

A Collaborative Filtering Recommender using SOM clustering on
Keywords

Joel Bennett

November 2006

A Proposal for a Project, submitted to the faculty of the Department of Computer
Science, in partial fulfillment of the requirements for the degree of Masters of Computer
Science in the

Golisano College of Computing and Information Science
Rochester Institute of Technology
Rochester, New York

Hans-Peter Bischof Graduate Coordinator

Jessica Bayliss Chair

Leon Reznik Reader

Zack Butler Observer

Abstract

This project comprises designing and implementing a hybrid recommender system for web–pages which uses data from a social tagging system to recommend interesting items to users. For the initial implementation, the tagging data will come from del.icio.us, the oldest and largest public social bookmarking system. The system will cluster items using a self–organizing maps (SOM) network and will include a new SOM visualizer that allows users to see and modify the system’s evaluation of their regions of interest.

The focus of the programming project will be a scraper for gathering tagged URLs from del.icio.us, a visualizer, and the recommender. The SOM network code will be based on existing implementations, and two recommenders will be built, with one using a single map for URLs *and* users and the other using separate maps to compare the relative quality of the recommendations.

1. Background

1.1 Recommender Systems

In the information age, one of the biggest problems facing users — and researchers in particular — is the need to find the useful and interesting items in the overwhelming flow of web pages, emails, instant messages, etc. Information filtering (IF) is focused on analyzing documents and selecting the best documents for users automatically [Good et al., 1999]. Recommender systems (RS) can be thought of as information filters with a particular focus on profiling individual users so that items can be picked out which are specifically relevant to them. However, RS are in some sense more general, being applicable not only to the field of information and documents, but to everything from on-line stores to music players [Burke, 2002].

The initial recommenders were passive systems like search engines which return lists of items matching keyword terms, but modern systems are expected to “learn” the users’ areas of interest as well. There are two major classes of recommenders, based on the differences in their algorithms, and there are additionally hybrid systems like FAB [Balabanović and Shoham, 1997] which use both. Content-based recommenders find items with similar content to other items that the user has previously rated highly. Community-based systems are called collaborative filters, recommending items based on the ratings that similar users have given them.

1.1.1 Content-based Systems

Content-based filters are by far the more mature technology, dating back to work done by H.P. Luhn at IBM in the 1950’s. Luhn introduced the idea of creating profiles for individual users which could be used in an exact match system to produce reading recommendations, and the user’s actual reading habits (this system was designed for use in a library) would then be used to update the user’s profile [Marchionini and Oard, 1996]. The main distinction is that they treat each user as though they were working completely on their own. Disregarding information from other users, if present, a content-based system filters and makes recommendations based on the items themselves, and knowledge

of the target individual.

Content-based systems are more closely related to the field of information filtering than collaborative filters, as they require the items to have intrinsic content that can be used in filtering. This becomes a major drawback to such systems, since some items we might wish to apply recommender systems to simply have no intrinsic content. Thus, content-based systems are primarily document classifiers, and don't generally work with other types of items like movies, restaurants, etc. Another problem with content-based systems is that they may be unable to generalize sufficiently: since they're designed to return items similar to those already rated by the user, they may be too restrictive [Balabanović and Shoham, 1997], and the user may miss out on interesting items outside the range of documents they have already rated.

1.1.2 Collaborative Systems

The second wave of information filtering research was born of the idea expressed by the Information Lens team that social factors could be effective in information filtering, including ratings from other users [Malone et al., 1987]. This idea was not actually implemented in Information Lens, so it was not until the Xerox Palo Alto Research Center (PARC) developed Tapestry that some level of social filtering was actually implemented [Goldberg et al., 1992]. Probably the most well known example of a collaborative recommender system is the GroupLens system from the University of Minnesota which was designed to recommend USENET articles based on ratings from all users.

The GroupLens project pushed collaborative filters forward in many ways, and the team working on it continues to do so today. They also identified the core problem which plagues collaborative filters. Variously known as the start-up problem, the ramp-up problem, or the cold start problem, it is basically a question of which comes first, users or items?

- **New users** have no correlation with existing users and thus get no recommendations. They must rate articles in order for the system to be able to recommend articles, but may abandon the system before they start receiving recommendations because they don't see the benefits.
- **New items** have no ratings and are therefore not recommended. This is particularly troublesome because it does not help to have these items rated if the users doing the rating are new users. That is, an early rating doesn't improve the item's chances of

being recommended, nor does it improve the user’s chance of being matched with other users.

This recommender will essentially be a collaborative system, but will behave as a hybrid system because it is based not just on discovering similarities between users, but also on discovering similarities between documents — based on who reads them and which tags/categories they assign to them.

1.2 Social Tagging

This project will effectively side-step these best known problems for both types of systems by using the data already collected from thousands of users by social bookmarking sites. As previously stated, we will work primarily with the data from del.icio.us, but the core methods could be applied to any of the social bookmarking sites, including language-specific sites, and specialized research sites. This data gives us a pre-existing set of users and items, and is basically all about documents that not only have their own intrinsic characteristics, but have certain characteristics ascribed to them by the users in the form of “tags.”

Tagging is the act of assigning free-form keywords to items with no list of allowed keywords, no built-in taxonomy, and generally no limits on the length or number of keywords that can be assigned to an item. Social tagging allows users to assign keywords to items and share them with others. The assumption in social tagging is that, given this freedom, different users will sometimes use different words to describe the same item, and that the collection of keywords used by all users is helpful in both finding items and browsing similar content. We theorize that similar users will use some of the same keywords in their descriptions of the same item, and will also use similar keywords on similar items, thus, we can take advantage of this to cluster both users and items.

Bookmarks have long been used by web surfers as a way of saving a local reference to specific pages they are interested in [[Millen et al., 2005](#)]. The desire to spread this solution from single users to social groups and enterprise teams has recently lead to a rise of web-based systems for bookmarking. These systems generally reject the traditional hierarchical “folder” hierarchy in favor of tagging systems, and so become social tagging systems where the items are web pages.

At first, these systems were sufficient on their own to allow users to explore other users’ bookmarks via the keywords and tracking users who bookmarked pages you considered interesting. However, with some of these systems (like del.icio.us) having hundreds of

thousands of users, and many users racking up tens of thousands of bookmarks, the task of finding interesting items or users is really quite out of reach for the common user. The goal of our system, therefore, is to deduce the user's preferences and find items that would be of interest from plus

1.3 Self-Organizing Maps

The most important piece of any recommender system is the clustering algorithm. In order to make recommendations in either the content-based or collaborative models, the system must group like items or users together. To cluster the web documents in our system we require a clustering algorithm which can take into account both contents (as represented by categorizing tags) and user ratings, and which can also provide some sort of *measurement* of similarity. The clustering algorithm needs to be unsupervised since there is no way to get direct feedback, and should work well with high dimensional datasets, since the number of possible tags in our bookmarking system is essentially unlimited.

One of the most popular unsupervised clustering algorithms in recent study is the Self-Organizing Map (SOM) neural network algorithm. It has attracted much attention, including over 5000 scientific papers that use or analyze the algorithm in the last 20 years, with hundreds of researchers using the algorithm for everything from pattern recognition to information theory and science [Oja et al., 2003].

The reasons for the popularity of Self-Organizing Maps are simple: not only is it one of the most successful unsupervised clustering algorithms, it takes high-dimensional inputs and represents them in an easy to understand way. A SOM clusters items which are similar so they end up topologically near each other in the resulting map, so they expose the existing relationships between items with high-dimensional descriptions even when there is so much data that the relationships are not otherwise discernible [Kohonen et al., 1996]. Since they work on high-dimensional vectors they are easily usable for our current task, and they create a simple distance metric that can be used to determine similarity.

1.3.1 Semantic Word Clustering

It is worth noting that in these statistical models, synonyms cannot be accounted for — that is, words which have similar meanings end up behaving the same as any other pair of words [Honkela et al., 1998]. One approach for dealing with this is semantic word clustering, which creates clusters of words with similar meanings. Using a clustering algorithm such as the SOM neural network algorithm already developed for document

clustering, we can group similar words together with the goal of eliminating synonyms and even reducing the dimensionality of the classification space. The original WEBSOM system used this method simply to keep the dimensionality reasonable. They did an initial SOM word clustering based on context (the frequency of appearance of words in the context around a word was the input vector). The documents are then encoded by mapping them onto the word map, creating a histogram of the “hits” to each node in the map [Lagus et al., 2004], [Honkela et al., 1998]. In the current project this can be implemented by creating the initial word map based on tag co-occurrence on a URLs, and should eliminate synonyms, but may require caution to avoid marking sub-categories as synonyms of their super-set words. In any case, this term map can then be used to encode the individual documents for placement into a document map.

1.3.2 Code–book model

Luo introduced a variation of the semantic word clustering that works at the level of letters. This method is simpler to apply to this project than to the general case, because the tags we’re dealing with obviate the need for most word-stemming and other pre-processing used by Luo, since the tags are already a small set of relevant words. The code–book system creates a SOM network trained on the frequency of letters appearing at a certain position in the words, essentially a code–book for the character patterns. It then creates a second intermediate SOM based on words for each document (similar to how WEBSOM creates its second SOM based on the first): each word is fed through the first level SOM and the inputs for the second level SOM are the histogram of “hits” —the neurons which fire in processing that word. Finally, the output–level SOM is trained on the documents based on a histogram of hits in the second level SOM [Luo and Zincir-Heywood, 2003]. This method may not work directly with *just the tags* that our system has available, but Luo reports very good results. When compared to the random projection simplification of the vector space that is used by Kohonen’s latest WEBSOM algorithm — an adaptation he found necessary for massive document sets — Luo reports this code–book model has nearly doubled the accuracy from 56% to 90% [Luo and Zincir-Heywood, 2003].

One of the highlights of the SOM algorithm, and certainly one reason for choosing it for this task is that on top of providing reasonable unsupervised clustering, it provides an interesting *visual representation*, and a useful way to allow exploring of the results. The research group at Helsinki University of Technology has encouraged

this by providing some interesting interfaces for visualizing and navigating the resulting maps, and the best is probably the web-based interface to WEBSOM available at <http://websom.hut.fi/websom/> which provides several levels of detail in a zoom-like fashion, and the ability to pan page-by-page around the map at each level of detail. Other good examples include the growing hierarchical SOM (GHSOM), SOMLIB digital library, and LabelSOM algorithms developed at the Vienna University of Technology and available at <http://www.ifs.tuwien.ac.at/~andi/somlib/>, where two examples of it's use are with music in the [SOM enhanced jukebox](#) and countries (based on data from the CIA World Factbook) [using GHSOM](#). This project will include the development of a more fluid visualization with the ability to view and modify the regions of interest that the recommender has identified for you.

1.3.3 Visualization

Most current SOM viewers are static 2D maps, or 2D overview summary maps which allow the user to select one area to get a more detailed map of that area. Some use perspective, pseudo 3D, or real 3D to visualize the number of items in an area of the map, or to compress the distance between clusters, but these map models don't scale very well to hundreds or thousands of items without a great deal of abstraction. There have been algorithms developed which provide a sort of summary for areas [[Rauber and Merkl, 1999](#)]. However, when there is a lot of data, what is really needed is an interface which allows you to zoom to full detail and still pan freely around the map. I intend to create such a zoomable interface for exploring SOMs which will *also* allow the user to select the areas of the map that they are interested in.

2. System Design

This project entails designing and implementing a recommender system which will use data scraped from a social tagging system to recommend interesting items to users. The system will attempt to emulate the benefits of hybrid systems using only the data that was already collected by a social tagging system. It will use a SOM neural network and will include a visualizer that allows users to see and modify the system's evaluation of their regions of interest. The system will cluster items based on the users who tag them and the tags that were used, and will additionally cluster users based on the tags they use and items they tag, allowing it to do both content-based and collaborative recommendations.

We will also create two simpler recommenders for use as a baseline in testing: a tag-based one which uses simple tag frequency, and a user-similarity one which recommends links bookmarked by similar users — who have bookmarked the same pages as the active user in the past.

A web scraper will be created to collect web pages from del.icio.us, parse them as though they were valid XML (even when they are sloppy HTML), and extract user names, URLs, and tags. A database will be created to store the data gathered by the scraper for use in training the recommender and doing a simulated playback for testing.

A SOM neural network will be created along with a data output adapter to feed data from the database to the SOM network in the vector format required. A method will be implemented for storing and distributing the trained network for use by users in parsing new items to determine whether a given user is likely to be interested. We will create a tool for visual navigation of the the trained neural network as a zoomable map, with radius indicators to show the areas where the users's regions of interest have been detected.

Finally, we will create a recommender which will process the bookmarks of an individual user and “learn” their regions of interest, and then pull the latest new links from del.icio.us' “recent” feed and process them according to the trained SOM network, feeding the user only those links which are within their regions of interest.

2.1 Functional Specification

2.1.1 A web downloader

1. Can download web pages, and support HTTP and HTTPS
2. Can support cookies
3. Can support basic http authentication
4. Can queue multiple web pages for downloading
5. Has settings for user authentication

2.1.2 A web scraper

1. Can parse RSS as well as HTML and XHTML web pages
2. Can retrieve data specified as XPath queries

2.1.3 A database for storing bookmark data

1. Can store user-name and tags per URL
2. Can preserve tag-order as entered by the user
3. Can query by URL, User, or Tag

2.1.4 A del.icio.us-specific crawler

1. Depends on the web downloader, scraper, database.
2. Results in storing the scraped data in the database
3. Can differentiate the del.icio.us user page, URL pages, and the recent page
4. Can parse the “recent” page at del.icio.us
 - (a) Finds URLs (using XPath and the scraper)
 - (b) Results in a list of URLs recently bookmarked
5. Can parse user pages at del.icio.us

- (a) Finds URLs (using XPath and the scraper)
 - (b) Results in a list of all URLs bookmarked
6. Can parse URL pages at delicious
 - (a) Finds user names who tagged the URL (using XPath and the scraper)
 - (b) For each user, finds the tags they used for the URL (using XPath and the scraper)
 - (c) Results in adding all users and tags to the database
7. Can convert user-names, tags, and URLs to their respective del.icio.uspage URLs
8. Can follow the trail of URLs, user-names or tags from a given start page
9. Has a setting to stop crawling based on the depth of a crawl
10. Has a setting to stop crawling based on the number of URL pages crawled
11. Has a setting to stop crawling based on the number of User pages crawled

2.1.5 A SOM neural network

1. Can train network on URL-term vectors
2. Can train network on URL-letter vectors (Code-book model)
3. Can train network on user-URL vectors
4. Can (de)serialize trained networks to/from files on disc
5. Can test single input vectors on trained network without affecting training
6. Can incrementally train by adding vectors

2.1.6 A SOM network visualizer

1. Can display trained networks from the serialized output of the SOM neural network
2. Can display an overview with only the most important terms visible (summary)
3. Can “zoom in” on sections of the map to display more and more items (detail)
4. Provides an interface for the user to navigate around the map at any zoom level

5. Can (de)serialize a user's regions of interest to/from file or database
6. Can display regions of interest as variable sized and alpha-blending overlays on the map to represent focus, threshold, and weighting
7. Can maintain the area of interest overlay at all zoom levels
8. Provides an interface for resizing (changing the threshold) the regions of interest
9. Provides an interface for moving (changing the focus of) the regions of interest
10. Provides an interface for weighting (changing the importance of) the regions of interest
11. Provides an interface for creating new regions of interest
12. Provides an interface for deleting regions of interest

2.1.7 A SOM-based Recommender

1. Can load trained networks from the serialized output of the SOM neural network
2. Takes into account both *who* recommended an item (collaborative) and *which tags* were used (content-based)
3. Can learn a user's regions of interest by feeding their items through the SOM
4. Can (de)serialize a user's regions of interest to/from file or database
5. Can accept creation of new regions of interest based on some node coordinate system (for the viewer interface)
6. Can retrieve new items from del.icio.us using the downloader and scraper
7. Can process new items to determine if they are in the regions of interest
8. Has a setting to control the maximum number (K) of items to consider interesting
9. Can filter the "interesting" items to only the *most* interesting K items
10. Can display links to the interesting items and *optionally*
 - (a) The tags that matched the user's interests
 - (b) The most popular tags by other users

- (c) The number of recommending users
- (d) Other data (to be determined) that we may find useful, such as the most similar users who recommended it, the most recent recommender, the date of the oldest recommendation, etc.

2.2 Architectural Overview

The system will be primarily written in C# with SQL Server for storage. There is a back end consisting of the database and the learning portion of the SOM neural network, and a front end consisting of the visualization and recommender portions. The web scraper is required on both ends. The back end uses the web scraper for the initial gathering of data and URLs, and to continue tweaking the training with new data as it comes in. The front end uses the web scraper to gather the recent links each time recommendations are requested, as well as to parse a specific user's links to train the recommender.

The back end and front end will be designed so they work together, but ideally should be designed to allow the front end to communicate with the back end as a web service. This means that the back end should be able to send the trained network to the front-end client via a web service interface, and the front end should be able to make recommendations without any access to the original training data, and ideally without transmitting the user's "favorites" data to the server.

2.2.1 WebDownloader

WebDownloader is a class wrapping the .Net HTTP classes to allow simple downloading of web pages based on URLs. Its primary purpose is to simplify the act of downloading multiple web pages or feeds using cookies for authentication.

2.2.2 WebScraper

WebScraper is a wrapper to an implementation of the .Net XMLReader which is actually SGMLReader. That is, it can read any SGML document (specifically, HTML web pages, even when they're malformed XML, unlike the default XML Reader). Its primary purpose is to allow the querying of web pages via XPath type queries so that flexible scrapers can be written for del.icio.us or other social bookmarking sites. It uses WebDownloader to fetch the pages it needs.

2.2.3 TagCollection

TagCollection is a specialized collection that allow all the data needed to be collected in memory. Instances of TagCollection can be passed directly to TagStorage for deposit into the database. It includes the item string, user, and tags, including the tag-order (and the date it was tagged?). Note that TagStorage requires TagCollection, but TagCollection does not require TagStorage.

2.2.4 TagStorage

TagStorage is a database wrapper that provides an interface for social tagging that could be implemented for any back end database. TagStorageSQL is a specific implementation for MS SQL Server (or SQL Server Express). It provides methods specific to the task of adding text items (URLs) and specifying the user and the tags that the user used. It deposits TagCollections into the database with the goal of allowing any and all useful data retrieved by the web scraper to be duplicated later if necessary.

2.2.5 DeliciousCrawler

DeliciousCrawler is the main tool for gathering data from del.icio.us. It uses WebScraper (not WebDownloader) to gather the data it needs, and stores the data it gathers in a TagCollection, but *does not* deposit it into TagStorage (or reference that class). It internally caches lists of the URLs, tags, or users that it still needs to check (based on user-settable thresholds) and crawls each of them in order, depositing all gathered data into the collection which is returned from the crawl method. This design means it can be used by the back end for the initial crawl, as well as by the front end for parsing the user's bookmarks or the recent links feed (which can't be put in TagStorage which is not available to the front end).

2.2.6 SOMInput

SOMInput formats data from TagStorage for use by the SOM network and SOM trainer.

2.2.7 SOM

SOM is the class that represents a working SOM network for serialization and for use as a classifier.

2.2.8 GHSOMTrainer

GHSOM is an implementation for .Net of the growing hierarchical SOM algorithm to allow training based on data passed from SOMInput rather than strictly from files, and output of trained SOM network objects rather than HTML files.

2.2.9 SOMRecServer

SOMRecServer is the server portion of the SOM Recommender application. It drives the DeliciousCrawler and GHSOMTrainer, and provides an interface for retrieval of the resulting maps by the client applications.

2.2.10 ROI

ROI is the region of interest class, which provides an object containing the necessary data to determine if a link passing through a trained SOM is of interest to a user.

2.2.11 SOMRecClient

SOMRecClient is the main client portion of the SOM Recommender. It initiates the first crawl of a user's bookmarked data, fetches the trained SOM from the server, and builds the ROIs from it. It is linked to the SOMViz for visualization and editing of the ROIs and to SOMRecServer for the server data.

2.2.12 SOMViz

SOMViz is the visualizer GUI. It's one of the client applications, and presents the user with the zoomable UI specified in [subsection 2.1.6](#) based on SOM objects, and allows creation, deletion and modification of ROI objects. It includes a client interface to the server for retrieving

2.2.13 SOMRecFeeder

SOMRecFeeder is the main app that would be used repeatedly by users. It's purpose is to use the ROIs and the trained SOMs, and the data from the recent links on del.icio.usto generate recommendations for users. It fulfills the visualization and feedback requirements of [subsection 2.1.7](#).

3. Project Outcomes

3.1 Deliverables

1. The SOM Recommender project will result in source code, sample data, and software which will run on Windows. It will be delivered via an installer, and will be packaged on CD along with the source code archive, for delivery to each committee member and the CS department.
2. A user manual will be produced in digital format that can be distributed with the installer or printed and bound. The manual will at a minimum detail the features of the software and explain the effect and possible values for settings.
3. An example user account will be created on del.icio.us with a set of bookmarks for use in evaluation when the user does not have, or want to create, their own account.
4. A design document will be created with special focus on the del.icio.us scraper (so it can be maintained)
5. A technical report on the construction of the software, and the results of testing and evaluation and comparison of the recommender output will be provided, along with the design document and user manual to each committee member and the CS department in digital and bound form.
6. All source code, installers, documentation and technical reports will be made available on my web pages at RIT <http://www.cs.rit.edu/~jhb3827/> and on my personal server at <http://JoelBennett.net/Recommender/>, where the regular updates to the project progress will be posted.

3.2 Current Status

I've finished all my background research, including doing some very basic testing of the basic theory. I've found and tested the C++ code for the growing hierarchical SOM. I've found and tested a LabelSOM algorithm for extracting "overview" information, and tested

it with data files based on a small sample of del.icio.us tags. I've found and implemented an SGML parser which implements the .NET XmlReader interface, and implemented some pieces of the del.icio.us scraper.

3.2.1 Potential Issues

I've identified a couple of potential problems based on our reliance on scraping delicious. The web scraper that I implemented for some initial testing was consistently "throttled" by the del.icio.us server, causing it to take an unexpectedly long time to gather any useful data. It may be necessary therefore to gather data slowly and with dates so that it can be "played back" for testing without using the live feeds. Once a data gathering tool has been created, it's also possible to work around the throttling by distributing it to multiple machines and user accounts.

Although this project will require database space and compute time, my current expectation is that this resource need is so small that it can easily be served by my personal computers at home. If we end up needing to distribute the web scraper work, the web scraper should be able to run on the Solaris systems within the RIT CS labs.

3.3 Schedule

The schedule assumes (at least) 16 hours of work per week.

Feature	Duration	Target Date	Risk, details
Web Downloader	16 Hrs	Dec 2 nd	Low risk, partly written
Web Scraper	16 Hrs	Dec 9 th	Low risk, refactoring to design.
Database Setup	16 Hrs	Dec 16 th	Low risk.
Del.icio.us Crawler	24 Hrs	Jan 6 th	Medium risk
Base Crawler	8 Hrs		
Recent pages	8 Hrs		
User pages	8 Hrs		
URL pages	8 Hrs		
Crawling	4 Hrs		
Settings	4 Hrs		
Integration	8 Hrs		Integrate downloader, scraper
SOM Neural Network	56 Hrs	Feb 3 rd	Low risk. Port C++ to .Net
Port C++ Code	16 Hrs		
Serializer	16 Hrs		
Vector Test	8 Hrs		Medium risk
Vector Add	8 Hrs		Medium risk
Code-book model	8 Hrs		Low risk
SOM Visualizer	56 Hrs	Feb 24 th	
Display UI	16 Hrs		High Risk. UI Code
SOM Overview	16 Hrs		Medium Risk. Port LabelSOM
ROI Display	8 Hrs		
ROI Editing	8 Hrs		
ROI Weighting	8 Hrs		
SOM Recommender	88 Hrs	March 31 st	High Risk. Biggest part
Tag-Based Engine	8 Hrs		Server-side SOM training
User-Based Engine	8 Hrs		Server-side SOM training
Learning Preferences	16 Hrs		Client-side SOM use
Serializer	8 Hrs		Client-side "saving" learning
Manual ROIs	8 Hrs		Inputs from Visualizer
New Items	16 Hrs		
Filtering	8 Hrs		
Thresholds	8 Hrs		
Info Display	8 Hrs		High Risk. UI Code
Simple Recommender I	16 Hrs	April 7 th	
Simple Recommender II	16 Hrs	April 14 th	
Documentation	36 Hrs	May 1 st	Low risk, just lots of typing
Comparison Testing	4 Hrs		
User Manual	8 Hrs		Just finishing...
Technical Report	16 Hrs		Much of this will be written
Design Documentation	8 Hrs		As we go along
Installer	4 Hrs	May 5 th	
Target Defense		May 10 th	

Table 3.1: Development Schedule

Annotated Bibliography

- [Arnt and Zilberstein, 2003] Arnt, A. and Zilberstein, S. (2003). Learning to perform moderation in online forums. In *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*, pages 637–641. Minimum description length (MDL) algorithm - probability of each word appearing in a document, given the documents length and the assumption that it belongs to a given category, trims non-discriminating words from the data. This paper discusses it's use in web forum moderation.
- [Balabanović, 1997] Balabanović, M. (1997). An adaptive web page recommendation service. In *AGENTS '97: Proceedings of the first international conference on Autonomous agents*, pages 378–385, New York, NY, USA. ACM Press. The introductory paper to the FAB hybrid system ... claims that pure collaborative recommendation solves all of the shortcomings of pure content-based systems, and content-based systems solve most of the problems inherent in collaborative systems. However, FAB doesn't fully implement a collaborative system, and doesn't overcome the content-based weaknesses.
- [Balabanović and Shoham, 1997] Balabanović, M. and Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72. The Stanford University digital library project has created “Fab,” a partially hybrid recommendation system for web pages. It tracks the preferences of users (they're required to “rate” pages they are given to read) and based on *content* (word frequency) of the pages they rate, it builds profiles. Users are recommended pages which match their profiles, or which are rated highly by other users with similar profiles. [1.1](#), [1.1.1](#)
- [Bielenberg and Zacher, 2005] Bielenberg, K. and Zacher, M. (2005). Groups in social software: Utilizing tagging to integrate individual contexts for social navigation. Master of science in digital media, Universitt Bremen, Bremen. A description of Bayesian classifiers which represent users are decision trees which lead to recommendations, of clustering techniques for users to weight their influence on collaborative recomenders,

and clustering items to use data from any item in the cluster to boost recommender quality.

[Breese et al., 1998] Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. Technical Report MSR-TR-98-12, Microsoft, Redmond, WA 98052. Discusses the classification of recommenders as memory-based (weighted average of other users opinions) or model-based (correlation of items, like FAB, or vector similarity like NewsWeeder).

[Burke, 2002] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370. General overview survey of recommender systems and hybrid systems. [1.1](#)

[Delgado et al., 1998] Delgado, J., Ishii, N., and Ura, T. (1998). Content-based collaborative information filtering: Actively learning to classify and recommend documents. In *CIA '98: Proceedings of the Second International Workshop on Cooperative Information Agents II, Learning, Mobility and Electronic Commerce for Information Discovery on the Internet*, pages 206–215, London, UK. Springer-Verlag. Content-based collaborative filtering. Tries to do collaborative filtering based on tf-idf: similarity of users is determined at the time of bookmarking based on the similarity of documents each user has bookmarked in the past (or rather, the keyword tf-idf profile extracted from them), with greater weight applied to terms from documents in the class of the document being bookmarked...

[Goldberg et al., 1992] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70. Tapestry. one of the first content filters which allowed a form of collaborative filtering. [1.1.2](#)

[Good et al., 1999] Good, N., Schafer, B. J., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J. L., and Riedl, J. (1999). Combining collaborative filtering with personal agents for better recommendations. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 439–446, Menlo Park, CA, USA. American Association for Artificial Intelligence. Author's claim: The next generation of intelligent information systems will rely on cooperative agents to play the fundamental role of actively searching and finding relevant

information on behalf of users ... They've created a system to support research by recommending bookmarks from one researcher to other researchers with similar research interests. It's a fairly simple system, but the paper is disproportionately interesting for it's assumptions and research background. [1.1](#)

[Honkela et al., 1998] Honkela, T., Kaski, S., Lagus, K., and Kohonen, T. (1998). Web-som - self-organizing maps of document collections. In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6*, pages 310–315. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland. A very slick visual representation of extremely large SOM maps for browsing and navigation. Multi-level zoom and links to actual documents. NOT an application of SOM to the web, but rather of the Web as a UI for SOMs. Of course, SOMs using full-text have already been applied to pretty much all types of documents you can imagine. [1.3.1](#)

[Keim et al., 2005] Keim, D., Schreck, T., and Mansmann, F. (2005). Mailsom - visual exploration of electronic mail archives using self-organizing maps. In *Second Conference on Email and Anti-Spam (CEAS 2005), Stanford University, Palo Alto, CA, USA, July 21-22*, Palo Alto, CA, USA. Stanford University. Used a WebSOM-like algorithm to create SOMs, and then (using a spam filter in combination with manual classification) they rate each portion of the map based on the percentage of emails in that region that are considered “spam.” Note that this is not a particularly good filter, since it necessarily leaves many emails on the fringes (there's no way to force a good separation), and doesn't grow. They have a couple other ideas for navigation and organization via the SOM.

[Kohonen et al., 1996] Kohonen, T., Hynninen, J., Kangas, J., and Laaksonen, J. (1996). Som pak: The self-organizing map program package. Computer Science Report A31, Helsinki University of Technology, Rakentajanaukio 2 C, SF-02150 Espoo, FINLAND. The original SOM package from Kohonen available in source and binary format, with documentation. [1.3](#)

[Kohonen et al., 2000] Kohonen, T., Kaski, S., Lagus, K., Salojarvi, J., Honkela, J., Paatero, V., and Saarela, A. (2000). Self organization of a massive document collection. *Neural Networks, IEEE Transactions on*, 11(3):574–585. A discussion of the special considerations needed for scaling SOM networks to map massive document collections. Notes semantic word clustering doesn't scale.

- [Konstan et al., 1997] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., and Riedl, J. (1997). Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87. GroupLens uses only collaborative filters to filter USENET. It’s partially a reaction to Tapestry being “closed.” Identified the cold–start problem.
- [Lagus et al., 2004] Lagus, K., Kaski, S., and Kohonen, T. (2004). Mining massive document collections by the websom method. *Inf. Sci.*, 163(1-3):135–156. Special methods for using SOM on massive document collections. Semantic Word Clustering, Singular value decomposition (SVD), etc. [1.3.1](#)
- [Lang, 1995] Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339, San Mateo, CA, USA. Morgan Kaufmann publishers Inc. Vector–based similarity using cosine of term frequency vectors.
- [Lemire and Maclachlan, 2005] Lemire, D. and Maclachlan, A. (2005). Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM’05)*. A discussion of correlation algorithms (a form of model–based algorithms) which derive models of the decision processes based on the distance between items, such as those used by Fab and RAAP.
- [Li and Kim, 2003] Li, Q. and Kim, B. M. (2003). Clustering approach for hybrid recommender system. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI03)*, pages 33–38. IEEE. This paper has particularly good background coverage of recommenders, from query-based systems through hybrid systems. The point of the paper is to introduce an enhanced collaborative recommender system using clustering techniques to bootstrap new items. This introduces a possibility to use content-analysis as a way of generating keyword tags for pages which haven’t been bookmarked or tagged by users. This has an interesting application in allowing the introduction of new pages (such as advertiser’s web pages) based on automated keyword assignment.
- [Luo and Zincir-Heywood, 2003] Luo, X. and Zincir-Heywood, A. N. (2003). A comparison of som based document categorization systems. In *Proceedings of the International Joint Conference on Neural Networks, 2003*, volume 3, 6050 University Avenue, Halifax, Nova Scotia. B3H 1W5. Dalhousie University. In information retrieval, a typical data representation phase uses the Vector Space Model (VSM), where the frequency of

- occurrence of each word in each document is recorded... this approach considers only term co-occurrences in documents and [ignores order and context]. The authors propose an alternative which they claim results in 90 vs 56 quality. I'm interested in this alternative, but I'm also very interested in how they determined the "quality." [1.3.2](#)
- [Malone et al., 1987] Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A., and Cohen, M. D. (1987). Intelligent information-sharing systems. *Commun. ACM*, 30(5):390–402. Information Lens suggested the need for social factors to be taken into account, specifically, that social filtering rely “not just on the characteristics of the author, but also on the references and recommendations of other people.” [1.1.2](#)
- [Marchionini and Oard, 1996] Marchionini, G. and Oard, D. W. (1996). A conceptual framework for text filtering. Technical Report EE-TR-96-25 CAR-TR-830 CLIS-TR-96-02 CS-TR-3643, University of Maryland, College Park, MD 20742. Among other things, this paper contains a good description of H.P. Luhn's original manual content-based recommendation system from the 1950's. [1.1.1](#)
- [Millen et al., 2005] Millen, D., Feinberg, J., and Kerr, B. (2005). Social bookmarking in the enterprise. *Queue*, 3(9):28–35. Corporate use of Delicious etc. Why you should... Also citeulike-article-id = 703553. [1.2](#)
- [Oard, 1997] Oard, D. W. (1997). The state of the art in text filtering. *User Modeling and User-Adapted Interaction*, 7(3):141–178. Statistics and information on the progress of the state of the art in text filtering or recommenders.
- [Oja et al., 2003] Oja, M., Kaski, S., and Kohonen, T. (2003). Bibliography of self-organizing map (som) papers: 1998-2001 addendum. *Neural Computing Surveys*, 3:1–156. A bibliography of all research papers which have used SOM networks. [1.3](#)
- [Ono et al., 2005] Ono, C., Motomura, Y., and Asoh, H. (2005). A study of probabilistic models for integrating collaborative and content-based recommendation. In *Nineteenth International Joint Conference on Artificial Intelligence (IJCAI05)*. Multidisciplinary Workshop on Advances in Preference Handling.
- [Rauber and Merkl, 1999] Rauber, A. and Merkl, D. (1999). Automatic labeling of self-organizing maps: Making a treasure-map reveal its secrets. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 228–237. [1.3.3](#)
- [Resnick et al., 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., and Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews.

In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina. ACM. Another paper on GroupLens, with specifics on the architecture and how it compares to other systems at the time.

[Riedl et al., 1999] Riedl, J., Schafer, B. J., and Konstan, J. A. (1999). Recommender systems in e-commerce. In *1st ACM. Conf. on Electronic Commerce (EC99)*. A good sources for who's using collaborative recommenders in e-commerce.

[Roh et al., 2003] Roh, T. H., Oh, K. J., and Han, I. (2003). The collaborative filtering recommendation based on som cluster-indexing cbr. *Expert Systems with Applications*, 25(3):413–423. This is probably as close to my actual project as anyone has ever written ... It uses “ratings” to map to other users “like me” but doesn't use keywords or contents to map to articles “like this one.” That is, they worked exclusively with individual's profiles: the SOM (Self-Organizing Maps) creates clusters of like-minded users based on their ratings for each article. They worked from the GroupLens data which has users' ratings of items, whereas I'll be working from bookmarking data, so I have keywords instead of ratings. Given a specific user, they only recommend articles which the user-cluster rated highly but which the user hasn't rated at all. I'll be able to recommend based on similar users and similar keyword documents. They use CBR (Case Based Reasoning), which I'm not very familiar with, to select areas from the SOM. I'm thinking about trying that, to follow their example, and keep the differences of my research as few as possible, however, I *had* been thinking about using simple probability based on proximity to the clusters, or possibly extending it to some sort of bayesian network or use dempster-schafer theory.

[Shardanand and Maes, 1995] Shardanand, U. and Maes, P. (1995). Social information filtering: algorithms for automating “word of mouth”. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co. Ringo, another system like GroupLens which was one of the first to send predictions without requiring users to create queries.

[Vucetic and Obradovic, 2005] Vucetic, S. and Obradovic, Z. (2005). Collaborative filtering using a regression-based approach. *Knowl. Inf. Syst.*, 7(1):1–22. Describes a rating system where the difference between a user's ratings for some items and several expert's ratings for those same items are quantitatively measured and weighted to predict how the the user would rate other items that the experts have already rated. Really good

article, thorough and comprehensive coverage of both recommenders in general, as well as their specific approach.

[Wikipedia, 2006] Wikipedia (2006). Principal components analysis — wikipedia, the free encyclopedia. Online; accessed 22-August-2006. The wikipedia has a good article on PCA (principal components analysis): a statistical method for reducing dimensions or extracting features.