Design and Anatomy of a Social Web Filtering Service

Michael G. Noll
Hasso-Plattner-Institut für
Softwaresystemtechnik,
University of Potsdam
14440 Potsdam, Germany
michael.noll@hpi.uni-potsdam.de

Christoph Meinel
Hasso-Plattner-Institut für
Softwaresystemtechnik,
University of Potsdam
14440 Potsdam, Germany
meinel@hpi.uni-potsdam.de

ABSTRACT

The greatest use of the Internet and new online technologies today is for constructive purposes. However, the use of the same technologies to spread illegal and objectionable content has been increasing dramatically during the last years. Internet users have begun to protect themselves and their wards by using so-called web content filters, which allow access to legitimate content and block access to objectionable and unwanted content. In this paper, we describe the design and anatomy of an open architecture for creating a social filtering service and show how it can be implemented. The system is designed to overcome the deficiencies of today's content rating and filtering systems and frameworks by empowering and involving end users, and to actively support user collaboration and cooperation by using techniques and methodologies with a low cost of participation combined with ease of use. The proposed work improves the quality of information shared by existing collaborative filtering systems by integrating tagging and folksonomy techniques, which we have extended in such a way that relations between document and rating metadata can be better expressed with only a minimal additional amount of required information. Data structures and workflows are optimized for scalability and fast service response times and allow the efficient computation and processing of even very large volumes of data. We show how such a service can be used by designing and implementing two client applications for filtering pornographic web content.

Keywords

World Wide Web, Web Filtering, Content Rating, Tagging, Collaboration, Social Filtering

Forth International Conference on Cooperative Internet Computing (CIC)
2006 Hong Kong

1. INTRODUCTION

A recent web server survey[1] counted more than 88 million active websites and an average increase of 2 million sites per month for 2006. The greatest use of the Internet and new online technologies today is for constructive purposes; however, the use of the same technologies to spread illegal and objectionable content has been increasing dramatically during the last years. Internet users have begun to protect themselves and their wards by using so-called web content filters, which allow access to legitimate content and block access to objectionable and unwanted content. For example, parents can use filtering software such as NetNanny or services such as AOL Parental Controls to safeguard their children from harmful websites. In a related context, the problem of spam emails has received increased public attention, and appropriate tools for filtering spam emails have been steadily integrated into the Internet's communication infrastructures and end-user applications. Compared to the fight against spam, the area of web content filtering is still in its infancy. To improve this situation, various international, governmental and public initiatives have started campaigns such as the European Commission's Safer Internet programme to increase the public awareness of objectionable Internet content and support the development of technologies and frameworks to tackle harmful material on the World Wide Web.

1.1 Rating by content providers

There exist rating systems for Internet content such as ICRA[2], which are similar to rating systems such as MPAA[3] for movies or ESRB[4] for computer software and games. The main idea of these rating systems is that human users add metadata to content. Most of the existing Internet content rating systems are legally voluntary and focus on the creators and providers of the content. Content providers can use these rating systems to manually classify their content with a common description framework and add the rating information in the form of digital labels to their websites. Internet users can then use special filtering software to allow or disallow access to websites based on this meta information.

Rating systems for Internet content sound promising on paper. Obviously, the availability of such content labels could make the filtering task per se rather trivial and theoretically more reliable than heuristic methods for content clas-

[1] Netcraft July 2006 Survey, http://www.netcraft.com/

[2] Internet Content Rating Association, http://www.icra.org/

[3] Motion Picture Association of America, http://www.mpaa.org/

[4] Entertainment Software Rating Board, http://www.esrb.org/

sification. However, the viability and success of any kind of content rating systems depend heavily on the actual usage of these systems and the accuracy and trustworthiness of rating information. We have shown in previous studies[13] that the usage of existing, provider-focused content rating systems in the Internet today is at best marginal and, at the example of using these digital content labels to filter pornographic websites, that the resulting classification performance of rating-dependent content filters is inadequate and their application not recommended in practice. In addition, provider-focused rating systems have a basic problem: differences in interpretation of depicted content results in different labels. We have found out that human users often disagree on how a web document "should" be rated because of differences in subjective interpretation; factors such as cultural, familial, educational and religious background has a high impact on how users perceive and rate content. The result is that labeler and viewer of a web document often disagree on rating information (see also [8]). The conflict of interest between provider and viewer of content aggravates this problem.

1.2 Rating by third parties

An alternative to content rating systems as described above are services offered by third parties. Internet service providers (ISPs) include filtering services such as AOL parental controls, and Internet security companies offer commercial whitelists and blacklists of websites to their customers. If detailed information of how these services are implemented is publicly available, they often use (though not always exclusively) manual classification approaches and dedicated teams of human experts for rating and categorizing web pages. However, this kind of manual processing will hardly scale with rapid growth of the Internet, the increasing number of new websites and changes to existing ones. Another drawback is that filtering is often not selective: instead of rating web content on a document basis, ratings are assigned to complete domains. A bad apple will often ruin the whole basket.

Rating by third parties has also similar problems as rating by content providers: differences in interpretation of depicted content in web documents will lead to different ratings and thus filtering decisions as the persons rating the content are different from the viewers (end users). Another issue and often highly debated point is that both rating by content providers and rating by third parties inherently do not focus on end users; rating and filtering "on behalf of the user" by third parties such as ISPs, governments or other institutions is often perceived as censorship and paternalism by end users. Very often, end users are not involved in the filtering process and complain about lack of feedback and transparency.

1.3 Rating and filtering by algorithms

Research and development in the area of web page classification has yielded several interesting results and advances in the last years. A multitude of potential algorithms are available for web filtering tasks, starting with simple keyword filtering (blocking a web page if it contains certain words) up to machine learning algorithms like SVMs or neural networks. However, rating and filtering Internet content by algorithms faces particular challenges. Firstly, it is not always easy to extract information from web documents because they may contain lots of different content types such as images, videos, Java applets, or Flash. While it is very easy for a human to analyze such content, it is a much harder task for algorithms even with modern processing power. For example, image processing algorithms may be able to identify human faces or nudity in images up to a certain reliability, but such techniques are often restricted to very specific problem domains. Secondly, results of machine learning algorithms depend heavily on quantity and quality of training input, and training input varies with a user's individual preferences and characteristics. An algorithm for binary classification will not yield optimal results if it is not trained with a sufficient number of samples of both classes, even though tricks such as PEBL[21] may help.

2. SOCIAL SERVICE FOR COLLABORATIVE RATING AND FILTERING

We have designed and implemented a social web filtering service for creating a true democracy on the web with regard to content filtering. It puts the focus on end users - the actual recipients of web content - and empowers them to decide what they do and do not want to see. On one hand, the proposed system shall improve the quality of information shared by existing collaborative filtering systems by integrating tagging and folksonomy techniques, which we have extended in such a way that relations between document and metadata can be better expressed. On the other hand, the system has been built to actively support user collaboration and cooperation combined with a low entry barrier and ease of use. The cost of participation is low, which gives an incentive to start and continue using the service - an important factor for a system whose benefit increases with number of users. The processes of joining the service, rating a web document and retrieving rating information has therefore been designed to be easy and fast for both users and technical infrastructure. We have optimized data structures and workflow for read and write access in constant time ($O(1)$) where possible, and the design allows the efficient computation and processing of large volumes of data with techniques such as MapReduce[7]. To the best of our knowledge, the work described in this paper is the first open architecture for such a system (see also section 7).

Our approach leverages the most flexible and powerful content processor available, the human brain, and is able to connect a multitude of participants via a collaborative networking. The first aspect allows to rate and filter non-trivial content such as images and videos on web documents[5] and overcomes limitations of automated algorithmic filters, the second aspect allows to scale much better with Internet growth and change. Focussing the system on end users also addresses the conflict of different interpretations of content - rater and viewer of a web document are the same person. The way rating metadata can be expressed allows to include and reflect contextual information.

Creating a social rating and filtering service has benefits for other areas, and we have wanted to support web research by

[5] An interesting side-effect is that in contrast to automated web spiders and filtering algorithms, human users can access restricted websites where authentication is required to access content. This increases the base data available for rating.

building such an open service. For example, collected data can be used as training input for automated machine learning algorithms or to build classified data sets and benchmarks for evaluation of classification algorithms with real-world data.

We have developed a working prototype of the system, which is being tested by internal user groups, and we are currently building a version that will be made accessible externally on the Web.

3. SYSTEM ANATOMY

In this section, we will give a high level overview of how the whole system works. Further sections will discuss the applications and data structures not mentioned in this section.

The social filtering system is implemented in Perl, and can run on a variety of platforms and operating systems. It provides three main interfaces to client applications:

- UID interface - for creating a random, unique identification number
- rating interface - for accepting rating information (metadata) about a document
- lookup interface - for querying rating information (metadata) of a document

The service assigns new clients a unique and random identification number (UID) when they try to access the service for the first time. Together with the UID, clients receive a shared secret which is used for authentication and authorization of all communication with the service. Known clients can submit ratings of web documents to the social filtering service. Whenever a client rates a web document, the rating information is stored in an individual database for each client and aggregated with other clients' ratings into a combined community rating for this web document. In addition to submitting ratings, known clients can query for rating information of a web document. The social filtering service will return up to three different types of rating information in this case: firstly, the client's own rating of the web document (if the client has rated the web document before); secondly, the community's rating; lastly, a "system" rating for the web document. The system rating information is special: it's metadata provided by the service operator and used mainly to prevent abuse and increase reliability of the service[6]. Of course, clients can opt to neglect each of the returned ratings, which implies that system ratings don't need to be honored in case of censorship concerns or other objections. After the client has received the rating information about a web document, it can decide whether to allow or block access to it (other actions are also possible).

The client applications we have implemented use a simple and intuitive order of preference: *client ratings > system ratings > community ratings*. A client's own ratings take preference over anything else. Since system ratings are used to prevent abuse, they should get higher priority than community ratings.

3.1 Client identification (UID)

All clients using the social filtering service require a unique identification number, the so-called UID. The UID is also randomized for security and privacy reasons. Whenever a new client accesses the service for the first time, it is assigned a UID and a shared secret, which is known only to the client and the service. The tuple of UID and shared secret is used for authentication and authorization of all communication between client and service. The identification number is also needed so that clients can retrieve their individually submitted information from the service.

3.1.1 User identification

By purpose, the social filtering service uses identification numbers for clients, not human users. Only in cases where human users use a single client application for accessing the service do client UIDs also identify users. There are two reasons for this design decision. Firstly, the service should not be restricted to human users but also allow access for computer programs and other services if needed. Secondly, the approach to use UIDs for identifying client applications instead of users is more generic: the ability to map multiple client UIDs to a single user[7] or multiple users to a single client UID[8] gives more opportunities to make use of the service. If user identification is a needed feature, it is very easy to implement an additional software layer which maps client UIDs to human users.

3.1.2 Example

Here is an example of a UID and shared secret as generated by our service implementation.

- UID: `A688C654-0C18-11DB-A342-7A1C118AA5B2`
- secret: `NnorMX4huH0`

3.2 Rating

3.2.1 Tagging

The recent emergence and success of tagging with services such as del.icio.us[9] or Flickr[10] has shown the great potential of this simple approach to add metadata to objects. Unlike traditional classification or categorization systems, the process of tagging is nothing more than annotating objects with a flat, unstructured list of keywords (tags). Users can browse or query objects by tags, and so-called tag clouds provide a rudimentary but often sufficient way to find popular and interesting content. Several studies have analyzed

[6] For example, the system rating can be used to "whitelist" websites such as ACM.org or Google.com or the service operator's own website. Or, it can be used to integrate information from web directories such as DMOZ to bootstrap the service on first operation when there is only a small number of active users.

[7] For example, a user with a work laptop and home computer.

[8] For example, a university using a single web proxy for internet access, which is configured to query the social filtering service for rating information about requested web documents. Section 4.2 shows how a web proxy can be configured to use the social filtering service.

[9] http://del.icio.us/

[10] http://www.flickr.com/

why tagging is so popular and successful in practice; a common argument is that tagging works because it strikes a balance between the individual and the community; the cost of participation, in particular entering data, is low for the individual, and tagging an objects benefits both the individual and the community.

Basically, tagging can be interpreted as a relation

$$R_{tagging} \subseteq D \times U \times T$$

where D is the set of documents, U the set of users and T the set of tags.

In the context of web content filtering services, frameworks such as ICRA for describing content depicted on web documents can be as overwhelming and difficult to understand for end users as hierarchical classification or sophisticated ontologies when compared with the simplistic but often very effective tagging approach. In particular, users may use their own vocabulary and define how they want to label content on web documents. The cost of the tagging progress is reportedly very low for a user, which - together with the benefits - gives an incentive to actively provide input and use tagging, all to the advantage of the community. We therefore consider the characteristics of tagging and folksonomies as very useful for supporting collaboration and cooperation.

3.2.2 From tagging to rating

We have decided to make rating a web document for filtering services similar to tagging and leverage its benefits. However, current tagging systems lack a convenient way to express non-relation of tags, or in other words the possibility to express negations. We have found out that it is often very useful to explicitly express an IS NOT relation, for example “this web page is not pornographic” or “this web page is not suitable for children”. Current tagging systems focus on IS relations of tags and do not allow to express logical negations. We have observed from test users that it is often easier and quicker to use negations: “I am unsure what it is, but I know for sure what it is not.”[11]

In addition, extending tagging systems in such a way allows to better include the context of a web document. Content rating systems such as ICRA include veto-type descriptors for this very reason, which for example can be used to denote that a web document shows pictures of a naked woman but in a medical (i.e. non-pornographic) context.

When clients rate a web document, they attach one or more tags to it and denote whether the document is or is not representative for a tag. We call the latter a “tag vote”, or “vote”. By voting, clients can provide information about IS and IS NOT relation of web document and tag. This at first glance subtle difference has in fact a large impact on the possibilities of the system with benefits for users as well as collaborative filtering systems. It is particularly interesting for problems where users might like to use negations for expressing what they (do not) want, e.g. exclude certain recommendations, and it allows logical combinations such as “document is about art photography but without nudity”, which is very different from “document has art photography tag but omitted nudity tag”. The former statement explicitly includes information that relevant documents do not contain nudity while the latter simply comprises documents where explicit information is missing that it does contain nudity[12].

Rating is thus an extension of tagging and can be described as a relation

$$\begin{aligned} R_{rating} &\subseteq D \times U \times T \times V \\ R_{rating} &= \{(d,u,t,v) \mid u \text{ rated } d \text{ with tag } t \text{ and tag vote } v\} \end{aligned}$$

where D is the set of documents, U the set of UIDs, T the set of tags and V the set of tag votes[13]. In our implementation, $V = \{0, 1\}$ and we use a value of 1 for denoting IS relation and 0 for IS NOT[13]:

$$vote(d,t) = \begin{cases} 1, & \text{if document } d \text{ is representative for tag } t \\ 0, & \text{else} \end{cases}$$

The medical example above could be easily expressed by rating the web document with tag/vote pairs `(nudity, 1)`, `(surgery, 1)`, `(porn, 0)` or even `(porn, 0)` if we just want to distinguish between pornographic and non-pornographic web documents.

3.2.3 Rating protocol

In our implementation of the service, client applications can chose from multiple technical interfaces and protocols to submit ratings. Currently, the system provides an HTTP/POST and an XML-RPC interface.

The following parameters are used:

- URL - address of the document to be rated in URL-safe Base64 format; a document is identified by its URL
- UID
- client rating - list of (tag, vote) pairs
- authentication string - HMAC of relevant protocol parameters
- protocol version - version of the rating protocol used by the client
- client version (optional) - name of the client software

[11] For example, a user might not be able to decide whether Helmut Newton’s photography of a naked woman is art or is not art while it might be clear to her that it is not pornographic.

[12] We have a similar problem as Schroedinger with his cat; tagging would leave Schroedinger in ignorance while rating would give him certainty.

[13] Again, one can see that tagging is a subset of rating, as it can be interpreted as $R^{*}_{tagging} \subseteq D \times U \times T \times \{1\}$.

- referrer (optional) - URL of the referring web document in URL-safe Base64 format

From the above listed parameters, some are used only for the technical implementation: protocol version, client version and, in particular, authentication string. The protocol version is used to identify how clients want to communicate with the service; it is also helpful to identify outdated client applications. The client version is used only for statistical purposes. The authentication string is discussed in section 3.4.1. The remaining parameters are needed for the actual rating functionality of the system as described above.

3.2.4 Rating example

Here is an exemplary rating submission using the HTTP/POST interface, written in GET notation for better readability:

```
...?uid=26AD3620-FF95-11DA-B006-9FC40806B13E
&url=aHR0cDovL3d3dy5ocGktd2ViLmRlL2luZGV4Lmh0bQ
&tag=porn
&vote=0
&auth=VQyMIinY8lMdi8uR91xLEQ
&protocol=1.0
&client=firefox-extension-1.0
```

This submission rates the home page of the HPI, http://www.hpi-web.de/index.htm, as non-pornographic by attaching the tag **porn** with a vote of zero.

3.2.5 Individual client rating databases

When a client requests rating information of a web document, a so-called lookup request, it is very useful to have quick read access to a client's individual ratings. Instead of having a unified database of ratings of all clients, our system creates an individual rating databases for each client referenced by its UID. Client rating databases are implemented as hash tables, which map URL to rating information.

$$\begin{aligned} UID_a.\text{db} \quad \Rightarrow \quad & d_i \rightarrow \{(t_{i_1}, v_{i_1}), ..., (t_{i_m}, v_{i_m})\}, referrer_i, ... \\ & d_j \rightarrow \{(t_{j_1}, v_{j_1}), ..., (t_{j_n}, v_{j_n})\}, referrer_j, ... \\ & \qquad\qquad ... \end{aligned}$$

Even though this may sound surprising at first, this approach significantly reduces system and database load and number of queries needed to retrieve rating information. Access time is constant: we can find a specific client rating database in $O(1)$ because we can compose the filename of a client's rating database by simply mapping the UID (supplied by the rating request) to a local file system path[14], and we can search for an URL in a hash table in $O(1)$. Client rating lookups are mostly limited by the system's I/O capabilities and file system performance only. If needed, caching techniques and tools such as `memcached`[15] help to soften potential performance bottlenecks. Another advantage of this approach is that it helps to recover from problems such as hardware failures or data(base) inconsistencies. Problems with one client's database does not affect any other service data.

Write access to client rating databases is very fast for similar reasons. The average write access rate of rating submissions can be expected to be lower than the average read access rate of lookup requests. Note that in contrast to the community rating database, client rating databases are updated in real-time so that clients have instant feedback on their input[16].

3.2.6 URL normalization

A web document is identified by its URL. URL normalization is the process by which URLs are modified and standardized in a consistent manner. The goal of the normalization process is to transform a "raw" URL into a normalized URL so it is possible to determine if two syntactically different URLs are equivalent (see [11] for more information). It is an important technique to avoid duplicate data entries. Normalization involves transformations such as converting scheme and host name of a URL to lower case or removing the fragment part. There are safe transformations such as removing default port specifications, and unsafe transformations such as removing the query part, which change a URL in such a way that it may or may not point to different content after the transformation[17].

When designing a social filtering service, it is necessary to decide if and how URL normalization is implemented because it is a pre-processing step for the aggregation of individual client ratings into community ratings. Normalization also has an impact on filtering performance with regard to false positives, i.e. incorrect aggregation of URLs, and false negatives, i.e. not aggregating URLs even though they point to the same content; if two URLs are normalized to a single entry, rating information of both URLs is merged, too. Generally, by using "aggressive" normalization techniques which tend to aggregate URLs, we can increase the average amount of rating information per web document but also increase the number of of false positives caused by over-aggregation; by using "defensive" techniques which tend not to aggregate URLs, we decrease average rating information and false positive rate but also increase the number of false negatives[18]. URL normalization can therefore be used to increase and decrease average rating information per web document[19] and is thus an important parameter of a social filtering service.

[14] For example, /pathprefix/26AD3620-FF95-11DA-B006-9FC40806B13E.db on Unix file systems.

[15] http://www.danga.com/memcached/

[16] This was a very important point for our test users. They were more likely to submit rating information when there was instant feedback on their actions.

[17] Some normalization techniques strip the query part of a URL because it often contains information which is not relevant for other users, for example session IDs or parameters for highlighting matching words on result pages of search engines. This information leads to a syntactical difference between two URLs even though they point to the same content.

[18] As a rule of thumb, a normalization technique is more aggressive when it uses more unsafe transformations to achieve a higher aggregation of URLs at the price of aggregation accuracy.

[19] Filtering services can also group or cluster clients based on common document ratings for calculating group ratings, which might better reflect a group member's preferences than a global community rating. In such a case, URL normalization has an additional impact because aggregation

Finding a balance between over- and under-aggregation is hard in practice and often has to be adjusted based on field experience. We are currently evaluating the effects of different normalization techniques on our implementation. For now, we are using a semi-aggressive normalization strategy and have chosen only safe transformations with the exception of query part removal. This strategy is a compromise between coarseness and fine granularity and is very accurate for static web pages and otherwise "clean" URLs. Dynamic web pages with URLs containing query parts might suffer from over-aggregation depending on the internal structure of the website.

When a client submits or queries rating information of a URL from the social filtering service, it passes the full, unnormalized URL with the request; only the service itself will perform normalization in subsequent steps. Even though avoiding client-side normalization might result in unnecessary communication with the service, it has the benefit that normalization techniques can be changed and adjusted during operation without modifications to clients.

3.2.7 From client ratings to community ratings

Community ratings are an aggregation of client ratings of a particular web document. The social filtering system periodically combines individual client ratings of a web document into an single community rating. In contrast to the client rating databases, the community rating database is not updated in real-time but in predefined time intervals for performance reasons. The final database used for answering lookup requests is implemented as a hash table, which maps URL to community rating information and answers searches for a specific URL in constant time, i.e. $O(1)$. Techniques such as MapReduce[7] help to compute community ratings for large volumes of data.

There are several options for calculating community ratings, for example thresholding approaches based on number of ratings ("last n ratings of a web document") or time ("ratings of a web document within the last m days"), or weighting approaches which prioritize trusted clients or clients with high reputation. There is no free lunch when choosing the "right" approach; it depends on the context of the system such as characteristics of documents to be rated, or number and type of clients/users, some of which are variables hard to estimate in advance. Time based approaches might be better suited for highly volatile web documents with rapidly changing type of topic and content such as news portals while expert networks might favor weighting schemes based on user reputation, and sparse problem domains where the expected average number of ratings per document is very low[20] might favor not to discard any ratings. There is also the possibility to infer recommended aggregation schemes from documents and their ratings, thus picking the "right" scheme on a per-document basis[21].

In addition, grouping of clients or documents are useful for situations where finer granularity than a global community rating is wanted. Algorithms for grouping, clustering or horting objects in collaborative filtering systems based on similarity measures are described and analyzed in [1, 15, 4].

Our service implementation computes the community rating of a document d as follows.

1. Create the set of all tags attached to document d.
2. Compute the average vote for each tag in the set.

The community rating of a web document is thus the set of all its tags with associated averaged votes. Community rating can be described as a relation

$$CR \subseteq D \times T \times V^*$$

where D is the set of documents, T the set of tags and V^* the set of community tag votes. In our implementation, $V^* = [0, 1]$. Representing rating data in such a way has the added benefit that efficient computation with techniques such as MapReduce[7] is possible; computation of large volumes of data can be split into smaller, manageable tasks and distributed to multiple computer machines for autonomous processing.

Here is a result of an exemplary community rating calculation for a document d based on information provided by three clients c_1, c_2 and c_3.

$$\begin{array}{ccc} (d, c_1, \mathrm{porn}, 0) & \Rightarrow & (d, \mathrm{medical}, 1.000) \\ (d, c_1, \mathrm{medical}, 1) & & (d, \mathrm{porn}, 0.333) \\ (d, c_2, \mathrm{porn}, 0) & & \\ (d, c_3, \mathrm{porn}, 1) & & \end{array}$$

There are parallels between our rating aggregation approach and Bayesian filters. Tags are similar to tokens and votes are similar to class information of documents. In our case however, votes are associated with tags not documents, i.e. a client can submit different votes for a document's tags. For example, Bayesian spam filters consider every token in a trained spam email as an indication for spam (only IS relation like tagging), while rating allows more granularity by using tag and vote pairs (IS and IS NOT relation). Another difference to traditional content filters is that tags are not extracted from a document itself but they are metadata supplied by collaborative clients.

3.3 Lookup

Clients can retrieve rating information of a URL from the social filtering service by sending a lookup request. Lookups

of URLs and associated rating information also aggregates clients.

[20] For example, where the number of documents to be rated largely exceeds the number of clients/users.

[21] For example, we have experimented with identifying highly volatile websites. Such websites are often characterized by a shift in rating information at a certain point in time. A very good indication is a high number of users who "change their mind" about a web document which they already rated before. Note that there is a difference between change rate of content and change rate of topic - a daily news portal about digital cameras might have a low topic volatility while a weekly personal weblog might have a high topic volatility.

are very similar to rating submissions except that clients don't include tag and vote information, which are returned by the service instead. The service divides returned rating information into three categories, or dimensions: *client*, *community*, and *system*. The client category contains the rating information of a URL as submitted by the client itself, the community category contains the aggregated community rating information, and the system category contains rating information as defined by the service operators, which is mainly used to prevent abuse and increase reliability of the service.

It is then up to the client to decide how it wants to make use of the rating information. The client applications we have implemented make filtering decisions based on an intuitive priority definition: client ratings > system ratings > community ratings. When the vote of a tag is greater than a predefined threshold, the rated document is considered to be representative for a tag. Our clients use a majority voting strategy, i.e. a document is considered representative for a tag if the tag vote is greater than 0.5.

As described in previous sections 3.2.5 and 3.2.7, data structures are optimized for fast read access and allow lookups in constant time, i.e. $O(1)$. The time from sending a lookup request to the reception of the service response is mostly limited by two factors only, network performance and I/O performance. The client applications we have implemented also use caching strategies to further increase responsiveness and reduce system load.

3.3.1 Lookup protocol

The client uses the following parameters for lookup requests:

- URL - address of the document to be rated in URL-safe Base64 format; a document is identified by its URL
- UID
- authentication string - HMAC of relevant protocol parameters
- protocol version - version of the rating protocol used by the client
- client version (optional) - name of the client software
- referrer (optional) - URL of the referring web document in URL-safe Base64 format

From the above listed parameters, some are used only for the technical implementation: protocol version, client version and, in particular, authentication string. The remaining parameters are needed for the actual lookup functionality of the system as described above.

The server responds with the document's rating information, divided into three categories:

- client rating - list of (tag, vote) pairs
- community rating - list of (tag, vote) pairs
- system rating - list of (tag, vote) pairs

3.3.2 Lookup example

Here is an exemplary lookup request using the HTTP/POST interface, written in GET notation for better readability:

```
...?uid=26AD3620-FF95-11DA-B006-9FC40806B13E
&url=aHR0cDovL3d3dy5ocGktd2ViLmRlL2luZGV4Lmh0bQ
&auth=MMoZb4zmhkzPIM0ympQqHA
&protocol=1.0
&client=firefox-extension-1.0
```

This lookup requests asks for rating information of the HPI's home page, http://www.hpi-web.de/index.htm.

Following up on our example from section 3.2.7, client c_3 would receive the following rating information from the service when looking up document d

- client: `(porn,1)`
- community: `(medical,1.000),(porn,0.333)`
- system: n/a

If client c_3 was configured to filter pornographic web documents, it would block access to document d because - in our implementation - client rating information would take precedence over both system and community rating information.

3.4 Security and privacy

3.4.1 Authentication and authorization

Any communication of client and service requires two pieces of information: a valid UID and a shared secret, which is known only to the client and the service. This tuple is used for authentication and authorization[22]. Whenever a client sends a message to the service, it has to include its UID and authenticate itself by calculating an keyed-hash message authentication code (HMAC) of the message parameters with its shared secret. The social filtering services verifies the validity of a rating or lookup request by re-calculating the HMAC and comparing it to the authentication parameter supplied by the client. Invalid rating submissions are discarded. The HMAC ensures that only legitimate clients can submit data to and retrieve data from the service.

A valid UID allows usage of the service, and a valid secret allows usage of the UID. So one of the most important questions for securing the service itself is how to secure the tuple of UID and secret, which is mainly a question of securing the UID request process and secret exchange. As we will see, securing the UID request process is mostly a conceptual problem while securing the exchange of shared secrets is a technical problem.

[22] Our current implementation only authenticates and authorizes messages from the client to the server, not vice versa. Attackers could use traditional spoofing or man-in-the-middle attacks to fake server responses on TCP/IP network level. While there are ways to secure the communication of client and server against these attacks, we do not focus on preventing such attacks in this paper for the sake of simplicity.

Authentication and authorization helps to secure the service infrastructure. However, it does not help to identify submissions of - intentionally or unintentionally - incorrect rating information (the content of a rating request), which is a completely different problem. After all, different subjective interpretations of content lead to different ratings. What may be profound art for one person might be explicit pornography for others, and advertisements for prolongation of body parts may be seen as either advances in pharmaceuticals or shameless trickery. We are still investigating on the best ways to identify submission of incorrect rating information, some of which are described in [6, 18].

3.4.2 *Controlling client registration*

The way UIDs are issued strongly influences how resilient the system is against rating spammers. Social services are generally targeted at human users and system operators often have an interest in keeping unwanted automated "bots" out. A popular tactic of spammers is to register a large amount of system accounts for submitting fake data to an information system. This has been seen with link farms for exploiting search engine algorithms [20, 10], e-Commerce websites [2] and spam email. In closed environments, service operators could decide to manually register client accounts. When the service environment is more open, time-consuming registration processes can be replaced by Turing tests such as CAPTCHAs [12, 19] or email activation to hamper or even prevent automated registration of new UIDs.

Our service implementation does not restrict client registration in any way and provides free access. This has the benefit that human users can start to use the service without any additional burden. Client applications register with the service transparently in the background without the need of user input.

3.4.3 *Exchange of shared secret*

Client and service share a common secret used for calculating an HMAC for rating and lookup requests, which allows authentication and authorization of clients. Securing the exchange of the shared secret is therefore a very important requirement of a social filtering service. There are various options for securing secret exchange. We have decided to use HTTP with SSL. Even though SSL connections are computationally more expensive than other means, our choice is justified by the fact that clients request a pair of UID and secret only once on first use of the client.

3.4.4 *Privacy*

Both rating and lookup requests may contain sensitive client (and user) information, so securing communication with the social filtering service is often recommended. Again, one possible approach is to use HTTP over SSL to encrypt all communication of the client with the social filtering service[23]. However, rating and in particular lookup requests happen much more frequently than UID requests and considering the expensiveness of SSL connections, a better approach is to use symmetric encryption algorithms to encrypt just the communication payload (UID, URL etc.), since client and service already share a common secret. Encryption candidates are stream ciphers such as RC4 but also block ciphers such as AES.

While adding encryption to communication helps to prevent eavesdropping, a client's rating and lookup information is still exposed to the social filtering service itself and thereby, indirectly, to other clients via (anonymous) aggregation of ratings. There are techniques [5] for tackling this part of privacy protection. But because the basic idea of a social filtering service is to share information between clients and support cooperation and collaboration, our work did not focus on this aspect of privacy. In our case, the service itself is considered as trusted by the user.

4. USING THE SERVICE

The social filtering service can be used for various problem domains and in different settings. We have implemented two exemplary client applications used for blocking access to pornographic web content: a browser extension and an interface for a web proxy.

4.1 Browser extension

The first client is an extension plug-in for Mozilla browsers. The client registration process is transparent to the user, so the only entry cost is to install the extension software, which is a simple one-click process. The extension adds a toolbar to the browser and provides two buttons for users: a "Rate as porn" button and a "Rate as not porn" button. By the means of these intuitive buttons, users can rate web pages as pornographic or not pornographic. Rating information entered by the user is sent to the social filtering service and processed as described above. Whenever the browser loads a new web page, the extension will automatically query the social filtering service in the background for rating information, cache it locally and, depending on the user's preferences, allow or block (filter) access to the web page being loaded as shown in figure 1. In the case that a web page is blocked, its content is obfuscated and a notification message is centered on the browser window, which provides information why the web page has been blocked[24].

A user's individual ratings overwrite community ratings so a user has full control over her web browsing experience. Based on test user feedback, we have added an "Ignore this warning once" option to the block message for cases when the user is unsure if she agrees with the community on the nature of the web document and wants to take a look herself without overwriting the community rating with a temporary non-pornography rating. Of course, users can chose to disable the filtering component of the extension and provide ratings only.

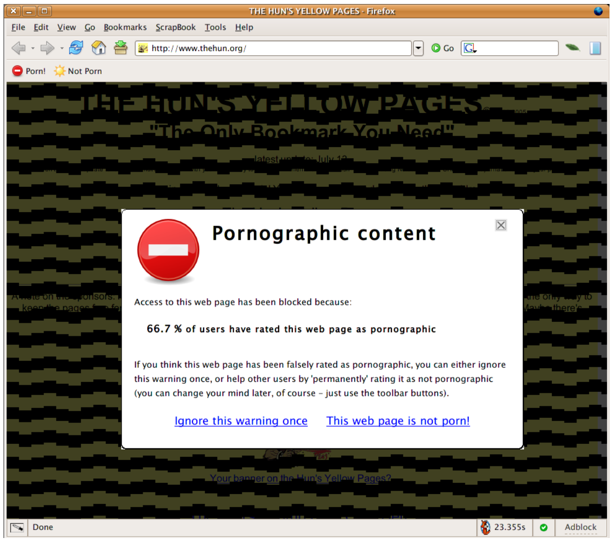
4.2 Web proxy integration

The second client we have implemented is an interface for the free, open source SQUID web proxy[25]. Here, only the lookup functionality of the social filtering service is used.

[23] Note that both rating submissions and lookup requests use HTTP POST, so the data payload is not leaked in the URL as query part.

[24] For example, "You have rated this web page as pornographic" or "82.3% of users have rated this web page as pornographic".

[25] http://www.squid-cache.org/


THE HUN'S YELLOW PAGES - Firefox
File Edit View Go Bookmarks ScrapBook Tools Help
http://www.thehun.org/
Go
Porn!
Not Porn
THE HUN'S YELLOW PAGES
"The Only Bookmark You Need"
Your banner on the Hun's Yellow Pages?
Done
23.355s
Adblock

Pornographic content

Access to this web page has been blocked because:

66.7 % of users have rated this web page as pornographic

If you think this web page has been falsely rated as pornographic, you can either ignore this warning once, or help other users by 'permanently' rating it as not pornographic (you can change your mind later, of course – just use the toolbar buttons).

Figure 1: The browser extension has blocked access to a web page rated as pornographic by the community.

SQUID has the ability to rewrite requested URLs and can be configured to pass every incoming URL through a redirector process that returns either a new URL, or a blank line to indicate no change. Our client application acts as a redirector and can read URLs on standard input, and write rewritten URLs or blank lines on standard output. Whenever the web proxy is asked by a user to download a web page from the WWW, it first passes the web page's URL to the redirector client, which queries the social filtering service for community rating information and locally caches the response in order to reduce load on the service. If the web page has been rated as pornographic by the community, the redirector client will modify the URL so that it points to a local CGI script, which is fed with the received rating information. As a result, the proxy will redirect the requesting user to the local CGI script, effectively blocking access to the original web page. The local CGI script will use the rating information supplied by the redirector client to show an informational notification message to the user.

The described proxy setup can also be combined with the browser extension client so that privileged users (e.g., parents at home, teachers in a school network, or system administrators in a University network) can easily adjust filtering information to protect other users of the system.

Of course, more sophisticated client applications can be built on top of the social filtering service. It could be used to filter comment spam on weblogs. Comment spam often links to "commercial", pornographic, gambling, or otherwise unwanted websites. Weblog applications could leverage the social filtering service to identify spam comments based on hyperlinks included in the comment which link to web pages deemed as inappropriate by the community. Of course, marking weblog comments as spam could also be sed to infer rating information about web documents hyperlinked in spam comments.

Search engines can retrieve rating information from the social filtering service about search query results in order to filter spam links and provide "safe searching" services similar to Google's feature but based on collaborative input from end users about objectionable content.

5. RESULTS AND ONGOING EXPERIMENTATION

We have developed a working prototype of the system, which is being tested by internal user groups. These tests led to improvements in both the overall system architecture and user interface design, in particular the browser extension. We are currently preparing a larger test with the goals of identifying any unexpected scaling issues and learning more about user behavior and service usage. A public release of the service is also planned for later this year, which should enable us to evaluate the system and its architecture in a real-world scenario.

The feedback we have received so far from test users is encouraging. The system does not show any slowdown even for users with a large volume of rating information, and response times for rating submissions and lookup requests have been fast enough that users have not noticed any delay in their browsing experience. Users have appreciated the ease of use of the system and the transparency and understandability of the approach. For example, it generally takes less than five minutes to explain new users how to use the browser extension as an interface to the service and what benefits can be expected. Most users could intuitively use the service without any kind of introduction on our part. The ability to overwrite community ratings with one's own ratings has been an important aspect for users, who feared that individuality might interfere with the "opinion of the mass".

6. SUMMARY AND CONCLUSION

We described in this paper the design and anatomy of an open architecture for creating a social filtering service and showed how it can be implemented. The system is designed to overcome the deficiencies of today's content rating and filtering systems by empowering and involving end users, and to actively support user collaboration and cooperation by using techniques and methodologies with a low cost of participation combined with ease of use. The proposed work improves the quality of information shared by existing collaborative filtering systems by integrating tagging and folksonomy techniques, which we have extended in such a way that relations between document and rating metadata can be better expressed with only a minimal additional amount of required information. Data structures and workflows are optimized for scalability and fast service response times and allow the efficient computation and processing of even very large volumes of data. We showed how such a service can be used by designing and implementing two client applications for filtering pornographic web content.

7. RELATED WORK

The collaborative filtering system described in [9] shares some similarities with our work. However, the focus is on creating distributed, automated software agents which support human users in exchanging content recommendations.

The complexity of the outlined system does also require more interaction between user and agents, i.e. the cost of using the system is higher without clearly visible added benefits when compared with our work. Resnick et al. [14] describe an open filtering system for helping people find interesting Usenet articles and movies. However, rating information is limited to a single score number from 1 to 5, and users cannot share any other information about rated objects. It should be noted that most collaborative filtering systems emphasize on recommendation of information while the proposed social filtering system has been designed and implemented with both recommendation and filtering (removing, blocking) web documents in mind.

The anti-phishing functionality in Google's Safe Browsing extension[26] for Mozilla Firefox browsers is similar to the usage of our social filtering service in combination with the browser extension as described above. However, rating information about phishing websites is not submitted by end users but derived from commercial as well as Google-internal sources (to the best of our knowledge). Websites are checked against a flat whitelist and blacklist database and access is granted accordingly. The lists do not contain any other information about the websites. If at all, the anti-phishing functionality is a subset of our service when combined with the browser extension client application.

The cold start is generally a problem for collaborative filtering systems. At the start of the system, only a small number of users and a small amount of rating information exists, which in turn might prevent other users from joining the service because of expected low benefits. Various techniques to overcome this issue have been proposed such as [17] but it is often still a problem in practice. In our case, existing databases with information about web documents such as the Open Directory[27] can be leveraged for bootstrapping the social filtering system up to a certain extent.

The problem of sparsity in data sets of collaborative filtering systems has been addressed in [16], and [3] discusses the problems associated with high dimensionality of data. Techniques for identifying intentionally incorrect rating submissions are described in [6, 18]. Alternatively, grouping of clients/users or documents can be used to exclude rating information of clients whose rating profile do not match a client's own ratings. Algorithms for grouping, clustering or horting objects in collaborative filtering systems based on similarity measures are described and analyzed in [1, 15, 4].

8. ACKNOWLEDGEMENTS

We would like to thank Alexandre Dulaunoy for many insightful discussions and feedback as well as his support during development of the system prototype. It has been greatly appreciated.

9. REFERENCES

[1] C. Aggarwal, J. L. Wolf, K. Wu, and P. S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the ACM KDD'99 Conference*, pages 201–212, San Diego, CA, 1999.

[2] R. Bhattacharjee and A. Goel. Avoiding ballot stuffing in ebay-like reputation systems. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 133–137, Philadelphia, USA, 2005.

[3] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Proceedings of the 15th International conference on Machine Learning (ICML)*, pages 46–54, 1998.

[4] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[5] J. Canny. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy*, pages 45–57, Oakland, CA, 2002.

[6] P.-A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 67–74, 2005.

[7] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Sixth Symposium on Operating System Design and Implementation (OSDI)*, 2004.

[8] W. I. Grosky, D. Sreenath, and F. Fotouhi. Emergent semantics and the multimedia semantic web. *SIGMOD Record*, 31(4), 2002.

[9] H. Guo, T. Kreifelts, and A. Voss. Soap: Social filtering through social agents. In *ECRIM Workshop Proceedings No. 98/W001 of the 5th DELOS Workshop on Filtering and Collaborative Filtering. The European Research Consortium for Inormatics and Mathematics.*, 1997.

[10] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, pages 271–279, Toronto, Canada, 2004.

[11] S. H. Lee, S. J. Kim, and S. H. Hong. On url normalization. In *Proceedings of the International Conference on Computational Science and its Applications (ICCSA)*, pages 1076–1085, 2005.

[12] W. G. Morein, A. Stavrou, D. L. Cook, A. D. Keromytis, V. Misra, and D. Rubenstein. Using graphic turing tests to counter automated ddos attacks against web servers. In *Proceedings of the 10th ACM conference on Computer and Communications Security (CCS)*, pages 8–19, Washington, USA, 2003.

[13] M. G. Noll and C. Meinel. Web page classification: An exploratory study of internet content rating systems. In *Proceedings of HACK 2005 conference*, 2005.

[26] http://www.google.com/tools/firefox/safebrowsing/

[27] http://www.dmoz.org/

[14] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Group lens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, Hong Kong, 2001.

[16] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl. Using filtering agents to improve prediction quality in grouplens research collaborative filtering system. In *Proceedings of the 1998 ACM conference on Computer Supported Cooperative Work*, pages 345–354, Seattle, USA, 1998.

[17] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, Tampere, Finland, 2002.

[18] X.-F. Su, H.-J. Zeng, and Z. Chen. Finding group shilling in recommendation system. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 960–961, Chiba, Japan, 2005.

[19] L. von Ahn, M. Blum, and J. Langford. Telling humans and computers apart automatically. *Communications of the ACM*, 47(2):56–60, 2004.

[20] B. Wu and B. D. Davison. Identifying link farm spam pages. *Special interest tracks and posters of the 14th international conference on World Wide Web (IW3C2)*, 2005.

[21] H. Yu, J. Han, and K. C.-C. Chang. Pebl: positive example based learning for web page classification using svm. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, Canada, 2002.