

Independent Study Report:  
A survey of SOM and Recommender techniques

Joel Bennett  
August 22, 2006

A research report, submitted to Jessica Bayliss.

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

© Copyright by Joel Bennett 2006  
All Rights Reserved

# Contents

|          |                                      |           |
|----------|--------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                  | <b>1</b>  |
| <b>2</b> | <b>Recommender Systems</b>           | <b>2</b>  |
| 2.1      | Content-based Systems . . . . .      | 2         |
| 2.1.1    | Search as Filter . . . . .           | 3         |
| 2.1.2    | Clustering as Filter . . . . .       | 4         |
| 2.1.3    | TF-IDF . . . . .                     | 4         |
| 2.1.4    | Minimum Description Length . . . . . | 5         |
| 2.1.5    | Content-Based Drawbacks . . . . .    | 5         |
| 2.2      | Collaborative Systems . . . . .      | 6         |
| 2.2.1    | Tapestry . . . . .                   | 6         |
| 2.2.2    | GroupLens . . . . .                  | 7         |
| 2.2.3    | Collaborative Drawbacks . . . . .    | 8         |
| 2.2.4    | Mitigating Factors . . . . .         | 9         |
| 2.2.5    | Collaborative Benefits . . . . .     | 10        |
| 2.3      | Hybrid Systems . . . . .             | 10        |
| 2.4      | Current Research . . . . .           | 12        |
| <b>3</b> | <b>SOM networks</b>                  | <b>15</b> |
| 3.1      | Processing Documents . . . . .       | 15        |
| 3.1.1    | Vector Space Method . . . . .        | 16        |
| 3.1.2    | Code-book models . . . . .           | 17        |
| 3.2      | Using the Results . . . . .          | 18        |
| <b>4</b> | <b>Confluence</b>                    | <b>19</b> |
| <b>5</b> | <b>Conclusion</b>                    | <b>21</b> |
|          | <b>References</b>                    | <b>22</b> |



# 1. Introduction

Recommender systems are machine learning algorithms which recommend interesting items to users based on their individual preferences. Through their application to electronic commerce and information filtering they have become essential tools for the information age, directing users toward those items that best meet their needs [Burke, 2002]. Research into recommender systems is considered a branch of information filtering that has the particular aims of developing classifier algorithms designed to sort items into “interesting” and “not interesting” classifications, and relies a great deal on good clustering and classification algorithms for success.

One of the most successful algorithms for unsupervised clustering (and by implication, classification) is the self-organizing map (SOM) neural network developed in the mid 1990’s [Kohonen et al., 1996]. Part of the reason for it’s success has been it’s visual nature, and much work has been done to create better visualizations for trained SOM networks. However, the SOM neural network algorithm is still new, and like many neural network algorithms, bewildering in it’s success, thus, much of the research regarding SOMs has been done specifically to test SOM algorithms and their capabilities, and not as a result of simply choosing the technique as part of a larger research project.

This paper focuses on an overview of research in information filtering as it applies to recommender systems, as well as the application of SOM networks to recommenders. We will start with a discussion of the progress of research in information filtering and recommender systems in [chapter 2](#), and continue with a discussion of the developments in of SOM networks and their visualization techniques in [chapter 3](#), and finally finish with a brief overview of one project which attempted to combine these two areas in [chapter 4](#).

## 2. Recommender Systems

In the information age, one of the biggest problems facing users is the need to find the useful and interesting items in the overwhelming flow of web pages, emails, instant messages, etc. Information filtering is focused on analyzing documents and selecting the best documents for users automatically [Good et al., 1999]. In a sense, recommender systems are a superset of information filtering: lacking the focus on documents and information, they are applicable to many different types of items. But in another sense they are a subset of information filtering that is focused on the individual users, being systems which produce individualized recommendations to guide users to useful or interesting items [Burke, 2002]. Developing recommender systems, therefore, can be thought of as developing information filters with a particular focus on profiling individual users so that items can be picked out which are specifically relevant to them.

Although some research in the area dates back to the 1950's, the actual term "information filtering" first came into use in the early 1980's, and research in this field has generally fallen into two main paradigms: content-based and collaborative or "social" filters. Although there has recently been much research into merging these two techniques, it is useful to examine them separately to understand the differences in their approach.

### 2.1 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 [Douglas W. Oard, 1996]. From that humble (and very manual) beginning they have always differed from collaborative filters in that they treat each user as though they were working completely on their own and thus can only filter or make recommendations based on the items themselves. Content-based filters are therefore

applicable almost exclusively to text document recommenders, since in most other areas computers are not yet capable of sufficiently thorough content analysis.

### 2.1.1 Search as Filter

The simplest and earliest such systems were essentially search engines, taking keyword queries and retrieving lists of documents which are relevant to those keywords [Li and Kim, 2003]. According to Oard, of the 60 such systems in operation in 1969, only four had implemented automatic profile updating, with the rest relying on the users (or professional staff) manually updating query terms [Oard, 1997]. These systems are therefore very simple, they are generally not tuned to individual users at all — they return the same list to any user who uses the same terms in a search, and in this sense they really don't fit our description of a recommender.

Additional research (and computing power) brought a new concept to the idea of information filtering: relevance. Defined by the American Heritage dictionary as “The capability of a search engine or function to retrieve data appropriate to a user's needs,” relevance can only be determined on an individual basis, and thus requires more power and information than a simple term-based search engine. It was in 1982 that the term “information filtering” was first used in an introduction of the ACM Transactions of Office Information Systems, and the focus of the field broadened to the concept of filtering incoming information rather than searching archives. The next generation of content-based recommender systems therefore brought systems which constructed simple individual profiles of keywords which were to be sought or avoided for a particular user, the most prominent examples of these were mail and USENET filters such as SIFT, InfoScope, and Information Lens [Oard, 1997].

All three of these worked with simple keyword-based filters and queries, but SIFT and InfoScope also introduced new techniques in their implementations. SIFT brought the idea of clustering user profiles such that articles could first be selected for the group, and then from that collection, items could be selected for individual readers and actually delivered to them. This allowed a significant improvement in efficiency over systems that treated each user's query individually, Infoscope brought a level of automatic query creation: it deduced the keywords you were interested in from the documents that you had saved, marked as interesting, or spent a lot of time reading. However, Malone and the others involved in the Information Lens project had a longer-term impact on the field of information filtering because, although they did not actually implement it at the time, they suggested the need for social factors

to be taken into account, specifically the fact that social filtering would rely “not just the characteristics of the author, but also on the references and recommendations of other people” [Malone et al., 1987].

### 2.1.2 Clustering as Filter

As the focus of new research moved to filtering this incoming data, content-based filter shifted to using clustering techniques to lighten the work of the recommenders. In document-clustering recommenders items are clustered according to some description of them (automatically created from the contents), and users provide some sort of profile of their preferences. The system can deliver documents to users based on whether the user’s profile indicates interest in a specific cluster.

The NewsWeeder USENET filter actually learned the user’s profile based on their rating of individual news articles, inferring their preference for keywords based on the contents of documents which they rated highly, obviating the need for users to manually create their filters. NewsWeeder also distinguishes itself from previous recommenders by requiring *active* feedback from users. Rather than rely on the amount of time a user spends reading an article, or whether they save it ... the user is specifically required to rate each article on a scale from 1 to 5. These ratings allow NewsWeeder to generate *predicted ratings* for new articles specific to the user, and thus, to present the recommendations in a sorted order according to their predicted relevance [Lang, 1995], but it has the downside of requiring the user to do extra work to create and maintain their profile.

In document clustering systems, documents and user-profiles are usually represented as vectors of keywords based on the frequency of appearance in the document (or in the case of users, in documents which they have rated as relevant). Of course, these vectors are very large, and require much processing power to use, but they allow systems to actually compute the similarity of documents, and thus, to recommend documents like those which a user has already found relevant [Delgado et al., 1998].

### 2.1.3 TF-IDF

Most of the recommenders in this classification use a system known as tf-idf weighting which stands for term-frequency and inverse document frequency. The idea is that the more often a word or phrase appears in a document (term-frequency), the more likely it is that that word describes that document — in other words, that the document is in the “category” described by that word. However, the more often that

word appears throughout all the documents (document frequency), the less useful that word is at discriminating between documents [Lang, 1995]. Tf-idf thus yields a vector which can be normalized to unit length and used with any of several different algorithms to calculate the similarity of documents based on nearest neighbor or cosine similarity and allow recommendations to be made.

#### 2.1.4 Minimum Description Length

The most significant contribution of NewsWeeder to the research in content-based recommender comes from the fact that it *does not* use tf-idf, but instead devised an improvement based on the minimum description length (MDL) principle. They based their MDL algorithm on the probability of each word appearing in a document, given the document's length and the assumption that it belongs to a given category, allowing them to leave words out of their models that don't have sufficient discriminative use.

Essentially, MDL, as applied in NewsWeeder, is very similar to work that is currently being done with Bayesian algorithms in the search for better spam-prevention. Using Bayes rule and a history of correctly classified documents, the system estimates the probability of each word appearing in a certain category, rather than the probability that it appears in a specific document, and then predict the overall probability of the document belonging to each category, based on the words that are in it. In limited experiments, Lang found that the the MDL implementation found 21% more relevant articles than the tf-idf implementation [Lang, 1995], and others have since used it with success for such diverse filtering tasks as movie recommendations [Ono et al., 2005] and moderating web forums [Arnt and Zilberstein, 2003].

#### 2.1.5 Content-Based Drawbacks

There are two major drawbacks to the use of content-based filtering. The first, and most obvious, is the fact that some items simply have no intrinsic content. As mentioned earlier, content-based systems are primarily document classifiers, and don't generally work with other types of items like movies, restaurants, etc. The second problem is that content-based systems 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.

## 2.2 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]. Since this was not implemented in Information Lens, it was not until the Xerox Palo Alto Research Center (PARC) developed Tapestry that some level of social filtering was actually implemented.

### 2.2.1 Tapestry

As with most of the information filtering research of the time, Tapestry was designed as an email filter. It worked on mailing lists, USENET and newswire stories, and was largely content-based, consisting of the same sort of rules as SIFT, which the user was actually required to create. However, Tapestry added two important features: The first was the ability to create a second level of filter rules which were run at the local client, and which supported scoring and ranking, so messages could, for instance, be marked as important based on a set of rules. The second and most important addition was the addition of a social aspect.

Tapestry's main function was that it allowed filtering on extra information about messages that was not normally visible to users, and specifically, that it provided for "annotations" by other users. Users could "endorse" messages by providing an annotation, and could use each other's endorsements in their filters [Goldberg et al., 1992]. With this enhancement tapestry moved from content-based filtering to become the first *social* filter: a Tapestry user could create filters to select messages endorsed by two or more people, or even by specific people, or people within their organization. Take these example filters:

```
m.to = 'BugReports'
  AND m.timestamp + [2 weeks] < now()
  AND NOT EXISTS(mreply:(mreply.in_reply_to = {m}))

m.to = 'FeatureRequests'
  AND EXISTS(a:(a.type = 'vote' AND a.msg = {m}))
  AND EXISTS(b:(b.type = 'vote' AND b.msg = {m} AND a <> b))

EXISTS(ml:(((ml.sender = {'Bill', 'Ray', 'Steve'}))
  OR (EXISTS(a:(a.type = 'vote' AND a.msg = {ml}
```

```
AND a.owner = {'Bill', 'Ray', 'Steve'})))))  
AND m.in_reply_to = {ml})))
```

These rules select, respectively, bug reports—that is, messages to the BugReports distribution list—older than two weeks with no replies; feature requests which received at least two votes; and messages sent in reply to either a message from Bill, Ray or Steve, or to a message which they endorsed. Although clearly very powerful, the tapestry system required the user to learn an extensive query language and to put some thought into exactly what they wanted to filter for. Although its authors coined the term “collaborative filtering” to describe what they were attempting to do, its implementation falls somewhat short of the modern understanding of the term, and Tapestry’s influence on further research was sadly restricted by its proprietary nature: having been developed at Xerox and used to filter internal messages, neither the details of the implementation, nor the datasets and results of its use were ever made available publicly.

### 2.2.2 GroupLens

Partly because of frustration over the proprietary problems with Tapestry, the GroupLens project which started that same year at the University of Minnesota was designed from the ground up to be an open system, and to rate only USENET, a publicly available dataset. Published in 1994, the original paper was entitled “GroupLens: An Open Architecture for Collaborative Filtering of Netnews” and emphasized from the beginning that its code and architecture was open: alternative clients can be (and have been) developed, alternative rating servers can be created, and it all inter-operates with the base GroupLens components [Resnick et al., 1994]. In fact, the GroupLens paper, in comparing GroupLens to Tapestry, states one of the two main differences is their open architecture, and this certainly had an effect on how much influence GroupLens has had on further research: to this day their datasets and measurements continue to be used as benchmarks. However, it was the second difference between the two that pushed forward the state of the art in collaborative filtering: automatic aggregate queries.

GroupLens uses user-aggregate queries, and actually correlates the ratings from all users to determine the similarity of individual users, so that a user does not need to know which other users’ endorsements to filter on, because the system automatically picks similar users and uses them to determine the ratings the user would give to a new item based on the known ratings of those other users [Resnick et al., 1994].

So GroupLens is not only the first real collaborative filter, but also one of the first to automate predictions without requiring user queries [Li and Kim, 2003], along with Ringo [Shardanand and Maes, 1995].

The GroupLens concept of collaborative filtering has defined most information filtering research since then, so it is worth looking at in more detail, but the essence is this: users who agreed in their evaluation of items in the past are likely to do so again in the future [Resnick et al., 1994]. The basic process works as follows:

1. Users rate individual items on a scale.
2. An algorithm correlates users based on the similarity of their ratings.
3. As new items come in and are rated by a user, the system predicts that other users with similar rating histories will rate them the same way, and presents these predictions to those users.

### 2.2.3 Collaborative Drawbacks

In the GroupLens system the correlation is done on servers, and the user interface is a series of modified USENET readers which connect to the GroupLens rating servers and can show the predicted (or actual) rating and sort by it. Correlation is done based on individual newsgroups, since early testing showed that users who agree on their evaluation of one “domain” such as a technical forum, don’t necessarily agree in another, such as a humor newsgroup [Konstan et al., 1997].

The GroupLens project also identified the core chicken and the egg problem which to this day still faces collaborative filters. Various known as the start-up problem, the ramp-up problem, or the cold start problem, this is actually a two part problem which applies to both users and 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.

In some cases this problem extends beyond start-up, and is more generally referred to as the sparsity problem. In the domain of USENET, for instance, if the system's current users are all "computer geeks" who read and rate primarily in computer-related newgroups, the items in philosophical, artistic, and even other scientific areas will not be rated. The cold start problem thus continues to exist in those areas where users are sparsely represented. New users who are interested in these under utilized areas will not receive recommendations if there are no other users like them, even after rating many items, and will most likely abandon the system, meaning that once again there will be no users rating items in those areas.

From this problem arise two issues which must be overcome in any social or collaborative filter: a critical mass of users must be present in order for new users to be able to find a cluster of similar users from which recommendations may be drawn, and systems which require explicit rating of items provide a disincentive for use—particularly in the early stages when the reward of good recommendations is not present. In other words: the problem with social or collaborative filtering is that it works best for users who fit into a group with many other users who have similar tastes—you have to have a social group to collaborate with—and you have to provide feedback about items so that the system can learn and give you recommendations.

As research has continued over the past ten years, although the clustering algorithm used to group users (and documents) continues to be the major area of research, these problems related to start-up have received much attention as well, and in many cases are the primary distinguishing factors in later projects.

## 2.2.4 Mitigating Factors

With all of these issues peculiar to collaborative filtering, one might start to wonder why collaborative filters continue to be so popular among researchers. Part of the reason is their simplicity, and part of it is due to the feeling that these shortcomings can be overcome, but the biggest reason is quite simply the fact that collaborative filters don't look at content when making recommendations.

For example, studies show the explicit rating problem may not be a necessary part of the equation. For instance, the GroupLens team found that *implicit* ratings such as "time spent reading" could be nearly as accurate as predictors as the explicit ratings created by users, and could potentially not only remove the burden of manual rating on users, but provide substantially more ratings [Konstan et al., 1997].

Additionally, the cold start problem can be solved in part—at least as it pertains to new items—by the use of hybrid methods, integrating multiple filtering systems to generate recommendations when the collaborative system falls short.

### 2.2.5 Collaborative Benefits

Totally aside from the question of relevance and accuracy, the most interesting reason for research into collaborative filters is that unlike content-based filters, they don't take the content into account.

Rather than spending time on document analysis, developing language parsing tools and word stemming algorithms, the designers of collaborative filters can leave that to other researchers and focus on the core of information filtering research: the clustering algorithms.

Rather than store huge amounts of term frequency data for each user and document, the design of collaborative filters can be somewhat simpler: user profiles are defined by the user's ratings for the items he has rated, rather than probability figures for every word in the English language. Typically the item profile consists solely of the item's actual content, since the rating data need only be stored once, with a reference to the user and the document it applies to, rather than another term frequency profile for each document.

Most importantly, collaborative filters have a vastly larger application domain: since they don't require any analysis of the item contents, they can easily be applied to any types of items. Collaborative filters have been used to filter documents or even books without regard to the actual contents, as well as *non-document items* like pictures, movies, or just items in a web store like Amazon.

## 2.3 Hybrid Systems

The driving motivation for hybrid systems is the desire to find ways to work around the various problems inherent in each of these systems individually. According to [Balabanović and Shoham, 1997], “pure collaborative recommendation solves all of the shortcomings [of] pure content-based systems” allowing the system to deal with any type of items and to recommend items which are—content-wise—completely different from those previously rated. And in turn, content-based systems solve most of the problems inherent in collaborative systems, with the exception of the “new user” problem which still exists since the content-based system still requires a user

profile. However, as Balabanović states, this can be somewhat worked around by providing those users with recommendations based on some average or combination of all users to provide them with a starting point.

Burke identifies seven possible ways that hybrid recommenders can integrate recommendations from different types of algorithms, which amount to six different approaches (he identifies “Feature Combination” and “Feature Augmentation” which are basically different ways of using the output of one or more recommenders as an additional input to another recommender) [Balabanović, 1997].

| Method     | Explanation  |
|------------|--|
| Weighted   | Combines the recommendations of multiple techniques as though they were votes, using a weighting for each recommender.   |
| Switching  | Chooses between recommender techniques depending on the situation (e.g.: falls back to a content-based system when there’s insufficient data for the collaborative system) |
| Mixed      | Uses multiple recommender techniques in parallel and presents all recommendations at once.   |
| Meta Level | Runs multiple recommender techniques in series, successively narrowing the field of documents.   |
| Feature    | Treats output from one (or more) technique(s) as additional inputs to another technique.   |

Table 2.1: Hybridization Techniques

Although in some sense Tapestry was the first filter to combine content and social filters, it does not really implement “collaborative filtering” as it is now understood, but merely allows manual filters to take into account the opinions of other users. Likewise, the earliest example of a system claiming to be hybrid is the Fab recommendation system, which claimed to combine content-based and collaborative filtering. However, at it’s heart Fab is really just a content-based filter. It’s collaborative element is limited to creating two levels of “agents” based on user’s preferences: one set with an agent for the individual user, and a second set with agents that are compositions of all user’s profiles. In that sense, the web-mining system which gathers web pages is a collaboration of the profiles of all users, but like Tapestry, it does not really use the full collaborative system in the fashion of GroupLens, and does not seem to actually overcome the weaknesses of content-based filters [Balabanović, 1997].

The Research Assistant Agent Project (RAAP) is one of the first published

systems that are actually true hybrid systems. Like Fab, it aims to classify and recommend from the web, but with a specifically limited subset of web pages. The intended users of the RAAP system are researchers who would manually add document URLs to the system, by bookmarking them and indicating the document's classification. The system does use an algorithm based on tf-idf analysis of the document to build term profiles of the classifications (rather than the users) and *suggest* cluster classifications for new documents based on their contents *and the user's history*, so it uses information about the user as well as the actual document to suggest the cluster, but the user actually chooses the cluster. The actual recommendations are made based on a modified version of the Pearson algorithm which takes into account the cluster that the bookmarks are in. Thus, the 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 [Delgado et al., 1998].

## 2.4 Current Research

Around this time collaborative recommender systems essentially “hit the big time,” starting to show up in public systems as a part of e-commerce sites like Amazon, CD-Now, and others [Konstan and Riedl, 1999], and the research changed from testing whether collaborative filtering would work, to attempting to devise better algorithms for performing the clustering and computing similarity.

Additional improvements to the correlation algorithm were made by Li and Kim, who observed that the way it was used by Fab and RAAP result in the quality of the predictions being dominated by the contents of the documents. They call their modifications Item-based Clustering Hybrid Method (ICHM) and the essence of it is that they use the user ratings *as well as* the item's information to calculate the similarity of items [Li and Kim, 2003]. The new system results in a three step process:

1. Analyze the documents: in this case, cluster documents and create a cluster-rating matrix.
2. Compute the similarity: in this case compute the clusters and items separately and combine them for a total similarity.

3. Make a prediction based on the similarity.

Continuing research in filtering and recommender systems has mostly followed this process. Much research has been done in recommender algorithms, but they can still be mostly classified into two main categories [Breese et al., 1998]:

**Memory-based algorithms** which use a weighted average of other users opinions to predict the users opinions (where the weight is the similarity between users) like nearest neighbor and others as used by GroupLens

**Model-based algorithms** which derive models of the decision processes can generally be subdivided into

- Correlation algorithms based on the distance between items, such as those described above and used by Fab and RAAP [Lemire and Maclachlan, 2005].
- Vector similarity algorithms like those used by NewsWeeder which calculate similarity based on the cosine of the angle between the term-frequency vectors [Lang, 1995].

Within these categories many different algorithms have been proposed. In the model-based approaches there are including Bayesian classifiers which represent users as decision trees which lead to recommendations, clustering techniques which generally cluster users to weight their influence, and dimensionality reduction which attempts to group items into clusters for the sake of increasing the amount of data for ratings [Bielenberg and Zacher, 2005]. In memory-based approaches the tf-idf model dominates, but there are several “improvements” that have been devised: using the user ratings instead *inverse document frequency*, to examine the *inverse user frequency* rating, using a default vote to assign scores (usually negative) for each item a user has not rated, and using amplification of scores in rating systems, so that higher scores are counted more.

The study done at Microsoft in 1998 indicates that of all of these, the most accurate filters for predicting user behavior are the Bayesian network and correlation methods, however, correlation seems to have a measurable advantage when less data is available. In terms of predicting user preferences (as a difference from the predicted “rating” of an item and the actual rating), once again the correlation algorithm outperformed the others.

This study also showed that the proposed “modifications” to the clustering algorithm do, in fact, improve it’s accuracy. Breese et. al. note a small but noticeable improvement in every test case when using the inverse user frequency method, and

again when using the “case amplification” to emphasize high weights, and even found that these improvements are additive [Breese et al., 1998].

An additional significant outcome of the Microsoft paper is that it has established a sort of benchmark for testing the effectiveness of recommenders, using the “*All but 1*” test: the recommender is presented with the data from all but one randomly chosen data point, and asked to predict that missing point, this is believed to be a good indicator of the algorithm’s performance in real-world use where the database has accumulated a significant amount of rating data from a user. They also established a series of “*Given-X*” tests for comparison where the recommender is presented with  $X$  randomly chosen items and asked to predict the remaining ones.

Recent research has also looked at performance issues. While the model-based systems are clearly more efficient than memory-based algorithms, they are (as previously show) not as accurate predictors. New algorithms are still emerging in this field, such as Slope One, which has been shown to perform on par with Pearson-based schemes in the *all but 1* test, while being simpler, more efficient, and easier to update [Lemire and Maclachlan, 2005].

## 3. SOM networks

The other end of the three step process set out by [Li and Kim, 2003] is the document clustering. This requires a clustering algorithm which can take into account both document contents and user ratings, and which can provide some sort of measurement of similarity. Of course, this clustering needs to be unsupervised, since there is no way to get direct feedback on this one segment of the process.

One of the most studied unsupervised clustering algorithms in recent years 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 them, and hundreds of papers published in areas like pattern recognition, information theory, and information science [Oja et al., 2003].

The reasons for the popularity of Self-Organizing Maps are easy to understand: 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. Since they work on high-dimensional vectors they are easily usable for the document clustering task because the term-frequency representation that is already common in content-based filters can be used as the inputs, and through a form of nonlinear regression the SOM maps them onto a two-dimensional map [Kohonen et al., 1996].

### 3.1 Processing Documents

One of the earliest applications of SOMs to documents was actually performed by the team from Helsinki University of Technology which created the algorithm, in an application known as WEBSOM. In particular, the WEBSOM study makes several observations that are critical to automatic document categorization:

- The number of words in any language's natural vocabulary is prohibitively

large for automatic analysis.

- Standard histogram-style representations lose much meaning by neglecting to account for the ordering of words.
- Synonyms, words which have similar meaning, can't be accounted for in histogram models, and end up behaving the same as any other pair of words [Honkela et al., 1998].

### 3.1.1 Vector Space Method

Of course, since the SOM algorithm takes vectors as input, it's tempting to simply use the term-frequency or even tf-idf representation of the document, but such representations grow to incredibly large dimensions, and the processing power required to run SOM algorithms (or any other clustering algorithm) on vectors of tens or hundreds of thousands of dimensions renders this option unfeasible. As a result, many different methods have been used to simplify the vector space, and it's worth discussing the most common methods as found in previous research.

#### 3.1.1.1 Principal Components Analysis

Principal components analysis (PCA) is a statistical method for general dimension reduction or feature extraction. It's basically a linear transformation to a reduced coordinate space so that the most significant terms (defined here as data points with the greatest variance) are retained [Wikipedia, 2006]. The problem with PCA is that it's really very complicated mathematically, and not well suited for huge datasets .

#### 3.1.1.2 Latent Semantic Indexing

Singular value decomposition (SVD) can be used on the actual term-frequency matrix (represented as documents-by-word) which allows the least significant elements according to the SVD to be discarded. SVD is very powerful, and can reduce documents to 100 or 200 dimensions and introduce similarities between word representations which occur together in documents [Lagus et al., 2004].

#### 3.1.1.3 Random Projection

Random projection is an extremely fast (linear-time) algorithm for computing low-dimension representations of documents by multiplying the term vector by a random

matrix, so that the size of the random matrix controls the size of the output vector, and has been shown to have only a minimal effect on the computation of similarities as long as the output isn't made too small. This method was actually used in the latest iteration of the WEBSOM algorithm in order to make the algorithm scale to the millions of documents they were trying to analyze [Lagus et al., 2004], [Kohonen et al., 2000].

#### **3.1.1.4 Semantic Word Clustering**

Semantic word clustering could be done many ways, the idea is to create clusters of words with similar meaning to reduce the number of dimensions in the dataset. This is obviously a task for which SOM networks may be well suited, and in fact that is what the WEBSOM system originally used. Clustering was done by means of a SOM method which maps the words based on their context, taking advantage of the fact that word order imparts meaning, and thus the inputs to the map are word-vectors instead of document-vectors, with the values being some number of terms surrounding the word, and the expected values of certain terms appearing in that position. This method worked well for WEBSOM and allows the documents to be encoded by mapping them onto the word map, creating a histogram of the "hits" on nodes in the map, which they "blur" to allow for small variations [Lagus et al., 2004], [Honkela et al., 1998].

### **3.1.2 Code-book models**

Recently, Luo introduced a variation of the semantic word clustering that works at the level of letters. This method pre-processes each document to eliminate obvious stop words and performs word stemming, and then chooses a small number (15, in their example) of the most common words in the document, and creates a SOM network for each document trained on the frequency of letters appearing at a certain position in these words, essentially a code-book for the character patterns. Luo then creates a second intermediate SOM based on words, in exactly the same way as WEBSOM creates its final SOM based on documents. That is: each word is fed through the first level SOM (corresponding to its document) and the inputs for the second level SOM are the histogram of "hits" —the neurons which fire in processing that word. Finally, the third-level SOM is trained on the documents based on a histogram of hits in the second level SOM [Luo, 2003].

Clearly, this is a much more complicated process than the semantic word clustering which Kohonen et. al. have used and discounted for very large datasets [Kohonen et al., 2000], but it appears to result in very good classifications: in Luo’s evaluation —based on correctly classifying news articles into the categories which Reuters assigned them— the code–book model results in nearly double the accuracy of the random projection simplification of the vector space that is used by Kohonen’s latest WEBSOM for his massive document sets (90% to 56%) [Luo, 2003].

## 3.2 Using the Results

WEBSOM’s purpose is to automatically organize arbitrary text documents, and it does so using an abbreviated vector representation of word frequency. WEBSOM is essentially the SOM algorithm scaled up to deal with huge amounts of high–dimensional data: their latest experiment mapped 7 million patent documents, described using vectors of 500–dimensional weighted word histograms, onto a million–node SOM [Kohonen et al., 2000], and previous experiments with WEBSOM have been performed on USENET, Reuters’ news articles, and even the Encyclopaedia Britannica [Lagus et al., 2004].

In most cases, the reason for using the SOM algorithm is that on top of providing clustering, it provides not only an interesting *visual representation*, but a good 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 [http://www.ifs.tuwien.ac.at/~andi/somejb/experiments/somejb2\\_col77/index.html](http://www.ifs.tuwien.ac.at/~andi/somejb/experiments/somejb2_col77/index.html) and countries (based on data from the CIA World Factbook) [http://www.ifs.tuwien.ac.at/~andi/somlib/experiments\\_wfb90.html#ghsom](http://www.ifs.tuwien.ac.at/~andi/somlib/experiments_wfb90.html#ghsom).

## 4. Confluence

Although the SOM algorithm is often chosen for its visual nature, it is, as we have said before, one of the leading clustering algorithms, and the clustering power of the SOM algorithm is obviously its most important feature. The WEBSOM project is a good example of how it could be used to cluster documents for the use of an information filtering or recommender system, and in addition the MailSOM project found that a SOM could be trained to discriminate well between spam from non-spam email, and even to generate key-word filters for spam [D. Keim, 2005].

One project in particular stands out as a good example of this use, having actually used SOMs as the clustering algorithm along with a case based reasoning (CBR) for a collaborative filtering recommender called SOMCBRF. This design from the Korea Advanced Institute of Science and Technology uses PCA to reduce the dimensions of the input items, and to determine the number of clusters they expected to find, and then use the SOM algorithm to find and detect clusters and create standard centroid users, that is, fake example users for each cluster.

Since SOMCBRF is a collaborative filter, the SOM algorithm actually maps the *users* rather than the documents, and so it is trained on the scores that users assign the documents, rather than on the contents of the documents (which eliminates all of the problems associated with using SOMs on plain text documents).

As a last step, the CBR algorithm is used to index the users into the map by placing them in a cluster with the centroid user they are closest to, and when recommendations are called for, the algorithm picks the users who are in the same cluster based on the CBR indexing, and predicts their rating for an item using Pearson-correlation.

The authors tested their implementation against the fairly standard MovieLens data set that is part of the GroupLens project, and compared it to the standard Pearson algorithm, several simple baseline predictors, and a couple of alternative designs using SOMs along with a neural network and an induction decision tree. Their experiments showed that their SOM and CBR model had better accuracy than any of the competitors with the sole exception that the SOM and neural network classifier showed a slightly lower normalized mean absolute error in the “all but 1”

test, although it was worse overall [Roh et al., 2003].

## 5. Conclusion

Recommender systems have come a long way in the last twenty years as text-analysis and clustering algorithms have improved, but there is still much room for improvement, with rating accuracies generally fairly low in most research and applications, and even the SOMCBRF showing predictions only within 15% of the actual values. However, there is still much room for improvement in both algorithms and methodology, as this SOMCBRF, for instance, does not take into account document contents at all, and the CBR algorithm may not be best suited for the task of clustering items, as is evidenced by the outstanding performance of a neural network in the “all but 1” test —which we have previously said is probably the most indicative of real world performance.

The SOM algorithm shows much promise in clustering all kinds of items, from music and countries to documents and even movies. It’s continued development is a presumptive fact, as much research is being focused on it’s use in many varied areas of research, and there is still much research to be done about the best way to represent documents for use with it. Finally, it’s visual representations are varied and there is much room for improvement there as well, with the state of the art in the WEBSOM project still being static images that are displayed one at a time as a user navigates from one web page to another. Last but not least, picking clusters and classifying items using a trained SOM is still an area for research, but there are many ways to do so which have been shown to work, from the simple distance algorithms to CBR and neural networks.

The combination of these techniques with sufficient data and a hybrid approach may lead to the best recommender algorithm yet, or it may not, but it’s clearly worth investigation and experimentation. Additional uses for SOMs will undoubtedly continue to be discovered due to the sheer volume of research in the area, so clearly better ways of visualizing the output cannot go amiss.

# References

- [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. [2.1.4](#)
- [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. [2.3](#), [2.3](#)
- [Balabanović and Shoham, 1997] Balabanović, M. and Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72. [2.1.5](#), [2.3](#)
- [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. [2.4](#)
- [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. [2.4](#)
- [Burke, 2002] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370. [1](#), [2](#)
- [D. Keim, 2005] D. Keim, F. Mansmann, T. S. (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. [4](#)
- [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. [2.1.2](#), [2.3](#)

[Douglas W. Oard, 1996] Douglas W. Oard, G. M. (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. [2.1](#)

[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. [2.2.1](#)

[Good et al., 1999] Good, N., Schafer, B. J., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J., 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. [2](#)

[Honkela et al., 1998] Honkela, T., Kaski, S., Lagus, K., and Kohonen, T. (1998). Websom - 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. [3.1](#), [3.1.1.4](#)

[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. [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. [3.1.1.3](#), [3.1.2](#), [3.2](#)

[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. [2.2.3](#), [2.2.4](#)

- [Konstan and Riedl, 1999] Konstan, J. B. S. J. and Riedl, J. (1999). Recommender systems in e-commerce. In *1st ACM. Conf. on Electronic Commerce (EC99)*. [2.4](#)
- [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. [3.1.1.2](#), [3.1.1.3](#), [3.1.1.4](#), [3.2](#)
- [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. [2.1.2](#), [2.1.3](#), [2.1.4](#), [2.4](#)
- [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)*. [2.4](#)
- [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. [2.1.1](#), [2.2.2](#), [2.4](#), [3](#)
- [Luo, 2003] Luo, X. Zincir-Heywood, A. (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. [3.1.2](#)
- [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. [2.1.1](#), [2.2](#)
- [Oard, 1997] Oard, D. W. (1997). The state of the art in text filtering. *User Modeling and User-Adapted Interaction*, 7(3):141–178. [2.1.1](#)
- [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. [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. [2.1.4](#)

- [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. [2.2.2](#)
- [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. [4](#)
- [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. [2.2.2](#)
- [Wikipedia, 2006] Wikipedia (2006). Principal components analysis — wikipedia, the free encyclopedia. [Online; accessed 22-August-2006]. [3.1.1.1](#)