DESKTOP RELATIONAL SEARCH OR ONE WAY AI COULD SAVE US FROM OUR FILES

D. R. Schlegel
Computer Science Department

Herein a new system for the automatic organization of user files based on metadata is presented. The system is based on a new derivative of Kohonen's Self-Organizing Map algorithm which is also discussed in detail. The system overall is one which aims to provide an end to end user experience for their file and metadata management.

I. Introduction

Consider the chore of searching for files on your computer. Consider the usual metaphors for search. You go to your computer, realize you've lost some document, open up a search tool, type in some sort of search string and wait as the computer finds the document you are looking for. What if you don't remember the file name? What if all you remember is that you worked on it with your roommate in college? With today's search paradigm you are out of luck.

The goal of this paper is to outline a new system which has been created to solve this kind of problem. This system is not based on searching for file names, it is based on the looser and less well defined search for relationships. Now imagine you go to your computer and type in a search string such as "Paper with Alex" and the system parses that and searches for it in some sort of relational store of your data. You would be returned a lot of things having to do with Alex himself, papers of his, pictures of him, etc… but tangentially related visibly could be a collection of things which you did together. This is the grand vision of search – and while the work shown here doesn't quite reach that goal, it does get us much closer.

Herein I will chronicle my attempt at a search system such as this. First discussed will be the problem with searching using today's file organization techniques and the concept of metadata. Next I will present the method I have used to solve this problem and its origins. Finally the search algorithm will be considered in context as part of a larger system from the point of view a user.

II. File Organization Overview

We are all familiar with today's methods of storing files on our computers. Our files are stored on drives, the drives are split into a folder hierarchy, and files are stored throughout this hierarchy. When this paradigm was developed disk size was measured in kilobytes, not

gigabytes or terabytes. In those days we used to label our disks with what specifically they contained. The main organization difficulty was losing the physical disks themselves, not losing our files.

Today our operating systems try to recreate this paradigm through the use of special folders like “My Documents” in Windows XP. Windows Vista takes this one step further with folders such as “Pictures” and the concept of stacks which are actively updated with all files meeting a certain criteria. These files are just deposited into these folders willy-nilly with no attempt to organize by content. Users don’t like to spend their time actively organizing their files, so there must be another way to find what they are looking for. Some applications such as Windows Media Player will manage your music folders for you, creating new ones when you rip CDs, but this does not suit every user. We are all different and like to access data in different ways.

Enter search. Since nearly the beginning of computing we have had search systems that would find something based on its file name or type. In the old days of DOS filenames using the “8.3” format (8 letters for the file name, and three for the extension) this made practically no sense. UNIX traditionally has fared better due to less strict limitations for file names, but file name doesn’t always convey what a file is really about. These search systems would manually traverse a directory tree looking for matches. This requires a complete disk traversal which is very costly.

Today we enjoy desktop search systems from Google, Microsoft, and Apple which catalog our files and even our e-mail in a centralized database. Desktop Search tries to emulate the internet search experience on the desktop. When we search on the internet we navigate to a search engine, enter our query and are returned a list of relevant pages from a pool of billions of documents. Therein lies the problem as yet unsolved – documents are not in standard easily searchable formats (a la HTML), and on the computers of today documents are no longer the principal file format.

Lately we seem to have a bit of an infatuation with the ease of internet search engines such as Google. When we search on the internet we look for a topic based on a search string, click on the results, and see if they contain what we are looking for. This paradigm does not easily extend to the desktop - search for our own files is a much more difficult job. We have to search for image and mp3 files directly without the benefit of related data that could be stored on a web page. Therefore extending this “search-and-browse” paradigm to the desktop cannot succeed.

Google has tried to change this internet search paradigm by adding video and image search to their engine, but just as on the desktop this requires a lot of manual tagging work, and the results are often faulty (Could Google News be Sued for Libel?, n.d.). They have spent countless man-hours on this technology for these two file types and it is useful but as it currently stands is not transferable to the desktop, and of course it only represents a small subset of files on a user’s machine anyway.

So how to do we fix this? The amount of files on our computers is growing all the time. Combine this with the laws of entropy and the ability of any organization system we devise becomes limited. Therefore we must seek out a new paradigm, a new, automatic, strategy

for organizing data based on what the data really means, based on the things we as users would use to organize our data.

III. Metadata

Contained within our data, without our even knowing it, is a resource just waiting to be tapped. In some cases this data is already there just waiting for us to look at it. In other cases it requires a bit more work to get at, but is still helpful. This resource I refer to of course is metadata. Loware defines metadata as "...our knowledge of data that we have interpreted as information in a particular decision-making situation (Laware, 2005)." So what really is metadata then? More simply, it is knowledge about our data.

When we talk about knowledge about data it follows that this knowledge needs to at some point either be explicitly defined or automatically generated. In the case of IDv3 tags for mp3 files it is explicitly defined by someone, and then propagated using a tool over the internet to our media players when we rip our CDs or sync our music with an external player. For a textual document on the other hand, the metadata is generally not going to be explicitly defined and requires some sort of a generation tool.

As time goes on it will be more important to discover ways to extract important features from our documents – whether it be image tagging as in social network sites like Facebook, or in rhythmic identification in our music files (Tenbergen, 2007). Once this new metadata is extracted there will be the need to organize it in a relational way, in a way which makes sense to the user.

If metadata is determined properly it should describe the data it represents. This highly filtered form of the data is much easier for our computers to process and compare versus whole documents of unknown type. There have been many attempts to solve this problem. On the web a search engine can employ algorithms to generate "snippets" of a web site. These snippets are essentially an abstract or the main idea of the article. Microsoft Word allows another way to auto-generate the idea of a document through its summary feature. So for at least a few document formats we have already started to solve the problem of generating the metadata. There of course is more work needed in this area but after we have the metadata putting it to use is all that is left.

So far we have discussed that we have a problem with organizing our data because we have so much of it. We have also discussed the fact that there is knowledge about this data we have available to us. Combining this knowledge with some maturing AI techniques a new, more natural paradigm can be developed. We have to combine metadata, search, and artificial intelligence techniques to help create a new paradigm for search.

IV. Self-Organizing Map

The Self Organizing Map, developed by Teuvo Kohonen, is an artificial neural network (ANN) based technique for organizing data in an unsupervised manner. This means that there is no user interaction while the algorithm runs, denoting a “black-box” type algorithm. It is generally represented by a two dimensional array of nodes (which in ANNs

have the biological counterpart neurons). These nodes contain some numeric data which represents what is being organized.The result of the map is similar to multidimensional scaling techniques which aim to recover the “underlying structure among stimuli which is ‘hidden’ in the data (Schiffman, Reynolds, and Young, 1981)."

When the algorithm is initially started the map is created using the specified size and is initialized with random data. This random data is stored in what is called a *weight vector*, and is of the same dimensionality of the input to the map.

Once the training of the map begins, input vectors in a training set are exposed to the network. When this happens, each node has a level of activation and that which is activated the most is called the Best Matching Unit. Generally to determine this level of activation Euclidean distance is used, and the one which generates the smallest distance is the BMU. This unit and its surrounding *neighborhood* nodes are then made to be closer to the input vector (Zavrel, n.d.) (How much closer is determined by a Gaussian function and a learning rate usually). The neighborhood is defined by a radius which shrinks over time.

This process is iterative, happening hundreds or thousands of times until the network is converged and smooth. This happens since over time the neighborhood shrinks. At the start of the run it may be as wide as the lattice, but by the end it is usually very small in relation to the map size. Some SOM algorithms split this into two phases. During the first, longer, phase the neighborhood shrinks very rapidly. During the second phase which lasts much longer, the neighborhood shrinks much more slowly until it is finally converged (Kaski n.d.). Kohonen mentions in his book on SOMs that since there are generally far less training data than numbers of iterations of training necessary "the samples may be applied cyclically or in a randomly permuted order, or picked up at random from the basic set (so-called *bootstrap learning*). It has turned out in practice that ordered cyclic application is not noticeably worse than the other mathematically better justifiable methods (Kohonen, 2001).”

The speed of convergence of the network is very important. If the network converges too quickly it will end up mosaic-like, with blocks of related data but not very smooth. If it converges too slowly then each node in the network will contain largely the same data. Either of these cases are not optimal and will only result in results which are not representative of the data.

The typical toy example for self organizing maps is to categorize colors based on an array containing their RGB values. This case is very simple since the color can already be represented with numbers without any extra abstraction and there are a finite number of values which fall into definite categories. The results from these maps are very successful and have been shown to work with any random initialization of the map (Honkela et al., 1998).

In addition to toy examples, there have been several real world applications of SOMs. SOMs have been used extensively in Biology to sort organisms into their respective families. Kohonen himself has spent many years working to organize large document collections using what is called the WEBSOM method (Honkela, et al., 1998). Other researchers have worked to create virtual library systems which involve creating “shelves” of similar data to be browed by a user on the web (Merkel & Rauber, 1999).

All is not rosy for the SOM though; there are a great deal of real world limitations when it comes to the algorithm itself. The self organizing map depends on a very structured model for feature vectors. It is easily applicable to systems where feature vectors are easily represented through numerical values. For textual values though, things are more difficult. For example, Kohonen mentions in his SOM book that most ways for organizing documents involves to some extent a word histogram. Methods are employed to simplify it, but it is still a histogram of thousands of words, with lots of empty space, and breaking any biological representation the SOM might have.

V. Abstract Self-Organizing Map

The ASOM or Abstract Self Organizing Map is different than a normal SOM in how it handles its features. The method for comparing feature vectors is not as obvious as it is with a word histogram or some other method since the input vectors may be only tangentially related (or not at all) to the training set. To deal with this the actual computation comparing vectors is done outside of the map itself where a weight showing the correlation between the input and the node on the map is generated. This abstraction is the reason for the algorithm being called the Abstract Self Organizing Map.

The ASOM (Fig. 1) has been designed to work with any set of data. The only requirement is that the author of that specific implementation devise a way to make the data comparable. A node in a self organizing map contains the numeric weights on which the algorithm will operate. In an ASOM the node is basically a container for the *Feature Vector* and its coordinates on the map.

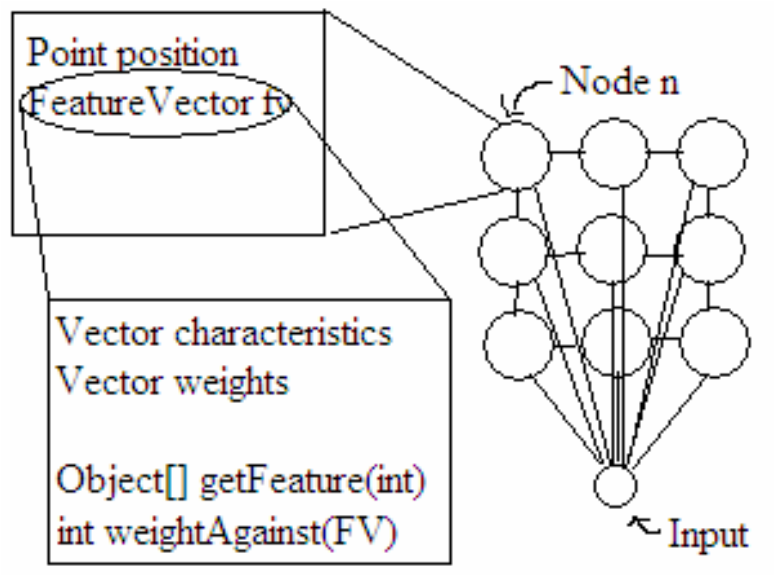

Fig. 1: ASOM Algorithm

The feature vector in an ASOM is really two vectors which are side by side. The first of these is a vector of *characteristics* and the second is a vector of *weights*. The feature vector also contains some method of comparing itself to another feature vector and returning a single value, this is called the *relative weight*. This relative weight function replaces the Euclidean distance method for determining similarity in Kohonen's SOM algorithm. It is important to mention though at this point, that an argument could be made against this algorithm being a neural network. It contains many of the same features of a neural network, but ideologically it breaks from the tradition of neural networks by requiring the weights to be computed and not stored concretely. Also, in the same way the SOM algorithm does, this algorithm goes against the highly interconnected view of a neural network. Indeed without an input vector, no connections exist between the nodes, which obviously is not biologically accurate and is not perhaps conducive to maintaining its status as a neural network based algorithm.

A characteristic is a defining aspect of whatever the implementer is trying to organize. It can be of any data type as long as the implementation can compare that type for likeness. The weights are a numeric value which say how important that specific characteristic is. These values are unbounded and should be tweaked to fit the needs of the specific implementation. Finally the relative weight is the outcome of the comparison. In general higher values denote a higher level of likeness, and lower values denote the opposite. This too is unbounded and should be taken as a relative scale in comparison to the entire map and never on its own.

Since the ASOM shares its main architecture with Kohonen's feature map, in many of the same ways that SOMs fail to represent biology, so does the ASOM. Even though this is the case, the ASOM may be more akin to how we consciously think about information. The SOM fails in mimicking our thought processes by forcing everything to be represented through weights with no real data attached to them. The ASOM though creates relative weights by actually comparing the data and not just a weight associated with it. This is more similar to how we draw connections between information cognitively, but is perhaps more an issue of semantics than an actual likeness.

Altohough this seems to be an improvement conceptually, there are still issues. It appears from initial testing that the rate of learning must be very carefully tuned to the amount of iterations of learning (and consequently the radius of the neighborhood – this is especially important in a small network). If either of these are disproportionate then the system over-converges, meaning that each node becomes nearly identical to every other. Obviously search breaks down at this point and results are nearly random.

V.a *Implementation*

In my test of the map I chose to use 100 Word 2007 documents which I had local on my machine as a sort of real world test. Based on the metric of ¼ the number of nodes as documents, the result was a 5x5 map. Looking at examples from other SOM work this seems to be a fairly natural value – Kohonen used 1/6 in his WEBSOM experiment but there is no tried and true method for determining this value. The advantage though to a

larger map is that there is more room for unrelated documents to separate themselves. A 5x5 map is fairly useful in analyzing whether or not the map works since there are several inner nodes and several edge cases.

After creating the map the next step was to define what the feature vectors were going to contain. For my feature vectors I extracted all of the words which are capitalized and aren't extremely common sentence starters (such as For, It, The, Then, etc…) and set the weight for each of these characteristics to the number of occurrences of that word. This is obviously a very crude method but for the purpose of testing the maps ability to organize data it is sufficient.

In order to compare these feature vectors the algorithm first finds all of the characteristics which are in common between the two FVs being compared. The weights of each in common characteristic are multiplied together and these values are summed. This determines the relative weight of these two feature vectors.

The implementation which I designed uses a Gaussian function for the purpose of converging neighbor nodes with the BMU. The Gaussian function is useful for this since it allows for the natural graded-ness of the map which we are seeking. The learning rate is two, and the terminal learning rate is 1/2. I ran the map for 100 iterations. Testing at different values determined this was optimal for this size map, but there is no known algorithm for determining what this value should be. At values much smaller than this (~50) the map would not have any gradient, and at values much higher (~300) the maps gradient was too large (as expected and discussed earlier).

V.b. *Algorithm Analysis*

To test the ability of the ASOM to organize data I used a technique which is used often with traditional SOMs. This method tries to determine whether the topology of the map has converged at the end of the training. This involves using "input samples to determine how continuous the mapping from the input space to the map grid is. If the two reference vectors that are closest to a given data item are not neighbors on the map lattice, the mapping is not continuous near that item(Kaski, n.d)." Testing that two nodes are neighbors is not enough though – the map should be smooth. If the map has a large spike when an input is presented and nearly no activation around it, then the map has not been trained over enough iterations.

To do this I applied some sample search strings to the converged map. In each case the node for which the BMU was the document I was looking for was found, and one node removed from it showed lower activation levels. Depending on the search string usually two nodes removed produced an activation level of zero which is expected for a map of this size. A sample of this is shown in Table 1. This has been tried with maps smaller than this, and the results seem to scale well from the limited tests I have performed thus far.

0	0	0	0	0
10	60	10	0	0
15	10	0	0	0
10	10	0	0	0
0	0	0	0	0

Table 1: A search gradient

Neural networks have a reputation of not being particularly speedy. The combined time of training the map (including getting the generated metadata from persistant storage) averaged around three minutes on an older Pentium 4M laptop. Previous tests have shown that smaller maps take far less time to compute. There is a nearly exponential increase in time as the training set grows larger, but there are several techniques for increasing the speed of SOM-like algorithms which have not been explored yet. In comparison, standard desktop search systems can take many hours or days to catalog just the document data on someone's PC.

The ASOM algorithm and method provide a great deal of needed improvement over the original SOM algorithm. The major improvements are reiterated below in a more easily digestible tabular form.

Abstract SOM	"Classic" SOM
Flexible design – Doesn't require changing the algorithm to sort different data types	The SOM is crafted and optimized for the target data type as needed
Can sort multiple types of data together as long as a method is defined to make them mutually comparable	Maps are created with a specific purpose in mind, there are few (if any) "in general" type solutions.
Uses feature vectors to store comparable data. Weights are relative, on no particular scale.	Does not have a defined method, sparse word histogram arrays are often used for text. Must be made to correspond to a 0-1 weight value

VI. The User Perspective

Making search relational is not enough, the user needs an intuitive way to interact with their files. I have thought hard about what types of applications this metadata-driven system might allow us to take advantage that we cannot today. One example may be a system to tag images. As the user adds tags to an image they influence the way search works with those files. This adds an extra dimension to the idea of search – essentially giving the user control over their search results. It is conceivable that in time this could be expanded with functionality to tag a group of related images together, or to propagate changes to "like" images. This flexibility will be appreciated by the user later when they search for their files.

This allows a user to impact search before they do it, but all that we have done here is so that the user can relationally search for their files, and an interface is needed for that. In the SOM Explorer interface (Fig. 2) there is no need for an "advanced search" interface – the user can simply type in what they are looking for and hit search. Since the SOM has already been created and has converged the search results are returned instantly. The nodes in the center of the image are very closely related to the search string, and the top ranked feature in that node is displayed. As you look further out in the image the items are less related. This is simply a view of the underlying neural network which the SOM algorithm has already ordered in this way for us.

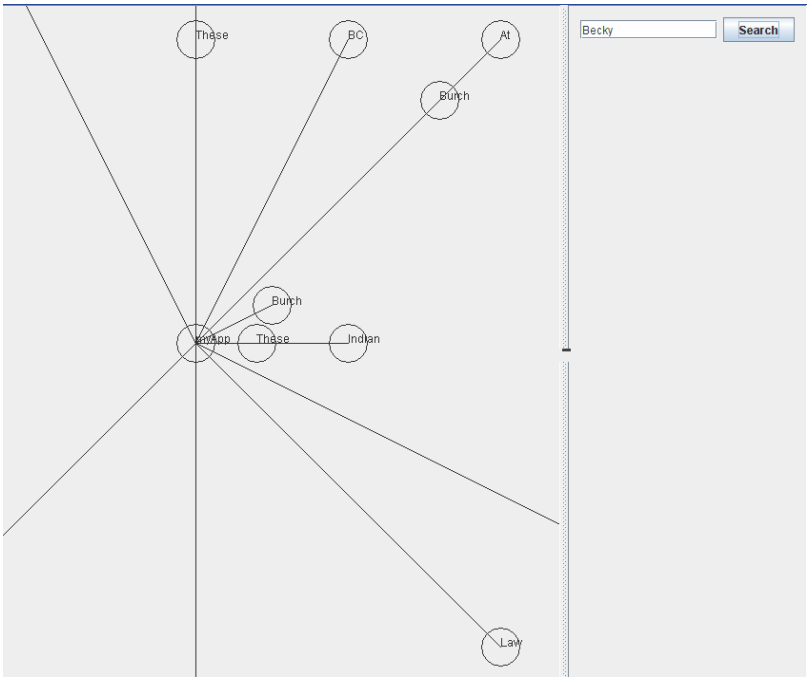

Fig. 2: SOM Explorer

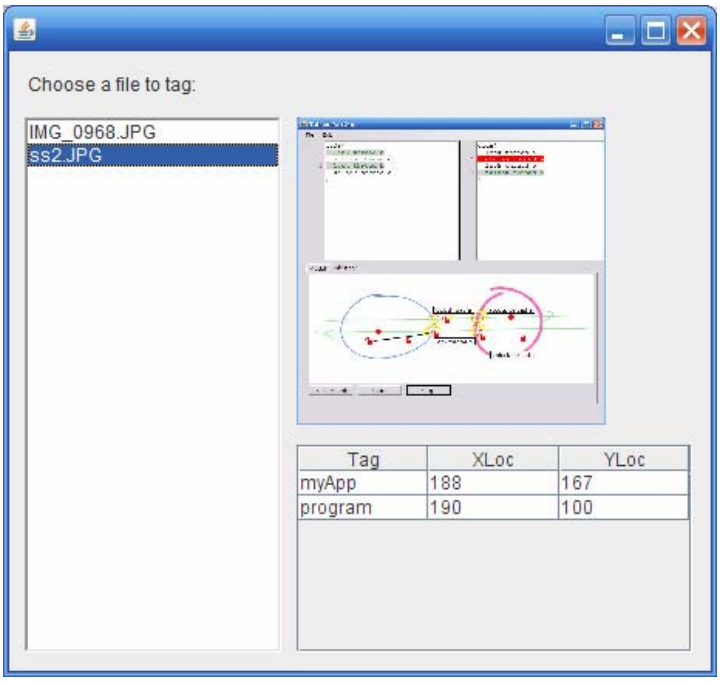

Fig. 3: Image Tagger

VII. Thoughts for Future Work

The list of further work I would like to do on this project easily is far more expansive than any of that which has been completed thus far. These changes range from finding methods to enhance the speed of convergence to finding better results through more useful metadata generation techniques.

One way to speed up the convergence could be to take advantage of multicore processors. Kohonen mentions some examples in which he uses many small maps of data to estimate the layout of the larger map. These smaller maps could each be computed separately in dedicated processors or processor cores which can then be combined and calculated as we are now.

In addition to the improvements in speed, there is a need for better data to test on. The feature vectors from the documents are formed in a very rudimentary way (choosing all of the “important” capitalized words). I can see several simple ways of improving this. Chopping off a words prefixes and suffixes would help the weighting be more accurate (Etzioni and Zamir, 1998), and grouping words that are capitalized that appear consecutively may also help since they are likely related.

I would also like to explore how we determine characteristics of documents more carefully. In addition to using proper nouns, I think the structure of documents should be considered. For example, the size of text, or whether or not text is in a bulleted list can provide a sense of importance to words or phrases. Using this kind of analysis we could not only compare documents with similar important words or phrases, but also documents with a similar structural format.

On top of these improvements more tests will be needed, on a larger data set. Obviously a 5x5 network is not appropriate to determine if an algorithm is a success or failure. Just because it works at a small scale doesn’t mean it will not break down or become far too complex at larger sizes. I aim next to test on a network at least an order of magnitude larger, with perhaps a thousand documents and two hundred nodes.

As I’ve mentioned multiple times throughout this paper, documents do not define all or perhaps even a large part of the files on a user’s computer. The project contains an early tagging system for images now, but there is much of room for improvement. Current AI techniques for facial recognition and comparison could be used to automatically tag images. In addition to images, work has been done to categorize music not only by tag but by the actual musical patterns. This type of system could be included into this system with relative ease.

VIII. Conclusion

Organization of user data has over the last few years become an exponentially more difficult problem as the ubiquity of the internet and downloading of files becomes obvious, and several attempts have been made to try to solve the problem. These solutions though are just extensions of research that has already been done for search on the web and do not translate particularly well to the desktop paradigm. There have been novel ideas such as

Microsoft's proposed relational file store (WinFS), but it was ultimately cancelled in 2006 (Clark, 2006).

In order to succeed researchers and companies alike need to begin thinking about metadata as a viable method for ascertaining the meaning of user data. There have been many problems trying to use metadata in the past because it is largely non-standard. We can no longer ignore what is in reality self evident though, we need to bite the bullet and write code for these non-standard constructs or work together to ensure our data has proper metadata and that it is properly used.

Algorithms such as the SOM have shown promise for document organization, but have been quite clumsy and slow. Over time our AI techniques will improve and I think therein lies the next paradigm of search. AI has consistently gotten a bad rap as consisting of a collection of very complex programs which don't work, but slowly AI based applications are making their way to the desktop. The solution I have presented here is just one example of this evolution. Although I don't think there will ever be a permanent solution for our user data woes, I hope the technique presented here will help to fuel the next generation research.

IX. References

Clark, Q. (2007). *Update to the Update – WinFS Team Blog.* June 26, 2006. http://blogs.msdn.com/winfs/archive/2006/06/26/648075.aspx (accessed November 13, 2007).

Could Google News be Sued for Libel? http://www.cyberjournalist.net/news/001346.php (accessed November 3, 2007).

Etzioni, O. and Zamir, O. (1998). "Web document clustering: a feasibility demonstration." *Proceedings of the 21st annual international ACM SIGIR conference on research and devoeopment in information retrieval*, 46-54.

Honkela, T, Kaski, S., Kohonen, T, and Lagus, K. (1998). "WEBSOM - Self-organizing maps of document collections." *Neurocomputing* 21, 101-117.

Kaski, S. (n.d.). (*Data Exploration Using Self-Organizing Maps.* PhD Thesis, Helsinki University.

Kohonen, T. (2001). *Self-Organizing Maps.* Springer.

Laware, G. W. (2005). "Metadata Management: A Requirement for Web Warehousing and Knowledge Management." In *Web Mining: Applications and Techniques*, edited by Anthony Scime.

Merkel, D., and Rauber. A. (1999). "The SOMLib Digital Library System." *Third uropean Conference on Research and Advanced Technology for Digital Libraries*, 323-342.

Schiffman, S. S., Reynolds, M. L., and Young, F. W. (1981). *Introduction to Multidimensional Scaling : Theory, Methods, and Applications.* Academic Press.

Tenbergen, B. (2007). "Similarity Clustering of Music Files According to User Preference in Proceedings." *MICIA*, 182-192.

Zavrel, J. . *Neural Information Retrieval: An Experimental Study of Clustering and Browsing Document Collections with Neural Networks.* PhD Thesis, University of Amsterdam.