funnyyyy, i adored them!*" gives, after lemmatisation, " *depp 's film be so funny , I adore them*" .

Chunking The chunking allows associating each word to a terminal. A terminal is a word informing about the grammatical role. Some words may firing several terminals. For example, the word " love" can get terminals " love/NOUN" and " love/VERB" . The choice of the good terminal is done according to the context of the sentence. With this labeling tool, we have the possibility to study adverbs, verbs and adjectives separately.

Dependency Analysis This analysis could be very interesting when we want to study adverb-adjective combinations (Benamara et al. [45]). Syntactic analysis in dependence allows to organize the words of the sentence previously cut out by analysis in chunking by constituting word groups as nominal groups or verbal groups. It allows also to clarify relations between words in these groups (subjects, complements etc.).

In TiLT, syntactic dependency analysis is represented in the form of tree (see figure 3.17). The arborescent representation of the figure 3.17 allows us to see that the head of the phrase is the conjugated verb (GV-PT) “ has” to which is attached a subject of personal pronoun type (PRN-S) then a complement of object formed by a determinant (GN-DI) “*a*” and a noun (GN-NC) “ *problem*”.

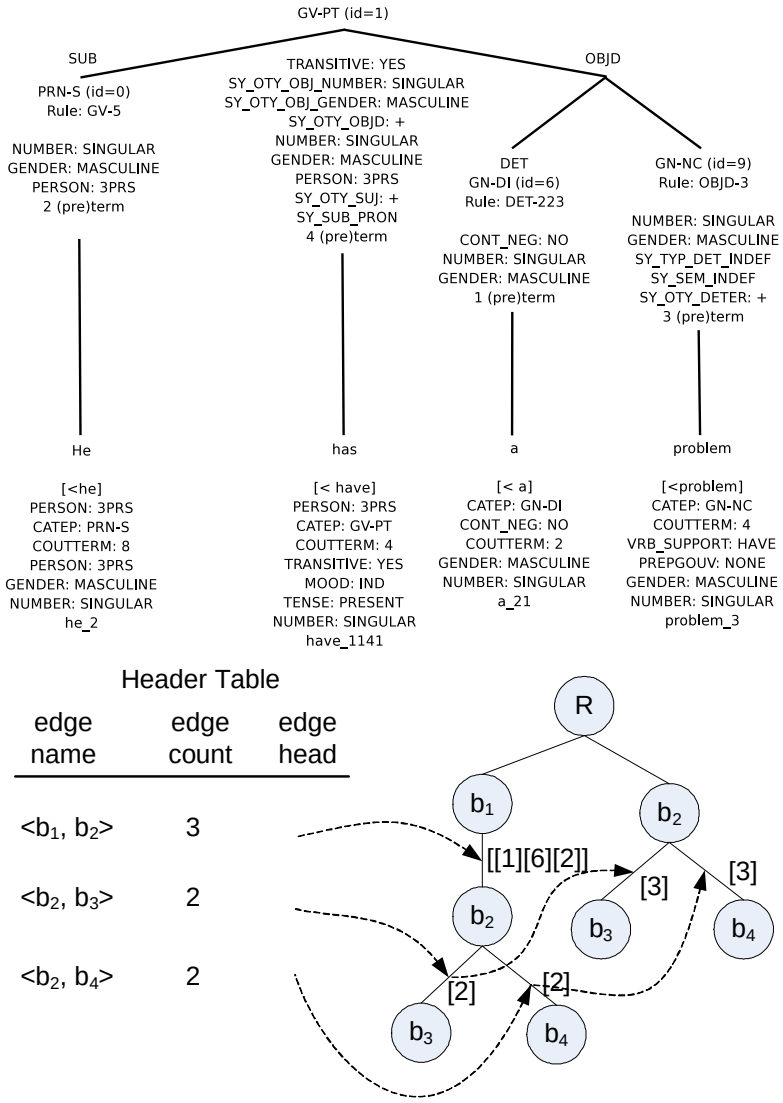

Figure 3.17: Analysis in dependence of the phrase “ He has a problem”

Dependency Grammar allows to give complementary information for each analyzed element. This information presents itself in form of morphological, syntactic and semantic grammatical features, and can be used in rules of dependence allowing to constitute word groups.

Semantic analysis with the thesaurus The semantic treatment is the last link in the chain of treatment. It concentrates information available in other modules to allow us *to understand* the text entered in the analyzer. Semantic data are gathered in the thesaurus which interacts with the lexicon to create predicates which will be used in semantic graphs. The thesaurus

is a multilingual catalog of lexicons where the objective is to accumulate information on the different meanings of words and on relations that maintain these meanings between them.

Our Proposed Opinion Mining Method

From our initial corpus, we keep a part for testing (30%), and use the rest to develop our training model (70%). In order to build a dictionary, we used TiLT to show the words and their frequency.

First, we classify each review according to its rating. We thus have created one file with all reviews rated with 1 star, one file with reviews rated with 2 stars and so on. We lemmatise each files and filtered words to be returned according to their grammatical category (Adjectives, adverbs and verbs). We then classify manually terms according to the opinion that they express and we have increase this list by using a synonyms dictionary (also manually). We kept words expressing only one type of opinion.

We have classified 221 words in 11 classes that we have identified:

- Adverbs:
  - Affirmation (absolutely, entirely, fully . . .)
  - Doubt (possibly, apparently, seemingly . . .)
  - Weak intensity (really, very, so, much . . .)
  - Strong intensity (only, a little, nearly, barely . . .)
  - Negation and minimizers (not, never, less, no . . .)
- Adjectives:
  - Satisfaction (good, great, funny, awesome . . .)
  - No satisfaction (bad, wicked, stupid, fucking, ugly . . .)
  - Provisional satisfaction (specially, disturbing, surprising, special . . .)
- Verbs:
  - Positive opinion (love, good mood, amaze, enjoy, recommend . . .)
  - Negative opinion (hate, disturb, fuck, shit, dislike, overeat . . .)
  - Provisional opinion (remember, feel, believe . . .)

In classes *Provisional satisfaction* and *Provisional opinion*, there are terms yielding opinions not semantically sufficient, i.e. which cannot be determined by themselves if the opinion is a positive or negative opinion. It is necessary to have other elements to determine opinion polarity of these words. We will see afterwards how machine learning techniques can help (see Sec. 3.3.2).

The next step of the analysis consists in marking the corpus. For each word in the corpus

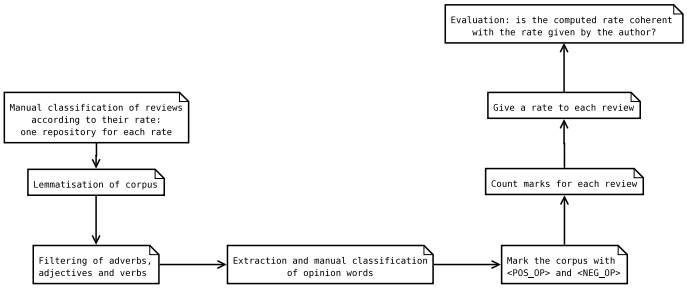

Figure 3.18: All steps for NLP approach

identified in the classified words set, the word class is added in the text (in a markup language way). At this stage of our research, we have used only two marks, **POS_OP** for positive opinion and **NEG_OP** for negative opinion, and also, we decided not to use adverbs which accentuate opinion transmitted by adjectives, adjectives of provisional satisfaction and verbs of provisional opinion.

The results are phrases of type: " *a <POS_OP>great<POS_OP> movie fill with all the <POS_OP>good<POS_OP> thing you <POS_OP>like<POS_OP>* !"

The last step consists in counting for each review, the number of marks **POS_OP** and **NEG_OP** in order to give a rate to the review.

These steps are indexed in figure 3.18.

The next step is the application of dependency rules, grouping, for example, adjectives and adverbs. Semantic interpretation of theses groups regarding expressed opinion has to be done. For example, when an adverb of negation class is found, the opinion of adverb-adjective combination is the reverse of the opinion expressed by the adjective which follows the adverb. In order to improve the classification process of words we will try to integrate and fusion the results of different methods stated in the state of art (see for example the study of adverb-adjective combinations in Benamara et al. [45]).

		With our dictionary (150 words)		With General Inquirer dictionary (4 207 words)	
		Number of reviews	Accuracy	Number of reviews	Accuracy
Rated reviews		46 013	0.66	53 594	0.77
Well rated reviews	All (70000)	39 537	0.56	40 875	0.58
	POS (52363)	36 806	0.70	34 942	0.67
	NEG (17637)	2 731	0.15	5 933	0.34

Figure 3.19: Results of the NLP method using 2 different lexicons

The results on table 3.19 show too many reviews in which no firing-word had been detected. We have to use a more exhaustive list of classified words according to expressed opinions.

With **positive** reviews, our method predicts a positive rate with an accuracy of 70%. Special improvements has to be made on the **negative** words.

In order to be more language-independent and more exhaustive, we show in the next section that machine learning techniques help in discovering the informative words used in the reviews and their associated expressed opinion.

3.3.2 Opinion analysis using machine learning techniques approach

For this machine learning techniques approach, we have done two distinct analysis. The first, using a supervised algorithm, allows to find the most informative words in term of opinion mining. For t*his analysis, we have used algorithms integrated in a tool named KHIOPS* developed by Boullé [100] at France Telecom R&D.

KHIOPS

KHIOPS is a tool allowing both supervised and unsupervised learning. It allows to perform univariate and bivariate descriptive statistics, to evaluate the predictive importance of explanatory variables, to discretize continuous variables, to group the values of categorical variables, to recode input data according to these discretization and value groupings.

This tools is used in the following way:

- The first step consists in building the dataset and choose the variables describing the data. Variable may be a categorical (male/female for example) or continuous (a real number). KHIOPS proposes an automatic layer discovering the variables after loading the data to analyze. In our case, the data are the reviews, and the variables are the presence/absence of each word extracted in the corpus. From this step, a database file is built.
- The second step is to check the correctness of the database file. In this step, the tool parses the database file and completely checks formatting or variable type errors.
- The third step, the most important, is to analyze the predictive value of the explanatory variables or pairs of variables. In supervised analysis, a target variable must be specified and KHIOPS evaluates the predictive importance (named “ level” in the results) of each variables. The predictive importance of a variable is a rate. The higher the rate, the better the variable allows to predict the target variable value. And conversely, more the rate is low, less the variable allows to predict the target variable value. In unsupervised analysis, there is no target variable specified and KHIOPS evaluates the correlation between any pair of variables. In the supervised mode, our target variable is the rate given by the review author.

Film	Notation		great	movie	i	loved	it	was	awsome
Troy	5	1	1	1	0	0	0	0	0
Memoirs of a Geisha			5	0	0	2	1	1	1
The Bourne Identity			5	0	0	0	0	0	0
Crash (2005)	5		0	1	1	0	0	0	0
Pirates of the Caribbean			5	0	1	3	0	2	
The Wedding Planner			5	0	0	2	0	1	0
Charlie and the Chocolate Factory				5	0	0	0	0	
Tim Burtons Corps Bride				5	0	0	0	0	0
Underworld: Evolution				5	0	0	3	1	1
Freedomland	5		0	1	3	1	2	1	1
Angel Eyes	5		0	0	2	0	0	0	1

Figure 3.20: Part of file for supervised learning method

Learning methods: Supervised vs. Unsupervised

Supervised learning is a machine learning technique for creating a function from training data. The training data consist of pairs of input objects (typically vectors), and desired outputs. The output of the function can be a continuous value (called regression), or can predict a class label of the input object (called classification). The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples (i.e. pairs of input and target output)[7].

For this method, KHIOPS uses the naive Bayes classification approach with selection of variables and average of models [101]. The supervised learning method consists, in our case, in learning which *variables*, i.e. words, are significant to predict the rate of the comment.

For this method, KHIOPS takes in entry a file describing a matrix (see figure 3.20).

Each line of this matrix represents a review about the film quoted in the first column. The second column contains rates of each reviews given by the author and to be predicted by the algorithm. Other columns contain the number of occurrences of each word.

The result of the process is the selection of informative words associated to each rating. Those informative words can then be used by the NLP process to classify a text in a rating set. By finding the informative words, KHIOPS builds a internal model up to classify new reviews.

The first experiment used 5389 reviews with 7087 different words. 70% have been used to train the classifier and 30% for the evaluation. On this dataset, KHIOPS selected 36 informative words. But results show that each word taken separately does not bring much information, and thus would not be so relevant to add to the NLP dictionnay. This statement explains the quite low classification accuracy (53%) obtained on the evaluation dataset.

As a first conclusion, we state that more data is required for better results (to get much more words and then extend the vector describing the reviews). For our first analysis, reviews were rated in 5 categories ([0, 1, ..., 5]). After splitting the reviews into two categories (Reviews with a rate higher or equal to 3 are positive, others are negative), results are improved (see Table

[7] Source: `http://en.wikipedia.org/wiki/Supervisedlearning`

	With 5 classes	With 2 classes (POS, NEG)
Accuracy	0.53	0.92
Informative variables	36	25

Figure 3.21: Results of our supervised learning method

Rank	R0002					
Variable	Continuous	love				
Discretization						
Value	1	2	3	4	5	Frequency
]-inf; 0.5[	0.0411728	0.0499064	0.130381	0.299231	0.478894	4809
[0.5; inf[	0.00689655	0.0137931	0.0603448	0.213793	0.705172	580
Total	0.0374838	0.0460197	0.122843	0.290035	0.503247	5389

Figure 3.22: Statistics of the word " love"

3.21), and we reach an accuracy of 91,5%.

Here is the list of the 36 most informative words, i.e. those which allow to predict a rate. They are listed from the more informative to the least one:
best, stupid, worst, ok, my, hated, okay, freak, word, inte, looks, love, so, seen, pretty, its, ever, loved, gr8, c, make, this, needs, such, picture, censored, babes, hop, tryin, bitching, moviee, each, married, kill, hip, breasted

For each informative word, KHIOPS returns detailed statistics. It gives the percentage for which the word is present or not in reviews rated one star, in reviews rated two stars and so on. For example, we can see that for all reviews where the word "love" is present, 70.5% had been rated five stars, 21.4% had been rated four stars, 6% had been rated three stars, 1.4% had been rated two stars and 0.7% one star (see figure 3.22. We can thus say that a review containging " love" has got more than 9-in-ten chances to be positive (three, four or five stars)(see the red framework on Fig. 3.22).

We conducted another experiment with $70,000$ reviews but processing time is much longer and KHIOPS is not adapted to the huge volume of variables describing the data (i.e. 90 000 words). KHIOPS has been designed for classical data mining analysis where data are described with few variables (e.g. salary, age etc.). Here, we face a text mining problem involving many descriptors. Thus, our second series of tests was not conclusive. But this experience shows us that bigger the dataset is, more informative words will be found, and more relevant they will be.

3.3.3 Conclusion and perspectives

First results of our experiments are encouraging but many problems still have to be solved.

The main problem is the data quantity. KHIOPS can not treat a corpus with 70,000 reviews.

To reduce the matrix (i.e. the number of the words describing each review), we can lemmatise the corpus, and correct many spelling mistakes and word abbreviations with TiLT. For example words as “ cooool” , “ Cool” , “ kool” can be reduced to the single word “ cool” . Applying lemmatisation by using NLP techniques reduce drastically the number of words to take into account.

The state of the art in Opinion prediction shows that combining Natural Language Processing techniques and Machine Learning techniques offer the best results. With the method and the tool used in our experiment, we can conclude that to predict the sentiment polarity (positive or negative opinion) regarding a textual comment about a movie, a supervised learning tool based on Naive Bayesian algorithms reach more than 91% of accuracy without any pre-treatment on the textual corpus.

We are therefore confident that NLP tools are necessary to improve the results:

- to lemmatise and go beyond the volume barrier: to train the machine learning algorithms with a greater lexicon (with 70,000 reviews, we count more than 90,000 different words),
- to clean the corpus and remove syntactic liberties users take when they write comments on multimedia content,

reduce the volume of variables

We still have to test if machine learning techniques, by stressing the informative words, can improve the NLP method consisting in counting opinion words and defining an heuristic to classify a review into a positive or negative class. The dictionary we used has been manually done, adding the informative words may offer better results to the NLP method. But this method, more complicated to use, is language dependent, which is not the case for our ML method.

In a different perspective of research, such as “ understand how” people like or dislike a movie, “ what or why ” they like or dislike a movie, then, machine learning will be useless. NLP ones, by analyzing the structure of the sentence will catch patterns such as “ <POS_OP>love/VERB<POS_OP> the image quality/OBJECT ”.

Finally, in the context of recommendation problems, our work can be used to label social networks with opinion. Our ML method is accurate enough to get precise opinion communities of people sharing the same interests in movies. By analyzing the metadata describing those movies, we will be able to generalize knowledge, and for example, learn what is common in the films appreciated by Johnny deep’s films, beyond the easy deduction that his presence is enough. Such studies will converge to fine communities profiles and fine recommendation taking into account polarity of opinion.

4 Algorithms For Personalization and Recommendation

4.1 Using User Generated Metadata For Music Recommendation

More and more companies start offering personalized services toward their users and online music recommender systems are one prominent example. Pandora[1], Last.fm[2], Foafing the Music[3] or Yahoo! Music[4] are a few of the currently online available music recommendation systems. These systems employ different approaches for recommending music tracks to their users, ranging from content based and collaborative filtering techniques to hybrid methods. While clearly useful to their users, these more conventional recommendation techniques still suffer from a number of problems: in the case of collaborative filtering, musical pieces with no ratings cannot be recommended because recommendations are based on actual user ratings. Besides, artist variety in recommended pieces can be poor, making these recommendations less satisfactory than they could be. Recommending tracks that are similar to users' favorites in terms of content induces unreliability in modeling users' preferences and besides, content similarity does not necessarily reflect preferences. Hybrid recommendation techniques combine the advantages of the two approaches and are thus better. However, to our knowledge, the only usable music recommender system using hybrid techniques is Foafing the Music, which relies heavily on FOAF profiles created by the users - not an easy task for non-expert users. Last, but not least, though many of these community sites allow tagging, these tags are not used for recommendation or any form of advanced search.

4.1.1 Related Work

Most available music recommender systems are based on collaborative filtering methods; i.e., they recommend music to a user by considering some other users' ratings for the same music pieces. This technique is quite widely utilized, including music shopping services like

[1] http://www.pandora.com/
[2] http://www.last.fm/
[3] http://foafing-the-music.iua.upf.edu/
[4] http://music.yahoo.com/

Amazon[5] or iTunes[6], and has proven to be effective. However, this recommendation method suffers from the *cold start* problem.

A better recommendation technique is described in [102]. The paper gives an overview of the Foafing-the-Music system, which uses the Friend-of-a-Friend (FOAF) and Rich Site Summary (RSS) vocabularies for recommending music to a user, depending on her musical tastes. Music information, such as new album releases, related news artists, or available audio pieces, is gathered from RSS feeds from the Web, whereas FOAF documents are used to define user preferences (i.e., for building the user profiles). Another hybrid music recommendation method is presented in [103], which simultaneously considers user ratings and content similarity and is based on a three-way aspect model, so that it can directly represent substantial (unobservable) user preferences as a set of latent variables introduced in a Bayesian network. Probabilistic relations over users, ratings, and contents are statistically estimated. Our approach differs from these previous ones by the fact that the user profile is inferred automatically from his desktop music data without any additional manual effort from the user's side, and that we rely on tag information instead of track-based profiles.

A totally different approach for producing music recommendations is presented in [104]. Their method is applied to an interactive music system that generates playlists fitting the preferences indicated by the user. For automatically generating music playlists, their approach uses a local search procedure in the solution space, based on simulated annealing: the algorithm iteratively searches the solution space moving from one solution to a neighboring solution, compares their quality and stops when an optimal solution is found.

[105] describes a system that queries web search engines for pages related to artists, downloads the pages, extracts text and natural language features, and analyzes these features in order to produce textual summary descriptions of each artist. These descriptions are used to compute similarity between artists and can be further used for producing recommendations. However, the paper does not present any evaluation experiments regarding the quality of the recommendations received by using this technique. Also, our approach differs from the one presented in [105] by the fact that they are searching the web for finding similar artists, whereas we rely on user tags (in particular from the Last.fm web site) for grouping tracks and artists and for building up user profiles.

The most similar approach to ours is [106], which uses collaboratively created data from the Web for making recommendations. However, their goal is generating personalized tag recommendations for users of social bookmarking sites such as Delicious[7]. Techniques for recommending tags do already exist and are based on the popularity of tags among all users, on time usage information, or on simple heuristics to extract keywords from the URL being tagged. Their approach complements these techniques and is based on recommending tags from URLs that are similar to the one in question according to two variants of cosine similarity metrics. We use tags as well, but use them for recommending interesting songs based on tag-based profiles.

[5] http://www.amazon.com/
[6] http://www.apple.com/itunes/
[7] http://del.icio.us/

4.1.2 Tag-Based Profiles vs. Track-Based Profiles

Tags are more and more widely used, which makes it easier to extract them for all types of information objects, including music tracks. We chose to extract available human annotations (tags) from Last.fm tracks or directly use those tags that the users applied themselves to other tracks or artists. Although most music recommendation methods use track-based algorithms to present the user with new interesting tracks, given the increasing tendency toward tagging all types of multimedia files on the Internet we wanted to investigate how these tags could be used for recommendations. We also wanted to avoid extensive manual ranking, so opted to construct user profiles from locally stored MP3 files. This usually works very well, as most users have quite a few music files in MP3 format on their desktop, usually much more than needed for a music recommender system to provide satisfactory results.

For the rest of this section, we will distinguish between profiles created for Last.fm users and for Non-Last.fm users. They differ in terms of the source of information which is used for building up the profiles: for Last.fm users the starting point is represented by their web pages on Last.fm, whereas in the case of the Non-Last.fm users we start from the information available on their desktops. For this type of users we extract metadata about each track existing on the desktop, and match artist and track name (extracted from filename or ID3 tag) against the Last.fm music database. This, as we will show in section 4.1.4 provides all data necessary to create comprehensive user profiles that accurately reflect users' music preferences. We will use the following notations:

$ITF(TG)$ = *Inverse Tag Frequency for Tag* TG

$p(TR, U)$ = *Preference of User* U *for Track* TR

$p(TG, U)$ = *Preference of User* U *for Tag* TG

$TR\_U\_Listened$ = *Number of times User* U *has listened to Track* TR

$TR\_Overall\_Listened$ = *Number of times Track* TR *was overall listened on Last.fm*

$TG\_UsedFor\_TR$ = *Number of times Track* TR *was tagged with Tag* TG *by all users*

$TG\_Used\_Overall$ = *Number of times Tag* TG *was used overall*

TG_UsedBy_U = *Number of times User* U *has used Tag* TG

$Tracks\_Containing\_TG$ = *Number of tracks on the User's Desktop that were tagged with Tag* TG

We distinguish between Last.fm and Non-Last.fm users using either *Last.fm* or *Non-Last.fm* indices in the corresponding formulas.

4.1.3 Track-based Profiles

Track-based profiles are defined as collections of music tracks with associated preference scores, describing users' musical tastes, as follows:

$$Profile\_Tracks(U) = \{< TR_i, P_i > \mid TR_i = \text{user's track}, P_i = p(TR_i, U)\}$$

Track-based profiles for Last.fm users

In the case of Last.fm users the profiles are inferred from the users' web pages available on the Last.fm site. Their collection of tracks includes all tracks the users have been listening to inside the system. Their associated scores are a function of the number of times users have listened to these music tracks. The algorithm for creating the track-based profiles for this type of users is described below:

Alg. 4.1.3.1: Track-based profile for Last.fm user

1: For each track TR in user's tracks list UTR
2: **Compute track's score** P
3: **Add** pair $< TR, P >$ to user profile
4: Return user's track-based profile

with $P = p(TR, U_{Last.fm}) = log(TR\_U_{Last.fm}\_Listened)$

Track-based profiles for Non-Last.fm users

For Non-Last.fm users the only available source of personal information is represented by their desktops. We first extract explicit metadata such as artist and track name either from the filename or from the ID3 tags (if any) of the music files existing on the desktop. This information is then matched against the Last.fm database and only tracks with a TFxIDF score above 0.9 are kept for further processing. This pre-processing step is described below:

Alg. 4.1.3.1: Get list of tracks

1: For each track (MP3) on user desktop
2: **Extract** artist name and track name from filename as *S1*
3: **Extract** artist name and track name from ID3 tag as *S2*
4: **Combine** *S1* and *S2* into *S*
5: **Search** with *S* on Last.fm index
6: **Retrieve** tracks *LT* matching query with Lucene TFxIDF score >0.9
7: **Add** tracks *LT* to the user's list of tracks UTR
8: Return UTR

Once the list of tracks for a Non-Last.fm user is created, the track-profile is realized in a similar manner as for a Last.fm user: algorithm 4.1.3.1 is applied on the list of tracks with the only difference that the preference scores for the tracks are now a function of the overall number of times tracks have been listened on Last.fm.

Alg. 4.1.3.2: Track-based profile for Non-Last.fm user

1: Create list of tracks UTR applying **Alg. 4.1.3.1**
2: Apply Alg. 4.1.3.1 on list of tracks UTR

with $P = p(TR, U_{Non-Last.fm}) = log(TR\_Overall\_Listened)$

4.1.4 Tag-based Profiles

Tag-based user profiles are defined as collections of tags together with corresponding scores representing the user's interest in each of these tags. The formal definition is given below:

$$Profile\_Tags(U) = \{< TG_i, P_i > \mid TG_i = \text{user's tag}, P_i = p(TG_i, U)\}$$

Again, we distinguish between profiles created for Last.fm and non-Last.fm users. For this type of profiles, the list of tags can be extracted either from the users' list of tracks (tags which have been used to tag the tracks) or directly from the tags the users have used themselves. The different variants of tag-based profiles for Last.fm and Non-Last.fm users are described in detail in Sections 4.1.4 and 4.1.4.

Tag-based profiles for Last.fm users

For Last.fm users, the first type of tag-based profiles can be created starting from the list of tracks the users have been listening to on Last.fm. For each of these tracks, we extract the list of all tags which have been used inside the system for annotating them. In this case, the preference associated to a tag is proportional to the number of times these tracks tagged as *TG* were listened by the user and to the number times this tag has been used by all users on Last.fm to tag those tracks. The description of the algorithm is given below:

Alg. 4.1.4.1: Track-Tag-based Profile for Last.fm User

1: For each track TR in user's track list UTR
2: **Extract** list of used tags TTG for TR
3: **Add** TTG to user's tag list UTG
4: For each tag TG in UTG
5: **Compute** tag's score P
6: **Add** pair $< TG, P >$ to user's profile
7: Return user's tag-based profile

---

with $P = p(TG, U_{Last.fm}) = [ITF(TG)] \cdot$
$log \sum_i (log(TR_i\_U_{Last.fm}\_Listened) \cdot log(TG\_UsedFor\_TR_i))$

$ITF(TG) = log \frac{\sum_i TG_i\_Used\_Overall}{TG\_Used\_Overall}$

Similar to the IDF value in Information Retrieval, in order to reduce the influence of tags which are very popular among users, but might not accurately reflect user's personal musical taste, we introduce an optional parameter in the preference formula, ITF. The formula penalizes tags which appear very often and boosts the preference for tags appearing more rarely.

The second possibility for creating the tag-based profiles for Last.fm users is to directly take the tags which the users have already used (information which can be found on their web pages) together with their frequency. The algorithm in this case looks as follows:

Alg. 4.1.4.2: Tag-based Profile for Last.fm User

1: For each tag TG in user's tag list UTG
2: **Compute** tag's score P
3: **Add** pair $< TG, P >$ to user's profile
4: Return user's tag-based profile

---

with $P = p(TG, U_{Last.fm}) = log(TG\_UsedBy\_U_{Last.fm})$

For this case, we do not need to introduce the ITF parameter in the preference formula, since now the profile is already very personalized – the user has directly used these tags.

Tag-based profiles for non-Last.fm users

For non-Last.fm users, the collection of tags building up their profiles is inferred based on the list of tracks the users have on their desktops. This list of tracks is compiled as presented in Alg. 4.1.3.1 and then transformed using Alg. 4.1.4.1. Preference scores for the tags are now computed as follows: the score depends on the number of times these tracks which are part of the users' profile and are tagged as *TG* have been listened by all Last.fm users and to the number times this tag has been used by all users on Last.fm. Again in the formula we have the optional parameter ITF used to decrease the bias towards very popular tags. Moreover, for Non-Last.fm users we keep in the profile only the top 100 preferred tags, after evaluating recommendation results with several such values for this algorithm ranging from 10 to 500.

Alg. 4.1.4.1: Track-Tag-based Profile for Non-Last.fm User

1: Create list of tracks UTR applying **Alg. 4.1.3.1**
2: Apply Alg. 4.1.4.1 on list of tracks UTR
3: Retain in the profile **top 100** preferred tags
4: Return user's tag-based profile

with $P = p(TG, U_{Non-Last.fm}) = [ITF(TG)] \cdot log \sum_i (log(TR_i\_Overall\_Listened) \cdot log(TG\_UsedFor\_TR_i))$

$$ITF(TG) = log \frac{\sum_i TG_i\_Used\_Overall}{TG\_Used\_Overall}$$

The second variant of the tag-based profile corresponding to a Non-Last.fm user looks similar to the previous one. In this case the preference depends on the number of tracks on the user's desktop that are tagged with tag *TG*.

Alg. 4.1.4.2: Tag-based Profile for Non-Last.fm User

1: Create list of tracks UTR applying **Alg. 4.1.3.1**
2: Apply Alg. 4.1.4.1 on list of tracks UTR
3: Retain in the profile **top 29** preferred tags
4: Return user's tag-based profile

with $P = p(TG, U_{Non-Last.fm}) = log(Tracks\_Containing\_TG)$

Since Non-Last.fm users did not use the tags building up their profile by themselves — they are just inferred based on the music tracks they have on their desktop — we chose to simulate the Last.fm profiles by maintaining only the top 29 preferred tags. From the data we have crawled we could see that the average number of used tags among the users is 29, therefore we keep in the tag-profiles we create with algorithm 4.1.4.2 only top-29 most preferred tags.

4.1.5 Music Recommendations

Using track-based and tag-based profiles we implemented and evaluated different algorithms for producing music recommendations. This section describes 7 algorithms which, based on the type of profile and the technique we used for getting the recommendations, can be grouped into three categories: Collaborative Filtering based on Tracks (Section 4.1.6), Collaborative Filtering based on Tags (Section 4.1.7) and Search based on Tags (Section 4.1.8).

4.1.6 Track-based Recommendations

CF based on TRacks (CFTR). Traditional music recommender systems use User-Item Collaborative Filtering methods with music tracks as items. This method is successfully used in Last.fm and other systems, though we still have the cold start problem, i.e. users have to listen (and possibly rank) a minimum number of tracks and tracks have to be ranked by some listeners, before recommendations are possible. We will use such an algorithm as baseline to compare our other algorithms against.

Alg. 4.1.6.1: Collaborative Filtering based on Tracks (CFTR)
Track Profile $\hookrightarrow$ Track Recommendation $\hookrightarrow$ Tracks

1: Create users **track-based profile** (Alg. 4.1.3.1/ Alg. 4.1.3.1)
2: Get track recommendations based on the Taste-Recommender Java library:
3: Compute **top 10 most similar users** SU with current user U
based on cosine similarity between track profiles
4: **For each** similar user SU_i
5: Get tracks TR_j with preference $p(TR_j, SU_i)$
6: **Combine** lists TR_j of tracks into $\Rightarrow RTR$
7: Recommend music tracks RTR

4.1.7 Tag-Based Recommendations

For the three algorithms proposed in this paper as tag-based recommendation algorithms, the matrix on which we apply collaborative filtering is a User-Tag matrix. In this matrix, line i corresponds to the tag-profile of user i and contains corresponding preference scores for tags which have been used by the user (and 0 for the other tags). In these algorithms what we obtain as result of applying CF on the User-Tag matrix is of course a list of recommended tags, based on what tags other similar users have used. What we want to achieve are music recommendations and not tag recommendations. Therefore with this list of tags we search which tracks have been tagged with most of these tags, taking into account their associated preference scores. We return the top 10 matching tracks, scored by cosine similarity, as recommended songs.

CF based on Track-Tags with ITF (CFTTI). The first algorithm we propose in this category uses tag-based profiles which have been extracted from the list of tracks users have been listening to (Alg. 4.1.4.1/ Alg. 4.1.4.1). For not biasing profiles towards highly used tags, when computing preference scores associated to the tags we also include the ITF parameter. The recommended list of tags obtained after applying CF on the User-Tag matrix is then used for getting the music recommendations:

Alg. 4.1.7.1: CF based on Track-Tags with ITF (CFTTI)

Tracks ↪ Tag Profile ↪ Tag Recommendation ↪ Search w/ Tags ↪ Tracks

1: Create **tag-based profiles** (Alg. 4.1.4.1 / Alg. 4.1.4.1 both with ITF)
2: **Get tag recommendations** based on the Taste-Recommender Java library:
3: Compute **top 10 most similar users** SU with current user U
based on cosine similarity between tag profiles
4: **For each** similar user SU_i
5: Get top 50 tags TG_j by preference $p(TG_j, SU_i)$
6: **Combine** lists TG_j of tags into $\Rightarrow RTG$
7: **Create** Query Q
8: **For** each tag TG_i in RTG
9: **Add** pair $< TG_i, p(TG_i, U) >$ to Q
10: **Search** with Q tracks being tagged with tags in $Q \Rightarrow RTR$
11: **Compute** cosine similarity between tracks in the Lucene index and Q
12: **Rank** resulted tracks RTR based on cosine similarity
13: **Recommend** music tracks RTR

CF based on Track-Tags No-ITF (CFTTN). This second algorithm differs from CFTTI by computing the tag-based profiles without the ITF parameter in the formula corresponding to tags' preference. Otherwise the steps in the algorithm are the same as in Alg. 4.1.7.1.

CF based on Tags (CFTG). For the third algorithm the user profiles on which the tag recommendation step is based, are more personal – users have already used those tags. In this case, line **1:** in Alg 4.1.7.1 is modified and the algorithm looks as follows:

Alg. 4.1.7.3: CF based on Tags (CFTG)
Tags ↪ Tag Profile ↪ Tag Recommendation ↪ Search with Tags ↪ Tracks

1: Create **tag-based profiles** (Alg. 4.1.4.2 / Alg. 4.1.4.2)
2: **Get tag recommendations** based on the Taste-Recommender Java library:
3: Compute **top 10 most similar users** SU with current user U
based on cosine similarity between tag profiles
4: **For each** similar user SU_i
5: Get top 50 tags TG_j by preference $p(TG_j, SU_i)$
6: **Combine** lists TG_j of tags into $\Rightarrow RTG$
7: **Create** Query Q
8: **For** each tag TG_i in RTG
9: **Add** pair $< TG_i, p(TG_i, U) >$ to Q
10: **Search** with Q tracks being tagged with tags in $Q \Rightarrow RTR$
11: **Compute** cosine similarity between tracks in the Lucene index and Q
12: **Rank** resulted tracks RTR based on cosine similarity
13: **Recommend** music tracks RTR

4.1.8 Tag-Based Search

In our last set of algorithms, we use the tags extracted through the previously presented methods for direct matching with other tracks. This is done by creating a disjunctive query of

clauses where each clause consists of a tag and its preference. The results are tracks ordered by cosine similarity between the vector of tags they have been tagged with and the vector of tags given in the query. Direct search using tags has the big advantage of being much faster than any collaborative filtering algorithm, the results being produced instantly. It also offers the user the possibility to enter keyword queries based on tags and get new tracks from different domains if wanted.

Search based on Track-Tags with ITF (STTI). Similar to CFTTI this algorithm is based on profiles created using the algorithms 4.1.4.1 and 4.1.4.1 and includes the ITF factor in the preference formula:

Alg. 4.1.8.1: Search based on Track-Tags with ITF (STTI)
Tags ↪ Tag Profile ↪ Search with Tags ↪ Tracks

1: Create **tag-based profiles** (Alg. 4.1.4.1/ Alg. 4.1.4.1 both with ITF)
2: Create Query Q
3: **For** each tag TG_i in the profile of current user U
4: **Add** pair $< TG_i, p(TG_i, U) >$ to Q
5: Search with Q tracks being tagged with tags in $Q \Rightarrow RTR$
6: **Compute** cosine similarity between tracks in the Lucene index and Q
7: Rank resulted tracks RTR based on cosine similarity
8: Recommend music tracks RTR

Search based on Track-Tags No-ITF (STTN). The second search-based algorithm is based on Alg. 4.1.8.2, just that we remove the ITF parameter in the preference formula.

Search based on Tags (STG). Like the CFTG algorithm, STG uses profiles created by alg. 4.1.4.2 and 4.1.4.2. Tags contained in the profiles are then directly used for searching for tracks which have been tagged with these tags:

Alg. 4.1.8.3: Search based on Tags (STG)
Tags ↪ Tag Profile ↪ Search with Tags ↪ Tracks

1: Create **tag-based profiles** (Alg. 4.1.4.2/ Alg. 4.1.4.2)
2: Create Query Q
3: **For** each tag TG_i in the profile of current user U
4: **Add** pair $< TG_i, p(TG_i, U) >$ to Q
5: Search with Q tracks being tagged with tags in $Q \Rightarrow RTR$
6: **Compute** cosine similarity between tracks in the Lucene index and Q
7: Rank resulted tracks RTR based on cosine similarity
8: Recommend music tracks RTR

Nr.	Algorithm	NDCG	Signif. vs. CFTR	Popularity	Novelty
1	**CFTR**	0.54	-	15,177	1.39
2	CFTG	0.25	High, $\mathbf{p \ll 0.01}$	4,065	1.83
3	CFTTI	0.36	High, $\mathbf{p \ll 0.01}$	6,632	1.72
4	CFTTN	0.37	High, $\mathbf{p \ll 0.01}$	13,671	1.74
5	**STG**	**0.60**	No, $\mathbf{p = 0.22}$	7,587	1.07
6	**STTI**	**0.73**	**High**, $\mathbf{p \ll 0.01}$	10,380	0.82
7	**STTN**	**0.77**	**High**, $\mathbf{p \ll 0.01}$	16,309	0.78

Table 4.1: Normalized Discounted Cumulative Gain over the first 10 recommended tracks, along with the average track popularity and average novelty

4.1.9 Evaluation & Results

We evaluated our algorithms with 18 subjects which had to provide two different scores: one measuring how well the recommended track matches their music preferences and one reflecting the novelty of the track. The quality of the recommended results was measured using the normalized version of Discounted Cumulated Gain (DCG) [107].

Table 4.1 shows the NDCG value, its statistical significance over the $CFTR$ baseline computed using T-tests, and the average popularity (number of times a track was listened to on Last.fm) of recommended tracks for each algorithm. All Collaborative Filtering algorithms based on tags ($CFTG$, $CFTTI$, $CFTTN$) performed worse than the baseline, as standard User-Item CF techniques already show high precision. All our search algorithms, though, show quite substantial improvements over track based CF (STG 12%, $STTI$ 37%, $STTN$ 44% as shown in Figure 4.1; $STTI$ and $STTN$ both highly statistically significant). This outcome is certainly positively influenced by the consistent usage of tags on Last.fm: Most frequently used tags denote the track's genre, so our search gets biased towards specific user preferred music genres. It was also interesting to note that the better people knew the tracks (i.e., a lower novelty value), the higher they rated the recommendations. We observed an almost perfect inverse correlation between these two scores, with a Pearson's correlation coefficient between average NDCG and Novelty values per algorithm of $c = -0.987$, and still a high inverse correlation of all preference and novelty marks with $c = -0.513$.

Another interesting result is that STG recommends much less popular tracks than our $CFTR$ baseline, but still of higher quality, so that it is suited for people demanding a higher diversity of music, not listening to the same tracks over and over again. We can thus suggest different algorithms, depending on the user's preference concerning popularity and novelty of tracks. Because of the high use of the "rock" tag (used twice as much as any other tag), many *hard rock* or *heavy metal* songs were recommended, mostly by tag-based CF algorithms. Further research has to be done in order to disambiguate tag meanings, and to reduce unwanted tag weights.

Nr.	Algorithm	NDCG	Signif. vs. CFTR	Popularity	Novelty
1	**CFTR**	0.60	-	19,717	1.33
2	CFTG	0.29	Yes, $\mathbf{p = 0.02}$	7,787	1.84
3	CFTTI	0.33	Yes, $\mathbf{p = 0.02}$	9,970	1.79
4	CFTTN	0.32	High, $\mathbf{p \ll 0.01}$	25,576	1.77
5	STG	0.55	No, $\mathbf{p = 0.29}$	7,799	1.11
6	**STTI**	**0.76**	Minimal, $\mathbf{p = 0.10}$	10,709	0.81
7	**STTN**	**0.80**	Minimal, $\mathbf{p = 0.07}$	15,664	0.61

Table 4.2: Normalized Discounted Cumulative Gain, average track popularity, and average novelty over the first 10 recommended tracks, only for users with less than 50 tracks on their Desktop

Nr.	Algorithm	NDCG	Signif. vs. CFTR	Popularity
1	**CFTR**	0.31	-	22,766
2	CFTG	0.19	No, $\mathbf{p = 0.21}$	4,852
3	CFTTI	0.24	No, $\mathbf{p = 0.42}$	5,758
4	CFTTN	0.29	Minimal, $\mathbf{p = 0.13}$	17,419
5	STG	0.27	No, $\mathbf{p = 0.25}$	21,111
6	STTI	0.26	No, $\mathbf{p = 0.36}$	29,196
7	**STTN**	**0.35**	No, $\mathbf{p = 0.25}$	44,490

Table 4.3: Normalized Discounted Cumulative Gain and average track popularity over the first 10 recommended tracks, only for items with high novelty

When looking only at people with less than 50 personal music tracks on their desktop (Table 4.2) - this was the case for 7 of our test subjects, the number of tracks ranging from 17 to 48 and averaging at 31 - we still find a gain of 26% and 33% over the baseline for $STTI$ and $STTN$, respectively. This indicates that our user tag profiles also work with less rich music repositories. Results presented in Table 4.3 only for recommended tracks with high novelty (i.e., novelty mark = 2), show a decrease in NDCG for all algorithms, mostly not statistically significant since users had different music knowledge. Still $STTN$ performs 13% better than $CFTR$, mainly because it recommends tracks with higher popularity.

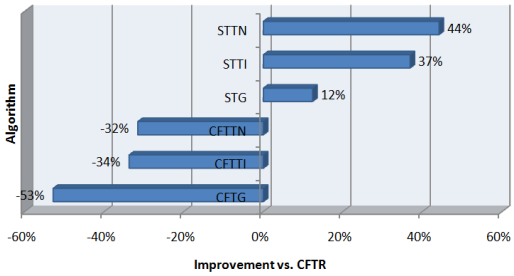

Figure 4.1: Relative NDCG gain (in %) over the $CFTR$ baseline for each algorithm.

4.2 Personalized Ranking using LSI

Since Latent Semantic Analysis (LSA) has been proposed in 1990 [64] plenty of research papers explored different aspect of LSA[8]. The idea behind LSA is the substitution of the term-document matrix by a low-rank matrix approximation. By reducing the rank of the matrix, we limit the number of underlying *latent variables* or *concepts* and remove noisy data. Experiments proved that this low-rank representation improves the quality of document ranking and clustering. The potential use of LSA is hindered by its computational complexity. The core of the method is the well–known Singular Value Decomposition (SVD), which computation is very expensive for large matrices. The modern clustering and ranking applications operate on matrices with millions of rows and columns and it can take weeks to perform SVD with such volumes of data, set aside the unrealistic memory requirements. The Web data changes rapidly and LSA must be updated reasonably often. Moreover, the new personalization techniques, which apply a user-specific profile to adjust ranking for a particular user, require precomputation of LSA for every user. The efficient method for SVD computation can improve result quality for the state-of-art search services. In this work we explore different ways to scale SVD computation to large matrices and personalize its computation based on the user profile.

4.2.1 Singular Value Decomposition (SVD)

So far we were discussing the high-level properties of SVD, now we would like to provide some more details on it. The SVD method stems from one of the basic theorems in linear algebra. Consider a real matrix $A_k^{m\times n}$ with m rows, n columns, and a rank k. The theorem says that there exist two orthonormal matrices $U \in \mathbb{R}^{m\times k^\star}$, $V \in \mathbb{R}^{n\times k^\star}$ and a diagonal matrix $\Sigma^{k^*\times k^*} = \mathrm{diag}(\sigma_1, \ldots, \sigma_{k^\star})$ with two following properties:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{k^\star} > 0, and \tag{4.1}$$

$$A = U\Sigma V^{\mathsf{T}}. \tag{4.2}$$

[8] In the information retrieval, it is also referred to as Latent Semantic Indexing (LSI).

The formula Eq.4.2 is essentially an SVD. Let us denote the i-th columns of U and V by u_i and v_i. Then SVD can be re-written as follows:

$$A = \sum_{i=1}^{k^\star} u_i \sigma_i v_i^\mathsf{T}. \tag{4.3}$$

The vectors u_i and v_i are called the i-th left and right singular vector of A, and associated σ_i is called i-th singular value of A. SVD is unique if and only if all singular values are different. SVD is commonly used for a low-rank approximation of A. Consider a rank k such that $1 \leq k \leq k^\star$ and two modified matrices $U_k^{m\times k}$ and $V_k^{n\times k}$ with extra right columns removed. The matrix with singular values is also reduced to $\Sigma_k := \mathrm{diag}(\sigma_1, \ldots, \sigma_k)$. It can be proved that $A_k := U_k \Sigma_k V_k^\mathsf{T}$ is an optimal rank-k least squares approximation of A or more formally:

$$A_k = \arg \min_{\substack{X \in \mathbb{R}^{m\times n}, \\ \mathrm{rank}(X) \leq k}} |A - X|_\mathsf{F}, \tag{4.4}$$

where, $X_k^{m\times n}$ is an arbitrary matrix of rank lower or equal to k and F is the Frobenius norm. An approximation error can be also expressed using the singular values only:

$$|A - A_k|_\mathsf{F} = \sqrt{\sum_{i=k+1}^{k^\star} \sigma_i^2}. \tag{4.5}$$

4.2.2 Latent Semantic Analysis

Now we elaborate a bit more on LSA method in application to information retrieval. LSA takes a low-rank approximation of the term-documents matrix, reducing the number of latent concepts in the document collection and improving document search and classification.

SVD is a main machinery behind LSA, which takes as an input a real term-document matrix $A_k^{m\times n}$ built from m terms and n documents. The actual values within the matrix depend on a particular weighting and normalization scheme applied to term statistics like term counts in a particular document, term counts in the whole collection, document length, etc. The triples (u_i, σ_i, v_i) can be seen as latent concepts or topics like “sport", “arts", etc. The intuition behind the method is that by reducing “weakly represented" topics with small singular values, we remove noise from the matrix and reinforce existing strong connections between topics, terms and documents. One has to empirically select the desired number of latent concept k (usually between 300 and 1000) and SVD returns a low-rank approximation $A_k^{m\times n}$ of the original matrix. The method is effective for numerous information retrieval tasks but computationally expensive and non-scalable.

4.2.3 Personalized VLSI

One of the recent techniques from [82] is called Variable Latent Semantic Indexing (VLSI). The proposed method achieves the same approximation quality as LSI with about 10 times less concepts selected. The intuition is that it gives a control over the noise reduction process so that we can reinforce the user-related concepts and preserve them from removing. Still, the exact effect of such a transformation is not yet well-explained and is a topic for future research.

Let us denote a query vector $Q \in \mathbb{R}^n$ as a random vector combined from all queries of a single user. The distribution of values within Q reflects user's search preferences. Such queries can be recorded in server logs and connected with the signed in user to create the user's query history. Now we can incorporate user's preferences into the approximation task and re-formulate a minimization problem presented in Eq.4.4 as minimization of the following expression:

$$\mathrm{E}\Big(\big|(A-X)Q\big|_F\Big), \tag{4.6}$$

over all matrices $X \in \mathbb{R}^{m\times n}$ of rank at most k as in original LSI. Now consider a matrix $C_Q^{n\times n} := \mathrm{E}\big(QQ^{\mathsf{T}}\big)$ is a co-occurrence matrix of Q. The $C_Q^{n\times n}$ is symmetric since $\big(QQ^{\mathsf{T}}\big)^{\mathsf{T}} = QQ^{\mathsf{T}}$. The intuition behind the so-constructed covariance matrix is that it captures the dimensions of the index most relevant to a user and assigns a magnitude to the co-occurance of query terms. For example, if the user searches for sport related material very often, terms such as *match, goal, footbal, tennis, ace, result, draw* etc will occur very often and can be mapped into the sport-related dimension of the index; parts of the index belongin to unrealted thesems e.g. legal terms, art & history will be deemed unimportant for the user.

As suggested by the eigen decomposition theorem, we can use identical the left and right singular vectors for symmetric matrices. In this particular setting we call them eigenvectors, while the singular values are called eigenvalues. The SVD for C_Q takes the following form:

$$C_Q = \hat{V}\hat{\Sigma}\hat{V}^{\mathsf{T}}. \tag{4.7}$$

Let $\hat{\sigma}_1 \geq \hat{\sigma}_2 \cdots \geq \hat{\sigma}_{l^\star} > 0$ be the eigenvalues of C_Q, where $l^\star$ is the rank of C_Q. Therefore, we define:

$$\hat{\Sigma}^{1/2} = \mathrm{diag}\big(\sqrt{\hat{\sigma}_1}, \sqrt{\hat{\sigma}_2}, \ldots, \sqrt{\hat{\sigma}_{l^\star}}\big), \tag{4.8}$$

$$C_Q^{1/2} := \hat{V}\hat{\Sigma}^{1/2}\hat{V}^{\mathsf{T}}, \tag{4.9}$$

$$C_Q^{1/2}C_Q^{1/2} = C_Q. \tag{4.10}$$

A matrix $C_Q^{1/2}$ is called the square root of C_Q and a matrix $C_Q^{1/2} := \hat{V}\hat{\Sigma}^{-1/2}\hat{V}^{\mathsf{T}}$, with $\hat{\Sigma}^{-1/2} := \mathrm{diag}\big(1/\sqrt{\hat{\sigma}_1}, 1/\sqrt{\hat{\sigma}_2}, \ldots, 1/\sqrt{\hat{\sigma}_{l^\star}}\big)$, is called the pseudoinverse of C_Q. Note that any symmetric matrix has both an associated square root and pseudoinverse.

It has been proved in [82] that the following equation holds:

$$\tilde{U}_k\tilde{U}_k^{\mathsf{T}}A = \underset{\substack{X\in\mathbb{R}^{m\times n},\\ \mathrm{rank}(X)\leq k}}{\arg\min}\ \mathrm{E}\Big(\big|Q(A-X)\big|_2\Big). \tag{4.11}$$

for any k, where $\tilde{U}\tilde{\Sigma}\tilde{V}^{\mathsf{T}}$ being a singular value decomposition of $C_Q^{1/2}A$. As proposed in the VLSI Approach, using the query distribution in the computation of the query process is more efficient than a query-oblivious latent factor analysis. If the query distribution used is that of a single user, the matrix $A\tilde{V}_k\tilde{V}_k^{\mathsf{T}}$ can be seen as a *personalized* rank-k approximation of A.

Use of Feature Space

We follow the approach of [82] with some modification: Instead of the original document–term matrix A we use A_k, for $k \approx 300$. Likewise, we deal with user queries in feature space, i.e. instead of the random vector Q we use the random vector $Q_k := V_k^{\mathsf{T}}Q$. Since k is small, it should be possible to make all required computations using standard algorithms.

A further advantage of this approach is that we can profit from the representation of data in feature space twofold: First, according to experiences with LSI A_k seems to be a better (i.e. less noisy) representation of the document–term structure. Second, we usually have to estimate the distribution of Q from a relatively small sample of queries. If this number is q, then usually $q \ll m$ and $q \ll n$ hold. Therefore, it will be difficult to make good estimations of Q's co-occurrence matrix. But q and k often will be of similar magnitude, what makes useful estimations of Q''s co-occurrence matrix feasible.

Avoid Computation of Co-Occurrence Matrix

In most cases, the distribution of Q is unknown. The only information available about Q is a finite sample from Q. Instead of making the long way round over estimating Q's co-occurrence matrix, we could directly make use of the samples we have.

There are learning algorithms that can incrementally estimate the eigenvectors and eigenvalues of the co-occurrence matrix for some random vector, only from samples of this vector. Therefore, it is possible to estimate C_Q's eigen decomposition from our set of query samples.

But we will use a slightly different approach. Define the random vector $\tilde{Q} := AQ$. With the algorithms mentioned above we can learn the eigendecomposition of

$$C_{AQ} := \mathrm{E}\Big((AQ)(AQ)^{\mathsf{T}}\Big) = \mathrm{E}\big(AQQ^{\mathsf{T}}A^{\mathsf{T}}\big) = A\,\mathrm{E}\big(QQ^{\mathsf{T}}\big)A^{\mathsf{T}}.$$

Now, denote the eigen decomposition of $C_Q = \mathrm{E}\big(QQ^{\mathsf{T}}\big)$ by $C_Q = \tilde{U}\tilde{\Sigma}\tilde{U}^{\mathsf{T}}$. We then have

$$C_{AQ} = A\tilde{U}\tilde{\Sigma}\tilde{U}^{\mathsf{T}}A^{\mathsf{T}} = \Big(A\tilde{U}\tilde{\Sigma}^{1/2}\Big)\Big(A\tilde{U}\tilde{\Sigma}^{1/2}\Big)^{\mathsf{T}}.$$

Since, for any matrix X, the singular value decomposition is equivalent to the eigen decomposition of XX^{T}, we are able to estimate the singular value decomposition of $A\tilde{U}\tilde{\Sigma}^{1/2}$. Note that $\tilde{U}$ and $\tilde{\Sigma}^{1/2}$ contain the eigenvectors and the square-roots of the eigenvalues of C_Q. There-

fore, $A\tilde{U}\tilde{\Sigma}^{1/2}$ can be seen as a reweighting of A according to the dominating interests of the current user.

If we now use only the first k singular vectors and singular values of $A\tilde{U}\tilde{\Sigma}^{1/2}$, this could be a good personalized approximation to A.

One could also replace A by A_k in this approach in order to make prior use of LSI's noise-reducing capabilities.

Conclusions and Future Work

The work presented above is under gradual improvement and is classifed as *work-under-progress*. We find the direction revealed by our analysis promising; the next version of the report will carry more details.

5 Conclusions & Future Work

This report aims at describing novel algorithms in Social media in the context of the PHAROS project. The vision for PHAROS has an important place for user-driven content; the use of personalization and recommendation is vital to PHAROS for enhancing the user experience. While various personalization and recommendation approaches have been identified (and described by us in Chapter 2), there are few large scale and generalized applications. The content-based search service of PHAROS will be served well by an accurate and efficient social-search service. What we have described in Chapter 3 represents the first wave of approaches which enhance search. Our approach is three pronged: using explicit feedback on content; using user generated content like tags and comments for semantic analysis; and finally, using user opinions expressed on external data sources like blogs and review sites. These approaches will be used to reinforce one another; combined approaches have been demonstrated to be more effective than individual approaches when it comes to classification (see Boosting [108]).

Research on improving access to Social Media is vital to the successful exploitation user feedback within PHAROS. Clearly, further research is needed to build a scalable social analytics module. We report here only the initial part of the solution; further research will be reported in the next version of the report. In parallel, there is design effort in Work Package 3.2 describing the social model and the storage components for user-related data.

Bibliography

[1] Waqar Ali Shah: What is web 2.0? (2007)

[2] Gruhl, D., Guha, R.V., Kumar, R., Novak, J., Tomkins, A.: The predictive power of online chatter. In: KDD. (2005) 78–87

[3] Golder, S., Huberman, B.: The Structure of Collaborative Tagging Systems. Arxiv preprint cs.DL/0508082 (2005)

[4] Bateman, S., Brooks, C., McCalla, G., Brusilovsky, P.: Applying Collaborative Tagging to E-Learning. Workshop on Tagging at WWW 2007 (2007)

[5] John, A., Seligmann, D.: Collaborative Tagging and Expertise in the Enterprise. Collab. Web Tagging Workshop in conj. with WWW2006 (2006)

[6] Berendt, B., Hanser, C.: Tags are not Metadata, but Just More Content–to Some People. ICWSM (2007)

[7] Marlow, C., Naaman, M., Boyd, D., Davis, M.: Position Paper, Tagging, Taxonomy, Flickr, Article, ToRead. Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland (2006)

[8] Golder, S., Huberman, B.: Usage patterns of collaborative tagging systems. Journal of Information Science **32** (2006) 198

[9] Zollers, A.: Emerging Motivations for Tagging: Expression, Performance, and Activism. Collaborative Web Tagging Workshop at WWW2007 (2007)

[10] Maala, M.Z., Delteil, A., Azough, A.: A conversion process from flickr tags to rdf descriptions. In Flejter, D., Kowalkiewicz, M., eds.: Business Information System. Workshop on Social Aspects of the Web. (2007)

[11] Karger, D., Bakshi, K., Huynh, D., Quan, D., Sinha, V.: Haystack: A Customizable General-Purpose Information Management Tool for End Users of Semistructured Data. Proc. of the CIDR Conf., January (2005)

[12] Gemmell, J., Bell, G., Lueder, R.: MyLifeBits: a personal database for everything. Communications of the ACM **49** (2006) 88–95

[13] Hammond, T., Hannay, T., Lund, B., Scott, J.: Social Bookmarking Tools (I). D-Lib Magazine **11** (2005) 1082–9873

[14] Schmitz, P.: Inducing ontology from flickr tags. Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland, May (2006)

[15] Halpin, H., Robu, V., Shepherd, H.: The complex dynamics of collaborative tagging. Proceedings of the 16th international conference on World Wide Web (2007) 211–220

[16] Hotho, A., Jaschke, R., Schmitz, C., Stumme, G.: Information Retrieval in Folk-

sonomies: Search and Ranking. Proceedings of the 3rd European Semantic Web Conference, Budva, Montenegro, June (2006)

[17] Catutto, C., Schmitz, C., Baldassarri, A., Servedio, V.D.P., Loreto, V., Hotho, A., Grahl, M., Stumme, G.: Network properties of folksonomies. AI Communications Journal (2007)

[18] Mika, P.: Ontologies are us: A unified model of social networks and semantics. ISWC **3729** (2005) 522–536

[19] Sood, S., Hammond, K., Owsley, S., Birnbaum, L.: TagAssist: Automatic Tag Suggestion for Blog Posts. Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007) (2007)

[20] Mishne, G.: AutoTag: a collaborative approach to automated tag assignment for weblog posts. Proceedings of the 15th international conference on World Wide Web (2006) 953–954

[21] Byde, A., Wan, H., Cayzer, S.: Personalized Tag Recommendations via Social Network and Content-based Similarity Metrics. Proceedings of the International Conference on Conference on Weblogs and Social Media (ICWSMŠ06), March (2006)

[22] Xu, Z., Fu, Y., Mao, J., Su, D.: Towards the semantic web: Collaborative tag suggestions. Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland, May (2006)

[23] Kaser, O., Lemire, D.: Tag-Cloud Drawing: Algorithms for Cloud Visualization. Arxiv preprint cs.DS/0703109 (2007)

[24] Michlmayr, E., Cayzer, S.: Learning User Profiles from Tagging Data and Leveraging them for Personal(ized) Information Access. Collaborative Web Tagging Workshop at WWW2007, Banff, Canada (2007)

[25] Dubinko, M., Kumar, R., Magnani, J., Novak, J., Raghavan, P., Tomkins, A.: Visualizing tags over time. Proceedings of the 15th international conference on World Wide Web (2006) 193–202

[26] Aurnhammer, M., Hanappe, P., Steels, L.: Augmenting Navigation for Collaborative Tagging with Emergent Semantics. 5th International Semantic Web Conference (ISWC 2006), Athens, GA, USALecture Notes in Computer Science, Springer (2006)

[27] Aurnhammer, M., Hanappe, P., Steels, L.: Integrating collaborative tagging and emergent semantics for image retrieval. Proc. of the Collaborative Web Tagging Workshop (WWWŠ06) (2006)

[28] Firan, C., Nejdl, W., Paiu, R.: The Benefit of Using Tag-Based Profiles. LaWEB 2007 (2007)

[29] Bao, S., Xue, G., Wu, X., Yu, Y., Fei, B., Su, Z.: Optimizing web search using social annotations. Proceedings of the 16th international conference on World Wide Web (2007) 501–510

[30] Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web (1998)

[31] Marchetti, A., Tesconi, M., Ronzano, F., Rosella, M., Minutoli, S.: SemKey: A Semantic Collaborative Tagging System. Collaborative Web Tagging Workshop at WWW2007, Banff, Canada (2007)

[32] Iturrioz, J., Díaz, O., Arellano, C.: Towards federated Web2.0 sites: the TAGMAS approach. Collaborative Web Tagging Workshop at WWW2007,Banff, Canada (2007)

[33] Kushal Dave, Steve Lawrence, David M. Pennock: Mining the peanut gallery: Opinion extraction and semantic classification of product reviews (2003)

[34] Satoshi Morinaga, Kenji Yamanishi, Kenji Tateishi, Toshikazu Fukushima: Mining product reputations on the web (2002)

[35] Peter D. Turney: Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews (2002)

[36] Theresa Wilson, Janyce Wiebe, Rebecca Hwa: Just how mad are you? finding strong and weak opinion clauses (2004)

[37] Tetsuya Nasukawa, Jeonghee Yi: Sentiment analysis: Capturing favorability using natural language processing (2003)

[38] Bing Liu, Minqing Hu, Junsheng Cheng: Opinion observer: Analyzing and comparing opinions on the web (2005)

[39] Hong Yu, Vasileios Hatzivassiloglou: Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences (2003)

[40] Fernando Pereira, Naftali Tishby, Lillian Lee: Distributional clustering of english words (1994)

[41] Dekang Lin: Automatic retrieval and clustering of similar words (1998)

[42] Vasileios Hatzivassiloglou, Kathleen R. McKeown: Predicting the semantic orientation of adjectives (1997)

[43] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, Katherine Miller: Introduction to wordnet: An on-line lexical database (1993)

[44] Minqing Hu, Bing Liu: Mining and summarizing customer reviews (2004)

[45] Farah Benamara, Carmine Cesarano, Antonio Picariello, Diego Reforgiato, VS Subrahmanian: Sentiment analysis: Adjectives and adverbs are better then adjectives alone (2007)

[46] Yu, K., Schwaighofer, A., Tresp, V., Ma, W.Y., Zhang, H.J.: Collaborative ensemble learning: Combining collaborative and content-based information filtering via hierarchical bayes. In: Proceedings of UAI, Morgan Kauffman (2003)

[47] Minqing Hu, Bing Liu: Mining opinion features in customer reviews (2004)

[48] Bo Pang, Lillian Lee, Shivakumar Vaithyanathan: Thumbs up? sentiment classification using machine learning techniques (2002)

[49] Robert E. Shapire, Yoram Singer: Boostexter: A boosting-based system for text categorization (2000)

[50] William W. Cohen: Learning trees and rules with set-valued features (1996)

[51] Thorsten Joachims: Making large-scale support vector machine learning practical

(1998)

[52] Theresa Wilson, Janyce Wiebe: Annotating opinions in the world press (2003)

[53] Ellen Riloff, Janyce Wiebe: Learning extraction patterns for subjective expression (2003)

[54] Peter D. Turney, Michael L. Littman: Unsupervised learning of semantic orientation from a hundred-billion-word corpus (2004)

[55] Kamal Nigam, Matthew Hurst: Towards a robust metric of opinion (2004)

[56] Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22** (2004) 5–53

[57] Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW'01: Proceedings of 10th International Conference on World Wide Web. (2001) 285–295

[58] Goldberg, D., D, N., Oki, B.M., Terry, D.: Collaborative filtering to weave and information tapestry. Communications of the ACM **35** (1992) 61–70

[59] Shardanand, U.: Social information filtering for music recommendation. Master's thesis, Massachussets Institute of Technology (1994)

[60] Shardanand, U., Maes, P.: Social information filtering: Algorithms for automaing "word of mouth". In: Proceedings of CHI'95. (1995)

[61] Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work, ACM, ACM Press (1994) 175–186

[62] Resnick, P., Varian, H.R.: Recommender systems. Commun. ACM **40** (1997) 56–58

[63] Montaner, M., Lopez, B., de la Rosa, J.L.: A taxonomy of recommender agents on the internet. Artificial Intelligence Review (2003) 285–330

[64] Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the American Society of Information Science **41** (1990) 391–407

[65] Gorrell, G.: Generalized hebbian algorithm for incremental singular value decomposition in natural language processing. In: EACL. (2006)

[66] Kolda, T.G., O'Leary, D.P.: A semidiscrete matrix decomposition for latent semantic indexing information retrieval. ACM Trans. Inf. Syst. **16** (1998) 322–346

[67] Papadimitriou, C.H., Tamaki, H., Raghavan, P., Vempala, S.: Latent semantic indexing: A probabilistic analysis. In: ACM Conference on Principles of Database Systems (PODS). (1998) 159–168

[68] Bingham, E., Mannila, H.: Random projection in dimensionality reduction: applications to image and text data. In: KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2001) 245–250

[69] Drineas, P., Frieze, A., Kannan, R., Vempala, S., Vinay, V.: Clustering in large graphs

and matrices. In: SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (1999) 291–299

[70] Frieze, A., Kannan, R., Vempala, S.: Fast monte-carlo algorithms for finding low-rank approximations. J. ACM **51** (2004) 1025–1041

[71] Drineas, P., Frieze, A., Kannan, R., Vempala, S., Vinay, V.: Clustering large graphs via the singular value decomposition. Mach. Learn. **56** (2004) 9–33

[72] Drineas, P., Drinea, E., Huggins, P.: An experimental evaluation of a monte-carlo algorithm for singular value decomposition. In editor = to be added, ed.: booktitle = to be added, publisher = to be added (year = to be added)

[73] Drinea, E., Drineas, P., Huggins, P.: A randomized singular value decomposition algorithm for image processing applications. In editor = to be added, ed.: booktitle = to be added, publisher = to be added (year = to be added)

[74] Grossman, D.: Experiments on an algorithm for fast matrix singular value decomposition and low-rank approximation (1998)

[75] Jiang, F., Kannan, R., Littman, M.L., Vempala, S.: Efficient singular value decomposition via improved document sampling. In editor = to be added, ed.: booktitle = to be added, publisher = to be added (1999)

[76] Tang, C., Dwarkadas, S., Xu, Z.: On Scaling Latent Semantic Indexing for Large Peer-to-Peer Systems. In editor = to be added, ed.: booktitle = to be added, publisher = to be added (2004)

[77] Gene Golub, Knut Solna, P.V.D.: Computing the svd of a general matrix product/quotient. In editor = to be added, ed.: booktitle = to be added, publisher = to be added (2000)

[78] Claver, J.M., Mollar, M., Hernández, V.: Parallel computation of the svd of a matrix product. In: Proceedings of the 6th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, London, UK, Springer-Verlag (1999) 388–395

[79] Bobda, C., Steenbock, N.: Singular value decomposition on distributed reconfigurable systems. In: RSP '01: Proceedings of the 12th International Workshop on Rapid System Prototyping, Washington, DC, USA, IEEE Computer Society (2001) 38

[80] Ye, J.: Generalized low rank approximations of matrices. Mach. Learn. **61** (2005) 167–191

[81] Kurucz, M., Benczur, A.A., Csalogany, K.: Methods for large scale svd with missing values. In editor = to be added, ed.: booktitle = to be added, publisher = to be added (2007)

[82] Dasgupta, A., Kumar, R., Raghavan, P., Tomkins, A.: Variable Latent Semantic Indexing. In Grossman, R., Bayardo, R., Bennett, K.P., eds.: Proceeding of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD 2005), ACM Press (2005) 13–21

[83] : (Interactive advertising burea)

[84] Bruner, R.E.: The decade in online advertising. In: http://www.doubleclick.com. (2004)

[85] Gruhl, D., Guha, R.V., Liben-Nowell, D., Tomkins, A.: Information diffusion through blogspace. In: WWW. (2004) 491–501

[86] Song, X., Tseng, B.L., Lin, C.Y., Sun, M.T.: Personalized recommendation driven by information flow. In: SIGIR. (2006) 509–516

[87] Guha, R.V., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: WWW. (2004) 403–412

[88] Kumar, R., Novak, J., Raghavan, P., Tomkins, A.: On the bursty evolution of blogspace. In: WWW. (2003) 568–576

[89] Gomez, L.M., Lochbaum, C.C., Landauer, T.K.: All the right words: Finding what you want as a function of richness of indexing vocabulary. JASIS **41** (1990) 547–559

[90] Hofmann, T.: Probabilistic latent semantic analysis. In: UAI. (1999) 289–296

[91] Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. In: NIPS. (2001) 601–608

[92] Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD Conference. (2000) 1–12

[93] El-Sayed, M., Ruiz, C., Rundensteiner, E.A.: Fs-miner: efficient and incremental mining of frequent sequence patterns in web logs. In: WIDM. (2004) 128–135

[94] Grarovetter, M.: Threshold models of collective behavior. Americal Journal of Socialogy **83-6** (1987) 1420–1443

[95] Jacob Goldenberg, B.L., Muller, E.: Talk of network: A complex systems look at the underlying process of word-of-mouth. Marketing Letters **12-3** (2001) 211–223

[96] Hipp, J., Güntzer, U., Nakhaeizadeh, G.: Algorithms for association rule mining — a general survey and comparison. SIGKDD Explorations **2** (2000) 58–64

[97] Bo Pang and Lillian Lee: Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. Proceedings of the ACL (2005)

[98] Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, Daniel M. Ogilvie and associates: The general inquirer: A computer approach to content analysis (1966)

[99] Edward Kelly and Philip Stone: Computer recognition of english word senses (1975)

[100] Boullé, M.: Khiops: A statistical discretization method of continuous attributes. Machine Learning **55** (2004) 53–69

[101] Marc Boullé: Compression-based averaging of selective naive bayes classifiers (2007)

[102] Celma, O., Ramirez, M., Herrera, P.: Foafing the music: A music recommendation system based on rss feeds and user preferences. Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR) (2005)

[103] Yoshii, K., Goto, M., Komatani, K., Ogata, T., Okuno, H.G.: Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR) (2006)

[104] Pauws, S., Verhaegh, W., Vossen, M.: Fast generation of optimal music playlists using local search. Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR) (2006)

[105] Whitman, B., Lawrence, S.: Inferring descriptions and similarity for music from community metadata. Proceedings of the International Computer Music Conference (ICMC) (2002)

[106] Byde, A., Wan, H., Cayzer, S.: Personalized tag recommendations via tagging and content-based similarity metrics. Proceedings of the International Conference on Weblogs and Social Media (ICWSM) (2007)

[107] Järvelin, K., Kekäläinen, J.: Ir evaluation methods for retrieving highly relevant documents. Proc. of the 23th Intl. ACM SIGIR Conf. on Research and development in information retrieval (2000)

[108] Schapire, R.: The strength of weak learnability. Machine Learning **5** (1990) 197–227