Design and Anatomy of a Social Web Filtering Service

by Michael G. Noll
Table of contents

- The problem
- State of the art
- Our approach: social web filtering
- Results and conclusion
The problem

“The Internet is not a safe place for users.”

=> how can we make it safe(r) ?
The problem

Possible answer: “Filter Internet content.”
The problem

Multitude of reasons to filter Internet content

- fight illegal and objectionable content: child porn, racism, violence, etc.
- protect users
  - UK: 66% of parents want improved Internet filters
  - US: 95% of public schools use filtering software
- protect technical equipment

...however
The problem

Multitude of reasons *not* to filter

- what is “illegal” or “objectionable” ?
- protection vs. censorship
- objectivity vs. subjectivity
  - different interpretation of same content because of culture, education, religion, ...
Art or porn?

Lena, Lena Soderberg
(picture: Playboy, 1972)
Bill Gates
(picture: Teen Beat Photospread, 1983)
State of the art

- Filtering of Internet content requires information about content, i.e. metadata
  - By content providers
  - By third parties
  - By computer algorithms

> Can we tackle the problem from a different angle?
Our approach

- the Wikipedia of Internet content rating
- “power to the people”
- human brain >> computer CPUs
- help users help themselves
- true democracy* of the web

> filter Internet content based on metadata provided by users
Concept sketch

www.picshunter.com

pseudo-metadata
“this is a porn page!”

“www.picshunter.com ?”
Our requirements

- improve quality of collaboratively shared information to get better metadata
  - traditional collaborative filtering: won't work
  - social tagging (folksonomy): so-so

- actively support user collaboration
  - make it easy => user wants to use system
  - make it fast & scalable => user can use system
Defining “rating” (1 of 3)

- **collab. filt.**: \( R \subseteq D \times U \times N \) ~ like/dislike
- **tagging**: \( R \subseteq D \times U \times T \) ~ metadata
- **rating**: \( R \subseteq D \times U \times T \times V \) more metadata
Defining “rating” (2 of 3)

\[ R = \{(d, u, t, v) | u \text{ rated } d \text{ with } t, v\}\]

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

- minimal impact on usability
- effect:
  - explicit IS and IS NOT relationship
  - human users: voting [sic!]
Defining “rating” (3 of 3)

Example: rating a medical website about plastic surgery after breast cancer

- (nudity, 1)
- (surgery, 1)
- (porn, 0)

www.plasticsurgery.org
Design and Anatomy

- first open architecture, coded in Python
- client - server
- three main components
  - UID interface = authentication & authorization
  - rating interface = WRITE
  - lookup interface = READ
- \textit{READ} \gg \textit{WRITE}
UID interface

- generates \((uid, shared\_secret)\) tuples
- authentication & authorization with HMACs
- UIDs for clients, not users
- RFC 4122

- **Ex:** \((A688C654-0C18-11DB-A342-7A1C118AA5B2, Up32xJAc30d)\)
Rating interface

- REST and XML-RPC over HTTP(S)
- parameters:
  - *url*, *uid*, list of (*tag*, *vote*) pairs
  - HMAC
  - optional params, e.g. protocol version
- Ex: `http://...?uid=26AD3620...&url=aHR0cD...&tag=porn&vote=0&auth=VQyMinY81Mdi8uR91xLEQ&protocol=1.0&client=firefox`
Storing rating information

- one rating database per client
  - referenced by UID, e.g. /path/<uid>.db
  - hash table: $d_i \rightarrow \{(t_{i1}, v_{i1}), \ldots, (t_{im}, v_{im})\}$
  - constant access time, $O(1)$
  - separation of user data
- bottlenecks: I/O, file system
- tricks: caching, e.g. memcached
Aggregation of ratings

- from client ratings to community ratings
- relevant clients, relevant ratings?

\[ CR \subseteq D \times T \times V' \]

- here:
  - community = all clients
  - community vote is average of client ratings, \( V' = [0, 1] \)
  - 1 community rating database, periodically updated
Aggregation of ratings

Example:

\((d, u_1, \text{porn}, 0)\) => \((d, \text{medical}, 1.000)\)

\((d, u_1, \text{medical}, 1)\)

\((d, u_2, \text{porn}, 0)\)

\((d, u_3, \text{porn}, 1)\)

- Tricks: load sharing, MapReduce (Hadoop)
Lookup interface

- analogous to rating interface
- three rating types:
  - client - “you”
  - community - “us”
  - system - “them”
- here: client > system > community
- constant access time, $O(1)$
Global topics

- security
  - authentication & authorization
  - abuse protection
- privacy
  - encryption of communication
  - trusted service
Using the social filtering service

Examples:
- Browser extension
- Web proxy setup
Results and conclusion

- new approach to tackle Internet safety
  > focus on end users for true web democracy
- new methodology: $R \subseteq D \times U \times T \times V$
- efficient design and implementation
  > ease of use + scalability + security
- evaluation and comparison
- tests by internal user groups