SOCIAL MEDIA BETA

Platform for searcH of Audiovisual Resources across Online Spaces

PHAROS

Project IST-45035

Deliverable D212 WP2.1

Deliverable
Version 1.0 – 30 June 2008
Document. ref.: pharos.D212.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 Centre (DE), Fraunhofer IDMT (DE), Ecole Polytechnique Fédérale de Lausanne (CH), Knowledge Media Institute Open University (UK), Technical Research Centre of Finland VTT (FI), Circom Regional (FR), Metaware (IT), Web Models (IT), SAIL LABS Technology (AT), Fundació Barcelona Media Universitat Pompeu Fabra (ES)
Document Number:	pharos.D212.L3S.WP2.1.V1.0
Work-Package:	WP2.1
Deliverable Type:	Prototype
Contractual Date of Delivery:	30/06/2008
Actual Date of Delivery:	30/06/2008
Title of Document:	Social Media Beta
Author(s):	L3S (Kerstin Bischoff, Ling Chen, Sergey Chernov, Ernesto Diaz, Claudiu Firan, Raluca Paiu, Avaré Stewart), VTT (Vainikainen Sari, Magnus Melin), FT (Cecile Bothorel)
Editor(s): …..……………………….	Raluca Paiu, Kerstin Bischoff, Ling Chen (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

Executive Summary

This deliverable titled *Social Media Beta* aims at describing the progress made in the second 9 months of the PHAROS project in the area of social media. The report has been authored by various project partners involved in WP2.1, and has been edited by L3S Research Center.

This report focuses on the engineering aspect and describes the development of *Social Media Bet*a in the context of WP2.1. Particularly, Chapter 1 briefly presents some background knowledge about social media and introduces the general architecture of Social Media Beta release. Four specific modules: User & Social Information Storage (USIS), User & Community Profiler (UCP), Social Networks & Blogspace Analysis (SNBA), and Personalization Module (PM) are described in depth with both architectural aspects, as well as algorithms in Chapter 2. For the purpose of further demonstrating the functionalities of these modules, in Chapter 3, a series of use cases corresponding to each module are described. Finally, we conclude this report with some summary remarks and future work discussions.

In this document we refer to some of the algorithms described in the deliverable D2.1.1 and discuss the changes we made since that time. The new algorithms are presented in detail and included in the Algorithms – related sections. The main new contributions are the following:

- The User and Social Information Storage (USIS) component
- Detailed analysis on the implications of tags on search (included in the Algorithms section of UCP)
- Community identification based on blog analysis (included in the Algorithms section of SNBA)
- The Personalization Module (PM) and the associated algorithms
- A series of use cases illustrating the functionalities of the modules building the Social Media Beta release

This report describes the current implementation status of social media modules in the PHAROS platform. The algorithms and implementation are subject to change, based on the feedback we will gather through the user evaluation of the Showcase, whose part the Social Media Beta will also be. Further research and engineering work will be continued to optimize the developed and employed algorithms and to improve the overall performance of *the Social Media Beta* release.

1. Introduction

As described in the first release of this document, social media refers to online technologies and practices that people use to share content, opinions, insights, experiences, perspectives, and media itself. Instead of creating online content, it focuses on creating the facilities and framework for non media professionals (i.e., “ordinary people”) to publish their own content in prominent places, such as blogs, wikis, online forums, and social network sites. Blogs, wikis, networking sites, such as MySpace (MySpace) and Friendster (Friendster), let everyone have their say on anything and publish it to the world at large. As Web applications become more sophisticated, people can easily develop elaborate personal Web pages, create blogs, upload their own opinions, photos, audios and video content, or augment news by reporting current events sometimes faster than professional journalists.

The large volume of user generated content, although sometimes amateurish, represents a valuable source of information for service providers. For example, companies and organizations can efficiently get feedback from consumers observing their online interaction with social media providers. Offering accurately personalized services is possible now, since users provide more personal information about themselves openly, which was previously much more difficult to perceive and measure.

In PHAROS, we also aim at exploiting the new and freely available data to improve users’ online experience with respect to their interaction with social media. We focus on building technologies, which bridge the gap between the availability of information (both in form of descriptions of content, such as annotations, and user interests and preferences) and the use of it, for augmenting traditional search and retrieval methods or for personalization purposes.

In the present document, we describe how this external information can be brought into PHAROS and how it is used to support users. We describe the multiple components supporting this process, discuss their connection with the rest of PHAROS components and present the different algorithms which rely on social media information.

1.1 Applying Social Media in PHAROS

Thanks to the various social media technologies existing on the Web to date, a large amount of different types of social metadata can be derived. In PHAROS, we currently take into account two types of important social media which is increasingly popular over the Web today: social annotations and weblogs.

Social annotation refers to the user-supplied tags, which are textual labels, to a piece of information on the Web, such as a picture, blog entry, a video clip etc. With the vast development of Web 2.0, social tagging has been a powerful and important feature provided by many social media applications, such as Flickr (Flickr), Del.icio.us (Delicious), and Last.fm (Last.fm). Consequently, large volume of social annotation data can be collected easily, which enables reliable and accurate knowledge discovery. The knowledge embedded in such user-supplied annotation data is believed to be useful in many applications such as intranet search (Dmitriev, Eiron, Fontoura, & Shekita, 2006), as well as internet search (Bao, Xue, Wu, Yu, Fei, & Su, 2007). In PHAROS, we also investigate the social annotation data to extract valuable knowledge. Particularly, we focus on studying the usage patterns between users and social annotations. We then further explore the usage of discovered patterns in personalized searching and recommendations.

Recently, weblogs have become one of the dominant forms of self-publication on the Internet. A weblog, or “blog”, is commonly defined as a Web page with a set of dated entries, in reverse chronological order, maintained by its writer via a weblog publishing tool. The contents of entries (a.k.a posts) are discussions and observations ranging from the mainstream to the very personal. The fast-growing popularity of the blogosphere offers new chances and challenges for web searching. For example, besides searching blogs, we can also analyze weblog communities, as a representative of our target audience, to predict the effectiveness of a new recommendation. In PHAROS, by using weblogs we aim at discovering communities, which consist of blog users discussing similar topics in a certain period of time. We then analyze the properties of the identified communities, such as the information diffusion patterns in a community, to create accurate and detailed community profiles. The discovered community information will be used to optimize the search and recommendation results for individual users.

1.2 Architecture of PHAROS Social Media Modules

We start with the description of the architecture of social media modules in PHAROS, to show how the social data is brought to the PHAROS platform and exploited for personalization purposes. There are five modules in total which belong to three layers: *Analysis*, *Storage* and *Processing*, as shown in Figure 1-1:

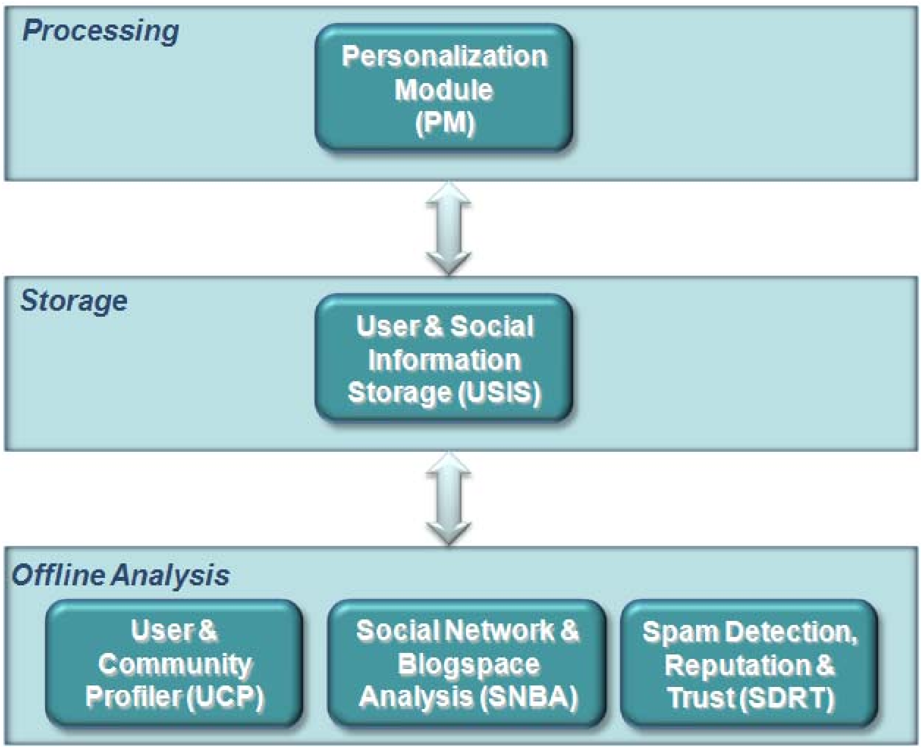

Processing
Personalization Module (PM)
Storage
User & Social Information Storage (USIS)
Offline Analysis
User & Community Profiler (UCP)
Social Network & Blogspace Analysis (SNBA)
Spam Detection, Reputation & Trust (SDRT)

Figure 1-1: Architecture of Social Media Modules in PHAROS

There are three modules inside the Offline Analysis layer: **User & Community Profiler** (**UCP**), **Social Networks & Blogspace Analysis** (**SNBA**), and **Spam Detection, Reputation and Trust** (**SDRT**). These modules retrieve related social metadata either from the PHAROS platform or from some other sources available on the Web. They further process the collected raw data to extract useful knowledge for other functionalities in PHAROS. In particular, the UCP module focuses on collecting and creating complete and accurate user profiles, as well as community profiles so that precise and personalized search and recommendation can be provided based on these profiles. The SNBA module aims at retrieving social network data, such as friendship network and blogspace information, and analyzes the social network both from a micro-perspective (e.g., the network of a user community) as well as a macro-perspective (e.g., the network of all PHAROS users). Due to the fact that current social media technologies are highly vulnerable to malicious users motivated by both private and commercial interests, the module SDRT is developed to improve the robustness of the PHAROS platform by detecting spam and assigning reputations and trust values to the users involved in social interactions. (Note that the details of this module will be covered by another deliverable, D2.2.3.)

The useful knowledge extracted from social metadata by all of the offline analysis modules will be recorded in the same storage, the **User & Social Information Storage** (**USIS**). Periodic updates are initiated in order to ensure that the stored knowledge is not obsolete. Beside the interfaces with offline analysis modules, USIS also interacts with the **Personalization Module** (**PM**), located inside the Processing layer, to provide requested information for search and recommendation. Usage information such as user click-through data will be also stored into the USIS module, and this information will be provided by the Query Interaction & Results Presentation (QIRP) component, located like the PM module inside the Processing Layer.

The **Personalization Module** (**PM**) module manifests the usefulness of social metadata in PHAROS. Various personalization strategies based on different extracted knowledge are developed to finally optimize the search results provided by PHAROS.

The details of each of the modules composing the Social Media Beta release will be described in depth in the following chapters.

2. Social Media Modules

2.1 USIS – User & Social Information Storage

2.1.1 Description

The **User & Social Information Storage (USIS)** plays a central role inside the architecture of the Social Media Beta, as this is the place where all user-related information is stored.

The functionality of the USIS can be divided into the following roles:

1. Metadata storage for PHAROS content objects;
2. User related data storage and processing;
3. External social interaction data storage

Metadata Storage

Each PHAROS content object can have different types of metadata attached. Here *metadata* stands for data provided by registered PHAROS users (rather than metadata produced by content object annotators extracting low-level features from the data itself). Each user having an account in the platform will be able to enrich content objects with metadata. This metadata can be viewed and searched by the users.

The USIS will contain information about PHAROS content objects in several ways:

- *Tags* – short textual content: For each content object the user will be able to quickly add a short textual remark. Although users can add any text they prefer, by analyzing the tags (done by Offline Analysis modules) or by performing Spam identification (done by the SDRT module) only accurate tags will be presented to all PHAROS users.
- *Comments* – long textual content: In contrast to tags which can be the same added by many users, here each user can write some sentences s/he thinks will be appropriate for that content. Comments can include opinions, usage descriptions, mistakes found while playing the content, etc. Similar to the tags, these comments will also used by the Offline Analysis modules to extract meaningful information about content objects and about the users who wrote the comments.
- *Ratings* – assigned marks (from -5 to 5): The simplest way to express an opinion about a content object is to assign a mark defining its content quality. These ratings will be displayed with the object, showing impressions by other users and can be also used for ranking or filtering purposes i.e., for showing most popular objects.
- *Favorites* – mark content objects or any other external resource as favorites: Similar to browsers, the functionality to have a list of objects stored as favorites is provided. These favorites can be both PHAROS content objects as well as URLs pointing to external resources. Although for the user the main functionality will be fast access to preferred resources, these resources can be also analyzed in order to further improve the user profiles.

User Related Data

All the information stored in the USIS will be later extracted and included into the personalization process. User and community profiling information in form of both preferences, as well as interaction with the PHAROS platform are stored in here.

Several parts will build up the user profiles and will be stored inside the USIS (see Figure 2-1):

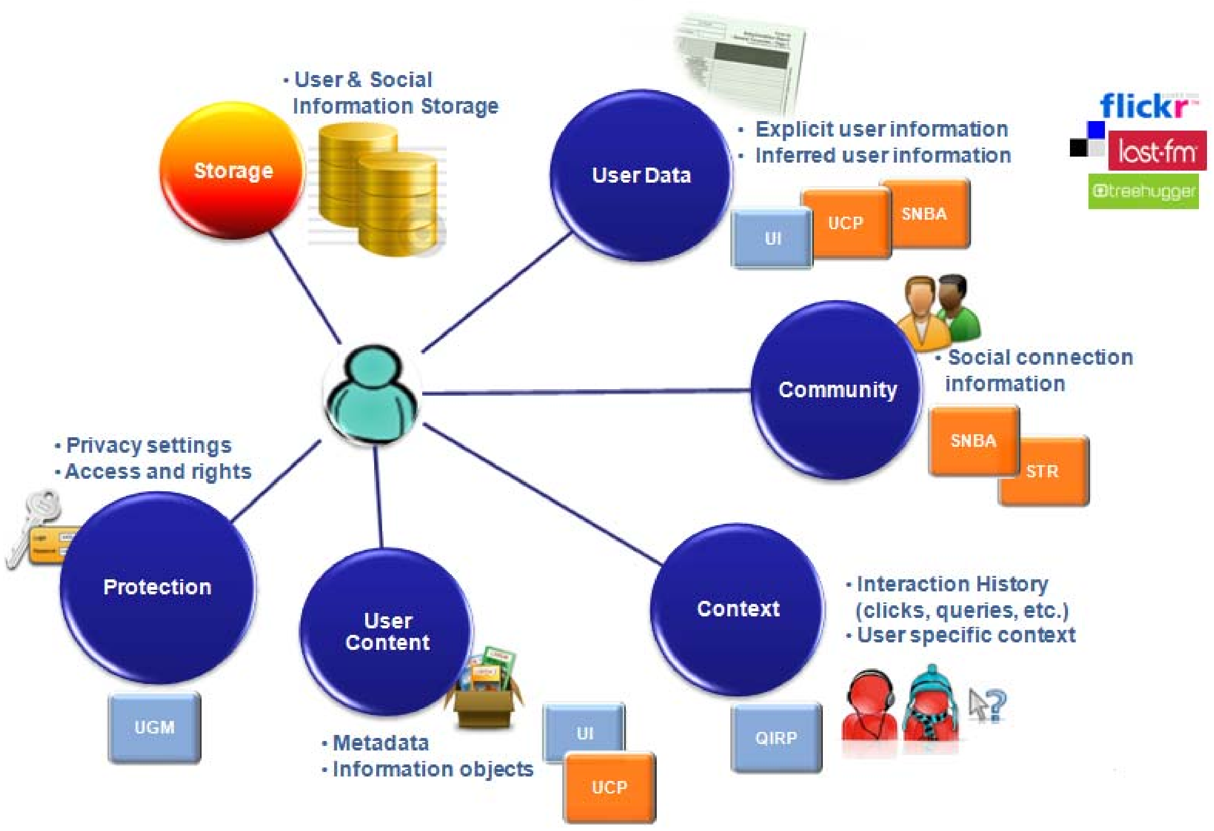


Figure 2-1: User profile's composing parts

- *User Data* – refers to both explicitly provided user information, gathered through form completions, as well as to the inferred user information. User preferences (e.g. preferred language for displaying the PHAROS pages) specified upon registration, or later when editing the existing profile information – i.e. by completing some forms – will be communicated to the USIS through the Query Interaction and Results Presentation (QIRP) component coming from the User Interface (UI) component. Users also have the possibility to specify their account IDs for other external applications they use, e.g. Last.fm, Flickr, del.icio.us. By giving away this information the User & Community Profiler (UCP) and the Social Networks and Blogspace Analysis (SNBA) modules can gather the profile information from the services specified by the users and store this knowledge inside the USIS. This feature is extremely useful especially for recently registered PHAROS users, because this way we can compensate the lack of user data in the system and personalized services can be supported from the very first moment of using the platform, thus avoiding the 'cold start' problem.

- *Community Information* – Users nowadays are not single individuals, but interact with each other, and this social connectivity data, coming both from inside and outside the PHAROS platform, is also included in their profiles. The Social Networks and Blogspace Analysis (SNBA) module will gather and analyze this type of social information. Possible trust and reputation levels established inside social networks are also an important for personalization and this knowledge will be stored into the USIS by the Spam Detection, Reputation and Trust (SDRT) component.

- *User Context* – For being able to personalize the users' interaction with the PHAROS platform, the profiles will also include contextual information – interaction history (query logs, click-through data, etc.) or user specific context, corresponding to the task at hand.

- *User Content* – The content of the users – information items and metadata associated to them – will be also included in the specification of the users' profiles, since the content the users are interacting with is a very good indication of their interests. QIRP and UCP components will manage this type of data.

- *Protection Settings* – Privacy, rights and access settings might be bound to the users' content – therefore this will be also part of the users' profiles and will be managed by the User & Group Management (UGM) module.

External Social Data

Data from external (not residing inside the PHAROS platform) sources like social networks or collaborative tagging Web sites can be also stored inside the USIS. This data can be used by any of the Offline Analysis modules in order to extract additional data relevant for PHAROS users or content objects. This storage is needed inside the USIS, as it is an online module which can be accessed at any time by the Offline Analysis components. These components themselves are not online and cannot provide this kind of services. This way, data analysis which requires several steps performed by three Offline Analysis modules (UCP, SNBA and SDRT) can be stored in between analysis steps.

2.1.2 Architecture

In the following we present the detailed architecture of the USIS module.

USIS is mainly a storage component, a database, providing several services depending on different data to be stored or requested. Several modules interact with USIS as shown in Figure 2-2.

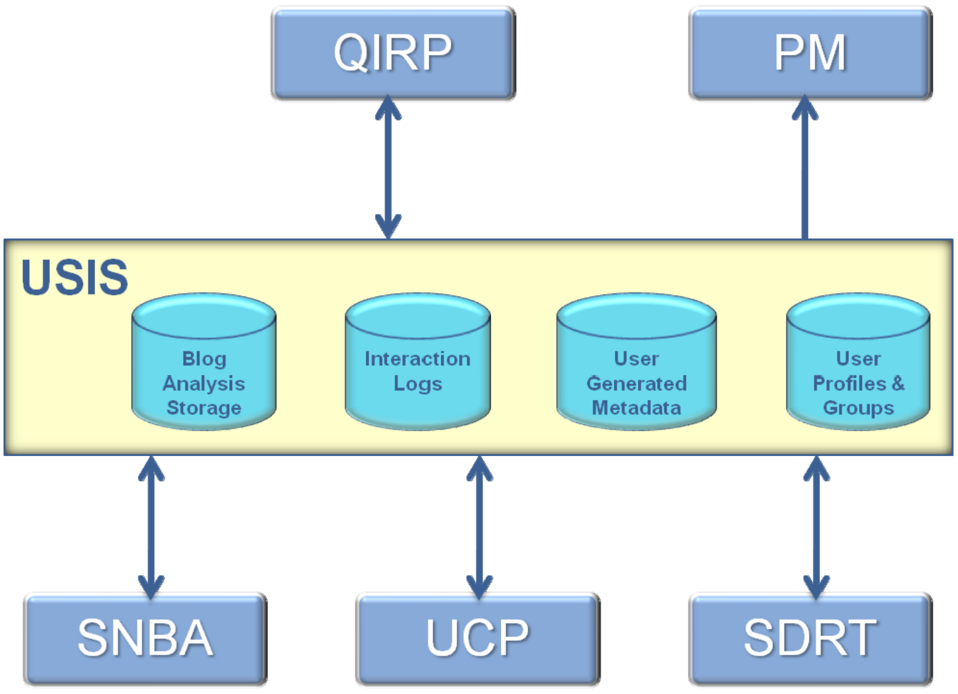


Figure 2-2: Architecture of the User & Social Information Storage (USIS)

Five components are interacting with the USIS, two online and three offline analysis modules:

Online modules:

- *Query Interaction and Results Presentation (QIRP)* – Functions as a bridge between USIS and UI for presenting user data and metadata to users
- *Personalization Module (PM)* – Provides personalized search and recommendations functionalities based on data retrieved from USIS

Offline analysis modules:

- *Social Networks and Blogspace Analysis (SNBA)* – Stores and analyzes data about social networks and blogs (from external sources) and combines it with the data about PHAROS users
- *User and Community Profiler (UCP)* – Creates and extends user profiles based on data from PHAROS users as well as from data external to PHAROS (e.g. external collaborative tagging systems like del.icio.us)
- *Spam Detection, Reputation and Trust (SDRT)* – Analyzed data entered by users and computes reliability scores and if necessary marks data or users as spam

Four major storage components reside inside the USIS:

- *Blog Analysis Storage* – First, blogs gathered from different sources will be stored here. These blogs will be further processed, and analyzed in detail by SNBA extracting interesting features and statistics needed by other components (e.g. UCP). The results of the analysis will also be stored in the Blog Analysis Storage. These results can then be accessed by UCP in order to update the profiles of the users with additional information from this external data source.
- *Interaction Logs* – User actions related to querying, receiving recommendations, and result handling will be monitored by QIRP (with some hooks in the UI) and will be stored at the end of a user session in the USIS. Information stored here includes: what query was entered by the user, what results were viewed, what results were clicked, if the visualization of the results was interrupted prematurely (e.g. a video was not viewed until the end), etc. All this information will be used by UCP in order to create and update the preferences of the user regarding resources and resource types (video, music, images).
- *User Generated Metadata* – All user generated metadata will reside in this storage component. This includes tags, comments, ratings and favourites. Information will be stored as to which user added what metadata to which content object. In this way the metadata can be filtered and statistics can be computed regarding a user, a piece of metadata (e.g. some specific tag), or a content object. Metadata will be entered by the user in the UI, passed to the QIRP, which in turn stores it in USIS. User Generated Metadata will be available for use by SNBA in case more data about blogs can be extracted from here, UCP in order to compute user profiles based on metadata, SDRT for marking metadata or users as inappropriate (spam), and not least by QIRP to present content objects accompanied by all relevant metadata.
- *User Profiles & Groups* – User profiles, user-user relationships, and user-group memberships will reside here. User profiles are constructed initially from the personal data a user enters when s/he creates his/her profile. UCP will add more data to the profile as the user starts using the platform. As sources for improving the user profile, UCP can use blog analysis results, the interaction logs of the user and the metadata the user has added to his preferred content objects. The interactions a user has with other users will also be stored in User Profiles & Groups. This includes manually adding users as “friends” or automatically computing similarities between users based on different criteria (collaborative filtering based on the content objects, similarity of the used tags, etc.). Also group memberships are a part of User Profiles & Groups: A user can create groups which other users may join. This way social networks inside PHAROS will be supported.

In the following we describe the services provided by USIS and the correspondences to the different modules served by them.

Services Provided

Following services are being provided to external modules:

- *Metadata Service* – Service for storing and retrieving information about tags, comments, ratings, and favorites;
- *User Data Service* – Storing and retrieving user profiles and PHAROS community data (friends of a user, similar users, social group assignments);
- *Analysis Results Service* – Service for storing and retrieving results from the Analysis modules (e.g., storing results performed on the tags a user was using in a collaborative tagging system external to PHAROS);
- *Personalization Service* – Providing a succinct representation of the user profile needed for personalizing search and presenting recommendations to a user;
- *Search Service* – Service for providing sets of content object IDs that comply to a certain criteria (e.g., retrieving a list of content object IDs that are all tagged by one or more specific tags).

These services are intended to be used by different components according to needed functionalities. The following table (Table 2-1) presents these relations between services and components.

Service Provided	Components Served	Functionality Description for the Component
Metadata Service	**QIRP**	- Store metadata (tags, comments, ratings, favorites) entered by the user for particular content objects - Retrieve stored metadata and present it to users
	UCP	- Retrieve metadata for a particular user and add it to the user profile
	SDRT	- Mark inappropriate metadata as spam
User Data Service	**QIRP**	- Store basic user data (provided manually by the user) in the user’s profile - Retrieve the user profile and present it to the user - Store the interaction of the user with the system as a structured log at the end of a user session - Retrieve the user’s manually created “friends” and present them to the user - Retrieve the automatically created similar users (“neighbors”) and present them to the user - Permit the user to create or join a social group - List all social groups to the user
	UCP	- Retrieve the user profiles for analysis - Store additional information (computed also using other sources) to the user profiles - Automatically create social groups based on user similarities - Automatically create lists of similar users for a user
	SNBA	- Analyze user profiles and create social groups if the users show the same characteristics as users in other (external to PHAROS) social networks, or as previously analyzed bloggers
	SDRT	- Analyze user profiles and user relationships in order to

		identify spammers
Analysis Results Service	**UCP**	- Store analysis results related to users and communities - Retrieve previously stored analysis results
	SNBA	- Store analysis results related to social networks and blogs - Retrieve previously stored analysis results
Personalization Service	**PM**	- Retrieve the part(s) of a user profile needed in order to perform personalized search or recommendations
Search Service	**QIRP**	- Provide search or filtering by metadata (e.g. retrieve objects tagged with a specific tag)

Table 2-1: Services provided by USIS and components served

Methods Provided for Services

Table 2-2 presents the methods provided by USIS within each service. Note that although at this point the services and methods are stable, methods might evolve over time as additional / different functionalities might be needed.

Service	Method and description
Metadata Service	**Tagging:**
	`void` **`addStringTagToObject`**`(String tag, String objectID, long userProfileID)` Conveniency method - calls addTagToObject
	`void` **`addTagToObject`**`(Tag tag, String objectID, long userProfileID)` Add a tag to a content object
	`void` **`deleteTagFromObject`**`(Tag tag, String objectID, long userProfileID)` Delete a tag from an object. Only the user that created the tag can delete it.
	`void` **`addTagToSiteObject`**`(Tag tag, String objectID, long userProfileID, String siteName)` Add a tag to a content object given a tag, the content URI, the user/profile ID and the name of the site/system (like PHAROS, Lastfm).
	`List<Tag>` **`getUserTagsForObject`**`(String objectID, long userID)` Get tags from a specific user for a content object
	`List<Tag>` **`getAllTagsForObject`**`(String objectID)` Get all tags for a given content object
	`HashMap<String,List<Tag>>` **`getAllTagsForObjects`**`(List<String> objectIDs)` Get the tags for a list of content objects. Conveniency method to be called only once instead of calling multiple times getAllTagsForObject
	`List<Tag>` **`getMostPopularTags`**`(int number)` Get a list of most popular tags used in PHAROS

	Comments:
	void **addCommentToObject**(String comment, String objectID, long userProfileID) Add a comment to a content object
	void **deleteComment**(long commentID) Delete a comment
	void **modifyComment**(long commentID, String comment) Modify a comment
	List<Comment> **getCommentsForObject**(String objectID) Get a list of comments for the given PHAROS object ID
	List<Comment> **getCommentsForUser**(long userID) Get a list of comments for the given PHAROS user id
	Ratings:
	void **addRatingToObject**(double rating, String objectID, long userProfileID) Add a rating [-5..5] to an object
	double **getAverageRatingForObject**(String objectID) Retrieve the average rating for an object
	Spam:
	void **flagObjectAsSpam**(String objectID, long userProfileID) Flag a content object as spam
	Favorites:
	void **addPharosContentAsFavorite**(String objectID, long userProfileID) Add a PHAROS object as favourite
	List<String> **getFavoritesForUserProfile**(long userProfileID) Get all favourites for this user profile
	List<String> **getFavoritesForUser**(long userID) Get all favourites for this user
User Data Service	**User Behaviour Logging:**
	void **storeUserActionsLog**(long userProfileID, String userActionsLog) At the end of a user session, receive all logged information about user actions.
	Users:
	void **deleteUser**(long userID) Notify deletion of user accounts to the USIS
	User Profiles:
	long **addUserProfile**(UserProfile userProfile) Create a new profile for a PHAROS user - Explicit user profile provided by the user
	void **deleteUserProfile**(long userProfileID)

	Delete a user profile
	List<Long> **listUserProfiles**(long userID) Get a list of all profiles for this user
	UserProfile **getUserProfile**(long userProfileID) Get a specific user profile
	void **setUserProfile**(UserProfile userProfile) Update the user profile in USIS
	UserProfile **getCurrentUserProfile**(long userID) Get the current / last used user profile for a user
	void **setCurrentUserProfile**(long userProfileID) Set the current user profile for a user
	List<String> **getUserProfileFields**(long userProfileID, List<String> fields) Get only specific fields of the user profile
	void **setUserProfileFields**(long userProfileID, Map<String,String> fields) Update only specific fields of the user profile in USIS
	Friends and Neighbours:
	void **addFriend**(long sourceUserProfileID, long targetUserID, double preference) Add a user as friend - any user can have a list of friends. The friends are actually attached to a user profile instead of to the user directly. A preference value is give [-5..5] to measure the relationship
	List<UserForUserPreference> **listFriendsUserProfile**(long userProfileID) Get a list of all manually assigned friends of this user profile
	List<UserForUserPreference> **listFriendsUser**(long userID) Get a list of all manually assigned friends of this user for all his/her profiles
	List<UserForUserPreference> **listNeighborsUserProfile**(long userProfileID) Get a list of all automatically computed neighbors of this user
	List<UserForUserPreference> **listNeighborsUser**(long userID) Get a list of all automatically computed neighbors of this user
	List<Long> **listNeighborsLastFm**(long lastFmUserID) Get a list of all last.fm neighbors that are in our users dataset
	Social Groups:
	long **addGroup**(String groupName, long userProfileID) Add a social group to PHAROS

	`void deleteGroup(long groupID)` Delete a social group in PHAROS
	`Group getGroup(long groupID)` Get a social group
	`void setGroup(Group group)` Modify a social group
	`List<Group> listAllGroups()` Get a list of all possible social groups
	`List<Group> listAllGroupsForUser(long userID)` Get a list of all groups a user is involved in
	`List<Group> listAllGroupsForUserProfile(long userProfileID)` Get a list of all groups a user profile is involved in
	`void addMemberToGroup(long groupID, long userProfileID)` Add a member to a group
	`void deleteMemberFromGroup(long groupID, long userProfileID)` Delete a member from a group
Analysis Results Service	`void storeTagAnalysisResults(long userID, String siteDomainName, List<AnalyzedUserTag> analyzedUserTags)` Storing the analyzed user tags (in PHAROS or other sites) to the USIS
	`List<AnalyzedUserTag> getTagAnalysisResults (Long userID, String siteDomainName)` Method for retrieving analyzed user tags stored in USIS (e.g. for visualization)
	`List<AnalyzedUserTag> getAnalyzedUserTags(long userID)` Get a list of tags already analyzed
	`List<AnalyzedUserTag> getAnalyzedUserDomainTags(long userID, String siteName)` Get a list of tags already analyzed from a different domain
Personalization Service	`String getPersonalizedSearchUserProfile(long userID)` Return succinct String representation of relevant part of the user profile for personalized search
Search Service	`Map<String,Integer> searchByTag(String tag)` Search USIS to find objects tagged with the given tag

Table 2-2: Methods provided to each service within USIS

2.1.3 *Underlying Storage Description*

USIS is backed up by an Oracle database. Following is a list of all tables with all fields used to store the previously described data in USIS (* marks a private key, → shows a relation to another table).

UserProfile	One of the profiles for a user	
* profileID	Integer	User profile ID
userID	Integer	user ID the profile belongs to
reputation	Real	trustworthiness of the user: 0(banned) … 1(default) … 10 (good user)
contextID	Integer	→ UserContext.contextID – current context
profileName	String(50)	enumeration – work, leisure, …
email	String(100)	email address
homepage	String(200)	URL of homepage
foaf	String(200)	URL of FOAF metadata
lastfmName	String(100)	username of Last.fm
deliciousName	String(100)	username of del.icio.us
flickrName	String(100)	username of Flickr
birthDate	Date	date of birth
gender	Boolean	gender 0=female, 1=male
country	String(50)	country
city	String(50)	city
registrationDate	Date	date of registration of this profile on PHAROS
language	String(50)	preferred language of the user

UserGroup	Social group of PHAROS users	
* groupID	Integer	Group ID
groupType	String(50)	enumeration – external (lastfm, treehugger, …), internal (friends, automatically generated)
groupName	String(100)	name of the group (crazy people, rockers, …)

GroupMembership	Manage group memberships: what user (by his user profile) belongs to what group(s)	
* groupID	Integer	→ UserGroup.groupID
* userProfileID	Integer	→ UserProfile.profileID

UserContext	Full history of contexts from which the user log in	
* contextID	Integer	
device	String(100)	device the user uses (laptop, pocket pc, …)
country	String(50)	country of access
city	String(50)	city of access
location	String(500)	additional access location details
gpsCoordinates	String(50)	GPS coordinates
ip	String(25)	IP of used device
timestamp	Date	timestamp of login

UserForUserPreference	The preference of one user for another user	
* sourceProfileID	Integer	→ UserProfile.profileID – the profile of the user setting the preference
* targetProfileID	Integer	→ UserProfile.profileID – the profile for which the preference is set
preference	Real	-5(dislike) to 5(love)
preferenceType	String(50)	type of the preference: manually assigned, automatically inferred

Favorites	Favorite URL, PHAROS content	
* favoriteID	Integer	
profileID	Integer	→UserProfile.profileID – profile of the user having the favorite
uri	String(200)	URL, PHAROS content ID
favoriteType	String(50)	enumeration: URL, PHAROS content

generatedType	String(50)	enumeration: manually, automatically

UserUserInteraction	**List of interactions between two users**	
*** userUserInteractionID**	Integer	
sourceProfileID	Integer	→ UserProfile.profileID
targetProfileID	Integer	→ UserProfile.profileID
timestamp	Date	time of the action
actionType	String(50)	enumeration: invite, add, remove, comment, tag, isSpammer
groupID	Integer	→ UserGroup.groupID
message	Text	message sent with the interaction

UserGroupInteraction	**List of interactions of a user with a group**	
*** userGroupInteractionID**	Integer	
sourceProfileID	Integer	→ UserProfile.profileID
targetGroupID	Integer	→ UserGroup.groupID
timestamp	Date	time of the action
actionType	String(50)	enumeration: join, leave, post
message	Text	message sent with the interaction
accessType	String(50)	enumeration: invitation, manually detected, automatically recommended

PlatformNavigation	**Clicking on buttons, not resources – capture what happens in the UI**	
*** navigationID**	Integer	
timestamp	Date	time of the action
profileID	Integer	→ UserProfile.profileID - which user did the action/navigation
clickedLink	String(1000)	the link clicked to navigate
UserComment	Comments made by users on content or users	
*** commentID**	Integer	
profileID	Integer	→ UserProfile.profileID
timestamp	Date	time of the action
commentText	String(4000)	the comment text
contentURI	String(200)	PHAROS content ID

Ratings	**This stores the rating of the users for objects inside the platform**	
*** ratingID**	Integer	
profileID	Integer	→ UserProfile.profileID
timestamp	Date	time of the action
contentURI	String(200)	PHAROS content ID
rating	Real	-5 … +5
generatedType	String(50)	enumeration: manual, automatic
isSpam	Boolean	marking the content as spam

AlternativeTag	**Maintains alternative (Synonyms) for tags**	
tagID	Integer	Tag.tagID
alternativeName	String(50)	alternative tag name (different writing)
occurrences	Integer	nr. of times this pair co-occurred
probability	Real	0…1, specifies how likely is that this alternative tag name is suitable for this tagID

AnalyzedTag	**Stores the tags assigned by users outside the PHAROS platform (e.g. delicious tags)**	
*** tagID**	Integer	
name	String(50)	tag string
generatedType	String(50)	enumeration: extracted, user generated

UserTagItem	Stores the tags that the users have used in the PHAROS platform	
*userTagItemID	Integer	
tagName	String(50)	the tag string
profileID	Integer	→ UserProfile.profileID
timestamp	Date	time of tag assignment
contentURI	String(200)	PHAROS content ID tagged
accuracy	Real	-1 (spam), 0 (default, unknown) … 1 (highly accurate) how accurate is this tag for this content

TagTagOverallCooccurrence	Stores analysis on tags regarding tags co-occurrences.	
* cooccurrenceID	Integer	
sourceTagID	Integer	→ AnalyzedTag.TagID
targetTagID	Integer	→ AnalyzedTag.TagID
frequency	Integer	how often they occur together

ProfileTag	Tags associated to a user profile	
* profileTagID	Integer	
profileID	Integer	→ UserProfile.profileID
tagID	Integer	→ AnalyzedTag.tagID

TagSiteDomain	Tags associated with tagging sites	
* siteDomainID	Integer	
siteDomainURL	String(200)	Url of the site where the tag was used
tagID	Integer	→ AnalyzedTag.TagID
frequency	Real	absolute frequency

Query	Stores queries used in the Platform	
* queryID	Integer	
queryText	String(1000)	actual query
queryObject	String(200)	contentURI (for the case of a query-by-example)

UserQuery	Stores user queries	
* userQueryID	Integer	
profileID	Integer	→ UserProfile.profileID
timestamp	Date	timestamp of the query
queryID	Integer	→ Query.queryID

ResultSet	Stores the list of results returned by a given query for a given user	
* resultSetID		
profileID	Integer	→ UserProfile.profileID
queryID	Integer	→ Query.queryID
timestamp	Date	timestamp of the results returning
contentID	String(200)	PHAROS content object ID
rank	Integer	rank of the contented in the result set

QueryResultClicked	User clicks on results (content objects) of queries	
*queryReClID	Integer	
profileID	Integer	→ UserProfile.profileID
queryID	Integer	→ Query.queryID
timestamp	Date	timestamp of the click
contentID	String(200)	PHAROS content object ID
rank	Integer	rank of the contented in the result set

2.2 UCP – User & Community Profiler

2.2.1 Description

In order to achieve extreme precision in ranking and recommending multimedia content, adaptation of the core technology to user preferences and specific user contexts is necessary. Since most users may be unwilling to explicitly fill in and maintain a personal profile or they might not be able to specify an accurate profile, automatically inferring interests is important. Moreover, for providing high quality personalized services, profiles must be kept up-to-date as interests may change over time. Accurate user profiles often also depend on the community[1] a user belongs to. Therefore, inferring user profiles has to be complemented with the construction of community profiles. The **User & Community Profiler (UCP)** component is in charge of modeling user preferences both inside and outside the PHAROS platform by collecting and automatically inferring information about users and communities they belong to. To overcome problems associated with modeling and finding communities adequately, it will build upon a model of user and community actions and interactions in social networks (SNBA).

This module takes advantage of the explicit profile information freely provided by the PHAROS users and at the same time extends it with publicly available profile data from the services indicated by the user. Moreover, interests or preferences are implicitly found in concrete user (inter)actions and can thus be modeled from logging user interactions and group behavior within the PHAROS platform. Given this diverse amount of (raw) data about users and communities, the challenge and main focus of research activities within this module is to develop advanced techniques for building user and community profiles detailed, recent and accurate/reliable enough to meet the challenges of precise personalized and context-specific retrieval and recommendation.

In detail, the user or community profiles will comprise:

- Explicit user information, given in the account/my profile section in the User Interface, and will provide basic data about users (gender, age, language) as well as some general interests;
- User generated metadata: Annotations (tags, comments or bookmarks) made within the PHAROS platform, as well as favorite tags and resources in external Web 2.0 platforms like last.fm, del.icio.us or Flickr will be analyzed and aggregated to infer preferences;
- A user's blog data or aggregated community blog data respectively provided by SNBA will be analyzed;
- Usage history: Issued queries and click through data from the interaction logs will be used to show recently accessed resources ('charts') and will serve as implicit feedback to infer likes and dislikes;
- Friendship or contact relations between users will both inform about similarities or common interests and may be used for restricting recommendations based on privacy concerns;
- Similar, but unknown users (neighbors) will be detected by combining data about users as described above.

Since users may have different preferences with respect to different situations, users can have multiple user profiles comprising the attributes listed – one for each of their various contexts (like 'work', 'leisure' etc.). For effectively supporting distinct user profiles, current active contexts have to be identified accurately to add the information to the right place. However, a default profile giving all information available will be supported. To take into account most recent user and community data, profile updates will be scheduled according to availability of new data. For example, for tagging data within PHAROS updates are planned after every user session.

The following section gives an overview about the general architecture of the module.

2.2.2 Architecture

As an offline module UCP does not offer services, but accesses services to fulfill its task of user profiling. Dependencies only exist with USIS, as all (raw) data needed for profile building will be gathered from USIS for analysis and the modeled profiles will afterwards be written back to USIS to be

[1] We use the term 'community' when referring to any external social network of people e.g. build on platforms like Flickr; SocialGroup is used when we refer to the groups that users are actually building within the PHAROS platform

provided to other services like personalization. Figure 2-3 shows the subcomponents of the module and how they interact. This general overview about the architecture of the UCP and the different analysis steps to be performed, are a good starting point for afterwards looking in more detail into the implementation of those algorithms.

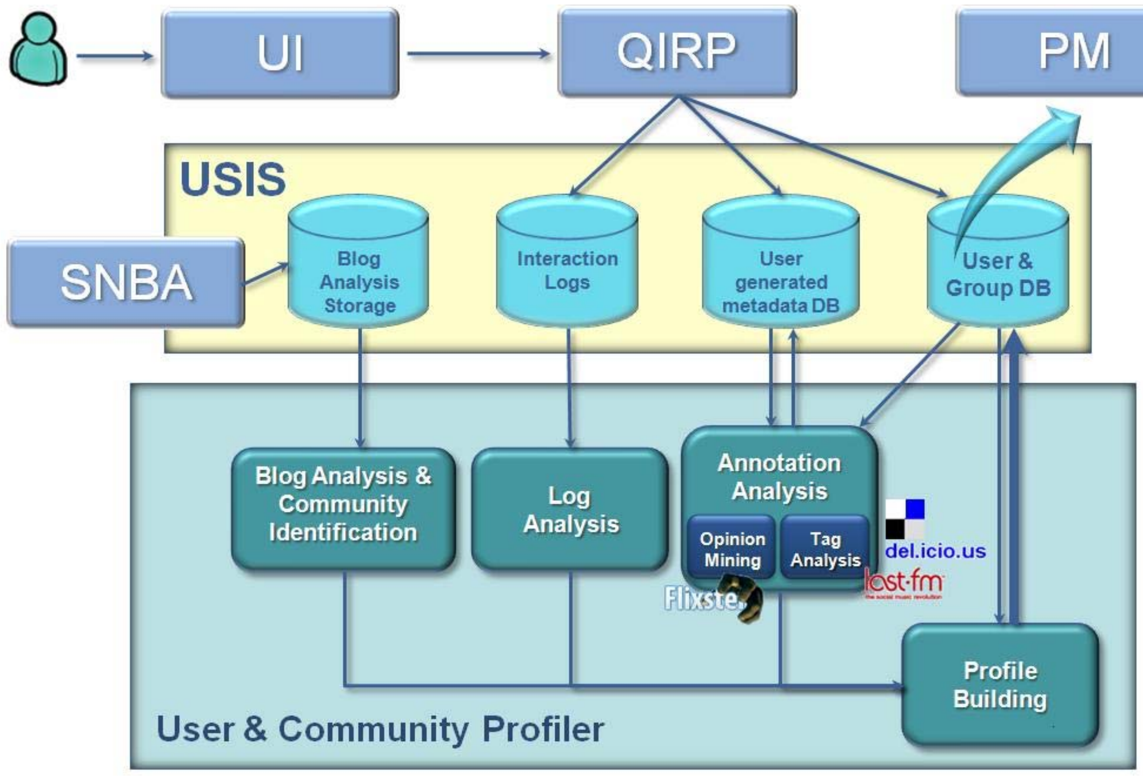


Figure 2-3: UCP architecture and interactions with other modules

All data necessary for analyzing and inferring user or community interests will be retrieved by UCP from USIS. For example, user interaction data will be sent via QIRP to USIS as log files. Similarly, explicit user data and annotations (tags, comments, and bookmarks) assigned to resources and users will go through QIRP to be stored in USIS for later analysis. Moreover, SNBA will perform the Blogspace analysis and store the results in USIS as well. Using the Metadata, User data and Analysis Results Service UCP retrieves these data, processes it and combines the single sources of valuable user information / inferred attributes to have one (or more context dependent) profiles. The profiles will be written back to USIS via the User Data Service in USIS. From there, they will later be returned to the PM to provide the basis for effective personalization.

In detail, the different subcomponents of UCP will be in charge of the following analyses:

Log Analysis

What resources a user searches for, which multimedia resources he actually accesses (for how long), whether he even recommends them to other people, as well as which persons he frequently interacts with, provides a lot of valuable data about a user's topics or preferences and probably typical behavioral patterns. In contrast to explicitly provided profile information such implicit feedback to resources and people comes in form of a huge amount of low level data of single user actions that have to be analyzed to infer meaningful and generalizable user attributes to inform e.g. personalization. Log Analysis as a subcomponent of profiling will retrieve the user logs, created by QIRP, from USIS. The different kind of actions will then be extracted, for example popular queries posted to the system. The user evaluation of the result set (skipped and clicked items) helps to infer new associated terms by getting keywords from resources implicitly judged as relevant. On the other hand, similar resources listened to or watched can be exploited to find similar queries or even super-ordinate topics. In general, just building the list of (recently) seen items - usage history of a user – alone is very useful for profiling interests with respect to finding similar users or similar resources. Interactions within groups and with friends or unknown people will be analyzed to model the social network of a user which can again be used to infer commonalities and preferences. Other patterns to be mined from action sequences (like system internal navigation paths) will help personalizing view settings and creating navigational short cuts as well as to inform about general usability issues to be improved. All these extracted and inferred information will be translated into specific attributes and written to the user profile.

Annotation Analysis

By adding tags, comments or bookmarks to multimedia resources seen within PHAROS, users tell (implicitly) us about what they like, don't like or what topics they are interested in, as they organize their resources around it. Therefore, Annotation Analysis will gather this kind of data and analyze it in depth to make it available for profiling and search.

Opinion Mining

For comments, stopword removal and term normalization are standard procedures, important keyword extraction and Sentiment Analysis / Opinion Mining more advanced techniques. The Sentiment Analysis and Mining module (SAM) aims at analyzing any free textual annotations about movies, such as the comments collected within PHAROS, or posts in forum, blogs, homepages, etc., or elsewhere on the Internet (in particular on the Flixster[2] website if external user account ID is available). The SAM module deduces from these textual annotations new tastes about cinema, and updates the USIS profile with this new knowledge. Such analysis comprises first a Named Entity recognition (what movie/actor/film maker is discussed in the comment); in a second step consists of the classification of the comment by a Machine Learning tool previously trained. In the third and last step we associate a rating to the named entities recognized.

Tag Analysis

For tags, such analysis will first of all comprise normalization and the inclusion of alternative, synonymous labels. Also absolute and relative frequency information will be calculated for individual tags as well as co-occurrence relationships that may help to dissolve term ambiguities or to refine queries by synonyms / strongly associated terms. To exploit the potentially huge effort a user already invested in bookmarking and tagging interesting web pages, songs or pictures in one or the other web 2.0 platform, annotation analysis will also fetch and analyze the user's tags from external sites like last.fm and del.icio.us. Thus, it is not only possible to distinguish preferences for certain media types (like music) but also to find commonalities and complement data from one source with further information about the user. Analyzing and classifying different kinds of tags is a further step to exploit especially useful tags (only).

Both the new cinema tastes and analyzed tag information will be written back to USIS. The user will be able to edit this tastes as well as her tag profile. For example, a visualization of tag profiles (tags, frequencies and co-occurrences) easily editable and adjustable will be displayed to the user within the UI. Here, a user can for example remove a tag completely or reduce its weight as it may not describe his current interests (in a particular context) very well. The updated tag information will be considered in next analyses phases. Besides advanced personalized search and recommendation, such tag profile data can be exploited to provide user generated keywords for query suggestion/auto completion. In the same way, a user may want to remove a particular rating, or refine the automatic-rating of a movie and complement the rating with tags.

Also related to users and communities interests is the analysis focusing on **Blogs and Communities Identification**. This analysis will however be undertaken in the Social Networks and Blogspace Analysis Module presented in Section 2.3 and will be centered around:

- Opinion mining, topic extraction for a single user, network extraction
- Community identification via adaptation of blogspace information diffusion

[2] http://www.flixster.com/

Profile Building

Finally, the results of the above mentioned single analyses have to be merged and enriched with the information explicitly given by the user. The basic (demographic) user data, his account information and the interests specified (for various contexts) will be retrieved from USIS via the user data service to be included into the profile. The USIS itself received this information from QIRP. Both types of information will go directly into the profile under the corresponding attributes, and they will complement the information automatically inferred as well. For example, this may mean merging or resolving conflicts about (the recency of) preferred topics identified in the single analyses. The final profile will then be written back to USIS to be used by other modules / services e.g. for personalization.

The next section describes in detail the algorithms currently available in UCP. Please note that algorithms for log file analysis and elaborate profile building will be developed after first usage data is collected from the showcase (after M18).

2.2.3 Algorithms

a. Creating tag based user profiles

The motivations for creating a tag based user profile and the corresponding algorithm is described in detail in D2.1.1. Here we give a short summary of the algorithm and a description of the updates to the algorithm, as well as a description of the implemented visualization of the tag based user interest profile.

2.2.3..1 Building a tag based user profile

The tag based user interest profile includes the user's interests defined as a collection of tags and their co-occurring tags with corresponding information about the tag, and tag co-occurrence usage frequencies. The interests of the users are ranked with the help of the tag usage frequencies. The more a tag is used, the more interested the user is assumed to be in the topic. The co-occurrences of tags indicate which aspects the user is interested in regarding the topic. The relative cumulative tag frequency is used for determining the threshold that is used for selecting tags for a user profile. This approach allows us to take into account the varying tagging habits of different users.

For analyzing user interests, our focus is on tags that have a recognized lexical class. Lexical analysis is used for cleaning and normalizing the user's tags. For doing the lexical analysis we used the "MIT Java WordNet Interface11" – library to categorize the used tags and we utilized the stemmer included in the library for lexical classification. To obtain the word root form from a tag, a stemming is applied. In the stemming plurals will be reduced to singular form ("blog" to "blogs"), and derived words can be combined into one ("blog" into "blogging"). Using WordNet (WordNet), normalized words are classified according to their lexical class (noun, adjective, and verb). For creating a user interest profile nouns are the most useful and descriptive.

The algorithm is updated to support tagging data from different services like del.icio.us, Last.fm and PHAROS itself. The APIs of the existing social media services are utilized for fetching the data. The example procedure for fetching and analyzing the del.icio.us tags is described below.

When the user provides in the PHAROS profile page a username for her del.icio.us account, the following sequence is executed using the del.icio.us JSON APIs:

- Fetch and store data about the 100 last tagged resources along with their tags from http://del.icio.us/feeds/json/<username>?params
- Fetch and store tag popularity data from http://del.icio.us/feeds/tags/<username>

This information is stored in a database for additional analysis. All the user's tags go through the following steps:

- Exclude non English words, and perform stemming for getting the word roots of the tag.
- Check WordNet to get the lexical class (noun, adjective, verb) of the tags.
- Store the resulting tag-tagroot-lexicalclass combination into the database.

After this process, the tags are analyzed and ranked to create the user's tag based interest profile.

For fetching users' tags from Last.fm the audioscrobbler API is used (Audioscrobbler).

The methods have been created for combining the results from different services in order to have one combined user profile. The result is visualized by the user as a tag cloud and the user is able to modify the profile. The annotation analysis uses its own database and it has been integrated as part of the Social media beta prototype using web service APIs.

The formal definition of the algorithm and steps relating to it are described below.

The profile of a user U can be formally defined as follows:

Profile (U)= {<Tag_i, F_i , RF_i , rel $_i$ <CoT_j, CF_j, rel $_j$>> }

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* F_i /Total frequency of all the user's (U) tags
- CoT_j = co-occurred tag of Tag_i
- CF_j = absolute tag usage frequency for co-occurred tag CoT_j
- rel $_{i,}$ = relevance of the tag expressed by a user U. Three level scale is used for the relevance (high, normal , none). The default value is normal. The numerical value of the relevance represents a scaling factor; 0 for none, 1 for normal, 2 for high.
- rel $_j$ = relevance of the co-occurred tag of Tag_i expressed by a user U.

***Steps of the algorithm*: c**reating a User Tag profile (steps 5 to 7 updated from D2.1.1)

1. Count tag usage frequencies (absolute and relative frequencies as well as relative cumulative tag usage frequency) for recognized tags, in other words 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 into the user profile. This threshold for the times a user has used a tag is used as criterion for including tags into the profile.

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 the 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. Repeat the steps 1-4 for tagging information from the different sites that the user has an account in. (Currently, del.icio.us, Last.fm and PHAROS have been implemented)

6. Visualize tag based interest profile to the user as a tag cloud. The visualization of the profile combines the results of the tag interest profiles from different sites.

7. Update the user profile based on the user input. The user is able to modify the profile: change the label of the tag, the relevance of the tag or the co-occurrence relationship weights and add other related tags.

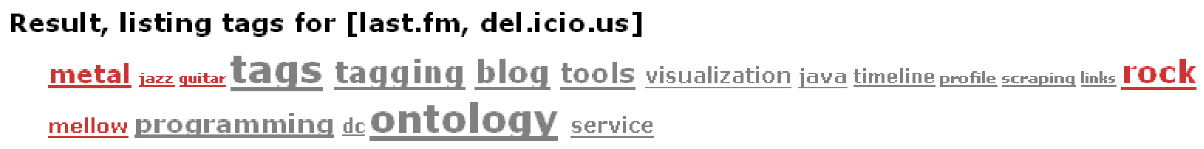
***Output*:** return user profile and update result to the USIS.

2.2.3..2 Visualization

The profile is visualized by the user (see Figure 2-4) and she is given an opportunity to modify it (see Figure 2-5). Only the tags that have a known lexical class have been selected for the profile and relative cumulative tag usage frequency has been used to select threshold for tags which were included in the user's profile. Because the original co-occurrence relations have been maintained, the tag label given by the user is shown instead of the analyzed word root of the tag. In future work, the different aspects relating to showing the word roots instead of the original label will be compared.

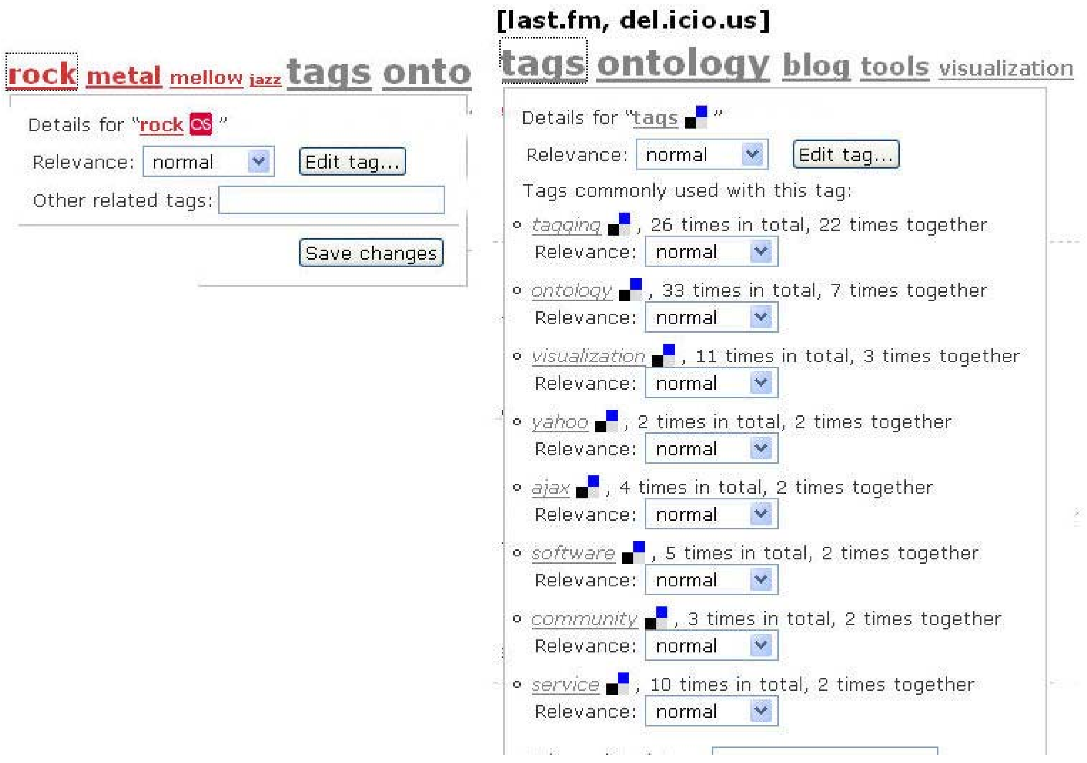
The visualization is implemented as a traditional tag cloud where the size of the tags implies the tag usage frequency and hence the importance to the user. The tags are obtained from the sites that the user indicates as his or her accounts. In the visualization, the tag origin is indicated using colors (and icons). By clicking or hovering with the mouse over the tag a pop-over widget appears. In the widget the user can see related tags and is able to modify the information that reflects the user's interest profile. The user is able to reduce or increase the relevance (weight) of the tag or the relevance of the co-occurrence relationships. The relevance is a three level scale; "normal", "high" and "none" with a default value of "normal". If the user feels that the tag is not describing his/her interests anymore, "removing" the tag is made by selecting the relevance "none". He can also modify/correct the label of the tag and add other tags that are closely related to the tag from his point of view. Information about the tag and tag co-occurrence usage frequencies are shown to user. The implementation is AJAX-based.

Since not all social media service APIs provide enough information to derive the co-occurrence, the visualization has to cope with different types of data. Figure 2-5 shows examples of modifying tags in one case where information about the co-occurring tags is available and in another where it is not.


Result, listing tags for [last.fm, del.icio.us]

metal jazz guitar tags tagging blog tools visualization java timeline profile scraping links rock mellow programming dc ontology service

Figure 2-4: Visualization of a user's tag interest profile as Tag Cloud. Colors indicate tags from different services (red - Last.fm, grey –del.icio.us).


rock metal mellow jazz tags onto

Details for "rock "

Relevance: normal | Edit tag...

Other related tags:

Save changes

[last.fm, del.icio.us]

tags ontology blog tools visualization

Details for "tags "

Relevance: normal | Edit tag...

Tags commonly used with this tag:

- tagging , 26 times in total, 22 times together
  Relevance: normal
- ontology , 33 times in total, 7 times together
  Relevance: normal
- visualization , 11 times in total, 3 times together
  Relevance: normal
- yahoo , 2 times in total, 2 times together
  Relevance: normal
- ajax , 4 times in total, 2 times together
  Relevance: normal
- software , 5 times in total, 2 times together
  Relevance: normal
- community , 3 times in total, 2 times together
  Relevance: normal
- service , 10 times in total, 2 times together
  Relevance: normal

Figure 2-5: Modification of tags a) for Last.fm and b.) for del.icio.us.

In Figure 2-5, the first example shows modification of a Last.fm tag where co-occurrence information was not available, and the second example modification of a del.icio.us tag with co-occurring tags.

The tag cloud presentation was selected because it is quite commonly utilized and it is an easy and illustrative way of describing the usage of tags. Another option was to use a graph visualization, but especially as co-occurrence information of tags in not available on every site, a tag cloud presentation was preferred. The strength of the graph visualization would be the visualization of the co-occurrences of tags. Possible future work would be to support different visualization approaches, so that users are able to select the visualization that they prefer. The visualization will be evaluated in a separate use study in connection to another project at VTT. The user test will utilize the VTT "Open web lab" Owela (http://owela.vtt.fi/).

The support for grouping tags into different profiles like work or leisure profiles will be considered in the next version. It is also important to show users the concrete benefits of creating and maintaining the interest profiles. In addition to more personalized and accurate search results and recommendations, these tags could be utilized as keyword suggestions in searching or in browsing the content.

b. Analyzing tag behavior to exploit relevant tag types

The tags users generate in various tagging platforms represent quite a few different aspects of the resources they describe and it is not obvious, whether and how these tags or subsets of them can be used for search and recommendation. Users' motivations for tagging resources, as well as the types of assigned tags differ across systems, and their potential to improve search remains unclear, despite initial investigations. First studies have started to investigate tagging motivations and patterns, usually for one specific collection, including some initial work on how to support the tagging process and improve information retrieval algorithms in general using tags. There are no studies so far investigating these questions across different collections, and there is only limited research regarding the usefulness of tags for search.

Thus, we conducted an in-depth study of tagging behavior for very different kinds of resources and systems (Bischoff, Firan, Nejdl, & Paiu, 2008) – Web pages (Del.icio.us), music (Last.fm), and images (Flickr) – and compared the results with anchor text characteristics. We analyzed and classified sample tags from these systems, to get an insight into what kinds of tags are used for different resources, and provided statistics on tag distributions in all three tagging environments. Since even relevant tags may not add new information to the search procedure, we also checked overlap of tags with content, with metadata assigned by experts and from other sources. Do tags provide new and reliable information about the content they annotate, or do they just replicate what is already available from content or other metadata? For discussing the potential of different kinds of tags for improving search, we also compared them with user queries posted to search engines as well as through a user survey. The insights gained will be used to enhance tag based user profiling as well as search and recommendation in general by both exploiting 'valuable' tags and by supporting the user in creating potentially search-relevant tags.

2.2.3..1 Tag Distribution across systems

Figure 2-6 presents a comparison of the collaborative tagging systems we analyzed. Usage of tags basically follows a power law distribution for each system. We observe a sharp drop at the end for the Flickr and Last.fm curves, due to the crawling mechanism which focused more on popular tags. Disregarding the exact number of tags (this was dependent on each system's architecture and the crawling methods) we analyzed the slopes of the different systems. A more abrupt slope shows that popular tags are being used more often while tags in the tail have less weight. A more gradual inclination indicates a more even use of tags throughout the collection. The most evenly distributed system is Flickr where people almost always tag only their own pictures, not much influenced by others. For Del.icio.us, influence of others is more visible as the slope gets steeper. Last.fm shows the steepest slope, with a few very popular tags and 60% of the top 100 representing genre information. Last.fm covers a very specific domain – music – which explains why tags are more restricted than in Flickr where images can include everything and in Del.icio.us which has an even broader range of topics (in the ODP[3] catalogue, about 4 million Web sites are filed into more than 590,000 different categories).

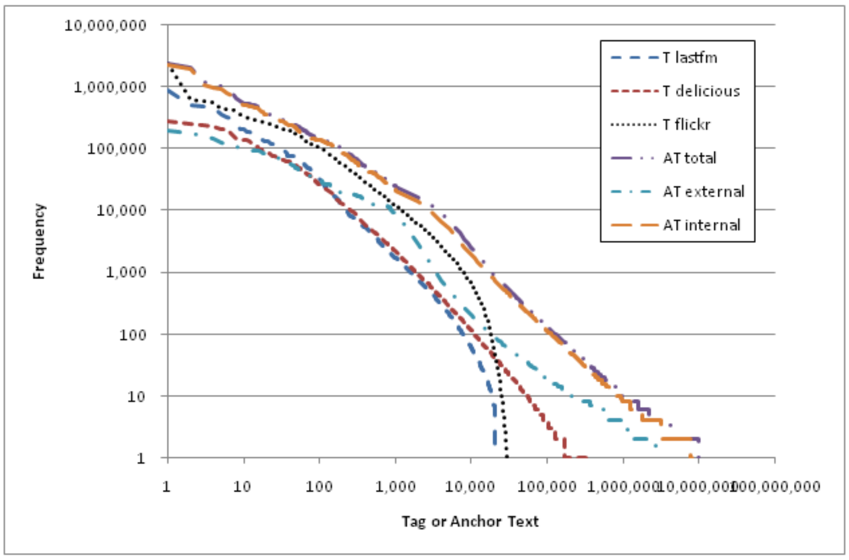


Figure 2-6: Frequency distribution log scale plot comparing tagging systems and anchor text usages

The anchor text (AT) distribution plot shows two visibly different parts with different slopes. The head (top 750 external AT; top 2,000 internal AT) is more even than for all three collaborative tagging systems, while the tail for the external AT is comparable to Del.icio.us tags, and for external AT is not a perfect power law distribution. We think this is mainly due to the fact that in our analyzed sample these top AT point to a small set of very popular Web pages. These are external AT sites like search engines, important news sites and portals, as well as internal AT links to key pages for the Web site like table of contents, site map and home pages. As the figure shows, all our datasets exhibit power law characteristics, so that even if the sizes of the collections differ, results are still consistent and comparable.

2.2.3..2 Usage of Tag Types in different systems

Tags serve various functions based on system features like resource type, tagging rights, etc. (Marlow, Naaman, Boyd, & Davis, 2006), and not all these tags are equally useful for the community or for interpersonal retrieval (Golder & Huberman, 2006). For being able to improve tag based search, we first need to know how tags are used and which types of annotations we can expect to find along with resources. For this purpose, we propose and use an extended tag taxonomy appropriate for different tagging systems. This builds on and extends previous work, which has discussed classification schemes for tags, restricted however to only a single tagging system or based on very small data samples. We then investigate tag distributions in different collections, based on our tag classification scheme.

Defining tag types: we started with an exploratory analysis of existing taxonomies, as well as possible attribute fields for the different resources to be considered. As a resource can be characterized by different attributes, tag types shed light on what distinctions are important to taggers. We kept and refined the most fine-grained scheme presented by Golder & Huberman (Golder & Huberman, 2006), adding the classes Time and Location, in order to make it applicable to systems other than Del.icio.us, which only focuses on Web page annotation. We went through several iterations to improve the scheme by classifying sample tags and testing for agreement between multiple raters. Our final taxonomy comprises eight classes, presented together with example tags from our datasets in **Table 3-3**. Topic is probably the most obvious way to describe an arbitrary

[3] “DMOZ” Open Directory Project, http://www.dmoz.org

resource, describing what a tagged item is about. For music, Topic was defined to include theme (e.g. “love”), title and lyrics. The Topic of a picture refers to any object or person displayed. While such Topic information can partially be extracted from the content of textual resources, it is not easily accessible for pictures or music. Tags in the Time category add contextual information about month, year, season, or other time related modifiers. This includes the time a picture was taken, a music piece or Web page was produced. Similarly, Location is an additional retrieval cue, providing information about sights, country or town, or the origin of a musician. Tags may also specify the Type, which mainly corresponds to file, media or Web page type (“pdf”, “blog”, etc.). In music this category comprises tags specifying format as well as instrumentation and music genre. For pictures, this includes camera settings and photographic styles like “portrait” or “macro”. Yet another way to organize resources is by identifying the Author/Owner who created the resource (author, artist) or owns it (a music and entertainment group like Sony BMG or a Flickr user). Tags can also comment subjectively on the quality of a resource, expressing opinions based on social motivations typical for free-for-all-tagging systems, or are simply used as rating-like annotations for easing personal retrieval. Usage context tags suggest what to use a resource for, or the context/task the resource was collected in and grouped by. These tags (e.g. “jobsearch”, “forProgramming”, etc.), although subjective, may still be a good basis for recommendations to other users. Last, Self reference contains highly personal tags, mostly helpful for the tagger herself. For comparison, we applied this tag classification scheme also to our AT collection – defining Self reference in terms of site internal and system-reference comprising frequent navigational AT pointing to pages within the domain or sections of a Web page.

Nr.	Category	*Last.fm*	*Flickr*	*Del.icio.us*	*AT*
1	Topic	*love, revolution*	*people, flowers*	*webdesign, linux*	*health, security*
2	Time	*80s*	*2005, july*	*daily, current*	*previous years, tomorrow*
3	Location	*england, african*	*toronto, kingscross*	*slovakia, newcastle*	*great lakes region, nederlands*
4	Type	*pop, acoustic*	*portrait, 50mm*	*movies, mp3, blogs*	*pdf, books*
5	Author/Owner	*the beatles, wax trax*	*wright*	*wired, alanmoore*	*musicmoz.org, elcel technology*
6	Opinions/Qualities	*great lyrics, yum*	*scary, bright*	*annoying, funny*	*mobile essentials*
7	Usage context	*workout, study, lost*	*vacation, birthday, science*	*review.later, work, travelling*	*event planning, research, entertainment*
8	Self reference	*albums i own, seen live*	*me, 100views*	*sonstiges, frommyrssfeeds*	*about us, home page*

Table 3-3: The classification taxonomy, applicable to different tagging systems

Distribution of tag types across systems: for the three different tagging systems as well as for our AT collection, we took three samples of 100 tags each to be manually classified. These three samples per system included the top 100 tags, 100 tags starting from 70% of probability density (based on absolute occurrences), and 100 tags beginning from 90%. These different samples based on rank percentages were chosen based on the results of prior work [9] which suggested that different parts of the power law curve exhibit distinct patterns. By classifying across systems, our goal was to provide descriptive statistics about tag type usage depending on popularity to formulate appropriate hypotheses based on relative frequencies of distinct tag types. The resulting distributions are shown in **Figure 2-7**.

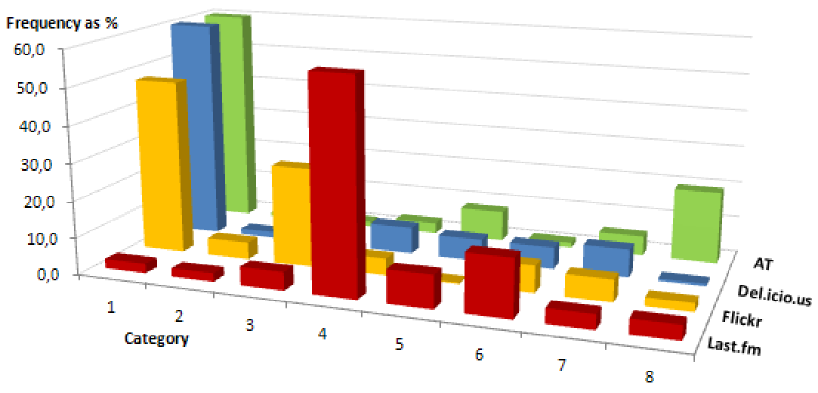


Figure 2-7: Tag type distributions across systems

The most obvious general conclusion is that tag types are very different for different collections. Specifically, the most important category for Del.icio.us and Flickr is Topic, while for Last.fm, the Type category is the most prominent one, due to the abundance of genre tags, which fall into this class. Obviously, genre is the easiest way of characterizing and organizing music – one of the rare exceptions was for the theme “love” and some parts of the lyrics / title. In contrast, a similar dominance can be observed for Topic in case of Web resources and pictures. Type is also common in Del.icio.us, as it specifies whether a page contains certain media. As Flickr is used only for pictures, Type variations only include fine grained distinctions like “macro” – most users do not seem to make such professional annotations. For pictures only, Location plays an important role. Usage context seems to be more used in Del.icio.us and Flickr, while Last.fm as a free-for-all-tagging system (with lower motivation for organization) exhibits a significantly higher amount of subjective / opinion tags. Time and Self reference only represent a very small part of the tags studied here. Author/Owner is a little more frequent, though very rarely used in Flickr due to the fact that people mainly tag their own pictures (Marlow, Naaman, Boyd, & Davis, 2006). For AT, specifying the Topic is the main functional category. External AT are mostly titles of pages or very similar to titles. Self (or system) reference is the second most important function for AT; AT for internal site navigation falls into this category. Time, Location and Opinions/Qualities are rare for AT.

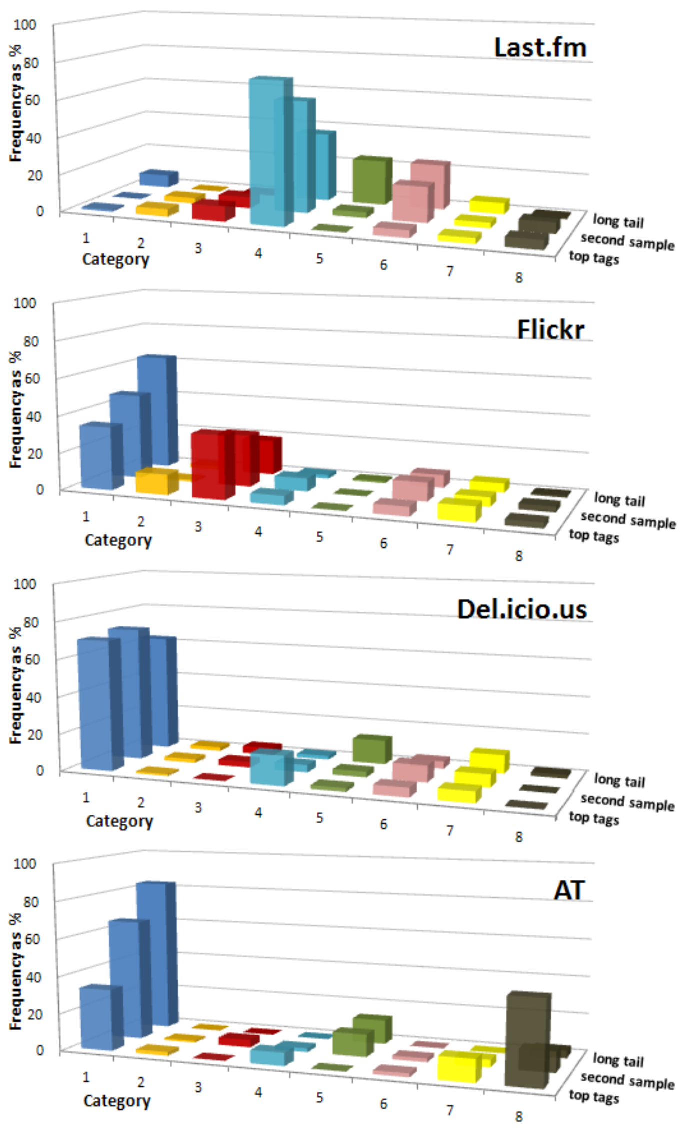


Figure 2-8: Tag type distributions across systems and samples

To better understand how importance of tag categories varies with tag popularity, **Figure 2-8** shows the distributions for all systems across all samples. Again, we observe Type as the predominant tag category for music, while for URLs and pictures it is Topic – mostly increasing across samples. For the long tail of the Last.fm sample, usage of Type category decreases, and opinion expression and artist labeling (Author/Owner) get more important. For AT, internal self- or system-reference decreases in importance for less frequent AT. This is probably related to the fact that the vocabulary for many navigational AT is highly standardized (“home”, “top”) and so highly ranked. The same argument holds for types of linked resources. The type distribution between systems shows a clear tendency of preferred tag functions that do not depend much on the popularity of the tags. With respect to search, it is encouraging to see, that most tags – Topics and resource Type in general, Topic and Location for pictures, and to a certain degree Type for music – are factual in nature, verifiable and thus potentially relevant to the community and other users. Subjective and personal tags (categories 6, 8) are only a minor part (except for category 8 in AT). Opinions/Qualities are only characteristic for social, free-for-all music tagging systems (like Last.fm), possibly because for young people (exposing) music taste is one important aspect in forming one’s own personal identity.

Accuracy of tag classification: clearly, such classification schemes only represent one possible way of categorizing things. To prove the usefulness of our proposed scheme, we evaluated inter-rater agreement, to get a quantitative measure on possible accuracy. From our initial sample we selected 75 tags per system (25 randomly chosen tags per range) plus 75 per anchor tags and had them also assessed by students unfamiliar with the tag categorization scheme. We computed Cohen’s Kappa (κ) (Cohen, 1960) which indicates the achieved inter-rater agreement beyond-chance, as the standard measure to assess concordance for our nominal data. Our raw agreement value for the κ calculation is about 0.79 given the sum of 0.77 for the by chance expected frequencies, resulting in a κ of 0.71 – considered as good and substantial inter-rater reliability. Looking more closely at the values for the individual systems, we found the classification for Last.fm the most consistent with actual agreement of 0.85 and κ value of 0.74 (Flickr: 0.8 and 0.69; Del.icio.us: 0.67 and 0.46; AT: 0.83 and 0.7). We observe that for more constrained systems, concordance seems to be higher. To account for the ambiguity in tag meaning and tag function for certain resources, we gave the rater a chance to name a second category that would fit as well. Taking into account this second possible category for a tag, our κ improved considerably to 0.80 – 0.76 for Last.fm, 0.9 for Flickr, 0.59 for Del.icio.us and 0.75 for AT respectively. Still, it is interesting to investigate how existing tag categorization schemes including ours can be improved further. The confusion matrix created for the κ calculation reveals several prominent confusion patterns for the Del.icio.us tags – always involving the ‘default’ Topic category.

2.2.3..3 Reliability and Information gain of user generated annotations

Given the huge amount of metadata created through collaborative tagging, another interesting question concerns its reliability: is it worth using tags for search, or should we use instead annotations produced by experts? To answer this question, we compared metadata created by experts against metadata produced by communities of users. The music domain is very suitable for this kind of analysis, since there are a lot of online available music reviews for albums, tracks, and artists, produced by human experts. At the same time, on the Last.fm portal, we can find most metadata in the form of tags, assigned to the same kinds of entities (tracks, albums and artists).

Tags in music reviews: in this experiment, we analyzed the overlap between tags assigned to Last.fm tracks and music reviews extracted from Google results for the same set of tracks. From the 317,058 Last.fm tracks in our original dataset, we randomly selected 8,130 tracks, for which we tried to find music reviews by sending queries in the form [*“artist” “track” music review -lyrics*] to Google. For each of the selected tracks we considered the top 100 Google results, and extracted the text of the corresponding pages to create one single document inside which we searched for the tags corresponding to the track. The tag distribution found was linear and 73.01% of the track tags occurred inside review pages. This overlap is rather high, and probably caused by the fact that most of the Last.fm tags represent genre names, which also occur very often in music reviews. Second, we investigated how many of the tags assigned to tracks occurred in the manually created reviews from www.allmusic.com. We randomly selected music tracks from our Last.fm dataset and crawled the Web pages corresponding to their AllMusic reviews. If no review was available for one track, we tried to find the review Web page of the album featuring that track. The resulting dataset consisted of 3,600 reviews. Following the same procedure as for the previous experiment, with reviews crawled from Google results, we found that 46.14% of the tags belonging to a track occurred on the AllMusic review pages. Again the tag distribution we found is linear. We hypothesize that the lower number of matches is due to the fact that AllMusic reviews are created by a relatively small number of human experts, which use a more homogeneous and thus restricted vocabulary than found in arbitrary reviews on the Web. A graphical representation of tag distributions for both Web and AllMusic reviews is given in **Figure 2-9**.

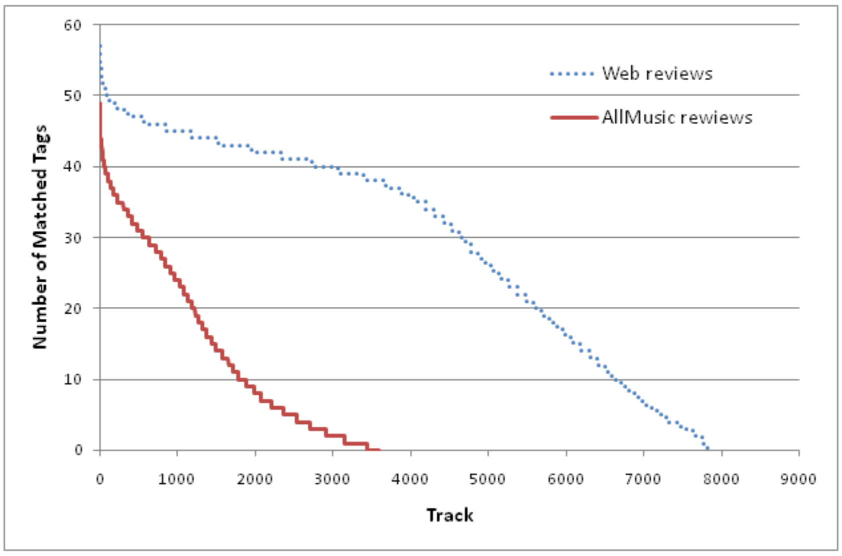


Figure 2-9: Tag distribution in Web and AllMusic reviews

It is also interesting to investigate the added value of tags: do they provide new information on the content they annotate, or just replicate what is already available from the content itself?

Tags and AT in Web pages: from the Del.icio.us crawl we extracted 20,911 URLs for which we had the full HTML page in the WebBase crawl. For these we counted how many tags appear in the Web page text they annotate and found that this is the case for 44.85% of the selected Del.icio.us tags. Comparing how many AT are present in pages they link to, we analyzed 8,614,990 AT and found that 44.7% of external AT and 81.24% of internal AT are already present in the linked page text. Results are thus similar to those of Del.icio.us tags as only external AT should be regarded as a collaborative annotation scheme comparable to Del.icio.us. We also computed the overlap between Bookmarks from Del.icio.us and the URLs from the Web crawl, and found 77,756 URLs present in both analyzed datasets. When manually comparing the tags to the AT of the corresponding pages it became obvious that most AT look like page titles, while tags relate to page content descriptions. We computed text matches between tags and AT. As Del.icio.us tags consist only of one word, we found a very low rate, of 4.71%, of the URLs that have at least one exact match between a tag and an AT, and 42.52% of the URLs that have at least one partial match, i.e., the tag was contained in the AT. For the same overlapping dataset we computed Pearson's correlation coefficient between the frequency of tags and that of AT. We found that these frequencies are uncorrelated (for internal AT p = 0:0002, for external AT p = 0:0411) although there is a slight, yet insignificant, increase in correlation for external AT.

Tags in track lyrics: to get an indication of how often tags are used to describe the theme of songs, we computed the overlap between track tags and track lyrics. The dataset used in this experiment consisted of the intersection of our Last.fm collection and a crawl of the site www.lyricsdownload.com. The intersection of the two sets consisted of 77,498 tracks, for which attributes, such as lyrics, name of the track, album featuring the track, and tags assigned by Last.fm listeners are available. To analyze how many of the tags assigned by the users describe what the songs are about, we took all tags corresponding to the tracks and tried to find them in the track lyrics. The curve follows a power law distribution: the maximum number of tags which also appeared in the lyrics text was 11, which was the case for only one track; approximately 3,000 tracks had more than 1 tag occurring in the corresponding tracks' lyrics; around 10,300 tracks had only 1 tag that could be exactly matched inside lyrics text and for the rest of about 63,000 tracks none of the tags was found in this "original" content. On average, 1.54% of the tracks' tags occurred in the lyrics – which is in line with our manual tagging classification results.

2.2.3..4 Exploring tags for search

Extending and complementing our final discussion in the previous section, we also explored how users' search and tagging behavior compare. We analyze how much a query log overlaps with tags and conduct a user study which shows what tag types users consider most useful for search and which ones they remember best - being thus easily available to be used as retrieval cues in search.

AOL query log analysis: in this experiment, we investigated how much current Web queries overlap with tags. We used the AOL query logs (Pass, Chowdhury, & Torgeson, 2006) to calculate overlap between Web queries and tags, and contrasted tag and query classes. First, we counted what percentage of queries consists of tags used in our three systems. Regarding Del.icio.us, 71.22% of queries contain at least one Del.icio.us tag, while 30.61% of queries consist entirely of Del.icio.us tags. Due to the significant overlap Del.icio.us tags may help finding Web resources matching queries to tags. For Flickr and Last.fm the numbers are 64.54% and 12.66%, and 58.43% and 6%, respectively. Here we have to take into account that our tag vocabulary contains 323,294 Del.icio.us tags, while we only have 32,378 Flickr tags and 21,177 Last.fm tags. Nevertheless, we notice that Del.icio.us tags (for tagging general resources) appear much more often in queries than Flickr or Last.fm tags (images or music related tags). Also, tags describing images are used almost twice as much in queries than music related tags.

For our comparative analysis of tags and queries we tried to find the tag classes established before within queries – investigating which kind of tags could best answer a given query. We built a frequency sorted list of all queries in the AOL log and took three samples, as in 2.2.3..2. For comparing system specific behavior, we similarly sampled 300 queries for music and 300 for image queries, by filtering the query log for queries containing a keyword (like "music", "song", "picture" etc.) or having a click on Last.fm or Flickr. The resulting queries were classified into our eight categories, with queries belonging to multiple classes in case they consisted of terms corresponding to different functions. The results are shown in **Figure 2-10**.

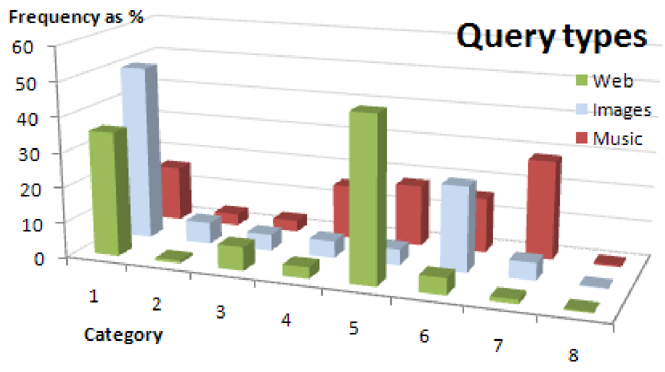


Figure 2-10: Distribution of query types for different resources

Not quite surprisingly, general Web queries often name the Topic of a resource – just like tags in Del.icio.us do to an even larger extent. The query distribution pattern seems to fit to tag types except for a clear difference regarding category 5 (Author/Owner). Usage context is more often used for tag based information organization than for search. For obvious reasons, Self reference is not a useful query type for public Web resources. For images, our tag type distribution almost perfectly corresponds to the query type patterns. As **Figure 2-10** shows, Topic accounts for about half of the queries, as well as of the tags in Flickr. Slight differences exist for Location, used more for tagging than for searching and Author/Owner being somewhat more important for queries than for tagging. Interestingly, there seem to be many more subjective queries beside the Topic asking for Opinions/Qualities like “funny”, “public” or “erotic” pictures. This however may also be influenced by our samples which often contained queries for adult pictures. With decreasing popularity of tags this category becomes somewhat less important – with increasing emphasis on Topic and Location. The biggest deviation between queries and tags occurs for music queries. While our tags in Last.fm are to a large extent genre names, user queries belong to the Usage context category (like “wedding songs” or “graduation songs”, or songs from movie or video games, category 7). Also, users search for known music by artist (category 5) and title or theme (category 1). This difference may be due to information value considerations: as artist and title are already provided in Last.fm as formal metadata there is no need in tagging resources with this information. In the less frequent tags of Last.fm these become more important, so our sampling of popular tags for this system may underestimate their importance. Lyrics are not frequently searched for. An interesting and surprising observation is that searching by genre is rare: Users intensively use tags from this category, but do not use them to search for music. One reason for this might be the fact that many music pieces get tagged with the same genre and thus search results for genre queries would contain far too many hits. Categorizing tracks into genre is also subjective to a certain extent, as it depends on the annotator’s expertise. The amount of subjective qualities asked for or tagged is comparable for the Last.fm system, with about 16% each.

User study: we had 30 participants, all researchers and PhD students in Computer science, in our user study inspired by (Naaman, Harada, Wang, Garcia-Molina, & Paepcke, 2004). The experiment consisted of two different parts: for the first part participants were asked to mentally recall 6 desktop items – 2 pictures, 2 songs and 2 URLs from the users’ bookmark list – which they did not access for a long time. For the case that study participants did not have some of the requested items on the personal desktop, we asked them to recall 2 photos which they once saw, 2 songs which they like hearing (e.g. on the radio) and 2 URLs of pages which they once visited and found interesting. Users had to write textual descriptions for each of them, BUT without looking at the pictures / Web pages and without listening to the music. They were requested to write descriptions as detailed as they could. Besides, for each of these resources, users had to provide a set of keywords best describing them. In the second part of the experiment, participants were provided with the tag category descriptions and examples of tags for pictures, music and Web pages corresponding to each category. They were then asked to answer 3 questions:

- *How useful do you think each of the 8 categories are for searching your own resources?*
- *How useful do you think each of the 8 categories are for searching other peoples’ resources?*
- *How well do you remember each of the 8 tag categories?*

For all questions, users rated on a 5-point Likert scale ranging from 0 – meaning “not useful at all” (for the first two questions) / “not remembering at all” (for question 3) – to 4 – corresponding to “very useful for search” and respectively “very good remembering”.

In the first part of the experiment we aimed at identifying which of the 8 tag categories were used by the users for describing resources they recall best. For each of the categories we measure both the frequency of appearance in the descriptions or keywords, and the order in which they appear. To do so we identified concepts as either a keyword or a phrase in the description and assigned a category to each concept. For this part we could only use the data of 24 participants, since for the remaining ones either descriptions or keywords were missing. On average, a picture had 6.06 concepts attached in the description and 4.33 in keywords. For songs the numbers are: 4.96 and 3.36 and for URLs: 4.65 and 3.42. The personal pictures elicited more memories written down in detail. This is probably partially due to their personal nature – in contrast to rather public songs and web pages. In general, descriptions contained many details about the Usage context and Self reference.

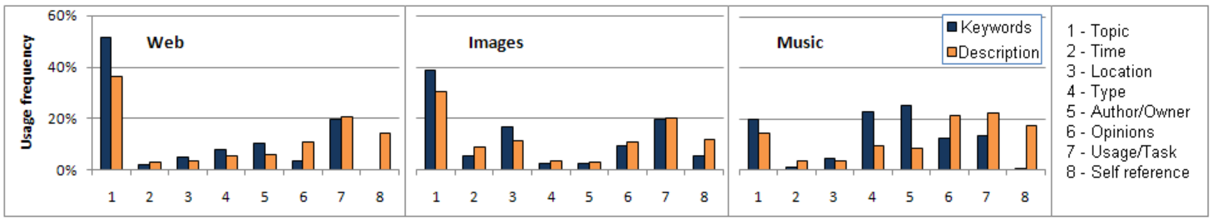


Figure 2-11: User study results - Comparison of category frequencies for keywords and descriptions

Figure 2-11 shows the relative frequencies of the different categories for descriptions and keywords. We observe that for URLs Topic and Usage context are most used in both descriptions and keywords; due to its factual nature Topic reaches over 50% for keywords. Self reference and Opinions/Qualities are also very frequently appearing in descriptions, but not as often in keywords. Looking at the descriptions, this is caused by people telling quite detailed stories about these resources and their associations. The category distribution for pictures is very similar to URLs except for a small drop in Topic and an increase in Location and Time. As we have already seen in earlier sections, the music domain is quite different. When describing songs, people tend to use much more Opinions/Qualities, Usage context and Self reference concepts then when using only keywords. Vice versa keywords are used more for Type and Author/Owner.

The keywords assigned by our participants thus exhibit similar characteristics found in the analysis of Flickr, Del.icio.us and Last.fm presented in 2.2.3..2. For web pages and pictures, actual relative frequencies deviate a little but ordering of category importance is almost the same except for the swap of Usage context and Location in Flickr. For music, we find more significant deviations: while in Last.fm Type is by far the most important category, the keywords are more often Author/Owner than Type and Topic. This is explained by Last.fm's system features, as artist and title are already provided as formal metadata. Independent of the resource type, Usage context is a very well remembered category, which certainly could be exploited and supported more in current tagging systems. Especially for pictures it provides new and only partially subjective information (e.g. "CHI2007", "Universiteit Twente"). Also for music, we found them useful as inter-personal recommendations or associations (e.g. "salsa course").

In the second part of the user study, we wanted to get an indication of the users' perception regarding the usefulness of different tag categories for searching both personal and non-personal resources (pictures, music files or Web pages). We also investigated which kinds of tags are best cues in order to recall a resource. **Figure 2-12**a presents a detailed comparison of the 30 users' ratings for usefulness and remembering of tag types for images. Ratings are very similar across the different activities of searching personal or public pictures and remembering – except Time and Type. Our participants remembered Time very well for their pictures and found it equally useful to search for them, but for public resources it is less valued as a retrieval cue. Often users do simply not know it. For Type it is opposite: Similar to the results of (Naaman, Harada, Wang, Garcia-Molina, & Paepcke, 2004) people seem to find Type more useful for searching others' pictures and also remember them, but they do not use it to search their own items – since they do not annotate or describe their pictures with such (semi)professional photographic aspects. Differences are even smaller for most of the categories for Web resources and music. The pair wise Pearson correlation coefficients between the three activities range from 0.89 to 0.94.

As the ratings are very similar across the different activities of searching personal or public resources and remembering, **Figure 2-12**b compares the 30 users' ratings only on usefulness for searching public resources across Web pages, images, and music. Values vary across resources. Topic is the most useful and best remembered type of information for Web pages, followed by Usage context, Author/Owner and Type. Self reference, Time, and Location are judged neither useful nor well remembered. For pictures on the other hand, while Topic is still the most valuable category, the next ones are Location and Time. Usage context and Type are judged least important, probably due to perceived subjectivity of context and low (semi-)professional photography knowledge.

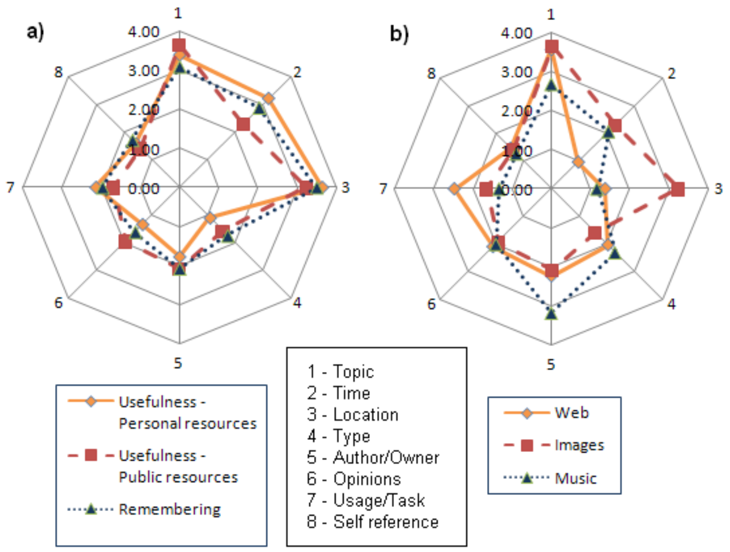


Figure 2-12: User study results - a) Usefulness for personal, usefulness for public, and level of remembrance of images; b) Usefulness for public resources for different resource types

For music, best ratings were given for Author/Owner, Type, and Topic, the others receiving a rather mediocre or low importance value. Opinions/Qualities is considered more useful for searching songs you do not have in your collection than it is for searching your favorite songs. As a surprise, it seems that people assume quite some agreement on subjective characteristics and opinions. We also found this tendency for URLs and pictures, though a less pronounced. Summarizing the results, substantial differences in perceived value of tag types exist only between resource types, each resource type having its own noticeable categories (e.g. Location for images). Concerning the activity (remembering or searching for own or other people's resources), there was only minor impact for Time or Type for pictures and Opinions/Qualities for music. However, for all resource types users rated 'factual' categories, especially Topic, very high. On the other hand, Usage context and Opinions/Qualities were valued higher than we had intuitively expected.

2.2.3..5 Results: Usefulness of tags in search

The results presented in 2.2.3..2 show that the tag distributions depend on the resource domain: pictures and Web pages can contain objects referring to any topic, whereas music resources are very restricted in content, leading to a much more focused set of top tags. Analyzing tag types, we were able to show that more than 50% of the tags in Del.icio.us, Flickr and AT are Topic-related keywords. As non-subjective annotations, these tags are usable for search by all users, not just the tagger. A probable motivation for using these tags is that Web pages and pictures can belong to any topic category, thus classifying these resources with topics is a very natural way to organize them. In contrast, for Last.fm the Type category is predominant: most of the tags correspond to music genres. In Last.fm we also find more opinion related tags, whose top tags might be useful for a majority of users, but not for people disagreeing with popular opinion. Opinion/Quality and Author/Owner are the second and third most used classes for tagging music resources. Regarding added informational value of tags we observe that Del.icio.us tags are, like AT, present in 45% of the pages they annotate. Only 43% of Del.icio.us tags are included in AT for the same URL they point to. This means that over 50% of tags bring new information to items they annotate or describe. In contrast, Last.fm tags are usually not contained at all in lyrics (the only textual original content available): the percentage of new tags is 98.5%. Regarding music reviews (another source of information about music, manually created by human experts), at least one Last.fm tag occurs in the review texts for almost all analyzed tracks. This proves tags to be a reliable source of metadata about songs, created more easily by a much higher number of users.

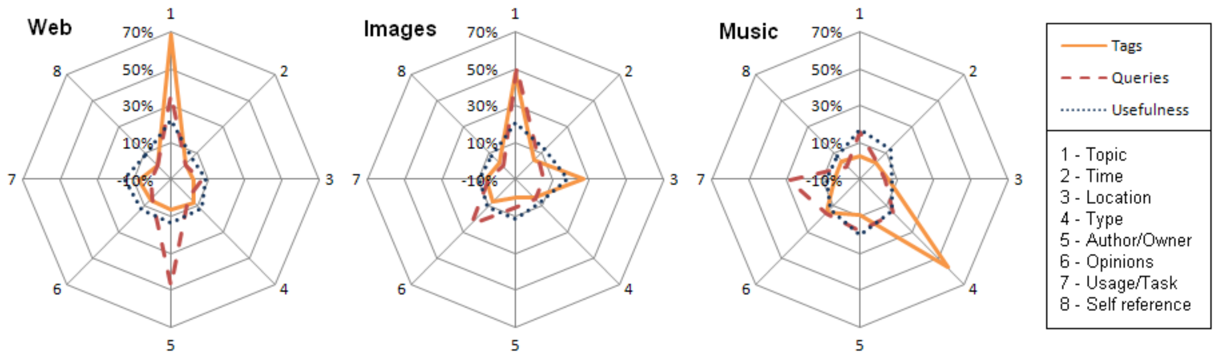


Figure 2-13: Comparison of category usage for tags and queries, and user usefulness assessment

Comparing categories of tags and queries as well as perceived usefulness provides some interesting insights: **Figure 2-13** summarizes the differences and commonalities of how our eight categories are used for the different resource types. It compares the usage of tags with the usage of queries and the perceived level of usefulness by the users for public resources for each of the categories. Most of the general Web queries are Topic-related queries (as most of the tags for Del.icio.us, Flickr and AT) and this category is also considered by far the most useful for search in particular for Web pages. Except for Topic users do not voice major differences in usefulness of the eight categories. However, we observe that some categories are more useful than others. For Web resources Topic tags are very useful, as over 30% of the queries target this category; but we also see that although users query the Author/Owner category, they usually do not tag in this way. For images the Topic category is considered by experiment participants very useful for search, and tags and queries of this kind are equally present. For images many queries are about Opinions/Qualities but users tend to add more Location tags than the needed Opinions/Qualities. So, even if users actually like to search for funny or strange pictures and judge them explicitly as partially useful for search, they often do not tag them in this way. As for the music domain, tags generally fall into the Type (i.e., genres) class, although more tags from Usage context and Topic categories would be needed (Author/Owner is already present).

To conclude, our experiments provide evidence for the usefulness of a common tag classification scheme for different collections, which allows us to compare the kinds of tags used in different tagging environments. We have shown that the distributions of tag types strongly depend on the resources they annotate: For Flickr, Del.icio.us and AT Topic-related tags are appearing in more than 50% of the cases, while for Last.fm the Type category is the most prominent one. Other interesting findings refer to the added value of tags to existing content: More than 50% of existing tags bring new information to the resources they annotate and for the music domain; this is the case for 98.5% of the tags. A large amount of tags is accurate and reliable, for the music domain for example 73.01% of the tags also occur in online music reviews. Regarding search, our studies show that most of the tags can be used for search and that in most cases tagging behavior exhibits approximately the same characteristics as searching behavior. We also observed some noteworthy differences: For the music domain, Usage context is very useful for search, yet underrepresented in the tagging material. Similar, for pictures and music Opinions/Qualities queries occur quite often, although people tend to neglect this category for tagging. Clearly, supporting and motivating tags within these categories could provide additional information valuable for search. These results are promising and provide more insight into the use of different kinds of tags for improving search. They also indicate potential extensions of tagging environments to add an incentive for encouraging users to provide potentially search-relevant tags.

2.2.3..6 Extensions: Current and Future work

Future interesting research questions include automatic tag classification, as well as investigations for which kinds of queries can be supported by which kind of information (from tags, content or other metadata). This will help us to strategically extend existing information – gathered from different sources – and provide better support to queries especially for pictures and music resources which cannot be handled well enough by existing techniques.

Thus, we are currently designing algorithms for automatically classifying the tags according to our tag type scheme. For example, the time category can be identified via regular expressions (numbers for years, names of months, etc.) while location is probably best kept via a table lookup. We used GATE's (University of Sheffield - Natural Language Processing Research Group) comprehensive tables for geographic named entities like cities, countries, etc. These are also available for languages other than English – and quite some tags we found during our analysis are non-English. Author/Owner is similarly identified by matching against a list of common first and last names also provided by Gate. For music, in addition knowledge about artist names can be used to classify these tags. Types as well are found via a list of typical file extensions or resource types like pdf or Blog. As self reference is defined as tags referring to the tagging person himself, clues like "I","my", "me" and "our" etc. help identifying these kind of tags. For Opinions/Qualities, one heuristic may be to explore their characteristic of being mostly adjectives. By using Natural Language Processing tools like GATE or matching against WordNet Part-of-Speech-Tagging can be applied to tags. Here, the difficulty is that tags are rarely complete sentences, which may provide problems to rule based systems like GATE. However, due to motivations like Opinion expression, performance and activism especially these Opinions/Qualities seem to be longer and more phrase-like than other types of tags (Zollers, 2007). More strategies to classify these tags as well as Usage context tags automatically will be experimented with; co-occurrence patterns will be one main feature. For topic, the broadest category of all, resource specific heuristics like comparing tags with the title or the lyrics of a song (given some threshold) may be used in addition.

For PHAROS, such automatic tag classification will be used internally for user tag profiling as well as in recommending and ranking content. We also plan to extend the tagging functionality in a way to support the creation of search-relevant and usually underrepresented tags. Related further work similarly explores machine learning of Usage context tags to recommend such tags to the user – for easy recognition instead of (more cognitive load through) generation.

c. Sentiment analysis and Opinion prediction in user created textual contents

This third set of experiments aims to catch users' preferences in user-created contents such as movies reviews, blogs or natural language annotations about multimedia contents. We focus on a method which analyses the natural language contents. As we propose an automatic classifier, we designed a learning tool which, in order to be able to be used for predictions, needs first to be trained. The training sets of comments had been provided by the community site Flixster (Flixster) gathering movie fans.

The challenge is to use the learned knowledge (pertinent words to predict ratings) in the particular context of PHAROS platform: we will reuse our opinion prediction system on PHAROS users' textual comments. As a result, the SAM module will be able to add new rated items to their profiles. This is particularly interesting to collect new tastes without having the users fill any forms.

2.2.3..1 Opinion Classifier

We consider here that textual contents delivering opinions or points of view can be modeled as a pair (Object;Opinion). With our current expertise, the Opinion Prediction problematic is addressed by predicting a positive or a negative rate with a prediction precision of 77%.

The tool we dispose of by now is the result of the study D. Poirier and C. Bothorel have conducted during April 2007 – June 2008. Two types of techniques have been developed:

1. The first one is based on a statistical machine learning tool (Boulle, 2004)
2. The second one is based on Natural Language Processing methods (using the France Telecom Suite Natural Language Processing "Tilt" - (Kervajan, Neef, & Véronis, 2006))

In order to conceive and develop both tools, we have crawled and analyzed a corpus of 60.000 reviews provided by Flixster. In each review, the "object" discussed (movie/actor/film maker) was known because they were manually described by the author: when a user writes a review about a film on the Flixster web site, he/she fills a form with the title of the film (resp. actor or film maker) and a rate [0-5]. Half of the reviews we have downloaded expressed positive opinions and the other half, negative opinions. We kept a set of 10,000 positive and 10,000 negative for the tests. Reviews with a rate lower than three stars were considered as negative reviews, other as positive reviews. During the test step, our aim was to predict the rate given by the author.

The main difficulty of this corpus is the small size of reviews (twelve words on average). This makes opinion extraction difficult even for human beings sometimes. Moreover, the corpus is composed of textual messages very similar to forum messages. They present common characteristics such as accumulation of punctuation (”!!!”), smileys (”:-)”), SMS language (”ur”, ”gr8”) or words stretching (”veryyyyy cooooool”).

We recall here quickly what has been already presented in the Deliverable 2.1.1. and published recently at LREC conference (Poirier, Bothorel, & Boullé, 2008). Since the previous deliverable, we acquired a better quality dataset and performed the comparison of our two methods on the same data in order to highlight pros and cons of both methods.

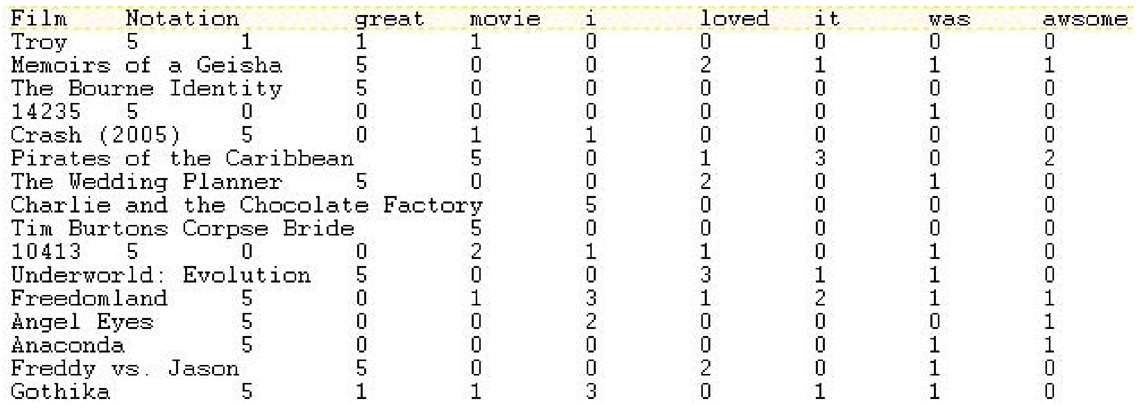
1. **Machine Learning Tool KHIOPS**: The tool is issued from a machine learning technique (KHIOPS) which has been trained on a sample corpus (with negative and positive reviews and their rates). During this training step, the tool has to find which words describe best positive or negative ratings. KHIOPS is a France Telecom homemade tool allowing both supervised and unsupervised learning. It allows performing uni-variate and bi-variate descriptive statistics, to evaluate the predictive importance of explanatory variables, to discretize continuous variables, to group the values of categorical variables, and to recode input data according to these discretization as well as value groupings.

   This tool is used as followings:

   • The first step consists in building the dataset and choosing the variables describing the data. A variable may be 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 predicting the target variable value. And conversely, the lower the rate, less the variable allows predicting the target variable value. Our target variable is the rate given by the review author. KHIOPS uses the naive Bayes classification approach with selection of variables and average of models (Boullé, 2007). The supervised learning method consists, in our case, of learning which variables, i.e. words, are significant to predict the rating of the comment. For this method, KHIOPS takes as input a file describing a matrix (see Figure 2-14 below).



```
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
14235   5       0       0       0       0       0       1       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 Corpse Bride        5       0       0       0       0       0
10413   5       0       0       2       1       1       0       1       0
Underworld: Evolution   5       0       0       3       1       1       0
Freedomland     5       0       1       3       1       2       1       1
Angel Eyes      5       0       0       2       0       0       0       1
Anaconda        5       0       0       0       0       0       1       1
Freddy vs. Jason        5       0       0       2       0       1       0
Gothika         5       1       1       3       0       1       1       0
```

Figure 2-14: Part of input file for the supervised training of the learning method.

As a result, the tool presents the more "informative" words allowing rating prediction. When a new review is presented, it shows as a result the calculated rating. With this approach we have no "a priori" on the data. Indeed we hang on all the reviews as the authors wrote them and process them as bags of words. We do not apply pre-treatment on the data with NLP tools. We only put the text in lowercase and delete the punctuation.

The tool found 305 informative variables out of the 24.825 words present in the learning corpus. Only a few of them are very informative as shown in the figure below. They are classified according to their level value. The level is directly related to the posterior probability of a discretization model, with a 0-1 normalization. Its value is 0 in case of a non informative input variable and is asymptotically equal to 1 in case of a perfectly informative input variable.

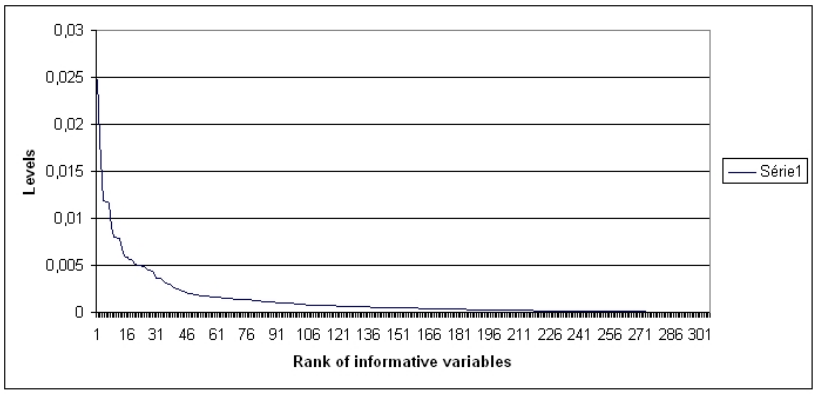
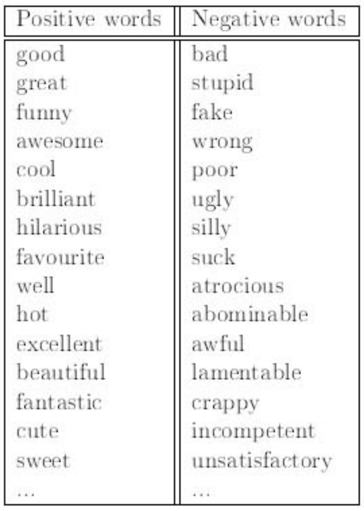


Figure 2-15: Only a few variables are informative, most of them don't bring much more information and decisiveness regarding the prediction of the rating.

These results allow learning opinion vocabulary but also information on the style of the reviews. For example, the presence of "and" is associated with a positive opinion, and when we analyze such reviews we find longer texts: this indicates that authors write longer texts with more details when they talk about a movie they appreciated. Users have tendency to be more prolix and detail their point of view on film features when they appreciate the movie.

2. **Natural Language Tool TiLT**: The tool uses a dictionary listing 183 "opinion" words, built from verbs and adjectives manually classified and using linguistic techniques (see a sample of our dictionary in the table below). When such a word (labeled positive or negative) is detected in a textual review, the tool counts positive or negative weight. For that we have to lemmatize the reviews and we have to keep only adjectives and verbs. One of the positive points using TiLT is to help in resolving typos, understanding abbreviations, smileys, and so on. Then, we are able to assign a polarity to reviews according to the majority number of positive words or negative words.



Positive words	Negative words
good	bad
great	stupid
funny	fake
awesome	wrong
cool	poor
brilliant	ugly
hilarious	silly
favourite	suck
well	atrocious
hot	abominable
excellent	awful
beautiful	lamentable
fantastic	crappy
cute	incompetent
sweet	unsatisfactory
...	...

Figure 2-16: Part of the hand crafted lexicon

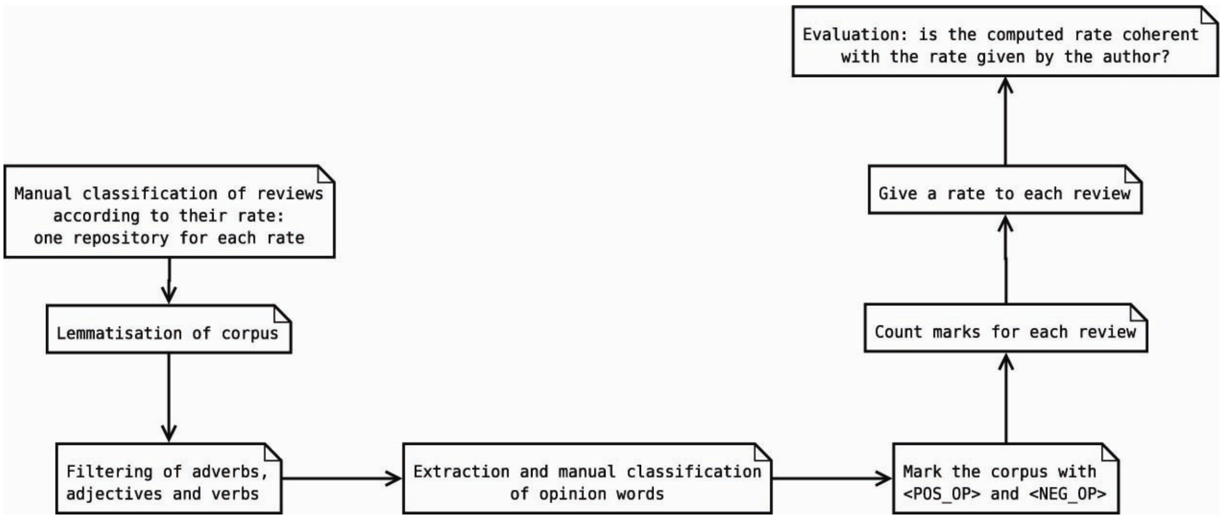

Manual classification of reviews according to their rate: one repository for each rate
Lemmatisation of corpus
Filtering of adverbs, adjectives and verbs
Extraction and manual classification of opinion words
Mark the corpus with <POS_OP> and <NEG_OP>
Count marks for each review
Give a rate to each review
Evaluation: is the computed rate coherent with the rate given by the author?

Figure 2-17: NLP process to evaluate the method

2.2.3..2 Results of Opinion Prediction module and further work

The Machine Learning technique offers a good quality: 0.77 for precision, 0.76 for recall. 70% of the positive reviews and 82% of the negative reviews are well classified, and all the reviews are classified with this first technique. On the contrary, only 74% of the reviews are rated (26% unclassified) with the second technique. Considering the F_{score} measure, both methods are equivalent (0.75). The NLP method is very reliable for the positive reviews (recall of 97%), but not good in classifying negative reviews as negative (43%). This phenomenon may be due to the dictionary we used: the positive category contains almost twice as many words as the negative category. The analysis shows that negative opinions are often expressed by using words carrying positive opinion associated with a negation. Since our linguistic approach ignores every negation, most of the negative reviews are labeled as positive ones. In further work, the best solution would probably be to proceed to a dependence analysis, not very costly to operationalize and which would bring much more reliability to this technique.

The main point characterizing ML techniques is that new datasets can be analyzed without any apriori knowledge (i.e. lexicon) and can then be quickly deployed with comfortable reliability on both positive and negative reviews. However the corpus has to be large enough to offer a consistent training dataset (we tried a learning set of 5000 reviews which was too small, more than 10 000 is suitable); but the main constraint is that the training set has to contain ratings to supervise the training, which is not always the case. This approach may also be used to detect pertinent words and thus help in building the dictionary, particularly in the context of Web Opinion Mining, where it is necessary to adapt the lexicon to the inventive vocabulary the Internet users' writings abound in. As a conclusion, we propose to use a low-level NLP approach when the corpus is too small to have a good training: the cost of building a lexicon (small ones bring satisfying quality) and designing negation detection remains reasonable. If the corpus is large enough, the ML approach will be easier to deploy.

In the PHAROS context, because we stay in the field of cinema, we assume that the training step we preceded on the Flixster dataset is pertinent, and as a consequence, that our ML tool can be used to predict rates to new PHAROS comments. Our NLP tools are also usable with our homemade dictionary.

2.2.3..3 PHAROS rating of new comments and profile enrichment

We presented in the previous paragraph two techniques to automatically rate a textual comment about a movie, an actor, or a film maker. We aim to collect and analyze textual annotations, such as comments, forums, and so on, and therefore deduce from the text new tastes that we can add to the PHAROS users' profiles. We consider that textual contents delivering opinions or points of view can be modeled as a list of (Object, Opinion). An opinion may be positive or negative (translated into a rate: 5 or 0 in PHAROS). Such pairs (Object, Rate) will be learnt and added to the USIS user profiles.

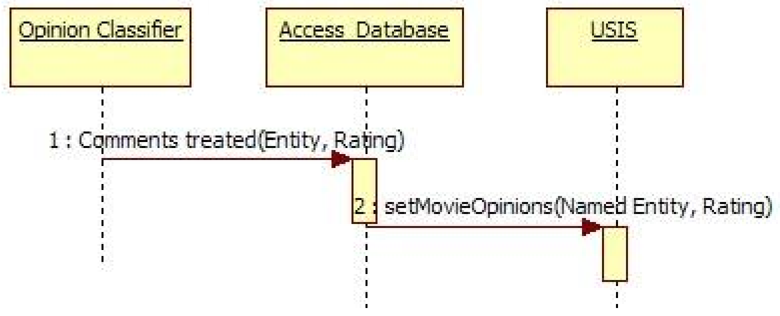

Opinion Classifier
Access Database
USIS
1 : Comments treated(Entity, Rating)
2 : setMovieOpinions(Named Entity, Rating)

Figure 2-18: SAM output: inserting a new rating about a cinema Entity into the USIS profile.

A PHAROS User (profileId) publishes comments (commentID + textual annotation) and discusses in those comments one or more "named entities" (actors/movies/film makers).

The SAM module takes as input a userID which allows fetching of unanalyzed comments. The analysis comprises the Named Entity detection and the Opinion Classification.

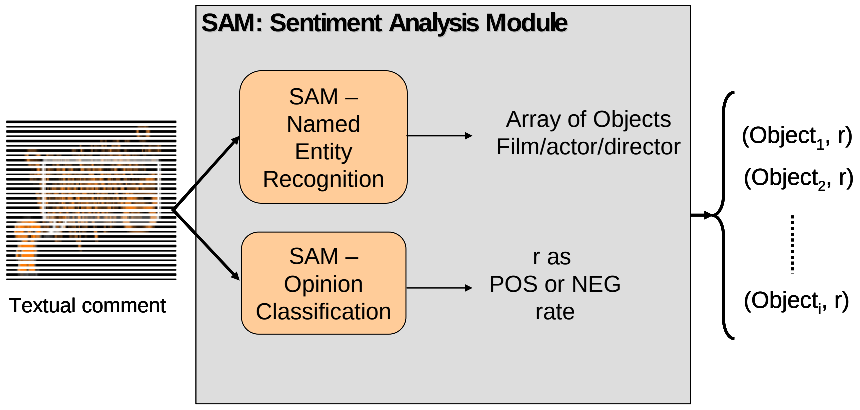

SAM: Sentiment Analysis Module
SAM – Named Entity Recognition
Array of Objects Film/actor/director
SAM – Opinion Classification
r as POS or NEG rate
Textual comment
(Object1, r)
(Object2, r)
(Objecti, r)

Figure 2-19: SAM global architecture.

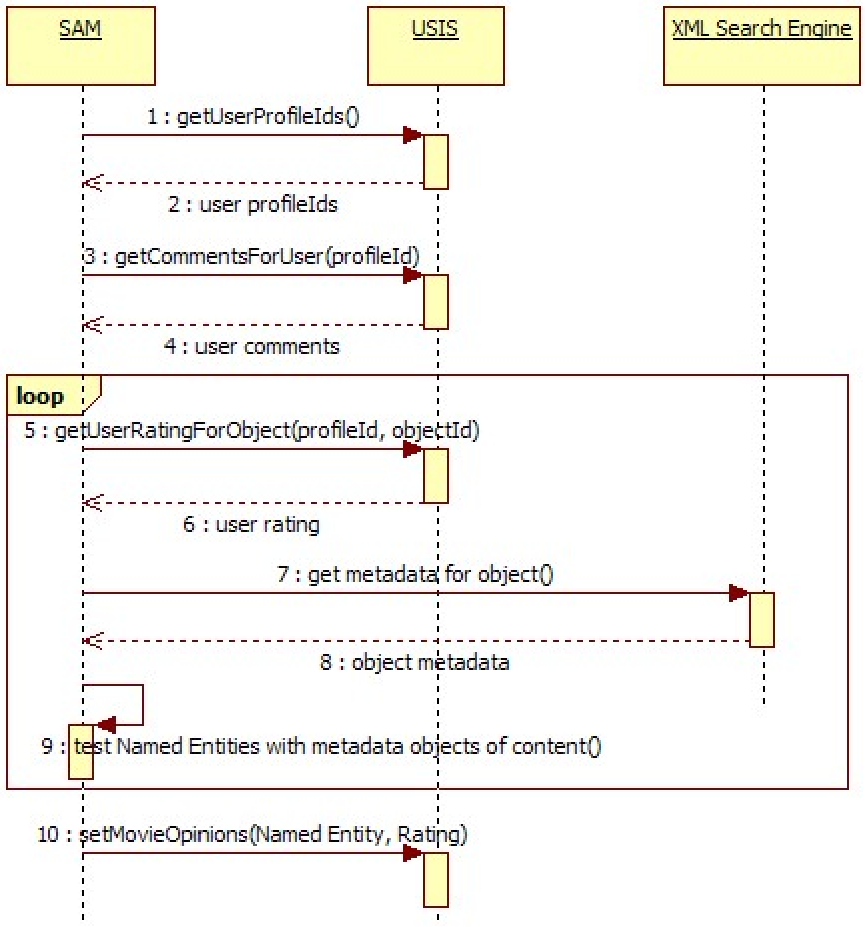
SAM - Named Entity Recognition: this software module takes as input a text and provides as output an array of Objects (Named Entities such as the title of a film, the name of an actor or a director). This operation of recognition is based on a database containing a set of movies. It comprises user discussions on 8620 film titles in Flixster. It is planned to complete the database by using the Internet Movie Database IMDB[4] as an additional source.

We begin by making an initial treatment of the text to bring it in a usable form (lowercase, no accent, alphanumeric characters). For each record from the database, we search the text (the comments are quite short). To design a more efficient component, we could possibly use Natural Language Process to select pieces of text as potential candidates and search if they match with records in the database. There are many typos and mistakes in what people write, so we have to use a fuzzy match and define a partial matching (threshold).

SAM – Opinion Classification: each classifier takes as input a textual comment and produces a positive or a negative opinion with a confidence value. The SAM module will launch both classifiers and combine the results to obtain reliable results. In case of contradiction, we will privilege the method which gives a result with a very high confidence (above a threshold).

Here is the overall process:

4 http://www.imdb.com/



SAM
USIS
XML Search Engine
1 : getUserProfileIds()
2 : user profileIds
3 : getCommentsForUser(profileId)
4 : user comments
loop
5 : getUserRatingForObject(profileId, objectId)
6 : user rating
7 : get metadata for object()
8 : object metadata
9 : test Named Entities with metadata objects of content()
10 : setMovieOpinions(Named Entity, Rating)

Figure 2-20: Sequence diagram showing how SAM interacts with the USIS.

From a user profileId, the SAM module gets the comments not already analyzed. The Named Entity Recognition reveals objects which may have been already explicitly annotated by the current user e.g. by ratings (through the UI form). SAM must download the multimedia objects metadata (from the Search Engine) as well as the social/ user generated metadata for the objects to test if the tastes discovered are really new or consistent respectively. In case of an object already explicitly rated by the user, the results produced by the SAM have to be ignored or merged. If the objects are not listed already, SAM inserts the new tastes into the USIS.

2.2.3..4 Future work and extensions

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 reaches an accuracy of 77% without any NLP pre-treatment on the textual corpus.

We are therefore confident that NLP tools will improve results:

• by lemmatizing and therefore going beyond the volume barrier: train the machine learning algorithms with a greater lexicon (with 70,000 reviews, we count more than 90,000 different words);

• by cleaning the corpus and removing (uncommon) syntactic variations users take make they write comments about multimedia content, and thereby again reducing the number of variables.

On the contrary, we have to test if machine learning techniques, by stressing the informative words, can improve the NLP method consisting in counting opinion words and defining a heuristic to classify a review into a positive or negative class. The dictionary we used has been manually built: 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 can be useful for a first exploration of the corpus (Poirier, Bothorel, Boullé, & Ferrandiz, 2008). As we have shown above, we have observed that longer comments are more often positive reviews, i.e. that users are more prolix about what they like. But if we want to go further in the analysis, NLP techniques, 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, we show here how to enrich users profile by adding new tastes about cinema discovered in user-created contents. But we are confident that our work can also 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 appreciated films with Johnny deep, beyond the easy deduction that his presence is enough. Such studies will converge to fine community profiles and fine recommendation taking into account polarity of opinion.

2.3 SNBA – Social Networks & Blogspace Analysis

2.3.1 Description

The **Social Networks & Blogspace Analysis (SNBA)** module aims at gathering and analyzing social network data coming from blogs or friendship networks for discovering additional knowledge which can be applied to improve the search and recommendation results in the PHAROS platform.

After a slow start, blogging rapidly gained in popularity, so that in December 2007 the blog search engine Technorati[5] announced tracking more than 112 million blogs. There are many different types of blogs, differing not only in the type of content, but also in the way that content is delivered or written. However, for our analysis we will focus on personal blogs, as this type of blogs – on-going diaries or commentaries by individuals – reveal the most personal information about their authors. Being so easy to create, personal blogs represent the traditional, most commonly found form of blogs. Personal bloggers usually take pride in their blog posts, even if their blog is never read by anyone but them. Blogs often become more than a way to just communicate; they become a way to reflect on life or works of art. And probably the most important aspect for our analysis is that they reflect a lot of personal aspects of their authors and give away some of their interests and preferences.

Since all blogs are on the internet by definition, they may be seen as interconnected and socially networked. Several features permit bloggers to link to each other's blog pages: the so-called 'blogrolls' lists one's favorite blog list in a frame inside their own blog page. These links represent other authors' blog pages that this author considers interesting and frequently visits for reading and / or directly commenting. In a sense this feature is similar to the in-links of a web page: they inject some importance to the blog pages they target by the fact that the author of the blog page lists these links on his own and indirectly shows that there are some trusted blog sources, worth reading. Besides, the blogrolling phenomenon is somewhat reciprocal. By linking to a blog, you are increasing your blog's chances of being linked-to by other weblogs. These links between Weblogs are the "currency" of the Weblog community. The more links you have pointing to your weblog, the more likely you'll get a growing audience and high rankings in search engines. A blogroll helps you get started earning links from other weblogs by expressing your affiliations. Permalinks are also a possibility to create social links among bloggers. Unlike blogrolls which point to a blog page, a permalink represents a link to a particular blog post inside a blog page (created by the author of this page) and allows other bloggers to use it to jump directly to this blog entry. Given the highly dynamic content change of the blog pages,

[5] http://technorati.com/

this feature is extremely useful if you want to re-read some very interesting post, which has already passed from the front page to the archives.

Given the aforementioned characteristics of the weblogs and of the blogosphere, it can be easily seen that blogging is inherently a social process, one in which information is created and diffuses (or flows), between bloggers due to bloggers influencing and being influenced by other bloggers.

Given this phenomenon, the overall objectives of the current release of Social Networks and Blogspace Analysis module is to:

- Determine ways to capture what is diffusing;
- Determine paths for who influences whom;
- Measure the extent of this influence;
- Exploit this knowledge for personalized ranking and search

These objects presents challenges for several reasons:

1) Diversity: There are many different styles and types of blogs – who aim to freely express themselves. Additionally, compared to scientific publications for example, writing style does not adhere to strict social practices.
2) No “paper trail”: The “readership” relationship among bloggers is largely unobservable. Links - hyperlinks, comments or blogrolls, citations - are power law in nature.
3) Evolution/Growth: The amount and redundancy of information is rapidly growing

In this deliverable, we touch each of the objectives presented above, describing the components, algorithms and current results for exploiting information diffusion for personalized ranking and search.

In the following we will present the architectural details of the SNBA module.

2.3.2 Architecture

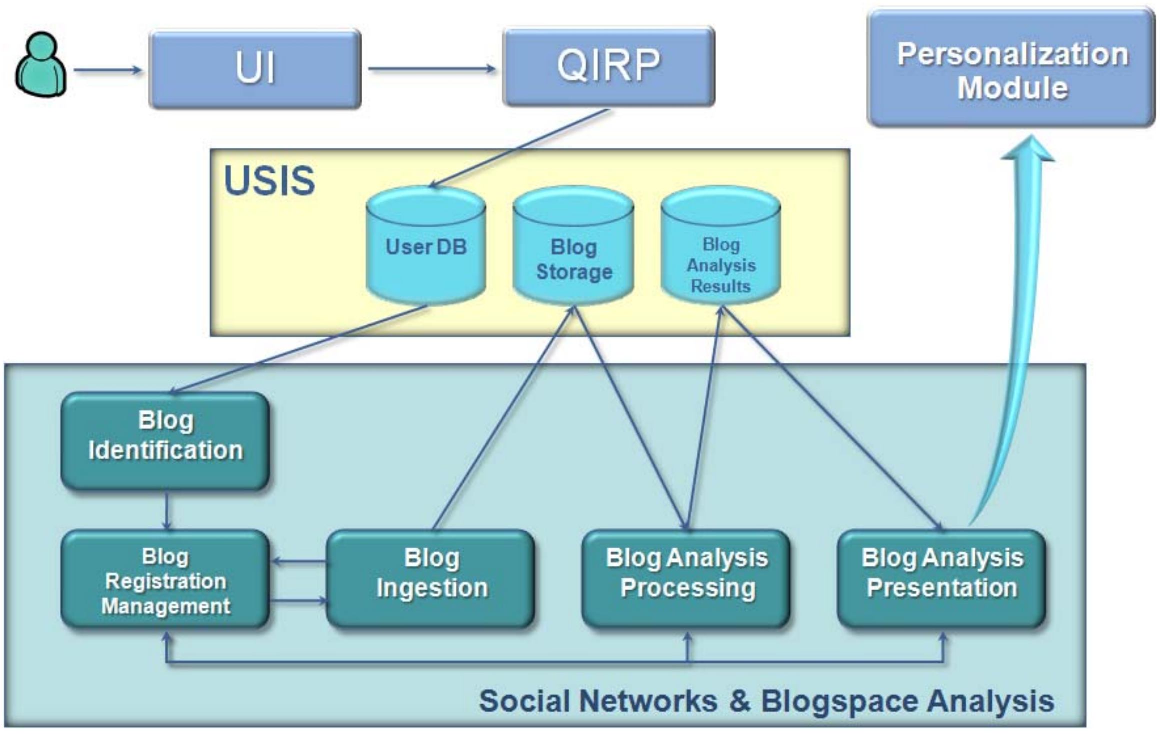


Figure 2-21: Overview of the Social Networks and Blogspace Analysis Components

As depicted in Figure 2-21, several components build up the Social Networks & Blogspace Analysis

(SNBA) module and the overall purpose for each component will be described below:

- *Blog Identification*: checked for the availability and crawl-ability of a blog URL provided by a PHAROS registered user.
- *Blog Registration Management*: maintains status information about every blog (e.g. whether the blog needs to be re-crawled, whether the crawling is finished already, etc.).
- *Blog Ingestion*: If the status of a blog indicates that it can be crawled, the Blog Ingestion starts to crawl it, parses and then stores content into the Blog Storage located in the USIS.
- *Blog Analysis Processing*: contains a suite of algorithms for mining knowledge from blogs.
- *Blog Analysis Presentation*: converts and exports the blog analysis results so that they can be used by other modules (e.g. PM for search and recommendations, UI for displaying the results of the analysis).

The Blog Analysis Processing component is further divided into different sub-modules (see Figure 2-22). The Blog Analysis Module is a three-stage process: at the lowest level, a static snapshot representation of the domain is obtained through the Text Mining Module. Based on this, higher level abstractions are built (Topic / Community Detection & Profiling). Finally, dynamics and evolution are handled within the Information Diffusion Module.

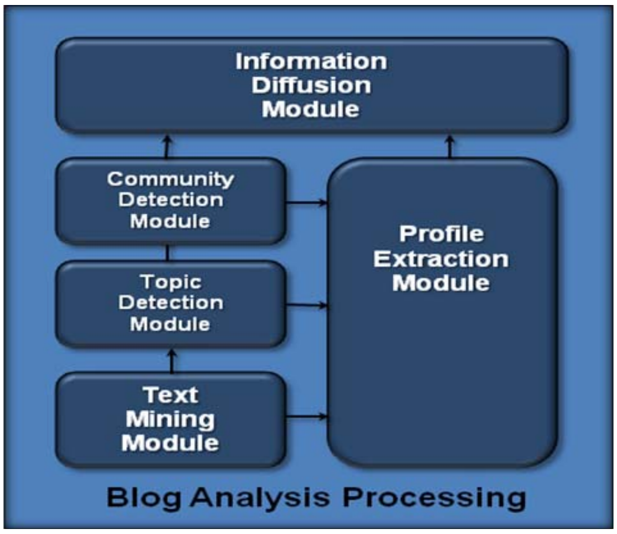

Information Diffusion Module
Community Detection Module
Topic Detection Module
Text Mining Module
Profile Extraction Module
Blog Analysis Processing

Figure 2-22: Blog Analysis Processing

The Text Mining Module uses pre-processed blog posts as input for the mining process. The output (typically described by a set a probability word pairs) is used as input for determining the topic of blog posts or of the blogs themselves. Community structures are then created from these underlying descriptions. Topics from blogs and posts written by a user are taken as representations for the user's and her communities' profiles. In the next step the Profile Extraction Module updates the created profiles with sentiment and time information. Once communities have been detected – the dynamics and evolution of information between individuals and communities are mined, in the Information Diffusion Module.

An important prerequisite for discovering information diffusion paths is that we must first be able to determine what information is flowing within / between blogosphere communities. With this in mind, one of our concerns was to design algorithms which can be easily adapted to support future extensions, and allow us to consider the different characteristics of the blogosphere. In the next section we will present an algorithm for mining information diffusion paths, based on unsupervised learning and making the following assumptions:

- inter-document relationships are unobservable

- separate blogs, even when they are related to a common topic, are for the most part not written with similar prose structure in mind
- blogs are written by different authors who may have overlapping concepts in mind

Domain Representation

Although not strictly part of the architecture, the Domain Representation is crucial to the analysis performed in the SNBA. This module is an internal representation on the real world data that is formatted as subsets useful for forming different types of analysis. Figure 2-23 represents an example snapshot of the repository that will be useful in the discussion that follows.

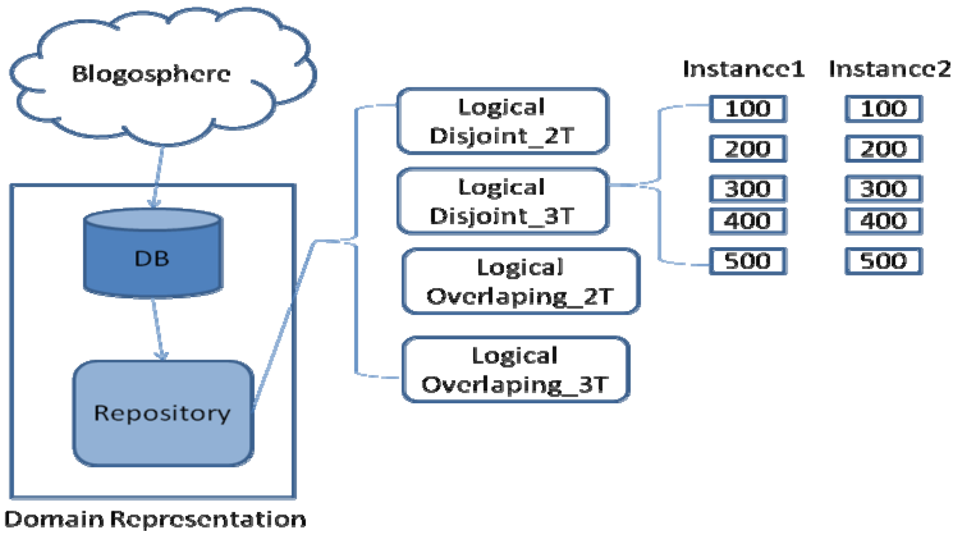


Figure 2-23: SNBA Repository

Data collected from blogs is stored in the USIS. Depending on the type of analysis, different subsets of the data in the database (we refer to this as our corpus) need to be extracted and converted to an algorithm-specific representation (which we refer to as a data set). For example, the repository may store logically disjoint or overlapping views of the underlying database, i.e. sets of posts which are thought to be related in some natural way - at a very high level (e.g. some blog sites – usually edited ones – actually have categories associated with blog posts, or a personal diary blog may be known, in general, to be a blog devoted to a given domain, such as cycling). This kind of information is useful for the analysis we perform. In the following we will focus on the algorithms used inside the SNBA module.

2.3.3 Algorithms

The intuition underlying the current set of algorithms developed in the SNBA for information diffusion is based, in part, on the paradigm of Cross Document Structure Theory (Radev, 2000). From this theory, we hypothesize that: a) information built from multi-documents summary (i.e. blog posts) can be traced back to the posts which were used to produce and influence it; b) this "trace" can be quantified; and c) the information itself is multi-topical. The approach we use to detect the multiple topics is inspired from the area of latent topic models.

Since blogs, are inherently text-based, we focus on text mining tasks which allow us to identify latent, intra blog topics. We attempt to learn the equivalence classes (communities) associated with the latent topics. In our study, we mine commonalities that exist across blog domains (via inter-blog topics) in order to better understand information flows between bloggers.

Community detection or clustering blogs posts is an important first step in our blog analysis. Most blog posts are not conveniently tagged or classified; even for those that are, the classification is usually too high level to adequately meet the information seekers needs.

Our approach (refer to Figure 2-24) first assumes a generative process for blogging: a user has a concept (one which is not observable) in mind and based on this she produces blog posts by choosing words that best represent this concept. Based on the produced artifacts, the SNBA module uses text mining techniques to determine: (1) the different latent concepts / topics used by the author; (2) the themes for each of the posts; (3) the set of words which best represent each topic. The mined knowledge is used to build profiles of users, and communities are later exploited in PHAROS by the Personalization Module to provide personalized ranking and recommendations.

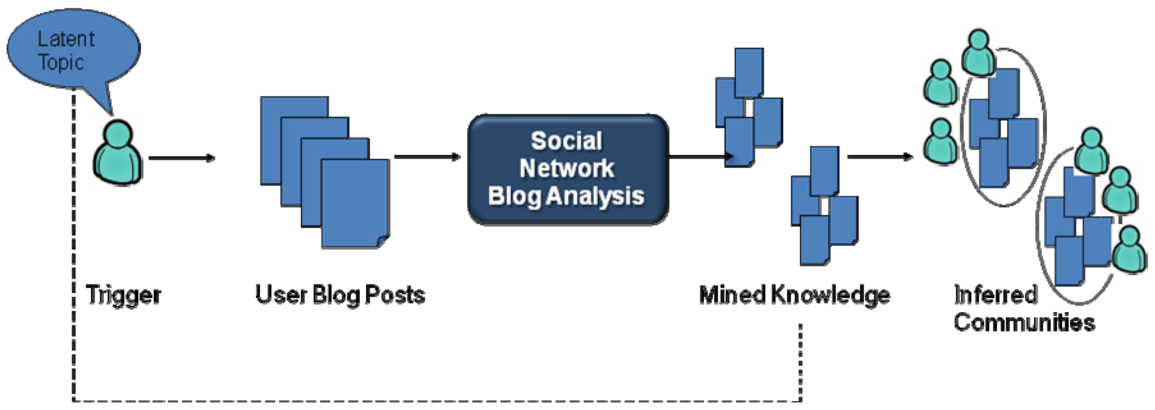


Figure 2-24: SNBA Community Mining Approach

One text mining algorithm used in SNBA is based on the approach presented in (Zhai, Velivelli, & Yu, 2004). This algorithm supports tuning a background model to help filter the level of noise considered to be present in the corpus. We believe this to be an important notion for blogs.

Conglomerate vs. Comparative Approach

Alternative considerations are made for the level of aggregation for representing blog artifacts. One alternative is to simply "flatten" all the blog posts into a single collection and then mine the topics across all blogs. In this manner, the "boundary" of a blog site is ignored in the computation. Alternatively, in a comparative approach, each blog site can be considered as a collection in which topics that are specific to a blog site, as well as topics that may be common among different blog sites can be mined.

Experimental Goals

In this section we discuss the experimental goals, design and current results for evaluating the selected comparative text mining algorithm. The experiments are divided into five parts:

- **Part I**: Conglomerate recovery of natural clusters – measure the conglomerate ability to recover natural clusters. We also present initial experiments, current observations and discuss performance issues
- **Part II**: Conglomerate vs. Common themes - compare the natural clusters from the conglomerate approach in Part I, with the common themes learned from the results of the comparative approach
- **Part III**: Common Theme vs. Multi-document Summary - common themes of the comparative approach in Part II are compared with a multi-document summarization
- **Part IV**: Collection-Specific Theme vs. Single -document Summary - compare the collection-specific themes with single-document summarization
- **Part V**: Parameter Tuning

Data Set

In order to test the algorithm, we collected data from blogspot.com by selecting a seed blogger in different domains on a single day, in April 2008. Following the links in the blogroll, all blogs starting from the seed blogger, were collected, going 2 levels deep. Note, that blogrolls point to blog pages of another blogger. Although blogrolls point to an entire blog, as opposed to only a blog post, we find this information useful for inferring natural communities. At the time we crawled this collection, no links between communities existed.

For each post we extracted: post permalink, author, textual content of the blog post, title and blog URL. Some summary statistics are shown below:

Domain	Date Collected	Total Bloggers	Total Posts	Avg. Time Span (days)	Avg. Posts (days)	Avg. Deter Level
gardening	2008-04-15_00.14.23	45	7642	487.9767442	177.5813953	0.403616544
beauty	2008-04-15_16.08.36	65	17087	271.3076923	262.8769231	0.965025741
cycling	2008-04-14_16.07.49	42	4778	435.804878	116.5121951	0.296660829

Figure 2-25: Data Summary

Experimental Results

In the following we present the experiments and the results for all five parts considered (and enumerated above).

2.3.3..1 Part I: Compare the Conglomerate recovery natural clusters

Part 1A: Preliminary Evaluations - How Well Can the Conglomerate Approach Recover Natural Clusters for 2 Domains?

In this experiment, we use precision and recall for documents taken from the domains of beauty and cycling. We want to measure how well the suggested algorithm is capable of recovering the natural clusters of blog posts belonging to a single equivalence class.

Procedure: the data set was built randomly, by sampling five, equal portions from all the posts in that domain. Posts were conglomerate and the learner applied to the entire data set. All posts were stemmed and stop words were removed. The actual equivalence class of each post is defined by the data collection procedure described above. The set of words produced by the learner is then subjected to human interpretation and used as the predicted equivalence class. *Precision* was computed using the number of posts actually belonging to the class (TP) divided by the total number of posts that were predicted as belonging to the class. *Recall* was computed using the number of true positive predictions, divided by the total number of posts that actually belong to the class. All results were averaged over the 5 data sets and the stopping condition for the algorithm was computed as absolute value of the difference between successive log values with a tolerance of 10^{-2}. An example output of the learner using 500 posts and 2 Topics is given below in Table 2-4: Example Distribution of words for 2 topics, 500 postsTable 2-4 and the confusion matrices are presented in Table 2-5:

Topic: 0	Probability	Topic: 1	Probability
race	0.042400	color	0.015446
ride	0.037841	skin	0.014866
bike	0.030079	product	0.012817
back	0.015450	pink	0.012426
gui	0.013716	design	0.011612
road	0.012344	black	0.011557
time	0.012032	free	0.010613
start	0.011139	lip	0.010401
rider	0.009657	ey	0.010276
climb	0.009276	leather	0.009518
wheel	0.009049	gold	0.008919
mile	0.007635	dress	0.008306
hour	0.006867	digit	0.007624
team	0.006841	bag	0.007545
lap	0.006807	brush	0.007324
head	0.006591	makeup	0.007173
front	0.006532	de	0.007169
good	0.006483	fashion	0.006863
move	0.006404	cream	0.006589
minut	0.006351	review	0.006337
car	0.006231	retail	0.006178

Table 2-4: Example Distribution of words for 2 topics, 500 posts

	Cycling	Beauty
Cycling	189	86
Beauty	2	114
	191	*200*

	Cycling	Beauty
Cycling	496	281
Beauty	3	218
	499	*499*

Table 2-5: Example Confusion Matrices for Cycling and Beauty, 200, 500

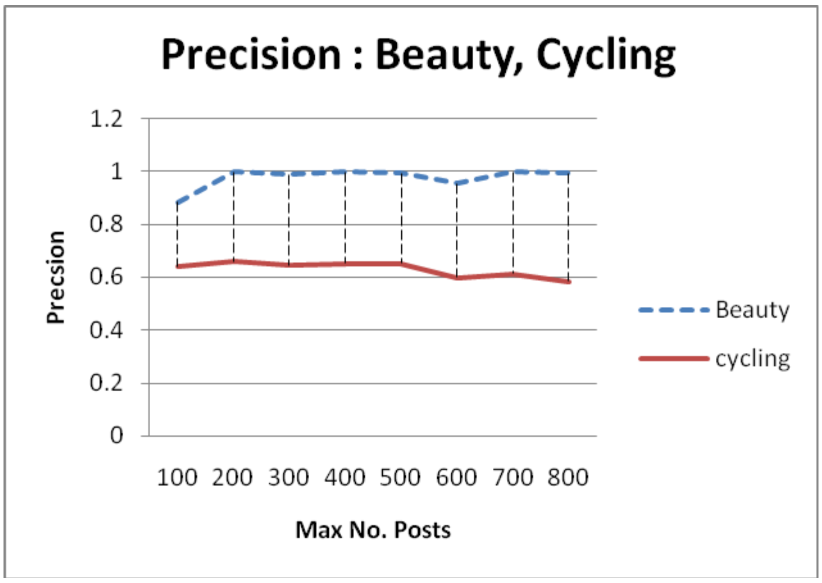


Figure 2-26: Precision Beauty, Cycling

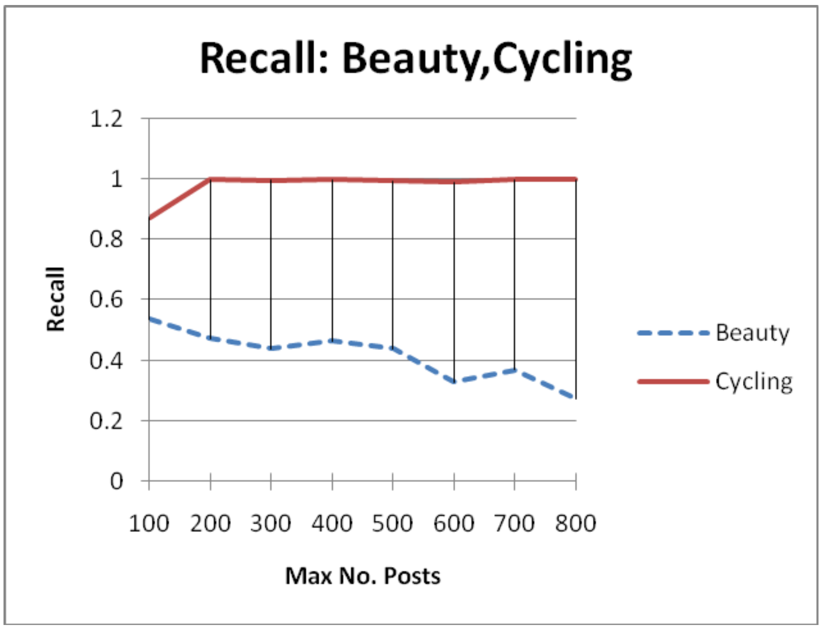


Figure 2-27: Recall Beauty, Cycling

Cycling has a very high recall and low precision, while beauty has a high precision and lower recall. This can also be seen in the confusion matrix where the actual number of documents for the cycling class is very high in comparison to beauty. This suggests some skew in the data set, probably also based on using only 5 samples and variants. This skew can be seen is all instances, except for Instance 2.

No. Posts = 100		beauty	cycling
Instance 1	beauty	57	2
	cycling	43	98
		beauty	cycling
Instance 2	beauty	56	59
	cycling	44	41
		beauty	cycling
Instance 3	beauty	41	0
	cycling	59	100
		cycling	beauty
Instance 4	cycling	98	48
	beauty	2	52
		beauty	cycling
Instance 5	beauty	62	0
	cycling	38	100

These characteristics of the dataset might also explain the drop in recall, as well as the stronger fluctuations for beauty. Additionally, depending on the initial state of the learner, we need to average together multiple samples.

Part 1B: Preliminary Evaluations – How Well Can the Conglomerate Approach Recover Natural Clusters for 3 Domains?

Below we can see the example Topics for 3 Natural Classes and 500 Posts:

Topic: 0	**Probability**	Topic: 1	**Probability**	Topic: 2	**Probability**
race	0.098764	code	0.089493	love	0.009653
bike	0.039804	coupon	0.077763	garden	0.008350
ride	0.033261	exp	0.050015	plant	0.008017
lap	0.030662	ship	0.045326	don	0.007785
sprint	0.025899	free	0.045233	ve	0.007531
rider	0.023011	stock	0.032729	time	0.006176
climb	0.018165	order	0.029887	thing	0.005719
wheel	0.015127	entir	0.029430	dai	0.004893
notebook	0.012226	purchas	0.020637	year	0.004522
team	0.011376	sale	0.014488	blog	0.004443
bicycl	0.009907	gift	0.013613	make	0.004348
racer	0.009574	deal	0.013381	good	0.004237
rode	0.009127	lash	0.012830	work	0.004022
speed	0.008705	save	0.010159	color	0.003995
lcd	0.008329	expir	0.009772	find	0.003790

gear	0.007924	gold	0.008979	friend	0.003714
brake	0.007712	appl	0.008918	peopl	0.003586
mph	0.007362	discount	0.008718	read	0.003578
pedal	0.006704	diamond	0.008592	ll	0.003531
trail	0.006046	suppli	0.007344	book	0.003471
crit	0.005803	oz	0.007305	leav	0.003470

From the table above we observe that example concepts for beauty include: oz - unit of measure for cosmetics, expiration date, product code, free samples and shipping, gift purchases, applying makeup, etc. Except for garden, the concepts are not as clear as those for cycling and makeup. This is consistent through the data set.

In the confusion matrix for the three topics, beauty is distinguished from the other classes with nearly perfect precision. On the other hand, the distinction between cycling and gardening is not clear at all.

No. Posts = 100	Cycling	Beauty	Gardening
Cycling	84	34	66
Beauty	0	24	0
Gardening	14	40	33
	98	*98*	*99*

No. Posts = 500	Cycling	Beauty	Gardening
Cycling	174	143	160
Beauty	0	74	0
Gardening	325	282	339
	499	*499*	*499*

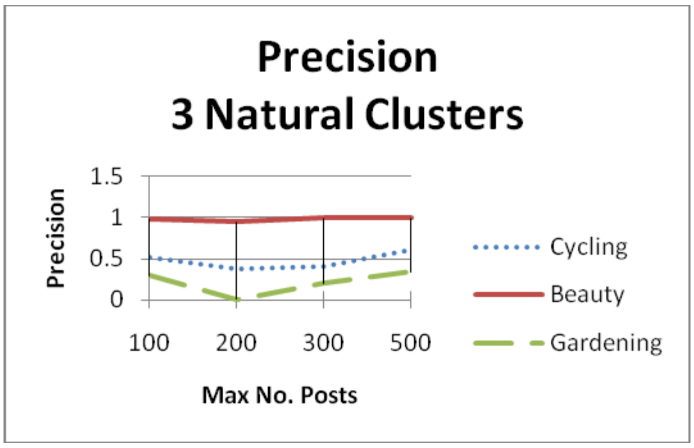


Figure 2-28: Precision 3 natural clusters

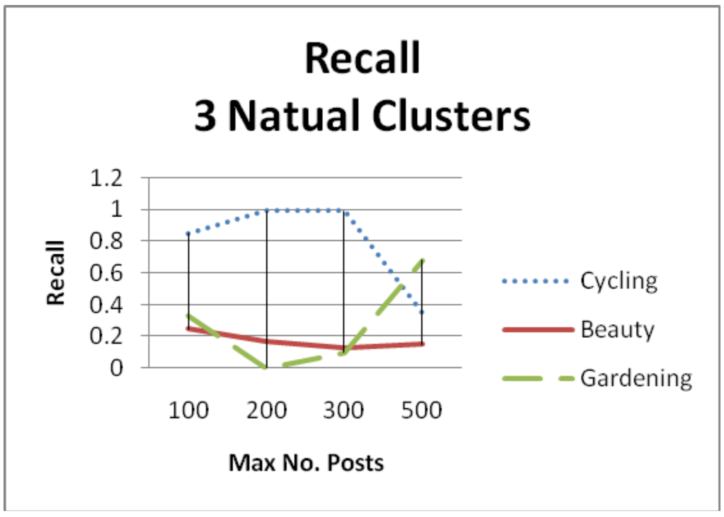


Figure 2-29: Recall 3 natural clusters

Using the conglomerate approach we see that the algorithm has a problem correctly classifying the documents for the gardening domain. The blogs in this domain were seeded with the blogger "obsessivegardener". However, upon closer examination of the gardening corpus produced at random, we see a large number of documents are about recent virus attack on their computer, creative writing, and speed walking.

Example:
(seed: obsessivegardener)
http://woodlandsworld.blogspot.com/2007/01/sorry.html
http://epiphanyprayer.blogspot.com/2007/04/creative-writing-wednesday-smells-like.html
http://natureremains.blogspot.com/2008/05/look-at-those-legs.html

One the other hand, it is apparent that some "noise", with respect to natural class cluster is introduced and this affects the conglomerate performance as whole, as the number and type of the bloggers discussion becomes more heterogeneous. However, it is clear, from observing the gardening data in the set, that this gardener is also interested in staying fit and walking (http://natureremains.blogspot.com/2008/05/look-at-those-legs.html).

The community-based collection process we followed, would thus categorize the blogger as interested only in gardening – based on blogrolls. This suggests two things: 1) that the "natureremains" blogger is interested in staying fit and 2) the bloggers, who listed this blog on their blog roll, might have some interest in this specific (non-gardening topic) on which the "natureremains" writes.

Work is ongoing to compare experiment on this issues and Part II: Conglomerate vs. Common themes, i.e. compare if the common themes from comparative text mining perform better.

2.3.3..2 Part III: Common Theme vs. Multi-document Summary

In the preliminary results presented in Part I and II, the number of topics considered is unrealistic is practice. We seek to determine how well the comparative mining is at determining common themes as the scale increases, while reducing the need for a human interpretation but mimicking it.

The evaluation involves considering the overall task from the perspective of text summarization. In essence, the learning algorithms seek to summarize blog posts and assign a probability distribution describing the words in all blog posts. In a parallel fashion, we assume that the posts are summarized by an agent in such a way that salience scores are assigned to sentences or paragraphs units of a blog post - extracting summaries with the highest scores. For this task we use MEAD[6] an existing open source tool, for multiple and single document summarization.

The design for this ongoing experiment is outlined below:

1. ***Training:*** Select 80% training and 20% test data for each domain
    a. Train on 80% portion of the data
    b. Select the top-K words for each topic, from the trained data
    c. Using the remaining 20% for testing: perform automatic text summarization on each document to extract the most salient-K keywords
    d. For each topic compute the Semantic Distance (discussed below) between the training set top-K and the test set salient-K. Order the computed semantic distances in increasing order. The topic whose set of words has the smallest semantic distance is taken as the best matching topic.
2. ***Prediction:*** Perform a prediction P(d|j): for each document we have a distribution over the topics.
3. ***Precision & Recall:*** both results for test and training sets are ranked and precision is used to evaluate the ranking.

Semantic Distance is a metric of relatedness that takes into account the relative position of two words as defined by a language and ontology. For example, the tem "nickel" is more related to "coin" than it is to "credit card" since "coin" is an immediate sub-summer of the term "nickel" in the WordNet hierarchy – even though all are mediums of monetary exchange. Although there is additional overhead in maintaining the sense of the words, from its original context, we use this approach to compare semantics between the top-N ranked words in the training and test set. We use the MEAD document summarization tools for determining the top-N salient words in the test set.

2.3.3..3 Part IV: Collection-Specific Theme vs. Single-Document Summary

Using comparative text mining, we are able not only to model common themes across blog sites, but also themes that are specific to a blog site. In this experiment, we want to measure the ability of the comparative text mining algorithm to detect collection specific topics as compared with the conglomerate approach. In the conglomerative approach, the notion of a blog site is lost by "flatten" all blog posts into a single representation. The procedure for carrying out this experiment is similar to the procedure outlined in Part III, except for the fact that the single (and not multi-document) summary of MEAD will be used.

2.3.3..4 Part V: Tuning

Finally in the 5th Part, Tuning, we experiment with finding the best values for the parameter of the algorithm. Specifically, we want to know: (1) What is the optimal value for the number of topics learn; (2) How high (or low) should the collection-specific coefficient be and under what conditions do we get an optimal setting and (3) likewise for the cross-collection coefficient. The first experiments performed in tuning parameters suggest that more processing power, than initially used is needed.

Exploiting the Mined Knowledge

Referring back to Figure 2-22, one of the goals is to provide input for Information diffusion, as well as community and user profiling.

The text mining algorithms presented here are crucial steps towards these goals. Specifically, it is not possible to model or exploit the results of information diffusion if we do not know: what information flows, between whom the information flows and how it evolved. A comparative text mining approach provides us with the needed tools for addressing these fundamental issued in the information diffusion task.

In the diffusion process, each node represents a blogger and each weighted hyper-edge represents the extent to which the topics that the two bloggers have in common diverged. The learner outputs topic distributions over all the blog posts and the distributions can be taken as weights on the edges

[6] http://www.summarization.com/mead/

between pairs of bloggers, for example, by using the Kullback–Leibler (KL) Divergence. KL is it is a non-commutative measure representing the information divergence, or difference between two probability distributions. We use the asymmetric measure since all blog posts are time stamped. We assume that, a blogger A can only influence another blogger B, if A posted before B. Further, the KL value is non-negative, and will be 0.0 only if the two distributions are identical. Again this conforms intuitively to the information diffusion task - information must diffuse along an edge leading away from a blogger node and is assumed not to loop back on itself. Finally, the knowledge mined from the SNBA is exploited to support search and recommendations within PHAROS.

2.4 PM – Personalization Module

2.4.1 Description

The **Personalization Module (PM)** focuses on providing personalized search and recommendation functionality. This component is especially important because it unlocks the value of personal information stored in USIS in order to improve the users' experiences inside the PHAROS platform. PM takes requests from the QIRP module and uses information stored in USIS as basis for performing personalization. The necessary information about user interests is computed offline in the UCP and SNBA modules and is then retrieved by PM both during the pre-computation of personalized ranking values, as well as during the model building phase of the recommendation engine, model to be later used to compute recommendations online.

The **Personalized Search** and **Recommender System** are the fundamental components of PM and below we describe each of them in detail.

Personalized Search Component

In addition to general search capabilities, the Search Engine component provides personalized search results matching the profile of the user or any groups the user belongs to. Personalization involves both a filtering and ranking of results. Result filtering is used to limit the result set to content, which fits the user information need, and content the user has permission to view. Ranking of results takes into account user and group preferences and ranks content, believed to be of high relevance to the user, higher than content which is of general interest. Ranking parameters are part of the query, and are inserted by the PM module. Different personalization techniques are developed within Information Retrieval field, like query re-weighting and query expansion, just to name a few. We provide details on implemented methods in the sub-section 2.4.3 - algorithms.

The personalized search capabilities assume re-ranking of the relevant multimedia items using information about previous user's interactions with a PHAROS platform. All data about previous user's queries, clicked results, tags in use, etc., should be exploited to provide a more precise search output. For providing high quality personalized services, user profiles must be kept up-to-date as interests may change over time. Accurate user profiles often also depend on the community a user belongs to. Therefore, inferring user profiles has to be complemented with the construction of community profiles.

Recommender System Component

Recommender Systems support people by identifying products or services they will appreciate, helping them to face the information explosion, where the complexity of offers exceeds the user's capability to survey them and reach an optimal decision.

Different approaches have been suggested for supplying meaningful recommendations to users and some of them implemented and deployed successfully over e-commerce and services sites like Amazon (Amazon), Netflix (Netflix), or MyStrands (MyStrands).

State-of-the-art Recommender Systems mostly use a variant of Collaborative Filtering (CF), an approach to solve the recommendation task that relies on historical data gathered from users, rather than using the information about content. The underlying assumption of the CF approach is that those who agreed in the past tend to agree again in the future, capturing human behavior: people searching for an interesting item they have little or no information, tend to rely on friends to recommend items they tried and liked.

The goal of the PM's Recommender System Component is to identify neighborhoods of users with similar taste, based on the profiles built by the UCP and SNBA modules and stored in the USIS. To build a user's neighborhood, the Recommender System Component relies on information of past user interactions (e.g., explicit ratings, tags assigned) or implicit grading methods based on user behavior actions, such as the time spent on a particular item web page. In order to provide recommendations for a given user, the system uses her corresponding neighborhood to compute a list of items interesting for her. A similar approach is also taken to consider neighborhoods of similar items to be exploited in order to provide recommendations of similar contents and resources.

The component architecture has been designed in order to support *pluggable* recommendation algorithms (Figure 2-31), that can be further developed and extended, for example to adapt the behavior of the PM module depending on the context or the user data available, and to try to complement some weaknesses and strengths of the algorithms themselves, by creating hybrid models that combine them.

2.4.2 Architecture

In case of the **Personalized Search Component** there are two critical factors for the effective personalization: the quality of the user profile and the query processing time. The user profiles are pre-computed by UCP and SNBA components and stored in the USIS module. The PM module communicates with USIS to fetch and transform these profiles into a format required by different personalization algorithms, see Figure 2-30.

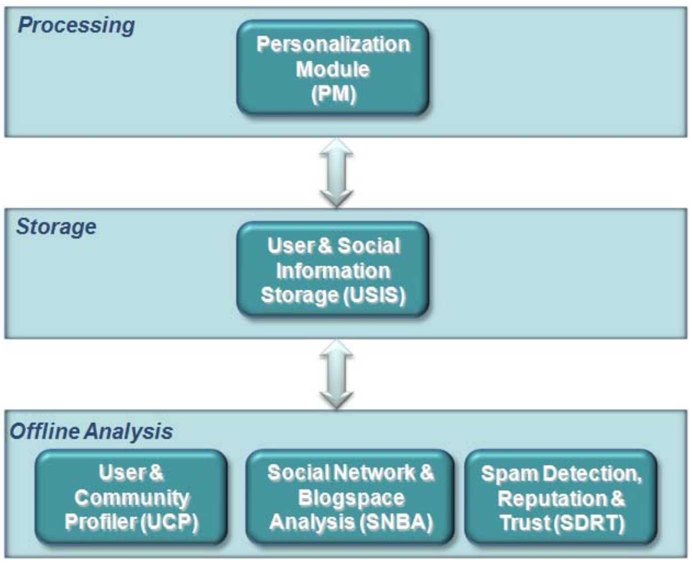

Processing
Personalization Module (PM)
Storage
User & Social Information Storage (USIS)
Offline Analysis
User & Community Profiler (UCP)
Social Network & Blogspace Analysis (SNBA)
Spam Detection, Reputation & Trust (SDRT)

Figure 2-30: Interaction between PM component and other Social Media Modules

During query time, QIRP sends a request with original query to PM and receives back a new query which includes the necessary modifications for a better ranking. Some of the variants of rank computation we provide, such as Community Rank and Object Rank require the full index of resources, which is stored in the Search Engine component. Therefore the PM module periodically calls the corresponding Search Engine service to update these values with respect to updated user profiles.

One part of the PM module is the *Query Personalization* component (see Fig. X.X.2). When a query comes from QIRP via a web service, the *Query Parser* transforms it into internal format for further processing. The *Personalization Selector* component chooses the requested personalization method and asks the *Profile Retriever* for a necessary user or community profile information. The *Query Personalizer* transforms the original query and sends the resulted personalized query back to QIRP.

The **Recommender System Component** also depends directly on the user profiles pre-computed by UCP and SNBA. Once these profiles are retrieved from USIS, the *Modeler* sub-component (Figure 2-31) builds a model of the user preferences. The computation is done offline periodically and the results are also stored back into the USIS.

Once the model has been built, the Recommendation Engine is ready to compute the necessary list of personalized recommendation for a given user, as well as her neighborhood (User-based recommendations).

Depending on the context, the Recommendation Engine is also capable to recommend similar items given a resource (Item-based recommendations).

In the following section we present the in detail the algorithms used by both the Personalized Search and Recommender System components.

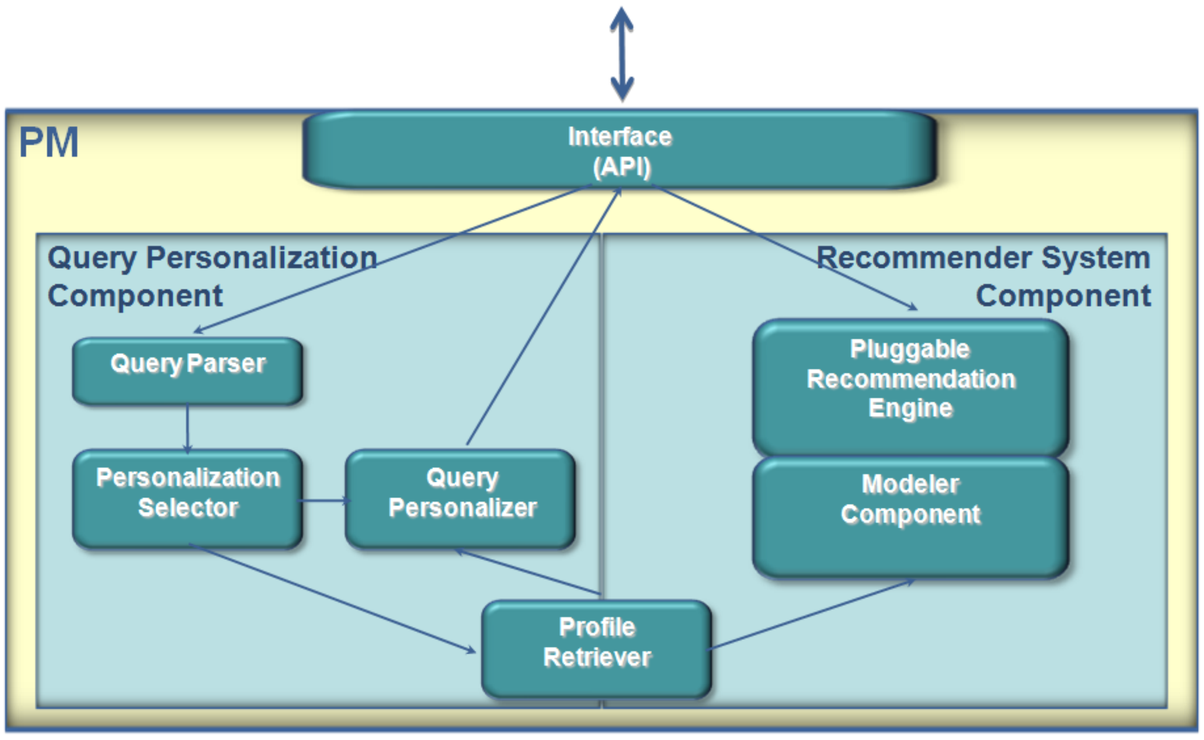


Figure 2-31: Architecture of the PM Module

2.4.3 *Algorithms*

For the **Personalized Search Component** we implemented 5 relatively standard information retrieval algorithms for relevance feedback and query expansion. The algorithms can be used as standalone methods as well as in combination with each other. The effectiveness of the proposed methods has been proved in general text and multimedia retrieval, while their practical usefulness depends on available data and quality of the user profiles:

1. **Fields Reweighting**. Results are initially ranked using default values for the given query fields. Based on previously collected information regarding user tags and associated ratings an algorithm can specify a different weight for each field and this information is then used for ranking. The frequently used user's tags and query fields receive higher weights and results are biased towards them. This relevance feedback technique is based on a well-known Rocchio method (Rocchio, 1971). Long-term feedback is obtained from tag usage of the user or

user group and is captured from UCP module in the form of a user profile, which tracks *user-specific weights* and other feedback-based parameters.
The vector-space representation of the query is modified so that more important term dimensions are emphasized and similarity between query and each item of interest is affected. Top-N most similar items are then presented to the user.

2. **Results Filtering**. Create restrictions based on the user profile, like removing from the ranked list of results the items that the user dislikes. We use Generalized Query Point Movement method (Ortega-Binderberger & Mehrotra), which previously received poor ratings from a user are used to extract tags which the user does not like. The result items containing such tags are moved down the ranking.
The algorithm has a similar mechanism to Fields Reweighting technique, but negative assessments are used to compute the user profiles. This personalization technique is effective when users explicitly mark items as non-interesting.

3. **Query Expansion**. Based on the users' tag usage patterns we compute tags' similarity. Additional keywords are added to the query based on preferences from the user profile. This method (Qiu & Frei, 1993) is one the most frequently used for personalization and can significantly increase recall in situations, where original query does not have enough results.
Query expansion is essentially adding new features to the query vector and re-ranking the results accordingly. The initial query terms are still of higher importance for the ranking. The precise values for the algorithm tuning have to be defined based on available data, which can be done as soon as first user profiles and interaction histories are collected.

4. **Community-Tag Rank**. This algorithm is inspired by work on Topic-Sensitive PageRank (Haveliwala, 2002) and (Chirita, Nejdl, Paiu, & Kohlschütter, 2005), but based on textual similarity rather than link-based similarity. Document model includes all the fields that can be extracted for multimedia content. We create a community-tag vector for each of the identified communities. To make computation scalable we assume that number of communities is significantly lower than a total number of users. A Community-Tag Rank represents a similarity between an item and a community vector, which is composed from tag usage statistics of all users belonging to the community. Communities can be defined based on explicit membership in particular communities and automatically computed clusters of users. A user belongs to one or more communities (topic groups) and we can compute a linear combination of the community vectors for items before query time, which is called Community-Tag Rank. A single score of Community-Tag Rank is associated with each multimedia item—community pair. During the computation of the item-query similarity the personalized Object Rank vector is used as a factor for ranking as a query-independent parameter.

5. **Community-Rating Rank**. This method is similar to Community-Tag Rank, but is based on a collaborative filtering rather than document model. A community profile for a Community-Rating Rank computation consists of previously issued users' ratings and independent of multimedia item tags. This average rating allows re-ranking retrieved items with respect to their overall popularity among community members and quality of each returned result. As with Community-Tag Rank, this value is query independent and it is pre-computed offline. During query time this value is added to the item relevance score.

The methods 1 – 3 require only interaction with USIS to obtain the necessary user profiles, which are prepared beforehand by UCP. The methods 4 – 5, Community-Tag Rank and Community-Rating Rank also need to store pre-computed values at the Search Engine component.

In the case of the **Recommender System Component** the following algorithms are provided, and are also combined with each other to provide the target functionality:

1. **Tag-aware Collaborative Filtering**. It exploits the tag-based profiles, in both dimensions (user, tag) and (item, tag) to build user and item neighborhoods in order to compute

personalized recommendations (Firan, Nejdl, & Paiu, 2007). Tag-based user profiles are defined as collections of tags together with corresponding scores representing the user's interest in each of these tags. Once the profiles have been computed, they are arranged in a User-Tag matrix structure, which is then used to derive the recommendations applying CF techniques that group similar users in order to suggest them valuable items that in turn have been inferred by their associated tags.

2. **Standard User-based Collaborative Filtering.** It supports (1) and can also be used alone or as part of other Recommendation Engine to exploit different kinds of profiles, and not only explicit ratings as in traditional CF (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994) (Konstan, Miller, Maltz, Herlocker, Gordon, & Riedl, 1997). The recommendations for each individual user are obtained by identifying a neighborhood of similar users and recommending items that this group of users found interesting. The design recommendations described by (Herlocker, Konstan, & Riedl, 2002) been also considered in the implementation of this algorithm.

3. **Standard Item-based Collaborative Filtering.** The recommendation task in this case is focused on the items' similarity, rather than on the users' similarity. It also supports (1) and its main objective is to produce a list of recommendations given a target item (Deshpande & Karypis, 2004). This recommendation algorithm uses the item-to-item similarities to compute the relations between the different items. It builds a model that captures these relations and then applies this model to derive the top-N recommendations for an active user. The model, which at the core is an item-item matrix representation, is built based on the original user-item matrix of user profiles that reflects their aggregated historical information of consumed items. Each item is associated with a vector in the users' space, and these vectors are then used to compute the similarity among the items. Once the similarities have been computed, for each item, just the most similar *k* items are kept on the model, where *k* is an input for the algorithm. The model computed is used during the recommendation step, where the goal is to recommend similar items for a given one.

3. Use case Scenarios

In this section we present a set of use cases which are supported by the current implemented functionalities of the modules composing the Social Media Beta release. The use cases described in the rest of this section encompass some of the most important functionalities of our modules. Some of them are new and some have already been described in deliverable D3.2.1. However, since those included in D3.2.1 have been updated in the meantime, we include them again in the present document. Part of the listed use cases are subject to be evaluated during the Showcase evaluation, which will provide us rich feedback on further possible refinements and extensions.
A list summarizing all supported use cases is given below and after that we describe each of them in detail:

1. User does a search by tag
2. User does a personalized search
3. Being aware by popular tags (or bookmarks)
4. User comments or manually annotates an item
5. User deletes an annotation
6. User sees or edits his tag profile
7. User edits or creates new user profile
8. User adds a friend
9. User removes a friend
10. User Joins Social Group
11. User Invites Other User to Join a Social Group
12. Get Recommendations (Pull)
13. Receive Personalized Recommendations (Push)
14. Explore Neighbourhood
15. Receive Recommendation of Related Content

3.1 Use case: User does a search by tag

3.1.1 Use case definition

USE CASE	User does a search by tag
Goal in Context	User submits a tag query to the system and expects relevant results
Scope & Level	PHAROS System
Preconditions	User id/ sessionId are known. If the user is not logged in, the user-id is a default setting
Success End Condition	User receives results if available according to access rights, terminal capabilities and indexed content
Failed End Condition	User does not receive results despite their availability
Primary actors	User
Secondary Actors	External content sources in case an external content object is used for querying (recorded multimedia, example picture, etc.)
Trigger	Query is entered at user interface and submitted

DESCRIPTION	Step	Action
	1	User constructs a query using a tag and specifies the preferred results view and personalization options.
	2	User submits his query.
	3	System returns results according to the access rights of the user and user generated metadata matching the tag query

EXTENSIONS	Step	Branching Action
	4	After a first result to a query the user filters the results by a tag (tag refinement see sequence diagram 3)
SUB-VARIATIONS		Branching Action
	1a	User searches both in keywords and tags at the same time (see sequence diagram 2)

RELATED INFORMATION	User does a search by tag
Priority:	highest
Performance	under 1sec
Frequency	Depending on application scenario up to many thousand times per second
Channels to actors	Interactive, online

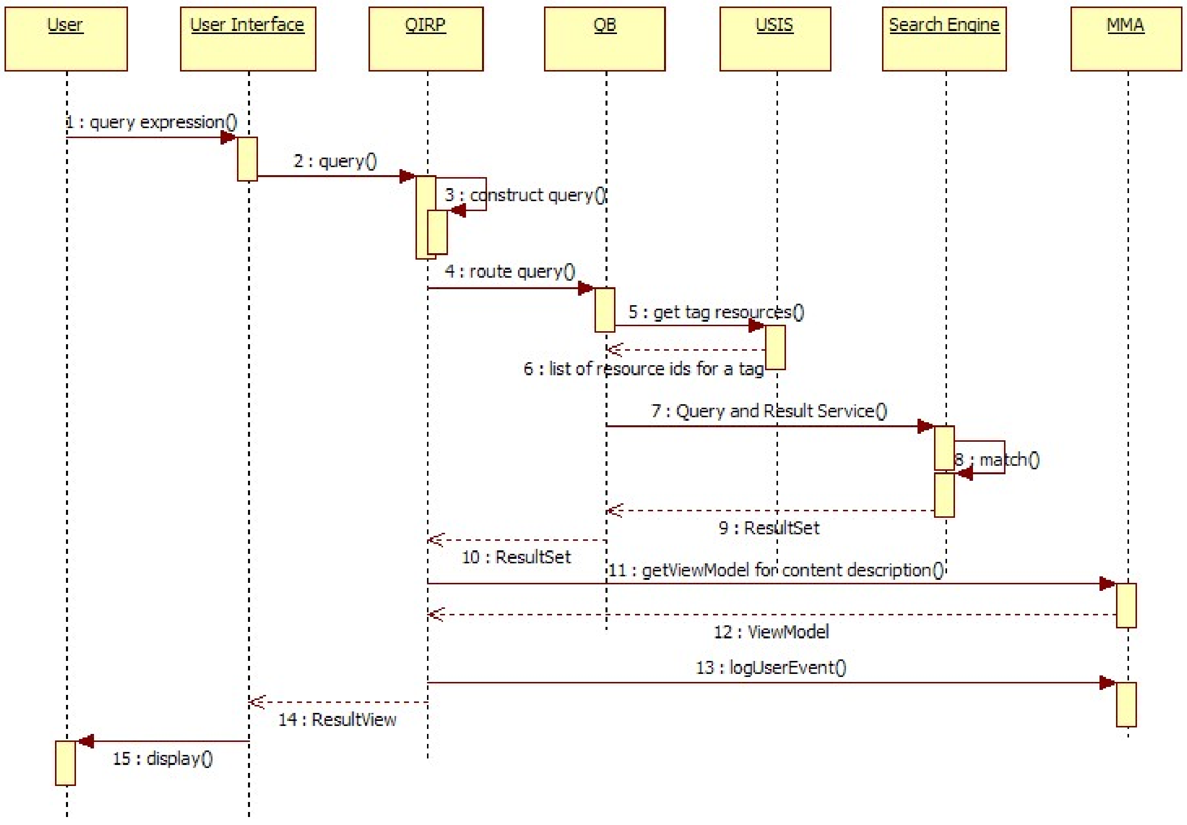
3.1.2 Associated sequence diagrams

Sequence diagram (1): User does a search by tag only

Name	Sequence Diagram (1)
Trigger	User enters a tag query
Actors	User
Participants	User Interface, QIRP, QB, USIS, Search Engine, MMA
Precondition	User has entered a query into the User Interface. The QIRP knows the group membership of the User (this was set at the start of the user session).

Step	Action	Comment
1	The User Interface sets the result viewing preferences for a particular user	
2	The User Interface passes the query together with the user context to the QIRP (the query contains the contentID to the content to be used for content-based search)	
3	QIRP constructs the query	
4	QIRP passed the query to QB to route it to the components involved (first annotations -> USIS, keywords -> Search Engine)	
5	QB queries USIS to get resources tagged with the specified tag	
6	USIS returns a frequency ordered list of resource ids tagged accordingly	
7	QB queries the Search Engine to retrieve the metadata and descriptions for the resource ids returned by USIS	
8	Search engine matches resource ids to get metadata and descriptions (optionally static ranking?)	
9	Search Engine returns the result set	
10	The result set is passed to QIRP	
11	The QIRP requests the result view for the result items in the Resultset from MMA, using the Retrieve Metadata Service	
12	MMA returns the Resultview	
13	The QIRP logs the user event to the USI Storage, using the User Behavior Submission Service	
14	The QIRP returns the result view to the User Interface	
15	The User Interface displays the results to the User	

User does a search by tag (only)

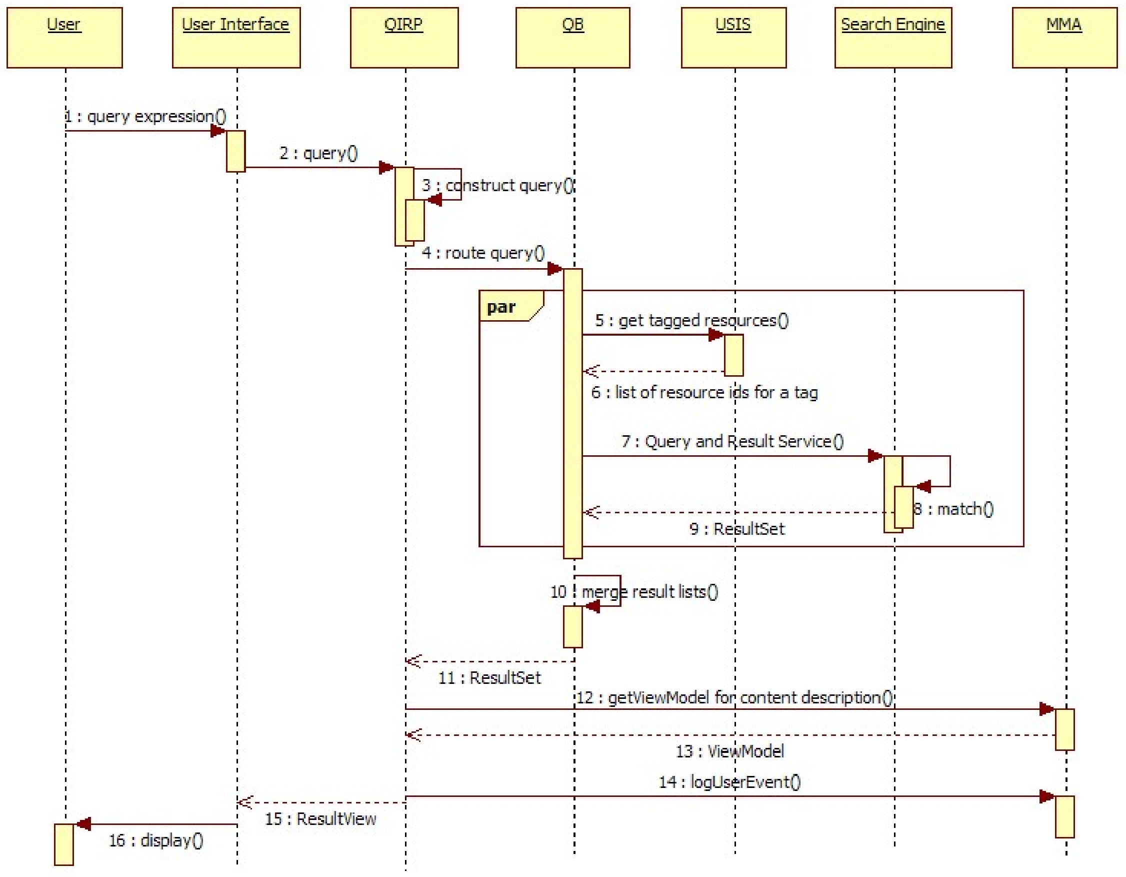


Sequence diagram (2): User does a search by tag and keywords

Name	Sequence Diagram (2)
Trigger	User enters a query for tags and keywords
Actors	User
Participants	User Interface, QIRP, QB, USIS, Search Engine, MMA
Precondition	User has entered a query to search both for keywords and in user generated metadata into the User Interface. The QIRP knows the group membership of the User (this was set at the start of the user session).

Step	Action	Comment
1	The User Interface sets the result viewing preferences for a particular user	
2	The User Interface passes the query together with the user context to the QIRP (the query contains the contentID to the content to be used for content-based search)	
3	QIRP constructs the query	
4	QIRP passed the query to QB to route it to the components involved (first annotations -> USIS, keywords -> Search Engine)	
5 par	QB queries USIS to get resources tagged with the specified tag	Querying USIS and the search engine should be done in parallel
6	USIS returns a frequency ordered list of resource ids tagged accordingly	
7 par	QB queries the Search Engine to retrieve matching results from the index	
8	Search engine matches the query terms against its index	
9	Search Engine returns his result set	
10	QB merges the different result lists coming from USIS and the Search Engine	
11	The result set is passed to QIRP	
12	The QIRP requests the result view for the result items in the Resultset from MMA, using the Retrieve Metadata Service	
13	MMA returns the Resultview	
14	The QIRP logs the user event to the USI Storage, using the User Behavior Submission Service	
15	The QIRP returns the result view to the User Interface	
16	The User Interface displays the results to the User	

User does a search by tag and keywords

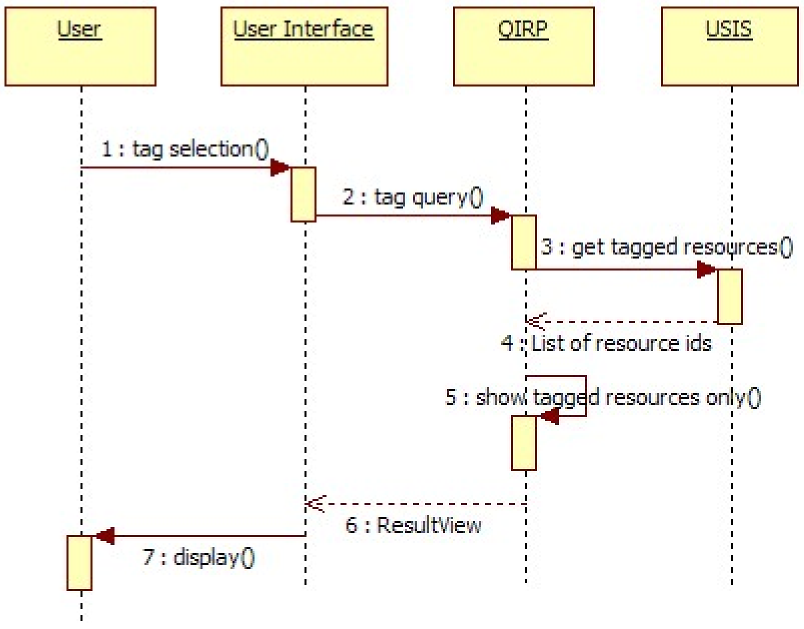


Sequence diagram (3): User refines a result set by tag

Name	Sequence Diagram (3)
Trigger	User does a search
Actors	User
Participants	User Interface, QIRP, USIS
Precondition	After a query user is viewing a Resultset intending to filter it by tag

Step	Action	Comment
1	Given a displayed search resultset by clicking on a showed tag the user wants to filter to show only the results also tagged accordingly.	
2	The clicked tag is send as query from the UI to QIRP	
3	QIRP queries USIS to retrieve resource ids tagged with this tag	
4	USIS returns a list of resource ids tagged accordingly. The list will be ordered by frequency (the most often tagged resources first)	
5	QIRP refines the initial resultset to only show resources annotated with the tag	
6	The QIRP returns the result view to the User Interface	
7	The refined resultset is displayed to the user	

User refines a Result set by tag



3.2 Use case: Personalized Search

3.2.1 *Use case definition*

USE CASE *new*	Personalized Search
Trigger	User does a personalized search
Actors	
Participants	User Interface (UI), Query Interaction and Result Presentation (QIRP), Personalization Module (PM), Search Engine: Query Broker (QB)
Precondition	User has entered a query and selected personalization options (or default ones are used) in the User Interface. The QIRP knows the group memberships of the User (this was set at the start of the user session).

3.2.2 *Associated sequence diagrams*

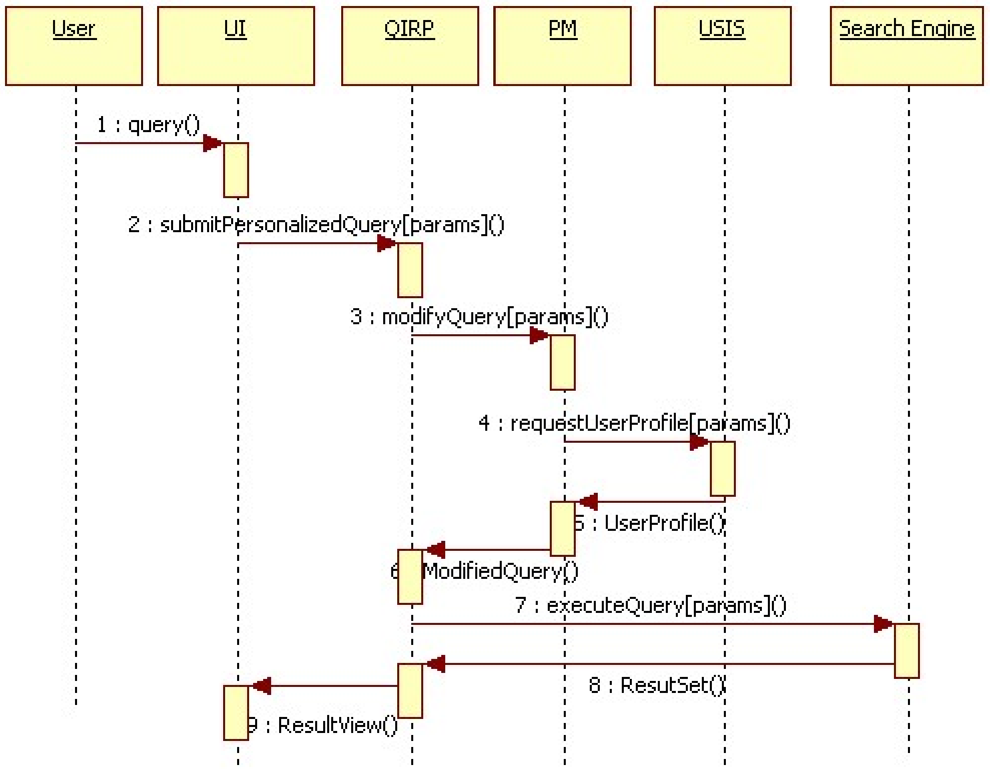
Sequence diagram (1): Query evaluation including personalization

Step	Action	Comment
1	User pre-selects personalization options (or uses default ones) and issues a query.	
2	The User Interface passes the query (containing a number of parameters) and personalization parameters.	
3	The QIRP sends a query enriched with personalization options to Personalization Module for query reformulation.	
4	Personalization Module requests User Tag Profile from the User & Social Information Storage.	
5	User & Social Information Storage returns User Tag Profile.	
6	Personalization Module returns modified (personalized) query to QIRP.	
7	The QIRP poses the query to the Search Engine using the Query and Result Service	
8	The Search Engine returns the result set to the QIRP	
9	The QIRP returns the result view to the User Interface	

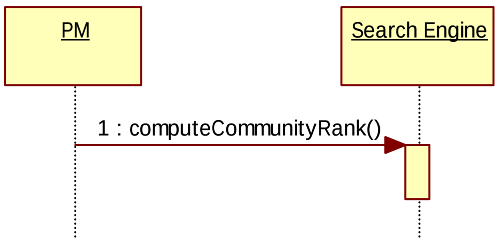
Sequence diagram (1) represents query evaluation which includes personalization of the query. The sequence diagram is valid for personalized search algorithms 1 – 5, with different request parameters for each method. At the step 2 parameters include algorithm names, which should be applied for personalization, for example (“Fields Reweighting, Query Expansion”). Based on these parameters, at the step 4 the Personalization Module requests different fields to be retrieved from the USIS.

At the sequence diagram (2) we show offline re-computation of the Community-Tag Rank and Community-Rating Rank values, stored at the Search Engine.

Query evaluation including personalization (1)


User
UI
QIRP
PM
USIS
Search Engine
1 : query()
2 : submitPersonalizedQuery[params]()
3 : modifyQuery[params]()
4 : requestUserProfile[params]()
5 : UserProfile()
6 : ModifiedQuery()
7 : executeQuery[params]()
8 : ResutSet()
9 : ResultView()

Update CommunityRank value (2)



3.3 Use case: Being aware by popular tags (or bookmarks)

3.3.1 Use case definition

USE CASE *new*	Being aware by popular tags (or bookmarks)
Goal in Context	To be aware of recent trends, the user wants to receive recommendations for popular content based on popular tags or bookmarks.
Scope & Level	PHAROS System
Preconditions	None
Success End Condition	Popular tags are displayed (together with the associated popular resources) to the user.
Failed End Condition	Popular tags cannot be retrieved and displayed (together with the associated popular resources). For bookmarks, no resources could be retrieved by bookmarking statistics
Primary actors	User
Secondary Actors	...
Trigger	User accesses the 'being aware' section/heading

DESCRIPTION	Step	Action
	1	User clicks 'being aware' to get recommendations based on popular tags or bookmarks from the PHAROS system.
	2	(The top k) Popular tags (with their associated resources) are shown to the user.

EXTENSIONS	Step	Branching Action
		...
SUB-VARIATIONS		Branching Action
	2a	For bookmarks, only the top k resources will be displayed (as bookmarks have no label)

RELATED INFORMATION	Being aware by popular tags (or bookmarks)
Priority:	high
Performance	Sub second response time
Frequency	once per user session
Channels to actors	Interactive, online

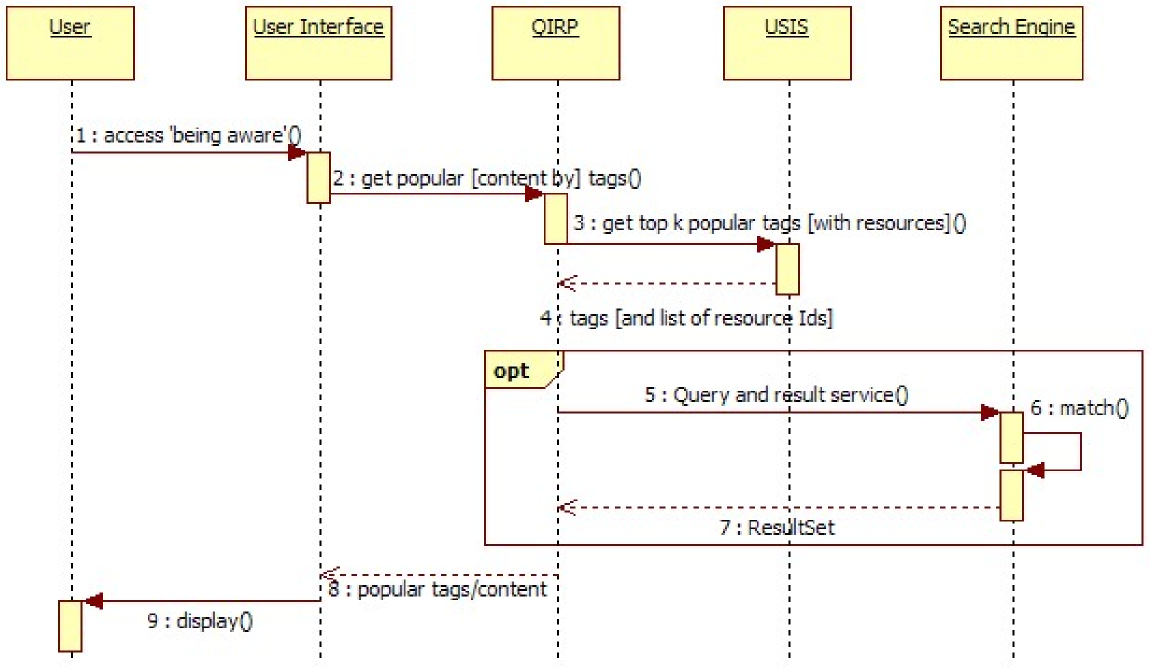
3.3.2 Associated sequence diagrams

Sequence diagram (1): Being aware by popular tags

Name	Sequence diagram (1)
Trigger	User accesses the 'being aware' section/heading
Actors	User
Participants	UI, QIRP, USIS (, Search Engine)
Precondition	None

Step	Action	Comment
1	User accesses the 'being aware' section/heading by clicking	
2	UI sends a request for popular tags to QIRP	
3	QIRP queries USIS to get the top k popular tags (and the associated resource ids)	
4	USIS returns the current top tags (with their top resources as frequency sorted list of resource ids)	only top tags or also the associated resources? Depends on UI and space. Probably a tag cloud is the best choice, then by 'SearchByTag' resources for this tag can be retrieved, by again calling 'SearchByTag' or 'RefineByTag' lists can be stepwise refined to narrow down the list
5 opt	If not only tags, but resources with tags should be recommended, QIRP send the list of resource ids returned from USIS to the search engine to get metadata and descriptions	Necessary, if not only tags as suggestions
6	Search engines matches resource ids to get content descriptions	
7	The search engine returns the result set	
8	QIRP returns popular tags (and the corresponding result set with content and descriptions) to the UI	
9	Popular tags are displayed to the user (together with their associated resources)	

Being aware by popular tags

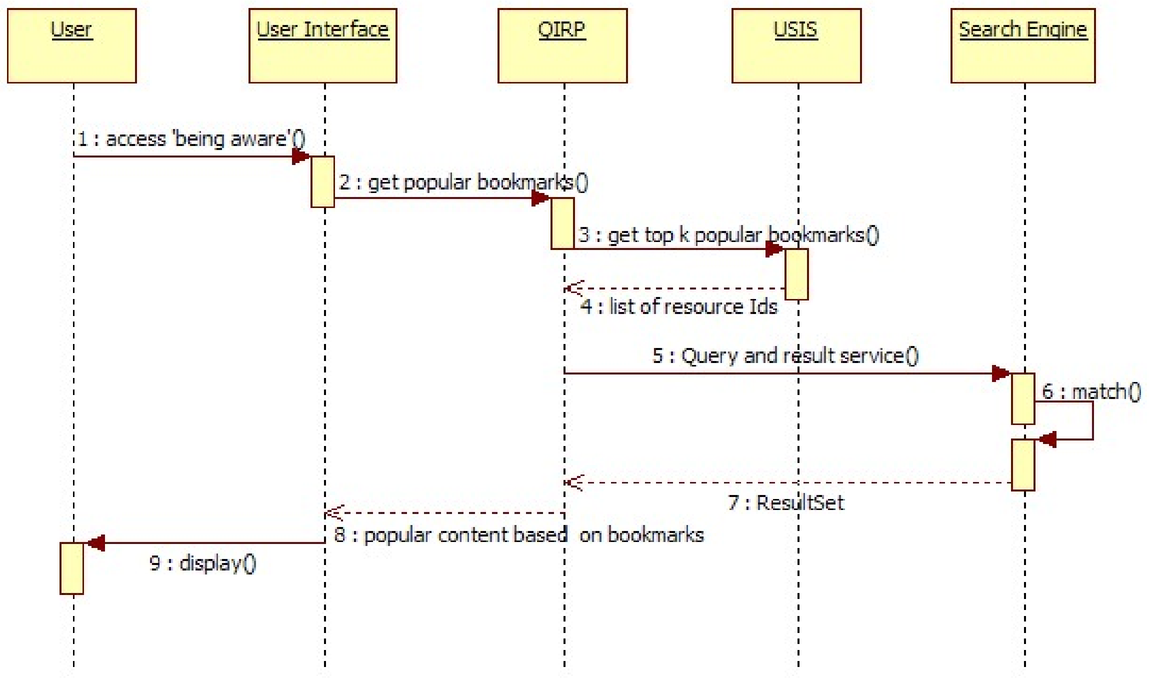

User
User Interface
QIRP
USIS
Search Engine
1 : access 'being aware'()
2 : get popular [content by] tags()
3 : get top k popular tags [with resources]()
4 : tags [and list of resource Ids]
opt
5 : Query and result service()
6 : match()
7 : ResultSet
8 : popular tags/content
9 : display()

Sequence diagram (2): Being aware by popular bookmarks

Name	Sequence diagram (2)
Trigger	User accesses the 'being aware' section/heading
Actors	User
Participants	UI, QIRP, USIS, Search Engine
Precondition	none

Step	Action	Comment
1	User accesses the 'being aware' section/heading by clicking	
2	UI sends a request for popular bookmarks to QIRP	
3	QIRP queries USIS to get the top k popular bookmarks	
4	USIS returns the current top bookmarks, i.e. a frequency sorted list of resource ids	Should they be grouped together (according to bookmarks)?
5	QIRP sends the list of resource ids returned from USIS to the search engine to get metadata and descriptions	
6	Search engines matches resource ids to get content descriptions	
7	The search engine returns the result set	
8	QIRP returns the result set of popular content to the UI	
9	Popular bookmarks are displayed to the user	

Being aware by popular bookmarks


User
User Interface
QIRP
USIS
Search Engine
1 : access 'being aware'()
2 : get popular bookmarks()
3 : get top k popular bookmarks()
4 : list of resource Ids
5 : Query and result service()
6 : match()
7 : ResultSet
8 : popular content based on bookmarks
9 : display()

3.4 Use case: User comments or manually annotates an item

3.4.1 Use case definition

USE CASE 1.19 in D3.2.1	User comments or manually annotates an item
Goal in Context	User annotates (comments, tags, rates or bookmarks) a resource, e.g. for future retrieval or information organization.
Scope & Level	PHAROS System
Preconditions	User selected/viewed a resource.
Success End Condition	Resource has a new annotation
Failed End Condition	Annotation got lost
Primary actors	User
Trigger	User enters an annotation in the user interface

DESCRIPTION	Step	Action
	1	User annotates a resource.
	2	System stores annotation as user generated metadata for an item (frequencies are updated).

EXTENSIONS	Step	Branching Action
	3	Annotation becomes visible to all users
	3a	For ratings, tags and bookmarks, frequencies must be updated before displaying them in the user interface. For ratings, average rating has to be recalculated as well

RELATED INFORMATION	User comments or manually annotates an item
Priority:	top
Performance	1 sec
Frequency	multiple times per user session
Channels to actors	online, interactive

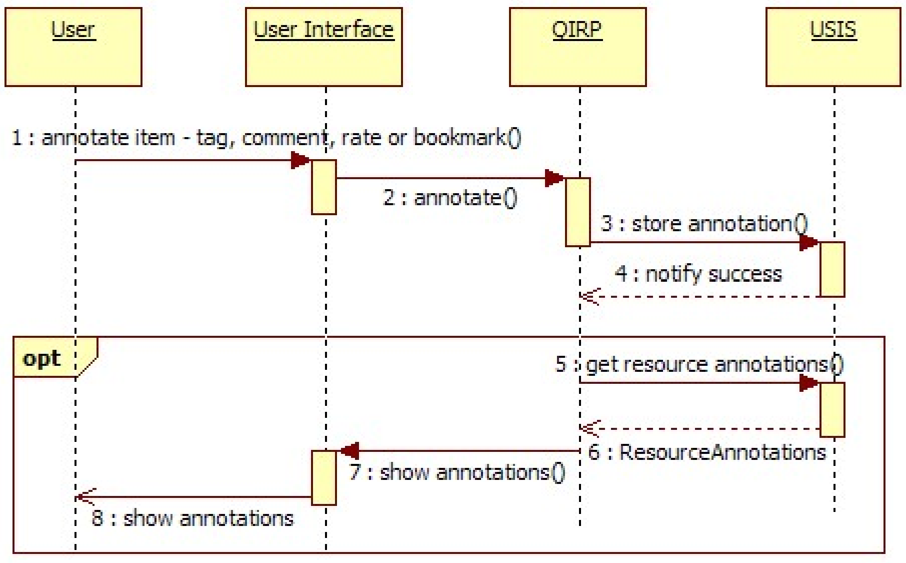
3.4.2 Associated sequence diagrams

Sequence diagram (1): User comments or manually annotates an item

Name	Sequence Diagram (1)
Trigger	Sequence diagrams 1.26, 1.29 or similar
Actors	User
Participants	UI, QIRP, USIS , UCP
Precondition	Logged in user views a resource

Step	Action	Comment
1	User annotates (i.e. tags, comments upon, rates or bookmarks) a resource	
2	Annotation is sent from the UI to QIRP	
3	QIRP sends annotation to USIS to be stored in the DB	
4	USIS returns a success notification (true/false)	
5 opt	All recent (user) annotations of the resource can be queried from the USIS storage	Optional, as depending on where the user is currently different actions seem reasonable after an annotation event occurred
6	USIS returns the recent annotations (for a resource [by the user])	
7	Resource annotations are presented in the User Interface	
8	The new information becomes visible to the user	

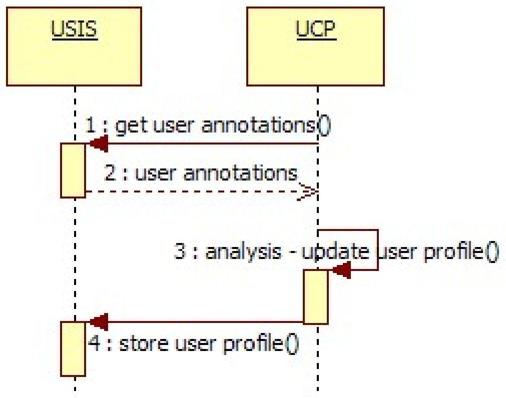
User comments or manually annotates an item



Sequence diagram (2): User profile update (offline, scheduled)

Step	Action	Comment
1	Based on a schedule to be specified, UCP will query all/recent annotations from USIS	Scheduled updates are currently planned for once after each session
2	USIS returns the annotations	
3	Internal analysis of annotations (e.g. normalizing or grouping tags, calculating frequencies, opinion mining and sentiment analysis from comments)	For tags, VTT's tag normalization/analysis involved. For comments, FT's opinion analysis will extract implicit user ratings for movies/actors/film makers
4	Updated user profile is written back to USIS	

User profile update: sequence diagram 1.32f


USIS
UCP
1 : get user annotations()
2 : user annotations
3 : analysis - update user profile()
4 : store user profile()

3.5 Use case: User deletes an annotation

3.5.1 *Use case definition*

USE CASE *new*	User deletes an annotation
Goal in Context	User deletes a personal annotation (a personal comment, tags, ratings or bookmarks) of a viewed resource
Scope & Level	PHAROS System
Preconditions	User must be logged in and selected/viewed a resource that already has annotations made by the user
Success End Condition	Resource annotation is deleted
Failed End Condition	Annotation is not deleted
Primary actors	User
Secondary Actors	...
Trigger	User requests deletion of an annotation

DESCRIPTION	Step	Action
	1	User deletes a personal annotation (tag, comment, rating or bookmark) for a resource
	2	System removes the annotation as metadata for the item (or frequencies are updated resp.).

EXTENSIONS	Step	Branching Action
	3	Annotation becomes visible to all users
	3a	For ratings, tags and bookmarks, frequencies must be updated before displaying them in the user interface. For ratings, average rating has to be recalculated as well
SUB-VARIATIONS		Branching Action
	2a	If an annotation was not by user requesting the deletion, the user will be informed and steps 2 and optional extension step 3 are skipped.

RELATED INFORMATION	User deletes an annotation
Priority:	high
Performance	1sec
Frequency	infrequent
Channels to actors	Online, interactive

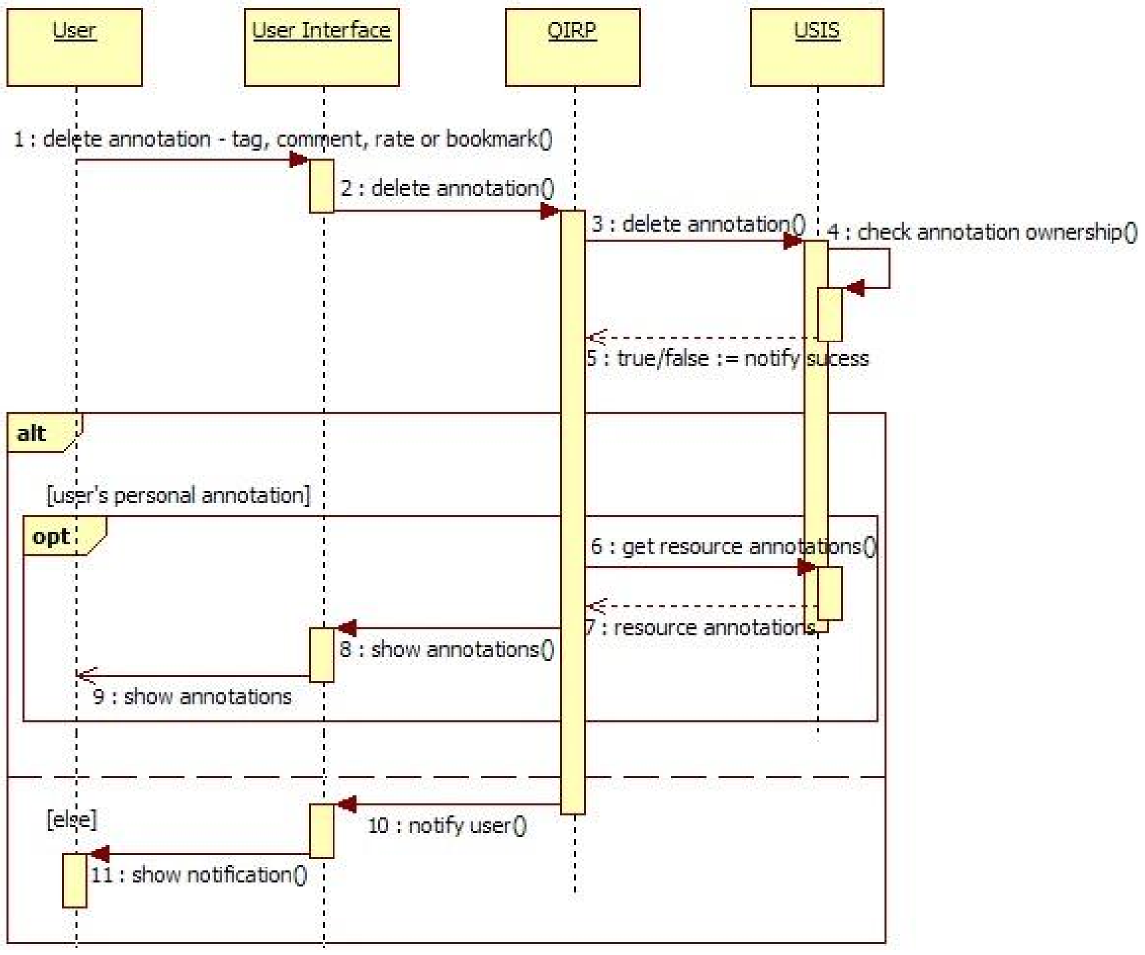
3.5.2 Associated sequence diagrams

Sequence diagram 1.32a: User deletes an annotation

Name	Sequence Diagram 1.32
Trigger	Sequence diagrams 1.26, 1.29 or similar
Actors	User
Participants	UI, QIRP, USIS (,UCP)
Precondition	Logged in user views a resource that has annotations made by the user

Step	Action	Comment
1	User requests the deletion of an annotation (i.e. tags, comments, ratings or bookmarks) of a resource	
2	Delete request is sent to QIRP	
3	QIRP sends deletion request to USIS to delete in the DB	Commenting, rating and bookmarking to be implemented
4	USIS checks whether the annotation was made by the user himself	
5	USIS returns a deletion success notification (true or false)	
6 alt [IF] opt	IF: If the user had the right to delete the tag, done changes (updated annotations of the resource) can be read from the USIS storage	
7	USIS returns all resource annotations	
8	Updated resource annotations are presented in the User Interface	
9	The changed information becomes visible to the user	
10 alt [ELSE]	ELSE: Notification is displayed in the user interface	
11	Notification about denied permission is displayed to the user	

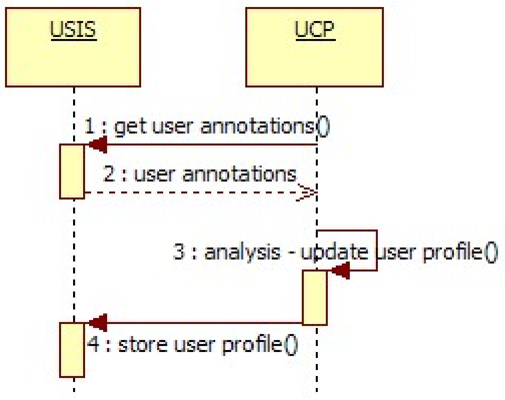
User deletes an annotation



Sequence diagram (2): User profile update (scheduled, offline)

Step	Action	Comment
1	Based on a schedule to be specified, UCP will query all/recent annotations from USIS	Scheduled updates are currently planned for once after each session
2	USIS returns the annotations	
3	Internal analysis of annotations (e.g. normalizing or grouping tags, calculating frequencies, opinion mining and sentiment analysis from comments)	For tags, VTT's tag normalization/analysis involved. For comments, FT's opinion analysis will extract implict user ratings for movies/actors/film makers
4	Updated user profile is written back to USIS	

User profile update: sequence diagram 1.32f


USIS
UCP
1 : get user annotations()
2 : user annotations
3 : analysis - update user profile()
4 : store user profile()

3.6 Use case: User sees or edits his tag profile

3.6.1 Use case definition

USE CASE *new*	User sees or edits his tag profile
Goal in Context	User views his tag based interest profile. User modifies the tag profile, adding or deleting tags and co-occurrences, increasing or decreasing the importance (weight) of the tag and tag co-occurrence.
Scope & Level	PHAROS System
Preconditions	User has tagged resources in PHAROS or has given his username to one of the supported social media sites (like del.icio.us, last.fm)..
Success End Condition	Modifications to tag based interest profile are stored in PHAROS
Failed End Condition	Visualization or modification fails
Primary actors	User
Secondary Actors	USIS
Trigger	User wants to view/modify his tag based interest profile

DESCRIPTION	Step	Action
	1	User sends a request (using a web interface) to the PHAROS system
	2	Visualization is shown to user.
	3	User modifies his tag profile.
	4	Updated tag profile is stored in USIS.

RELATED INFORMATION	User sees or edits his tag profile
Priority:	high
Performance	2 sec
Frequency	occasionally
Channels to actors	online, interactive

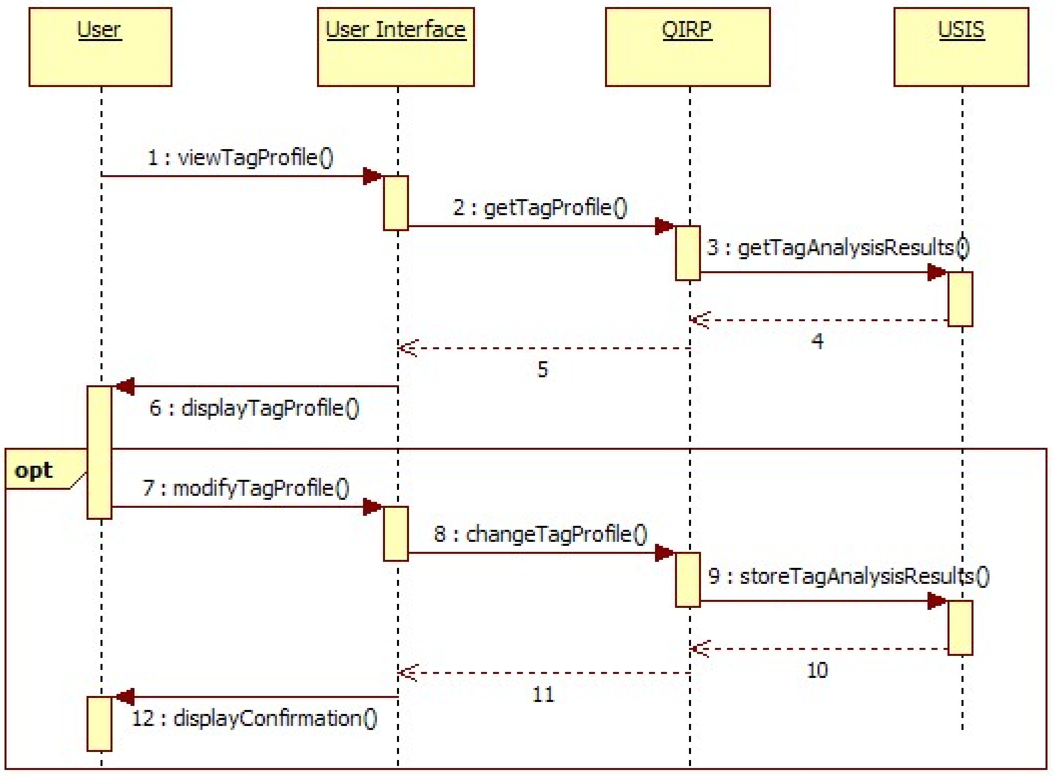
3.6.2 Associated sequence diagrams

Sequence diagram (1): Use sees or edits his tag profile

Name	Sequence diagram (1)
Trigger	User request
Actors	User
Participants	UI, QIRP, USIS, UCP
Precondition	Logged in user views his tag profile

Step	Action	Comment
1	User requests to view his tag profile	
2	UI submits user requests to QIRP	
3	QIRP queries USIS to get the tag profile of the user (the current user profile active)	
4	USIS retrieves tags analyzed before (scheduled, offline) by VTT's tag analysis algorithms	
5	QIRP returns tag profile information to UI	
6	display visualization to the user	
7 opt	User modifies his tag profile	
8	Change of tag profile is send to QIRP	
9	QIRP calls USIS to save modifications	
10	USIS returns confirmation about update success	
11	QIRP returns confirmation to UI	
12	UI displays confirmation to the user	

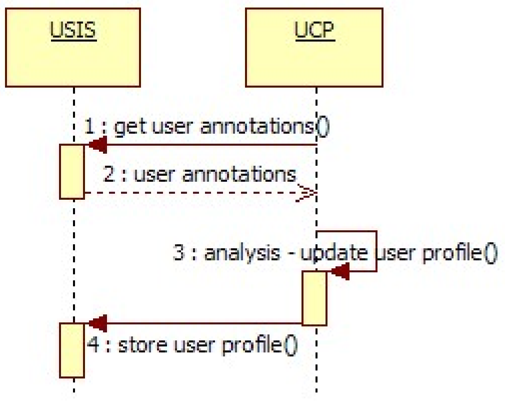
User sees or edits his tag profile



Sequence diagram (2): User profile update (scheduled, offline)

Step	Action	Comment
1	Based on a schedule to be specified, UCP will query all/recent annotations from USIS	Scheduled updates are currently planned for once after each session
2	USIS returns the annotations	
3	Internal analysis of annotations (e.g. normalizing or grouping tags, calculating frequencies, opinion mining and sentiment analysis from comments)	For tags, VTT's tag normalization/analysis involved. For comments, FT's opinion analysis will extract implict user ratings for movies/actors/film makers
4	Updated user profile is written back to USIS	

User profile update: sequence diagram 1.32f


USIS
UCP
1 : get user annotations()
2 : user annotations
3 : analysis - update user profile()
4 : store user profile()

3.7 Use case: User edits or creates new user profile

3.7.1 Use case definition

USE CASE *new*	User edits or creates new user profile
Goal in Context	User changes information in his user profile(s) or he adds a new profile.
Scope & Level	PHAROS System
Preconditions	User is registered in PHAROS. The user is authenticated.
Success End Condition	Profile data has been changed successfully.
Failed End Condition	Constrains on data are not respected: account data are not changed.
Primary actors	User
Secondary Actors	none
Trigger	User wants to change specific data related to his user profile

DESCRIPTION	Step	Action
	1	User provides profile data to be changed (web interface).
	2	Modified or new profile information is sent to USIS where it is stored, i.e. the user profile is updated
	3	Updated profile information is displayed to the user

EXTENSIONS	Step	Branching Action
	1b	In case the user wants to have an additional user profile (e.g. 'work') he creates a new profile first by clicking on some button 'Create new profile'. He can then fill in the fields to provide the information.

RELATED INFORMATION	User edits or creates new user profile
Priority:	high
Performance	1 sec
Frequency	Infrequent, see trigger.
Channels to actors	Interactive, online, web browser

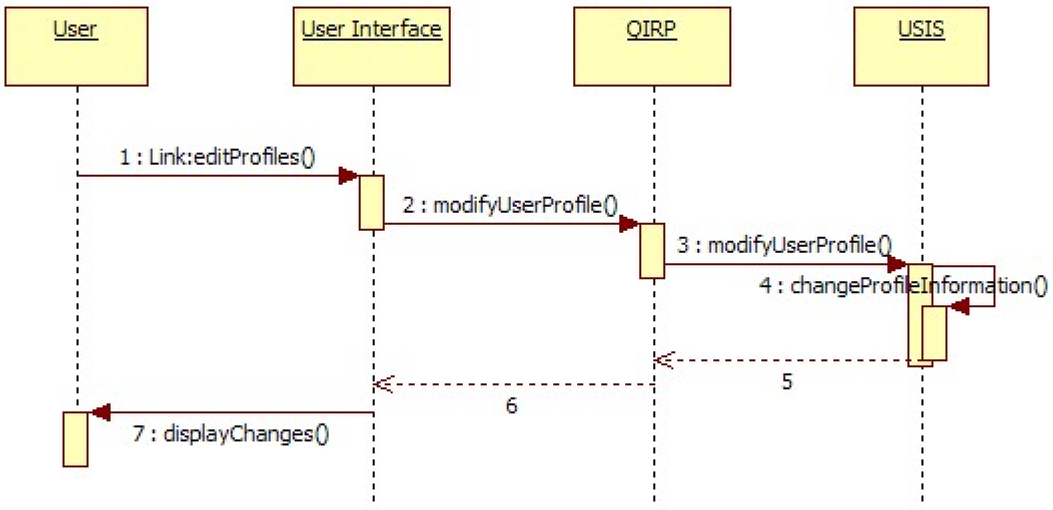
3.7.2 Associated sequence diagrams

Sequence diagram (1): User edits or creates new user profile

Name	Sequence diagram (1)
Trigger	User requests to add or to change a user profile
Actors	User
Participants	UI, QIRP, USIS
Precondition	The user is registered and authenticated

Step	Action	Comment
1	User requests the UI to execute the user profile information update or he requests creating a new profile respectively. The user also provided the edited profile information.	
2	UI sends request to modify profile information to QIRP	
3	QIRP sends to USIS the updated/new profile command along with the current profileID of the user requesting it and the edited profile info	
4	USIS sets the new profile information	
5	Confirmation (USIS replies to QIRP with operation confirmation)	
6	Confirmation (QIRP forwards operation confirmation to the UI)	
7	Confirmation (UI shows to user the new, updated information)	

User edits or creates new user profile



3.8 Use case: User adds a friend

3.8.1 *Use case definition*

USE CASE *new *	User adds a friend
Goal in Context	A user wants to add another person registered in PHAROS as his personal friend.
Scope & Level	PHAROS System
Preconditions	Both users have a PHAROS account. Inviting user must be logged in.
Success End Condition	User has the invited other PHAROS user as a new friend.
Failed End Condition	Friendship connection could not be build.
Primary actors	User
Secondary Actors	Another PHAROS User
Trigger	Request to add some PHAROS user as a friend

DESCRIPTION	Step	Action
	1	For example visiting the profile of some user in PHAROS, the user requests to add this person as his friend.
	2	A notification informs the user that his request has been sent to the person.

EXTENTIONS	Step	Branching Action
	3	After the other person accepted the friendship request, the new/updated friendship information will be visible in the heading "My Friends"
SUB-VARIATIONS		Branching Action

RELATED INFORMATION	User adds a friend
Priority:	high
Performance	1sec
Frequency	infrequent
Channels to actors	online, interactive

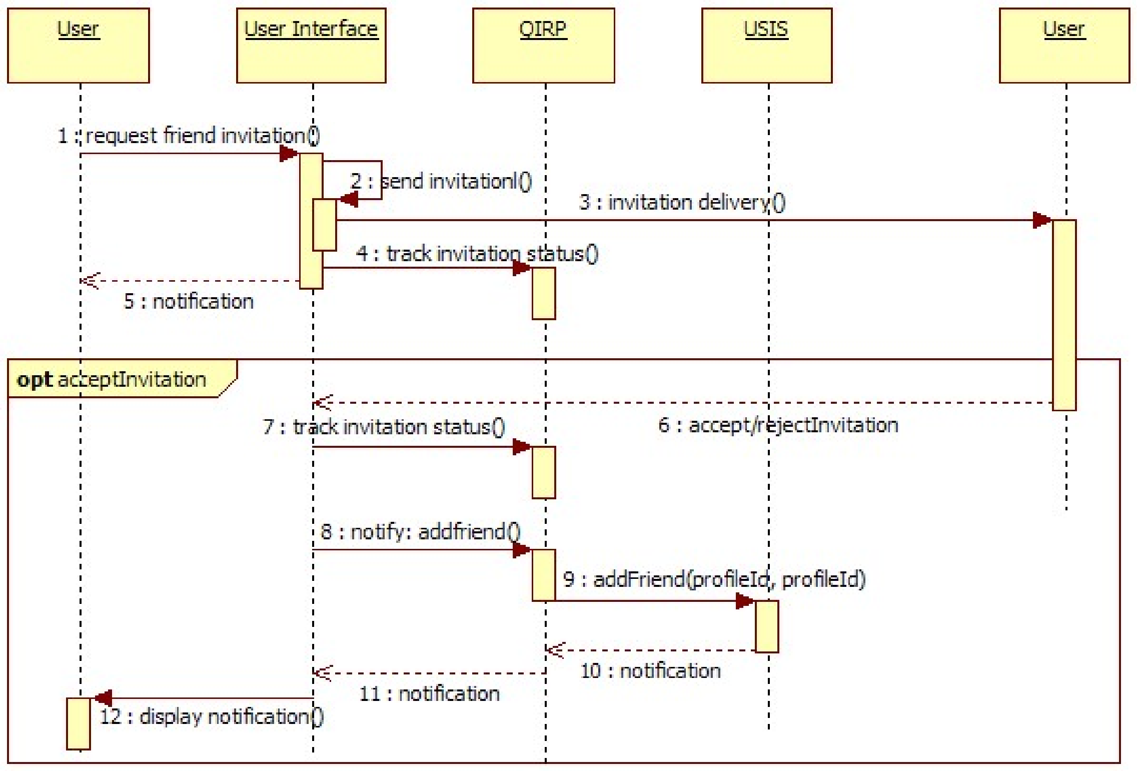
3.8.2 Associated sequence diagrams

Sequence diagram (1): User adds a friend

Name	Sequence diagram (1)
Trigger	User requests adding a person as a friend
Actors	
Participants	UI, QIRP, USIS
Precondition	User is logged in. Person to be added is a PHAROS user with a profile

Step	Action	Comment
1	User requests to invite another PHAROS user to become his friend.	
2	Via the User Interface the invitation request is sent.	
3	The invitation is delivered to the other user (e.g. via email)	
4	QIRP tracks and logs the action of the user inviting another user.	
5	The User Interface displays a notification about the message being sent.	
6 opt	The other person accepts the friendship invitation.	
7	The changes invitation status (accepted) is tracked and logged within QIRP.	
8	QIRP sends the request to add the two as friends to USIS.	
9	USIS returns a confirmation notification	
10	QIRP returns notification.	
11	User Interface displays the notification about (successful) adding.	

User adds a friend


User
User Interface
QIRP
USIS
User
1 : request friend invitation()
2 : send invitationI()
3 : invitation delivery()
4 : track invitation status()
5 : notification
opt acceptInvitation
6 : accept/rejectInvitation
7 : track invitation status()
8 : notify: addfriend()
9 : addFriend(profileId, profileId)
10 : notification
11 : notification
12 : display notification()

3.9 Use case: User removes a friend

3.9.1 *Use case definition*

USE CASE *new*	User removes a friend
Goal in Context	A user wants to remove another registered PHAROS from his friend list.
Scope & Level	PHAROS System
Preconditions	Both users have a PHAROS account and are friends. User must be logged in.
Success End Condition	The other user is removed from the user's friend list.
Failed End Condition	Friendship relation was not removed.
Primary actors	User
Secondary Actors	
Trigger	Request to remove some PHAROS user from one's friend list

DESCRIPTION	Step	Action
	1	Visiting his friend list, the user requests to remove a particular friend from the list.
	2	A notification about the removal is displayed to the user.

EXTENTIONS	Step	Branching Action
	3	The user will not get awareness notifications or recommendations or similar with respect to the former friend any more.
SUB-VARIATIONS		**Branching Action**

RELATED INFORMATION	User removes a friend
Priority:	medium
Performance	<1sec
Frequency	very infrequent
Channels to actors	online, interactive

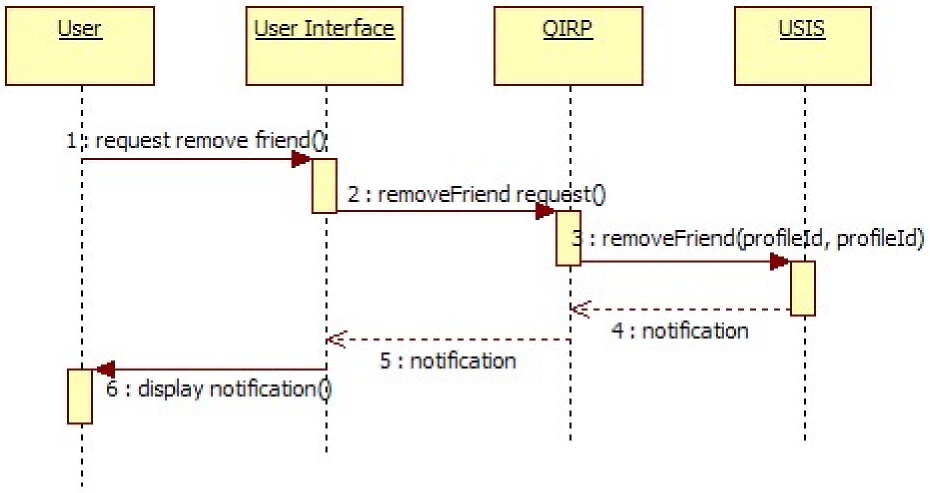
3.9.2 Associated sequence diagrams

Sequence diagram (1): User removes a friend

Name	Sequence diagram (1)
Trigger	User requests adding a person as a friend
Actors	
Participants	UI, QIRP, USIS
Precondition	User is logged in. Person to be removed is in the user's PHAROS friend list.

Step	Action	Comment
1	User requests to remove a friend from his friend list.	
2	The user request is sent to QIRP where the action is tracked/logged.	
3	QIRP sends the request to USIS.	
4	USIS returns a confirmation notification.	
5	QIRP returns notification.	
6	User Interface displays the notification about (successful) removal.	

User removes a friend


User
User Interface
QIRP
USIS
1 : request remove friend()
2 : removeFriend request()
3 : removeFriend(profileId, profileId)
4 : notification
5 : notification
6 : display notification()

3.10 Use case: User Joins Social Group

3.10.1 Use case definition

USE CASE	User Joins Social Group
Goal in Context	User requests to be made a member of a PHAROS social group
Scope & Level	PHAROS System
Preconditions	The use is not already a member of the request group
Success End Condition	PHAROS System has added the user to the group profile
Failed End Condition	PHAROS System isn't able to perform invitation, consequently the invitee isn't added to the profile of the inviter and vice versa
Primary actors	User
Secondary Actors	
Trigger	User who wants to become a member of a group

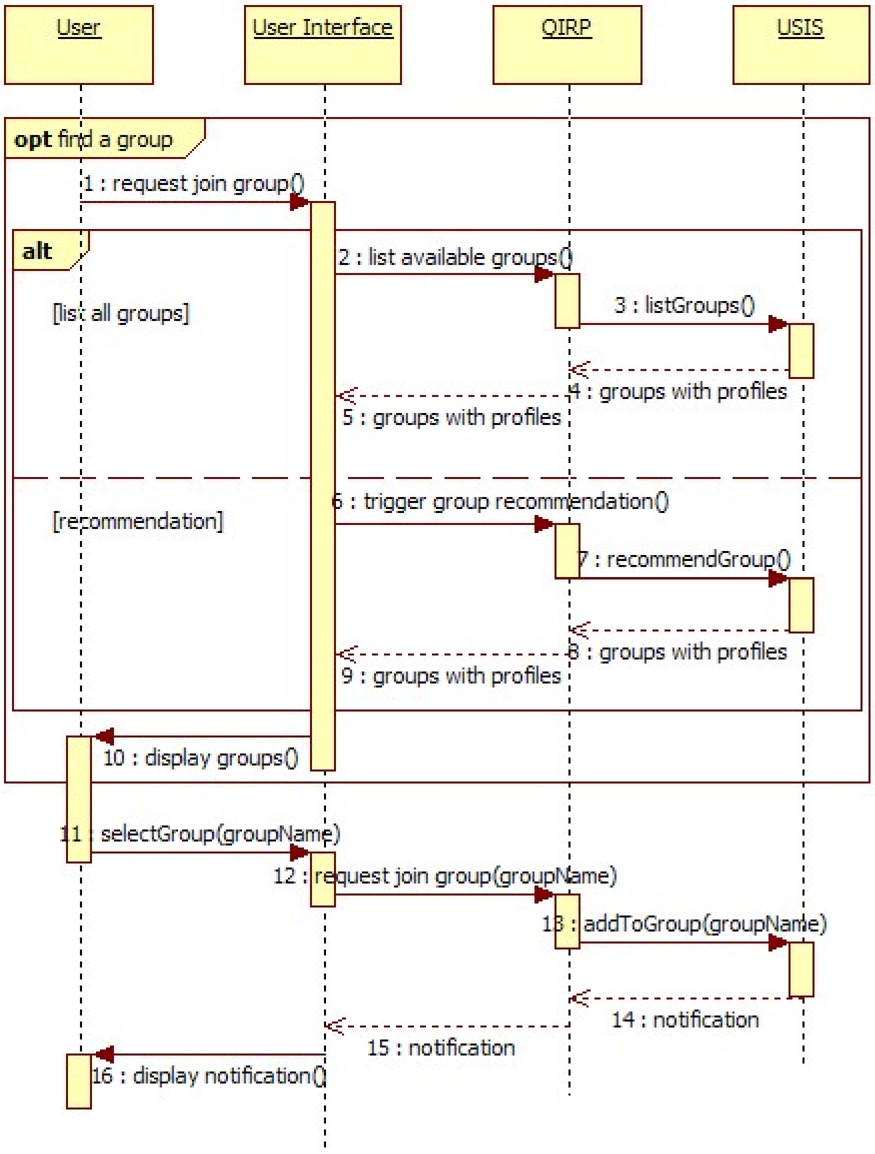
DESCRIPTION	Step	Action
	1	User wants to join a PHAROS group
	2	A list of available groups are presented to the user
	3	Alternatively, the user may receive a recommendation for groups to join
	4	After the user selects the group, the group name and user information is sent via QIRP to the USIS
	5	The user is added to the group and notified of the request status

3.10.2 Associated sequence diagrams

Sequence Diagram: User Joins Social Group

Step	Action	Comment
1 opt	User request to join a group	
2 alt IF	The user interface make a request for the list of available groups	
3	QIRP sends a list group request to the USIS	
4	The USIS returns the group and profile as a reply	
5	Group and profile are sent to the user interface for presentation	
6 ELSE	Upon receiving the group profile, the User Interface triggers the group recommendation request from the QIRP	
7	The QIRP sends a request for a group recommendation to the USIS	
8	The USIS responds with group profiles to the QIRP	
9	The QIRP sends the group profile to the User Interface for presentation	
10	The group information is displayed to the user	
11	Once the user selects a group, via the user interface	
12	The user interface triggers a join group request to the QIRP	
13	QIRP makes a request to add the user to the group	
14	After the USIS adds the group, a notification is sent back to QIRP	
15	QIRP notifies the User Interface of the status of the request	
16	The user interface displays the status of the request to the user	

User Joins Social Group



3.11 Use case: User Invites Other User to Join a Social Group

3.11.1 Use case definition

USE CASE	User invites other user to join a social group
Goal in Context	User invites other PHAROS user to join a social group he is already a member of
Scope & Level	PHAROS System
Preconditions	The inviter is member of a social group that the invitee is not a member of
Success End Condition	PHAROS System has added the invitee to the social group
Failed End Condition	PHAROS System isn't able to perform the invitation and the invitee does not get added to the social group
Primary actors	User: role of inviter
Secondary Actors	Use: role of invitee
Trigger	User wants to invite another PHAROS user to join a social group

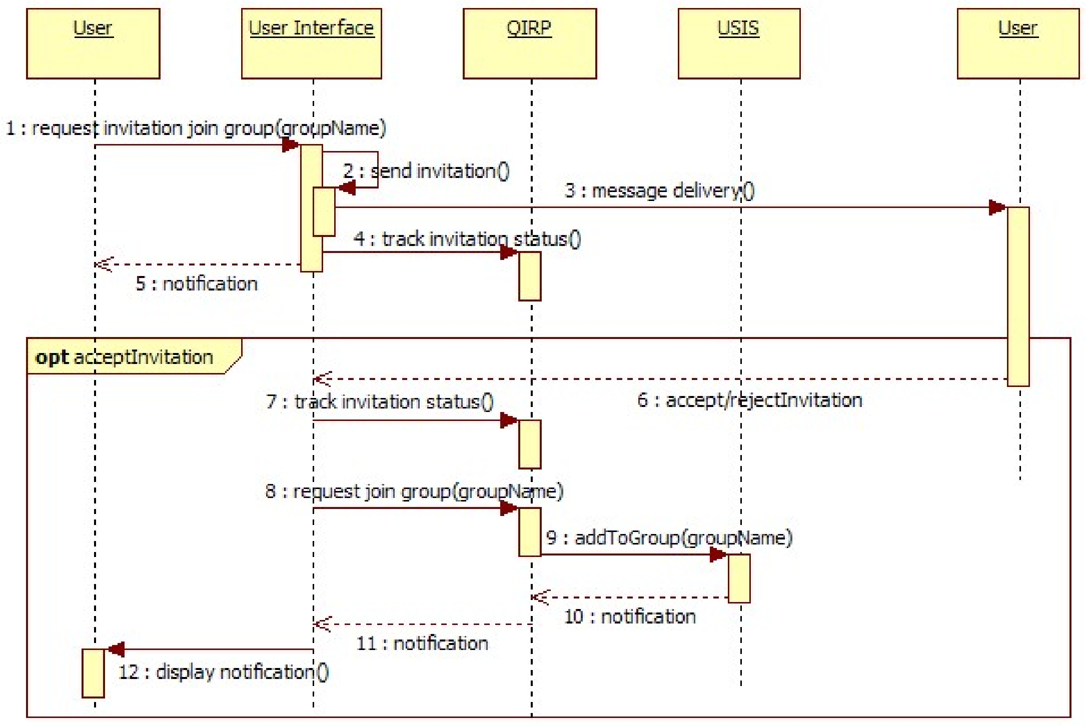
DESCRIPTION	Step	Action
	1	User decides to invite another PHAROS user to join one of his social groups
	2	A message is sent to the invitee, about which the user gets notified
	3	If the invitation is accepted, the invitee is added to the social group and will be visible as a member to the invitee and other persons in that group

3.11.2 Associated sequence diagrams

Sequence diagram: User invites other user to join a social group

Step	Action	Comment
1	User requests to invite another user to join one of his social groups	
2	The user interface triggers an invitation to be sent	
3	A invitation message is sent to the invitee	
4	The status of the invitation is tracked/logged by QIRP, pending an accept or reject by the invitee	
5	Notification is sent to the inviting user, informing him of the message delivery	
6 opt	The invitee responds to the invitation	
7	The accept or reject status is sent to QIRP for update and logging	
8	If the user accepts the invitation, the user interface notify the QIRP to trigger the addToGroup action	
9	The QIRP calls USIS to add the user to the specified group	
10	The USIS sends a notification about the status to QIRP	
11	QIRP notifies the User Interface	
12	The user interface displays a notification to the inviter.	

User invites other user to join a social group


User
User Interface
QIRP
USIS
User
1 : request invitation join group(groupName)
2 : send invitation()
3 : message delivery()
4 : track invitation status()
5 : notification
opt acceptInvitation
6 : accept/rejectInvitation
7 : track invitation status()
8 : request join group(groupName)
9 : addToGroup(groupName)
10 : notification
11 : notification
12 : display notification()

3.12 Use case: Get Recommendations (Pull)

3.12.1 Use case definition

USE CASE *new*	Get Recommendations (Pull)
Goal in Context	System provides recommendation to the user upon request
Scope & Level	PHAROS System
Preconditions	User Logged in
Success End Condition	User receives a set of recommended items
Failed End Condition	No items can be recommended to the user
Primary actors	User
Secondary Actors	N/A
Trigger	Pull: The User asks for recommendations

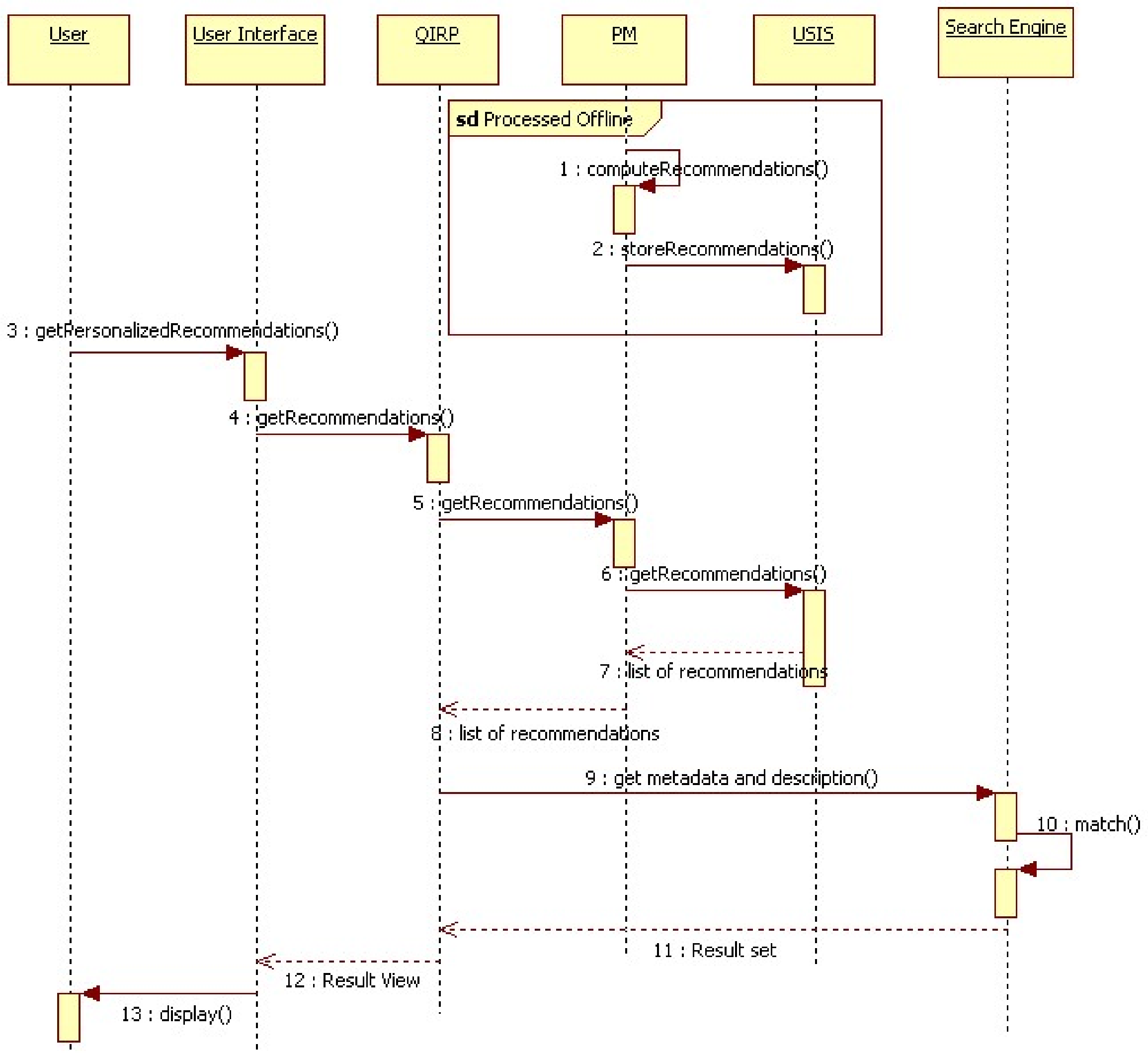
DESCRIPTION	Step	Action
	1	The User selects the option My Recommendations
	2	The System presents a list of personalized recommendations for the active User
	3	If the System cannot compute personalized recommendations, then it presents a list of most frequent or promoted items
	4	The user then can select to play one of recommended items and the corresponding Use Case is executed

3.12.2 Associated sequence diagrams

Sequence Diagram: Get Recommendations (Pull)

Step	Action	Comment
1	PM computes offline a list of recommendations, based on the data existing in USIS	
2	The results of the recommendation computation are stored in the USIS	
3	User request a personalized recommendation	
4	The UI sends the request to QIRP	
5	QIRP sends the request to PM	
6	PM requests the precomputed recommendations from the USIS	
7	USIS sends the recommendations to PM	
8	PM sends the list of recommendations to QIRP	
9	QIRP requests from the search engine the metadata and the descriptions for the items in the list of recommendations	
10	The search engine searches in the own storage the items requested from QIRP	
11	SE sends the result set back to QIRP	
12	QIRP sends the results to the UI to be displayed	
13	UI displays the recommendations to the user	

Get Recommendations (Pull)



3.13 Use case: Receive Personalized Recommendations (Push)

3.13.1 *Use case definition*

USE CASE *new*	Receive Personalized Recommendations (Push)
Goal in Context	System provides personalized recommendation to the user
Scope & Level	PHAROS System
Preconditions	User Logged in
Success End Condition	User receives a set of recommended items
Failed End Condition	No items can be recommended to the user
Primary actors	User
Secondary Actors	N/A
Trigger	Push: The system suggests the recommendations

Push Scenario: The system suggests the recommendations

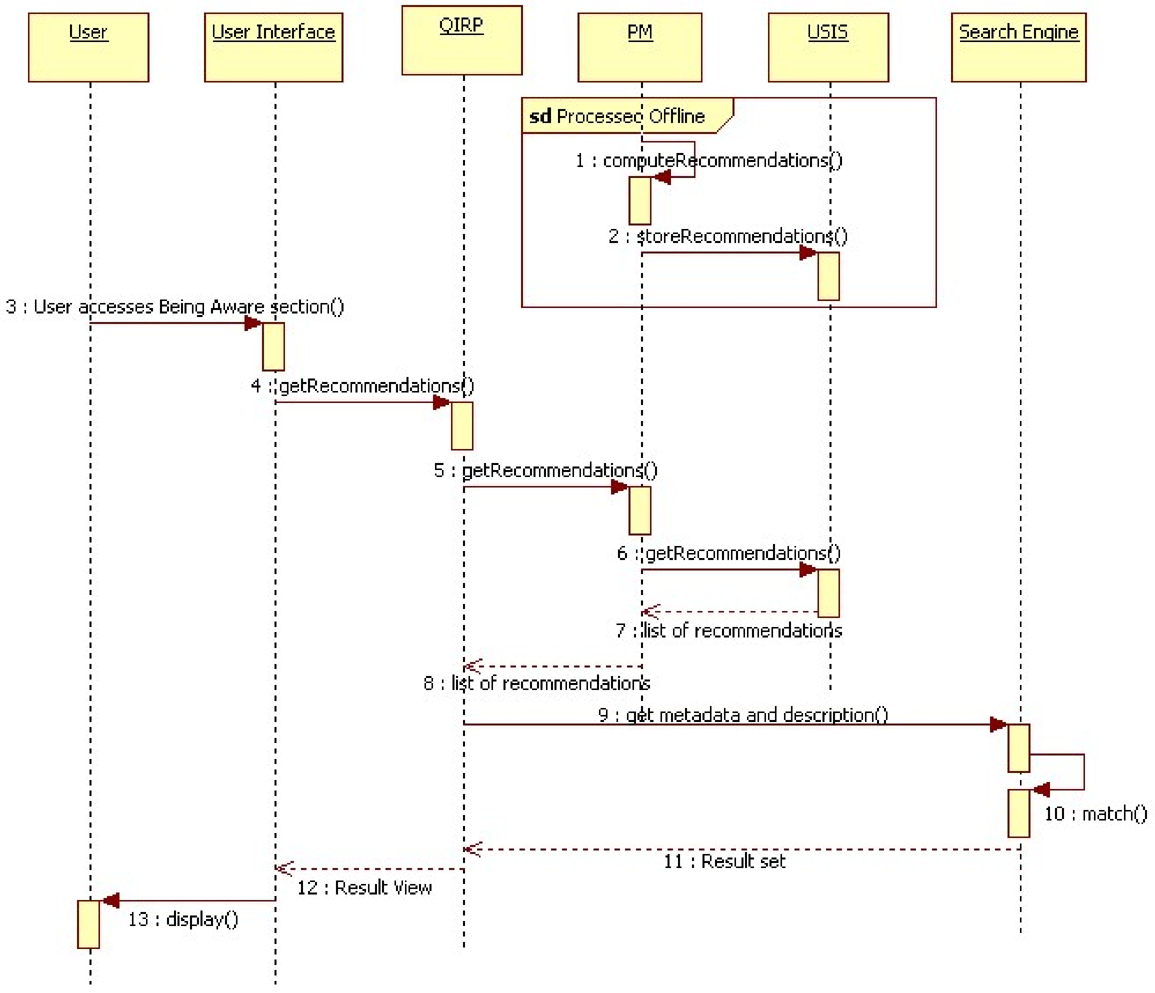
DESCRIPTION	Step	Action
	1	The System computes a list of recommendations for the active user based on the neighborhood computed
	2	The System presents the list of recommendations on the user's active page
	3	The User then can select one of recommended items and the corresponding Use Case is executed

3.13.2 Associated sequence diagrams

Sequence Diagram: Receive Personalized Recommendations (Push)

Step	Action	Comment
1	PM computes offline a list of recommendations, based on the data existing in USIS	
2	The results of the recommendation computation are stored in the USIS	
3	User accesses the ‘Being Aware’ section in his PHAROS page	
4	The UI sends the ‘getRecommendations(userId) request to QIRP	
5	QIRP sends the request to PM	
6	PM requests the precomputed recommendations from the USIS	
7	USIS sends the recommendations to PM	
8	PM sends the list of recommendations to QIRP	
9	QIRP requests from the search engine the metadata and the descriptions for the items in the list of recommendations	
10	The search engine searches in the own storage the items requested from QIRP	
11	SE sends the result set back to QIRP	
12	QIRP sends the results to the UI to be displayed	
13	UI displays the recommendations to the user	

Receive Personalized Recommendations (Push)



3.14 Use case: Explore Neighbourhood

3.14.1 Use case definition

USE CASE *new*	Explore Neighborhood
Goal in Context	System provides recommendation to the user based on the *neighborhood* of similar users computed by the system
Scope & Level	PHAROS System
Preconditions	User Logged in
Success End Condition	User receives a set of recommended items
Failed End Condition	No items can be recommended to the user
Primary actors	User
Secondary Actors	N/A
Trigger	User selects the option *My Neighborhood* (Similar Users) in order to explore it

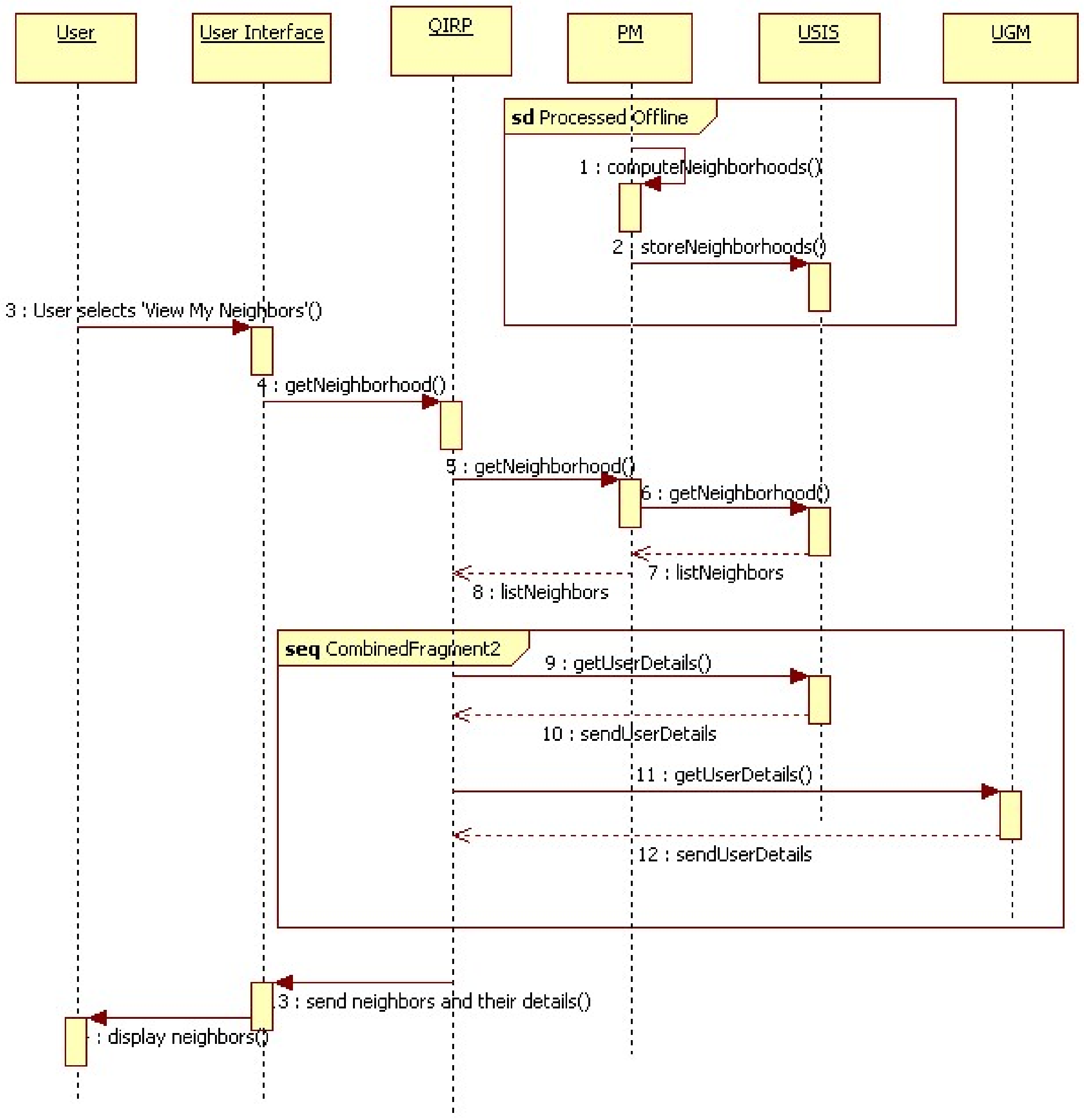
DESCRIPTION	Step	Action
	1	The System (pre)computes the *neighborhood* of the users
	2	The User selects the option *My Neighborhood* (Similar Users) in order to explore it
	3	The System presents a list of users (neighbors) to the active user.
	4	The user then can select one of them to explore his profile and/or to add her as a *friend* (The corresponding Use Case is executed)

3.14.2 Associated sequence diagrams

Sequence Diagram: Explore Neighborhood

Step	Action	Comment
1	PM computed offline the user neighborhoods based on the existing information in USIS	
2	The computed neighborhoods are stored in USIS	
3	User selects ‘View my Neighbors’ section in his PHAROS page	
4	The UI sends the request ‘getNeighborhood(usedId) to QIRP	
5	QIRP sends the request to PM	
6	PM requests the user’s neighbors from the USIS	
7	USIS sends the user’s neighbors to PM	
8	PM sends the list of neighbors to QIRP	
9	QIRP asks for each of the users in the neighborhood the corresponding details from USIS	
10	USIS sends to QIRP the details for each of the users	
11	QIRP asks for each of the users in the neighborhood the corresponding details from UGM	
12	UGM sends to QIRP the details for each of the users	
13	QIRP sends neighbors and all their details to UI	
14	UI displays the neighborhood to the user	

Explore Neighborhood



3.15 Use case: Receive Recommendation of Related Content

3.15.1 Use case definition

USE CASE *new*	Receive Recommendations of Related Content
Goal in Context	System provides recommendation to the user based on related content that she is playing
Scope & Level	PHAROS System
Preconditions	User Logged in
Success End Condition	User receives a set of recommended items
Failed End Condition	No items can be recommended to the user
Primary actors	User
Secondary Actors	N/A
Trigger	User start playing an item

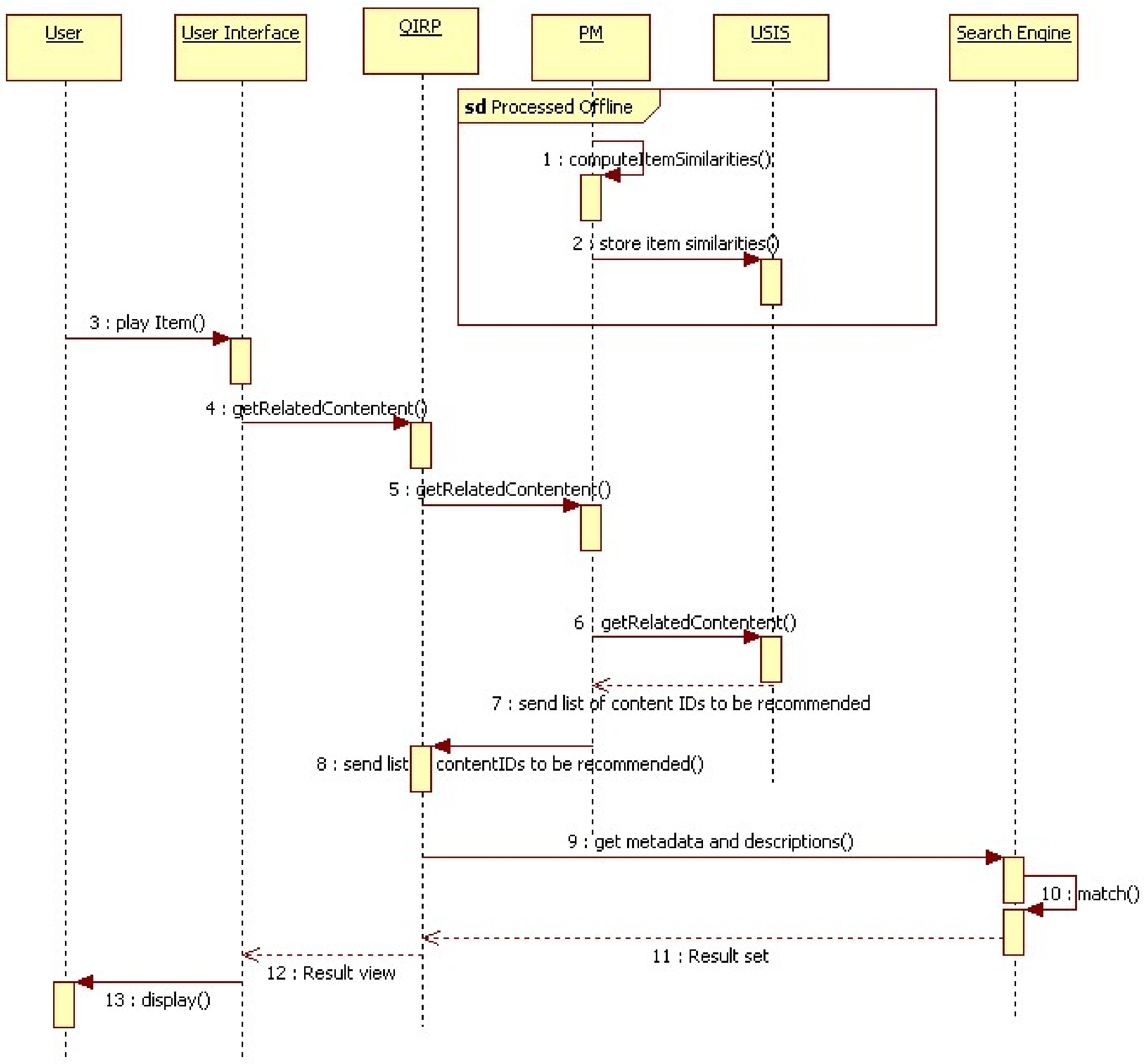
DESCRIPTION	Step	Action
	1	The User start playing an item (e.g., a video)
	2	The System presents a list of personalized recommendations to the User, based on related content to the one she is playing now
	3	If the System cannot compute personalized recommendations, then it presents a list of most frequent or promoted items

3.15.2 Associated sequence diagrams

Sequence Diagram: Recommendation of Related Content

Step	Action	Comment
1	PM computes offline content similarities for items based on the available information from USIS	
2	The computed content similarities are stored in USIS	
3	User plays an item in the PHAROS platform	
4	The UI sends the request ‘getRelatedContent(itemId)’ to QIRP	
5	QIRP sends the request to PM	
6	PM requests from the USIS similar content played by the user’s neighbors	
7	USIS sends the list of items to be recommended to PM	
8	PM sends the list of items to be recommended to QIRP	
9	QIRP asks the search engine for metadata and descriptions for this list of items	
10	SE searches the content storage for the requested items	
11	SE sends the metadata and descriptions for the requested items back to QIRP	
12	QIRP sends the result set to UI to be displayed	
13	UI displays the recommended items to the user	

Recommendation of Related Content


User
User Interface
QIRP
PM
USIS
Search Engine
sd Processed Offline
1 : computeItemSimilarities()
2 : store item similarities()
3 : play Item()
4 : getRelatedContentent()
5 : getRelatedContentent()
6 : getRelatedContentent()
7 : send list of content IDs to be recommended
8 : send list contentIDs to be recommended()
9 : get metadata and descriptions()
10 : match()
11 : Result set
12 : Result view
13 : display()

4. Conclusions & Future Work

In this report, we focused on describing the details of social media modules in PHAROS. As noted before, the content based search service of PHAROS will be served better by an accurate and efficient social based search service. Consequently, social media data analysis and processing plays an important role in the PHAROS platform.

There are currently five social media related modules developed in this project. Three analysis modules, User & Community Profiler (UCP), Social Networks & Blogspace Analysis (SNBA), and Spam Detection, Reputation & Trust (SDRT), aim at performing analytic study on various user-generated social media data and extracting knowledge relevant to users' interests. This extracted knowledge is further exploited by the processing module, Personalization Module (PM), to enhance users' personal search experience. A data storage module, User & Social Information Storage (USIS), is also provided to accommodate not only raw social media data but also processed and extracted information from the raw data. While SDRT was not subject of this deliverable, we described the other four social media modules: USIS, UCP, SNBA and PM. The technical details of each module (described in Chapter 2) include not only the specific algorithms we developed and employed in analyzing and processing social media data, but also the architecture of each module (e.g., internal architecture as well as the interaction with other modules). Moreover, related use cases of each module are also discussed in Chapter 3 to further clarify the functionalities provided by each of the modules.

There are still a few open issues in successful exploring social media data within PHAROS, such as scalability and robustness. Our ongoing work include improving the algorithms described in this report to address these issues, as well as conducting more research and developing work in optimizing the functionalities provided by social media modules by large scale. Moreover, the feedback obtained during the Showcase evaluation, which will include also the evaluation of the Social Media Beta release will provide more insights into possible extensions and refinements of these modules. The related efforts will be reported in next version of this report, deliverable D2.1.3 due in month 30.

5. Bibliography

Amazon. (n.d.). Retrieved from http://www.amazon.com

Audioscrobbler. (n.d.). Retrieved from http://www.audioscrobbler.net/data/webservices/

Bao, S., Xue, G.-R., Wu, X., Yu, Y., Fei, B., & Su, Z. (2007). Optimizing web search using social annotations. *WWW.*

Bischoff, K., Firan, C. S., Nejdl, W., & Paiu, R. (2008). Can All Tags Be Used for Search? *Submitted to CIKM 2008.*

Boullé, M. (2007). Compression-Based Averaging of Selective Naive Bayes Classifiers. *Journal of Machine Learning Research 8 , 8*, 1659-1685.

Boulle, M. (2004). Khiops: A Statistical Discretization Method of Continuous Attributes. *Mach. Learn.*, 53-69.

Chirita, P. A., Nejdl, W., Paiu, R., & Kohlschütter, C. (2005). Using ODP Metadata to Personalize Search. *Proceedings of the 28th ACM International SIGIR Conference on Research and Development in Information Retrieval.* Salvador.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement , 20* (1), 37-46.

Delicious. (n.d.). Retrieved from http://del.icio.us/

Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems* , 1-34.

Dmitriev, P. A., Eiron, N., Fontoura, M., & Shekita, E. J. (2006). Using annotations in enterprise search. *WWW.*

Firan, C., Nejdl, W., & Paiu, R. (2007). The Benefit of Using Tag-Based Profiles. *LA-WEB '07: Proceedings of the 2007 Latin American Web Conference*, (pp. 32-41). Santiago.

Flickr. (n.d.). Retrieved from http://www.flickr.com/

Flixster. (n.d.). Retrieved from http://www.flixster.com/

Friendster. (n.d.). Retrieved from http://www.friendster.com/

Golder, S. A., & Huberman, B. A. (2006). Usage patterns of collaborative tagging systems. *Journal of Information Science , 32* (2), 198-208.

Haveliwala, T. H. (2002). Topic-sensitive PageRank. *Proceedings of the Eleventh International World Wide Web Conference.* Honolulu.

Herlocker, J. L., Konstan, J. A., & Riedl, J. (2002). An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. *Information Retrieval*, (pp. 287-310).

Kervajan, L., Neef, E. G., & Véronis, J. (2006). *French Sign Language Processing: Verb Agreement.* Springer Berlin / Heidelberg.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, (pp. 77-87).

Last.fm. (n.d.). Retrieved from http://www.last.fm/

Marlow, C., Naaman, M., Boyd, D., & Davis, M. (2006). Ht06, tagging paper, taxonomy, flickr, academic article, to read. *Hypertext 2006.*

MySpace. (n.d.). Retrieved from http://www.myspace.com/

MyStrands. (n.d.). Retrieved from http://www.mystrands.com/

Naaman, M., Harada, S., Wang, Q., Garcia-Molina, H., & Paepcke, A. (2004). Context data in geo-referenced digital photo collections. *Multimedia.* New York, NY, USA: ACM.

Netflix. (n.d.). Retrieved from http://www.netflix.com

Ortega-Binderberger, M., & Mehrotra, S. Relevance Feedback in Multimedia Databases. In *Handbook of Video Databases: Design and Applications* (p. 2003). CRC Press.

Pass, G., Chowdhury, A., & Torgeson, C. (2006). A picture of search. *Infoscale.*

Poirier, D., Bothorel, C., & Boullé, M. (2008). Two possible approaches for opinion analysis in film reviews: statistic and linguistic. *Sentiment Analysis: Emotion, Metaphor, Ontology and Terminology (EMOT-08), Workshop held at LREC 2008.* Marrakech, Morocco.

Poirier, D., Bothorel, C., Boullé, M., & Ferrandiz, S. (2008). Exploring film reviews with a non-parametric co-clustering method. *Submitted to ADKDD.*

Qiu, Y., & Frei, H. (1993). Concept Based Query Expansion. *Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval.* Pittsburgh: ACM Press.

Radev, D. R. (2000). A common theory of information fusion from multiple text sources step one: cross-document structure. *Proceedings of the 1st SIGdial workshop on Discourse and dialogue* , pp. 74-83.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *Proceedings of Conference on Computer Supported Coorperative Work* , (pp. 175-186).

Rocchio, J. J. (1971). *Relevance feedback in information retrieval.* Prentice-Hall.

University of Sheffield - Natural Language Processing Research Group. (n.d.). *Gate - General Architecture for Text Engineering*. Retrieved from http://gate.ac.uk/

WordNet. (n.d.). Retrieved from http://sourceforge.net/projects/jwordnet

Zhai, C., Velivelli, A., & Yu, B. (2004). A Cross-Collection Mixture Model for Comparative Text Mining. *Proceedings of KDD* .

Zollers, A. (2007). Emerging motivations for tagging: Expression, performance, and activism. *WWW Workshop on Tagging and Metadata for Social Information Organization.*