

# **Categorization of High-Dimensional Knowledge-Based Representations of File Data Using Abstract Self-Organizing Maps**

**By Dan Schlegel**

Computer Science Department, State University of New York at Oswego

**Abstract:** Here a new algorithm based on the Self Organizing Map is presented which allows for easier categorization of textual data. The algorithm is based heavily on the concept of metadata and its extraction and providing an abstract look at that data. The algorithm is titled the Abstract Self Organizing Map for that reason. The paper discusses the current and potential impact of metadata, the current paradigm for file organization, Teuvo Kohonen's traditional SOM algorithm along with my new adaptation of it. Preliminary results are presented based on organization of real world file data. These results show that the algorithm is in fact effective in these situations and an outlook towards the future based on these results is discussed.

## **I. INTRODUCTION**

Self Organizing Maps have been theorized to be useful in the categorization of everything from music collections [1] to titles of books [2]. The map clusters the data according to the relatedness of it by a weighting metric and it does so in an unsupervised manor. A previously unexplored application of this is to organize a user's data on their computer. This involves using the untapped resource of file metadata. Metadata can be generated by the file on its creation, generated from analyzing the file, or defined by the user manually as in image tagging. Storing all of the generated metadata in a single area such as a database allows multiple files metadata to be compared to each other and a similarity between them determined. Once this data is stored and organized, the user can use the system to search for their files by a standard search string or by selecting existing metadata from a list. I will show here an organization/search system based on a Self Organizing Map which does exactly this. Since the user is able tag data as well as relying on the systematically retrieved metadata the user can have a direct hand in their search results. Since data is organized by file content the experience of using a computer and finding user data can be improved greatly, and as a consequence of the

algorithm used, the data is arranged relationally providing better search results.

## **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 formats on our PCs.

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 can not 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 [3]. 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 [4]." 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 [1]. 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 solved 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.

In summary 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 will result in this new, more natural paradigm. We have to combine our ideas about metadata and search to help push this new paradigm.

#### IV. THE SELF-ORGANIZING MAP

The Self Organizing Map, developed by Teuvo Kohonen, is an artificial neural network 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 [5]."

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 [6] (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 [7]. 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 [8].”

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 [9].

There have, though, been several real world applications of SOMs. It has 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. Other researchers have worked to create virtual library systems which involve creating “shelves” of similar data to be browsed by a user on the web [10].

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 [8] 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.

The SOM is useful for organizing data but it is not strictly speaking biologically accurate. In the brain when data is presented there is not one winning node as there is in the SOM algorithm. More biologically precise is the kWTA or winner takes all for multiple winners, algorithm [11].

SOMs with very high dimensional data such as large document collections do not scale particularly well. In one example using the WEBSOM2 [12] method, Kohonen mentions that it took several weeks to organize 6 million patent documents on a map containing one million nodes.

There have been several different versions of the self-organizing map created by researchers to solve different problems with it. Some examples being the growing self-organizing map [13] which doesn't have a concrete size allowing for more flexibility when it comes to how different the data applied are. Another example is the CSOM or cyclical self-organizing map. This concept is much like a cyclical queue in which the sides of the SOM are attached – removing the “edge” nodes and leaving only inner nodes. One example which tries to speed up the algorithm is a “bagging” SOM in which there are predefined categories [14]. A last example is the enhanced self organizing map which tries to make the SOM adhere better to the results from kWTA techniques. None of these though have attacked what I think is a major issue with the algorithm – the weighting system.

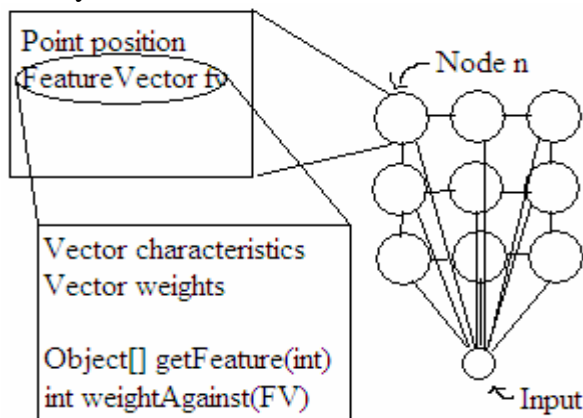
## V. ABSTRACT SELF-ORGANIZING MAP

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 and returns a weight showing the correlation between the input and the node on the map. This abstraction is the reason for the algorithm being called the Abstract Self Organizing Map.

The ASOM 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.

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 the removal of traditional node weights makes this algorithm no longer strictly a neural network.



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.

Another unique aspect of the abstract self organizing map is having a terminal value for the learning rate. This means that as the learning rate decreases over time, it will not decrease below this value. This allows the map to retain some plasticity after organization is complete so that the exposure of an input vector not in the initial training set will still be allowed to affect the map without retraining. The same is true of the radius of learning. Obviously a radius of zero wouldn't be helpful. This will prolong the usefulness of the map, but is not a replacement for occasionally recreating the map (which could be done aside from the current map so when it is done they are swapped out, transparent to the user).

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.

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). 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.

A second issue which may also plague the original SOM but which I could find no documentation on is what I call the size-reliability correlation. With the size of the map relatively small, it isn't possible to train the map for a significant amount of iterations without reaching the over-converged state mentioned earlier. Therefore a smaller map which is made up of data all of extremely high dimension then without the amount of training necessary will remain under-converged!

## **VII. IMPLEMENTATION AND TEST PARAMETERS**

In my test of the map I chose to use 36 Word documents which I had local on my machine. Based on the metric of  $\frac{1}{4}$  the number of nodes as documents, the result was a 3x3 map. Looking at examples from other SOM work this seems to be a fairly natural value – Kohonen used  $\frac{1}{6}$  in his WEBSOM experiment. A 3x3 map is difficult since the initial training phase places the center of the map in the position to be very generic and not very helpful in the organization.

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.

At this point it is worth noting that SOMs and ASOMs can have different purposes. In some cases the goal is only to see what categories exist in the map, and in others they would like to know what inputs belong to each of these groups. This project falls into the later case obviously and therefore the BMU is also stored with each input to the network. The BMU changes as the map organizes so storing it in this location is most efficient.

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  $\frac{1}{2}$ . 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.

## **VIII. ANALYSIS**

To test the ability of the ASOM to organize data I used a technique which is used often with traditional SOMs. One method to determine whether the topology of the map has converged at the end of the training is "to use 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, Phd) Testing that they 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 been trained too fast.

So to do this I used the 3x3 map mentioned earlier and applied some sample search strings. 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.

0	0	0
15	15	0
35	10	5

Depending on the search string usually two nodes removed produced an activation level of zero which is fine for a map of this size. A sample of this is shown in the given table. How this would fare in a larger map will require further testing.

Neural networks have a reputation of not being particularly speedy. The combined time of pulling the 36 feature vectors from the database and training the map averaged only 15 seconds on my Pentium 4M laptop. Again, scalability is still unknown but this is a promising result. In comparison, standard desktop search systems can take many hours or days to catalog just the document data on someone's PC.

## IX. FUTURE WORK

The system as presented here is very basic and no where near ready for consumption by anyone other than a researcher. While the ASOM algorithm itself is fairly complete, but there are several more features and experiments that are needed to make the system usable by the average user.

First it is worth mentioning that there is room for algorithmic improvements. As mentioned earlier when creating the map the nodes are populated with features from files on the disk. As of now these features are only added, not removed from the nodes. It is possible that allowing removing of features would make the map converge quicker and be more feasible on the desktop.

In order to make this algorithm feasible on a users PC, there must be some method to persist the map once it has been made and to dynamically load the features to apply. This will reduce the amount of memory required and allow the user to load the map on demand when needed. One technique that might be very powerful is actually storing the files themselves in the database with their feature vectors and creating a transaction layer between the operating system and the database.

A final interesting algorithmic idea to pursue is to exploit the existence of multi-processor systems. Kohonen mentions some examples in which he uses many small maps of data to estimate the layout of the larger map. By estimating the layout speed can be increased greatly. These smaller maps could each be computed separately in dedicated processors or processor cores. Another multithreaded application may be to computer more than one larger map and compare the validity of the map using techniques mentioned before for measuring smoothness and relatedness.

Next, it must be mentioned that the tests done on the map are inconclusive in many ways. Obviously a 3x3 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 break down or become far too complex at larger sizes. This work is the next thing I aim to get done with the project. Kohonen used a set of 6 million patents which mapped onto a SOM with a million nodes to test his SOM and it took weeks to run. I aim to test on something smaller, with perhaps a thousand documents and two hundred nodes.

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 [15], and grouping words that are capitalized that appear consecutively may also help since they are likely related.

Since the goal if this system is eventually for it to be used on desktop PCs by the normal user, a self explanatory user interface is going to be necessary. I envision a system where the organization happens in the background and a user can search by entering a string of natural language. The words the user enters would be deemed important and a feature vector would be created. This also gives the user a chance to filter the results and add a supervised aspect to the otherwise unsupervised algorithm.

## X. CONCLUSION

Organization of user data has recently become a much bigger problem than it has in the past, 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 work particularly well on the desktop. There have been novel ideas such as Microsoft's proposed relational file store (WinFS), but it was cancelled in 2006 [16].

In order to succeed researchers and companies need to begin thinking about metadata as a viable method for ascertaining the meaning of a file. There has 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.

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 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.

## XI. REFERENCES

- [1] Tenbergen, Bastian. *Similarity Clustering of Music Files According to User Preference* in Proceedings from MICIA 2007, p182-192.
- [2] Lin, Xia, Marchionini, Gary, and Soergel, Dagobert. "A Self-organizing Semantic Map for Information Retrieval." In *Proceedings of the 14th Annual international ACM SIGIR Conference on Research and Development in information Retrieval* (Chicago, Illinois, United States, October 13 - 16, 1991). SIGIR '91. ACM Press, New York, NY.
- [3] Could Google News be Sued for Libel? *CyberJournalist.net*, Accessed November 3, 2007. <<http://www.cyberjournalist.net/news/001346.php>>
- [4] Laware, Gilbert W. *Metadata Management: A Requirement for Web Warehousing and Knowledge Management* in Web Mining: Applications and Techniques edited by Anthony Scime, 2005.
- [5] Reynolds, Schiffman, and Young, *Introduction to Multidimensional Scaling : Theory, Methods, and Applications* Academic Press, 1981.
- [6] Zavrel, Jakub, *Neural Information Retrieval: An Experimental Study of Clustering and Browsing Document Collections with Neural Networks*. (PhD Thesis, Univ. of Amsterdam).
- [7] Kaski, Samuel, *Data Exploration Using Self-Organizing Maps* (PhD thesis, Helsinki Univ).
- [8] Kohonen, Teuvo, *Self-Organizing Maps* Springer, 2001.
- [9] Honkela, Kaski, Kohonen, and Lagus, *WEBSOM - Self-organizing maps of document collections in Neurocomputing* 21 (1998) pp. 101-117.
- [10] Merkel, D. and Rauber, A., *The SOMLib Digital Library System* in *Proceedings of the Third*

*European Conference on Research and Advanced Technology for Digital Libraries*, pp. 323-342, 1999.

[11] O'Reilly, Munakata, *Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*, MIT Press: 2000.

[12] Honkela, Paatero, Kaski, Kohonen, Lagus, Saarela, and Salojarvi, *Self Organization of a Massive Document Collection in IEEE Transactions on Neural Networks*, Vol. 11, No. 3, May 2000, pp. 574-585.

[13] Alahakoon, D., Halgamuge, S. K. and Sirinivasan, B. (1998) *A Self Growing Cluster Development Approach to Data Mining* in Proceedings of IEEE International Conference on Systems, Man and Cybernetics, San Diego, USA, pp 2901-2906

[14] Georgakis A., Li, H., and Gordan. M., *An ensemble of SOM networks for document organization and retrieval*, in *Proc. of Int. Conf. on Adaptive Knowledge Representation and Reasoning (AKRR'05)*, pp. 141--147, Espoo, Finland, June 2005.

[15] Etzioni, Oren and Zamir, Oren. *Web document clustering: a feasibility demonstration in Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp 46-54, 1998.

[16] Clark, Quentin. *Update to the Update – WinFS Team Blog*, June 26, 2006. Accessed November 13, 2007.  
<<http://blogs.msdn.com/winfs/archive/2006/06/26/648075.aspx>>