01 Jan 1996

Concepts of Adaptive Information Filtering

Daniel R. Tauritz
Missouri University of Science and Technology, tauritzd@mst.edu

Concepts of Adaptive Information Filtering

by
Daniel Tauritz
Last revised July 11th, 1996

Index

Preface

This paper was written for the project study “Adaptive Information Filtering” at the Department of Computer Science, Leiden University, The Netherlands. The assignment was to write an introduction to Adaptive Information Filtering (AIF), based on the author’s ideas for his M.Sc. thesis, and with as large an audience as possible in mind. In addition to a simple introduction to AIF, this paper should also provide easy introductions to clustering algorithms, evolutionary computation, and n-gram analysis.

Chapter 1 - Introduction

What is Adaptive Information Filtering?

Adaptive Information Filtering (AIF) is the process of filtering incoming data streams in such a way that only relevant data (information) is preserved. The relevancy of the data is dependent on the changing (adaptive) needs of a particular person or group of persons with a shared interest. Think for example of a newspaper. From all the news in the world available to the reporters a selection is made based on what is deemed to be of interest (relevant) to the readers of the newspaper, the rest is filtered out. As the information needs of the readers change the reporters must adapt their selection criteria correspondingly or the newspaper will fail.

What is an AIF system?

An AIF system is the complete system necessary to perform the process of AIF. One example is the system employed by our newspaper, radio and television news reports employ similar systems. The type of system discussed in this paper will be a software system. An AIF system consists of three main components. The data stream, the transformation from data to information through filtering, and the adaptive behavior. In the case of a software system the data stream might for example consist of Internet news, the transformation be performed by a computer program, and the adaptive behavior by the interaction between the user and the computer program.

The data stream

In the case of a software system with as incoming data stream Internet news, the data stream consists of textual documents with a specific classification. For example, if a document was posted to newsgroup x, then one may assume that the document can be classified corresponding to the topic of newsgroup x. This is of course not always the case, but except for groups with a very low S/N (signal to noise) ratio it will hold for the majority of the posted documents. Another common incoming data stream is E-mail which also consists of textual documents but which lacks a specific classification. In the rest of this paper textual documents will be assumed. To deal with the data stream it is necessary to be able to compare the documents with the interests of the user at a given time. This implies the necessity of storing the interests of the user and the need for a transformation of documents and user interests to a common space where they can be compared. Such transformations will be discussed in chapter 2.

The filtering transformation

The basic idea is that any document which differs “too much” from the user’s interests will be filtered out, thus leaving only relevant documents for the user to peruse. There are however a number of reasons for desiring a more sophisticated approach to the filtering process. First of all, users do not tend to make a sharp distinction between relevant and non-relevant data, rather their interest gradually declines as the subject matter increases in distance from the subjects of their choice. And, secondly, they will often have varied interests and in this case will prefer to have documents pertaining to each particular interest grouped together. One method is to cluster the incoming data stream and represent the user interests as points in the space of clusters. Clustering methods will be discussed in chapter 2.

The adaptive behavior

In order to adapt to the changing information needs of the user, interaction with that user is necessary. On two important issues input from the user is needed. First of all input is required concerning the user's interests, and secondly the user must indicate which documents have been clustered correctly and which incorrectly. One possibility is that the clusters themselves represent the user's interests. A mechanism particularly suited to facilitating this adaptive behavior is evolutionary computation, which will be discussed in chapter 2.

Chapter 2 - Methods

Paragraph 2.1 - Document and user interest transformations

Semantical versus syntactical spaces

In order to compare documents and user interests they must be transformed to a common space. A space in which both documents and user interests are expressed in terms of their semantics would be ideal. The distance measure would then indicate, for instance, the degree to which a document and a user interest correspond semantically, precisely the comparison one would hope for. The semantical analysis of written text is, however, still an open problem in computer science. Even lexical analysis is extremely complicated and requires so much computational power that it is not feasible for an AIF system where typically huge numbers of documents have to be processed in real time. An alternative is to choose as common space a syntactical space. This would greatly simplify the transformation because no understanding of the text is necessary. A practical choice for common space is a vector space where the vectors describe certain syntactical characteristics of the documents and the user interests. A vector space is a practical choice because many well known distance measures exist for it, such as:

- Euclidean distance: $d(\bar{x},\bar{y}) = \|\bar{x} - \bar{y}\|$
- Cosine correlation: $d(\bar{x},\bar{y}) = \frac{\bar{x} \bullet \bar{y}}{\|\bar{x}\| \cdot \|\bar{y}\|}$

where $\bar{x}$ and $\bar{y}$ denote vectors, $\bar{x} \bullet \bar{y}$ their inner product, and $\|\bar{x}\|$ is the norm of $\bar{x}$. For more information on measurements in the information sciences see [Boyce94].
The difficulty with syntactical transformations lies in the problem of choosing one for which the distance measure in the vector space reflects the semantical distance as accurately as possible. Two possible transformations will now be discussed.

Keyword analysis

An often used transformation is one based on keywords. In the simplest form a predetermined set of keywords is used. The transformation projects documents on vectors consisting of elements representing the frequency with which the keywords occur in the document.

Example

keyword set = {tree, house, sun, winter}
document = {In the summer the house gets hot from the sun. In the winter however the sun does not have this effect.}
vector = (0, 1, 2, 1)

The problem with using a predetermined set of keywords is twofold. First of all it is necessary to know beforehand which keywords should be included in the set. This requires domain specific knowledge. Secondly it precludes adapting to changes in the domain which would require the addition of new keywords to the set. To overcome both problems one could take the set of all words as keyword set. Unfortunately, this would require vectors too large for practical use. Also, new words are constantly being created, adding still further to the problem.

N-gram analysis

A completely different approach is the so-called n-gram analysis. The n stands for a positive integer. In 1-gram analysis the occurrence of single letters is determined, in 2-gram analysis that of pairs of letters, in 3-gram analysis that of triplets (in this context called trigrams), etc.

Example

document = {In the summer the house gets hot from the sun. In the winter however the sun does not have this effect.}
1-gram vector = (1,0,0,1,14,...)
a b c d e
2-gram vector = (..., 3, 1, 1, ...)
er es et
3-gram vector = (..., 5, 0, 0, ...)
the thf thg

The larger n, the more accurately the distances between n-gram vectors correspond with the semantical distances of the original documents [Scholtes93]. The price paid for larger values of n is an exponentially increasing demand for memory, namely proportional to 26^n (or 27^n if you include the space), see figure 1.

n	# elements
1	26
2	676
3	17576
4	456976
5	11881376

Figure 1. N-gram analysis memory requirements for various values of n.

For $n \leq 3$ the memory demands are still moderate enough for practical use, $n = 4$ is for the moment only interesting for research purposes, and $n > 4$ is currently not of any use. In practice this means only $n = 3$ is normally used as the results for $n < 3$ are not accurate enough to be of any use [Scholtes93]. The great advantage of trigram analysis is that it is domain independent and the set is small enough to be used entirely so that the problem of having to expand the set because of domain changes is circumvented.

Paragraph 2.2 - Clustering methods

What is clustering?

Clustering is the process of dividing a set of objects into multiple sets of objects, called clusters. For an overview of clustering algorithms see for example [Backer95].

Example

original set = {apple, cake, strawberry, lemon, pie, cookie}
set no.1 of clusters = {[apple,strawberry,lemon], [cake,pie,cookie]}
set no.2 of clusters = {[apple,cake,lemon,pie], [strawberry,cookie]}

In this example, the first set of clusters divides the original set into two distinct sets where the clustering criterion was, is it fruit or something you bake? Note that this requires knowledge about the objects to be clustered. The second set of clusters also divides the original set into two distinct sets, this time with as clustering criterion, is it a word of more than 5 letters, or not? Note that this requires no knowledge about the objects to be clustered, it is purely syntactical.

Incremental versus batch clustering algorithms

There are many clustering algorithms which all differ with respect to complexity, performance, and clustering properties. An important distinction can be made between clustering algorithms which require the whole set of objects to cluster in advance and those which can process one object at a time and present the results of the clustering after each newly processed object. The first class of algorithms will further be referred to as batch clustering algorithms, the second as incremental clustering algorithms. As a typical AIF system has a perpetual incoming data stream, the rest of this paper will be restricted to discussing algorithms of the second class.

A simple clustering algorithm shell

To illustrate how a clustering algorithm works a simple algorithm shell (shell indicates that the structure of the algorithm is described, but not precisely defined) will be described, after which a possible instantiation of that algorithm shell will be presented.
In this particular algorithm shell a common vector space has been defined to which all the objects to be clustered are transformed. The clusters are represented by hyperglobes in the common vector space. A hyperglobe is described in terms of its center, represented by a vector in the common vector space, called a prototype vector, and a parameter r indicating its radius, which is the same for all clusters.
The algorithm shell works as follows. Initially no objects have been clustered, thus no clusters exist yet. Then the first object is presented, which is transformed to the common vector space. As this is the first object it defines the center of the first cluster. Then a second object is

presented and transformed. Now a choice has to be made. Should this object be clustered as belonging to the first cluster, or should a new cluster be formed? This depends on whether or not the new object falls within the hyperglobe representing the first cluster. If it does, then the center of the first cluster will be moved a bit in the direction of the second object. Otherwise a new cluster is created with its center being defined by the second object. Let us assume the latter was the case. There are now two clusters. To determine how the third object should be clustered it is first necessary to determine which cluster is closest. If it is close enough, meaning that the object falls within the cluster's hyperglobe, the cluster is moved in the direction of the third object. Otherwise again a new cluster is created, its center being defined by the third object. This process continues for all further objects presented. The algorithm described by the above is the following given in figure 2.

Algorithm: Cluster Generic

Step 1 - Initialization
- Start with no clusters

Step 2 - Present and transform new object
- Let v := transformation (new object)

Step 3 - Find the closest cluster (if any exist)
- Let c := prototype vector of closest cluster (v)

Step 4 - Is the closest cluster close enough?
- If c is too far from v, or if there are no clusters yet, then create a new cluster with prototype vector equal to v; goto step 2

Step 5 - Update a matched cluster
- Update the matched cluster by moving it closer to v; goto step 2

Figure 2. Clustering algorithm shell

To obtain an actual algorithm it is necessary to create an instantiation of the algorithm shell by defining what an object is and specifying procedures for carrying out the transformation, finding the center of the closest cluster, determining if c is too far from v, and moving the matched cluster closer to v.

A possible instantiation of the clustering algorithm shell

As this paper is principally concerned with the clustering of textual documents, defining an object as a textual document is an obvious choice. It is then necessary to define what a textual document precisely is. In the rest of this paper a textual document will be considered to be a list of one or more lines, each line in its turn being composed of words separated by exactly one space (acting as separator symbol) and each word consisting of one or more lower-case letters of the Latin alphabet. We further restrict this simple class of textual documents by requiring the documents to be syntactical and semantically correct English non-fiction. The transformation is the in paragraph 2.1 discussed trigram analysis. To find the closest cluster and to determine if c is too far from v we require a distance metric. For distance metric we will use the euclidean metric. Moving the matched cluster closer to v will be accomplished by taking a linear combination of v and c. The algorithm is then as shown in figure 3.

Algorithm: Cluster Euclidean

Step 1 - Initialization
- Start with no cluster prototype vectors

Step 2 - Present and transform new object
- Let V = $(v_1, v_2, \ldots, v_n)$:= trigram analysis (new object)

Step 3 - Find the closest cluster (if any exist)
- Find the C = $(c_1, c_2, \ldots, c_n)$ to minimize d(P,I) = $\sqrt{\sum_{x=1}^{n} \left(C_x - V_x \right)^2}$

Step 4 - Is the closest cluster close enough?
- If d(C,V) > r, or if there are no cluster prototype vectors yet, then create a new cluster, with prototype vector equal to V; goto step 2

Step 5 - Update a matched cluster
- Let C := $(1 - \lambda) \cdot C + \lambda \cdot V$; goto step 2

Figure 3. Instantiation of Cluster Generic

The following example will illustrate the algorithm.

Example

In this example we consider a sequence of the following three objects:
object 1: {apples and cake}
object 2: {apples with cake}
object 3: {applesauce}

The trigram analysis of the objects results in the following vectors:

vector	ake	and	app	auc	cak	esa	ith	les	ple	ppl	sau	uce	wit
1	1	1	1	0	1	0	0	1	1	1	0	0	0
2	1	0	1	0	1	0	1	1	1	1	0	0	1
3	0	0	1	1	0	1	0	1	1	1	1	1	0

Note: All other trigrams are zero for all three vectors.

The clustering algorithm is executed with radius r=2.0 and adaptation parameter λ *=0.2.*

cycle	*prototype vectors*
start	-
object 1	*(1,1,1,0,1,0,0,1,1,1,0,0,0)*
object 2	*(1,0.8,1,0,1,0,0.2,1,1,1,0,0,0.2)*
object 3	*(1,0.8,1,0,1,0,0.2,1,1,1,0,0,0.2), (0,0,1,1,0,1,0,1,1,1,1,1,0)*

Figure 4. Clustering cycles

The cycle starts off with no clusters having been formed. Then the first object is presented. Its vector transformation defines the first prototype vector (see second row of figure 4) which represents the first cluster. Next the second object is presented. The distance between vectors 1 & 2 is $\sqrt{3}$*, which is smaller than the radius. Thus the second object is assigned to the first cluster which moves a bit in its direction (see third row of figure 4). Last the third object is presented. It does not fall within the first cluster, thus a second cluster is formed, its prototype vector being defined by the vector transformation of the third object. So now two clusters exist with the prototype vectors given in the last row of figure 4*

Paragraph 2.3 - Evolutionary computation

How to facilitate adaptive behavior?

In order to facilitate adaptive behavior we need parameters in our AIF system which can change dynamically in order to adapt to the evolving demands of the user. A system based on the clustering algorithm presented in paragraph 2.2 would have, for instance, at least two parameters, namely the radius of the clusters and a parameter indicating the degree to which clusters adapt to vectors being assigned to them (called λ in the algorithm of paragraph 2.2). A simple mechanism would involve providing the user with two options after each presentation of a document, one option to indicate satisfaction, one to indicate dissatisfaction. On the basis of the choice the user made, the parameters could be changed if necessary. For example, if the user was dissatisfied the change should be made in such a way that the chance of the previous presentation recurring is decreased. This does present the problem that it is necessary to know precisely how the parameters should be changed to improve the results. If there are many parameters this is often extremely difficult if not impossible. Evolutionary computation offers a way to solve such complex parameter optimization problems.

What is evolutionary computation?

Evolutionary computation is the process of finding (near) optimal solutions for computational problems using methods inspired by evolution theory. The general idea is that a population of potential solutions is maintained, this population evolves, and some members of the population are replaced based on some kind of selection process by new members which are the offspring of a selected group of members of the old population. Before discussing the various components of this process, an example will be given to illustrate how evolutionary computation works.

Example

Via the SETI (Search for Extra Terrestial Intelligence) project, NASA has come across some kind of radio signal from outer space. The problem is however that the frequency of the signal changes over time according to some kind of periodic function. The technician charged with modifying a radio to compensate for these changes is alas not at all good at math, but does manage to get the radio hooked up to a computer. Being very pleased with himself for getting it hooked up he decides to ask a friend of his, who happens to be a computer scientist, to program the computer in such a way that the radio will work as desired. His friend agrees to come and have a look.
After having studied the situation for a few minutes she says that there is an elegant way of solving it which will be easy to extend if the signal turns out to be much more complex than is now thought, and a simple way which is barely worth her time. The technician thinks for a moment and decides that an elegant extendable solution will surely impress his superiors much more than a simplistic one, so he asks her to go for the first option. After a couple of hours she calls him and says the program is ready but they will need lots of radios, all hooked up to the computer, if he thinks he can manage that. Well, you are a technician or you aren't, so he goes off and gets a hundred radios and hooks them all up to the computer. Then she starts the program and within an hour she announces that the problem has been solved. He is of course very pleased and asks her if she would explain how she did it. She explains that the periodic function they are looking for is of the form:
*frequency = a + b * sin (c * (time + d))*
with a, b, c, and d being the unknown parameters. The hundred radios start off at random frequencies, depending on the randomly selected values for a, b, c, and d stored for each radio separately in the computer. Every ten seconds the computer checks how clear the signal is for each radio. The five worst tuned radios have their values for a, b, c, and d replaced by slightly altered values of the five best tuned radios. Clearly the chances are good that these new combinations will be better as they are close to combinations already performing well. The slight alterations insure that new combinations are continually being tried. As the population of combinations evolves the radios get better and better tuned, until a (near) perfect solution is found. The technician has listened in amazement and declares her to be a genius. Before she leaves he asks her what the simple solution would have been. "Oh" she says, "because it is such a simple function we could of course have done a few measurements with one radio and simply calculated the right values. But this was much more fun, wasn't it?"

Besides demonstrating how evolutionary computation works, and showing that it is a bit of an overkill for problems which can be easily solved analytically, it is also worth mentioning that this example is also illustrative of how evolutionary computation can facilitate adaptive behavior. Imagine for instance that the periodic function of the radio signal very slowly changed in time in a completely random way. Then there would be no analytical function which could describe it, but the system of the hundred radios would have no problem adapting to the changes by following the same procedure as it did to fine tune initially.

Discussion of the components

Before the evolutionary process can commence, the initial members of the population have to be specified. This can be done either by using predetermined values, or completely at random (as in the radio example). The next step is to choose which members of the population will be replaced. This can be done at random, but applying Darwin's principal of "survival of the fittest" it is in general better to select the weakest members (or at least let the weakest members have a

greater chance of being selected). To determine which of the members of the population are the weakest a function is required called the fitness function (in the example this function returned higher fitness values for better tuned radios). Once the members to be replaced have been determined, the parents of the new members have to be selected. While this could be done at random, applying Darwin's principal of “mating of the fittest” it is in general better to select the fittest members (or at least let the fittest members have a greater chance of being selected). Selecting parents based on their fitness causes an effect called selective pressure. The stronger the preference for fitter parents is, the higher the selective pressure. The danger of too high a selective pressure is premature convergence in a local optimum due to loss of genetic diversity in the population. Note also that if both selecting the members to be replaced and selecting the parents of the new members, are done at random, the whole process boils down to a very complex way of performing simple random search, so this is obviously not advisable. Having selected the parents the next step is to create offspring. The simplest method is to take a copy of a parent and slightly mutate it. Another very popular method is to apply cross-over to two parents, creating a new member by taking some genes from the one parent and some from the other. This completes the circle and can continue until the perfect solution has been found or the process is terminated accepting the best found solution so far. There are a number of parameters influencing the process. One is the already mentioned selective pressure. Others include the population size, the number of members to be replaced each generation, the size of the mating pool, the mutation rate, etc. Finding the optimal values for these parameters is extremely difficult, one method is to include the parameters in the evolutionary process. For further information on Evolutionary Computation see for instance [Michalewicz94].

Chapter 3 - Constructing a simple AIF system

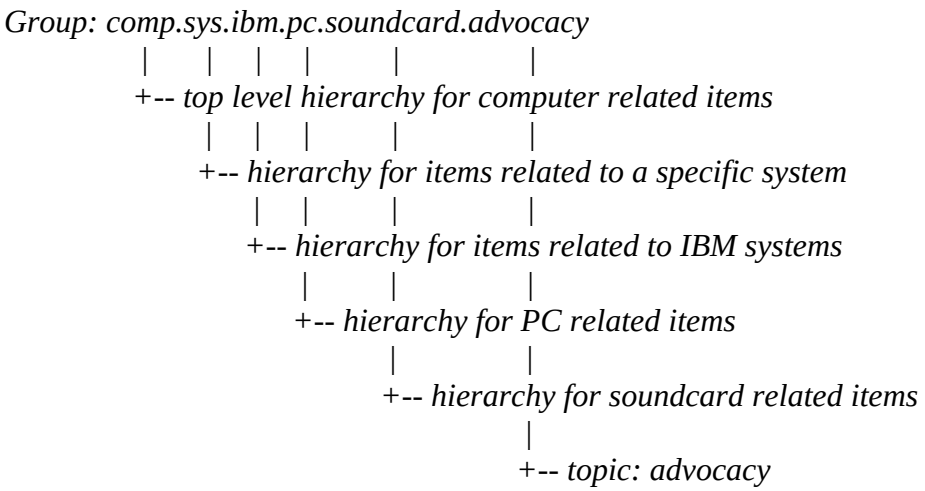
Synopsis

In chapter 1 we discussed what an AIF system consists of, in chapter 2 some methods were explained which could be employed in such a system. In this chapter we will try to bring it all together by describing how one could construct a simple AIF system to solve an existing information filtering need. The real world problem for which our software system will be constructed is the filtering of Internet news.

What is Internet news?

Internet news is a hierarchical system of discussion groups, each concerned with a specific topic. One can participate by sending a message to any particular group, either in response to an already posted message, or to introduce a new topic (which should of course be a sub-topic of the general topic of the group you are sending to). That message will then be distributed to all the computers in the world which are set up to receive that newsgroup.
In turn others then have the chance to react to the new message.

Example



So this group has as topic, advocacy for soundcards for IBM-compatible PC's.

There are a number of problems associated with Internet news. First of all one has to find the appropriate newsgroup. If your topic doesn't have a dedicated newsgroup it may be split over a large number of groups. Another problem is that the more popular groups have such a high traffic rate that finding the information you are interested in can be very time consuming. To deal with these problems the ideal system would be one which can rearrange messages according to the specific topic interests of the user, and filter out all non-relevant data. This is precisely what an AIF system attempts to do.

Interaction with the user

We will assume that the incoming data stream for our system consists of newsgroup messages (textual documents) retrieved from a particular set of Internet newsgroups. The user controls which groups the set consists of. At regular intervals the specified groups are polled and any new messages are retrieved. After retrieval they are clustered. To minimize the amount of effort required from the user, user interests are identified with certain clusters. The implication is that the clusters can be ranked according to how pertinent they are to the user. This can be inferred by counting the number of accesses per cluster. As the user peruses documents in various clusters, feedback to the system can be given by indicating per document approval or disapproval of it being assigned to that particular cluster. The user provides input to the system in three ways:

- Control of information sources
- Ranking of clusters (inferred from number of accesses per cluster)
- Approving and disapproving of the clustering of particular documents

Weighted trigram analysis

The clustering process is based on the algorithm described in paragraph 2.2 employing trigram analysis as described in paragraph 2.1. This combination does not, however, provide sufficient discriminating power to accurately cluster textual documents as shown in [Tauritz96]. In that same paper it was, on the other hand, demonstrated that weighted trigram analysis does have sufficient discriminating power. In weighted trigram analysis each trigram is assigned a weight. The discriminating power can be substantially increased by choosing the right weights. The

following example will demonstrate how weighted trigram analysis works.

Example

Suppose that the following three vectors were found:

vector	*aaa*	*bbb*	*ccc*
1	*1*	*0*	*1*
2	*1*	*1*	*0*
3	*0*	*1*	*1*

Note: All other trigrams are zero for all three vectors.

The distance between vectors 1 and 2 is the same as the distance between vectors 1 and 3. However, if vectors 1 and 2 represent objects which are semantically similar, and vector 3 represents an object which is semantically dissimilar to objects 1 and 3, it would be preferable if the distance between vectors 1 and 2 was smaller than the difference between vectors 1 and 3.

With weighted trigram analysis we can accomplish this by assigning a larger weight to a difference in frequency of trigram 'aaa' than to the other two trigrams. Step 3 of cluster euclidean then becomes:

Step 3 - Find the closest cluster prototype vector (if any)

- Find the $C = (C_1, C_2, ..., C_n)$ to minimize $d(C,V) = \sqrt{\sum_{x=1}^{n} \left(\left(C_x - V_x \right) \cdot W_x \right)^2}$

with $W = (W_1, W_2, ..., W_n)$ being the weight vector.

With weight 2 for 'aaa', and 1 for the other two trigrams, we find the following distances:
vectors 1 & 2: 1.4
vectors 1 & 3: 2.2
vectors 2 & 3: 2.2

Normalized trigram analysis

When comparing two documents of the same topic, one wants them to be considered similar enough to be grouped together, independent of the length of each document. However, using trigram analysis as described above, the results of the clustering process are not at all independent of the length of each document, as the following example will show.

Example

Suppose that the following three vectors were found:

vector	*aaa*	*bbb*	*ccc*
1	*1*	*0*	*1*
2	*3*	*0*	*3*
3	*0*	*1*	*0*

Note: All other trigrams are zero for all three vectors.

Assume that vectors 1 & 2 represent objects concerning topic A, and vector 3 represents an object concerning topic B. The Euclidean distances between the vectors are as follows:
distance vector 1 & 2: 4
distance vector 1 & 3: 1.7
distance vector 2 & 3: 4.4

While one would want the distance between vectors 1 and 2 to be smaller than the distances between vectors 1 and 3 and between vectors 2 and 3, this is clearly not the case. The problem is that we are measuring the difference in frequency, which is clearly length dependent, instead of the difference in frequency distribution. We can easily solve this problem by normalizing the vectors using the cumulative frequency as normalizer. This is done by dividing all the elements of the original vector by the normalizer. The results of normalization applied to the above example are as follows:

cumulative frequency vector 1: 2
cumulative frequency vector 2: 6
cumulative frequency vector 3: 1

normalized vector	aaa	bbb	ccc
1	0.5	0	0.5
2	0.5	0	0.5
3	0	1	0

distance normalized vector 1 & 2: 0
distance normalized vector 1 & 3: 1.2
distance normalized vector 2 & 3: 1.2

Now the distance between vectors 1 and 2 is smaller than the distances between vectors 1 and 3, and between vectors 2 and 3, as we would expect.

Implementing the adaptive behavior

Using normalized weighted trigram analysis we can implement the adaptive behavior through the use of evolutionary computation. We maintain a population of weight vectors. The user is presented with the results of clustering with the weights belonging to the fittest member of the population. Each time the user indicates approval of a particular clustering, the fitness of all weight vectors which support that clustering is increased. Likewise, each time the user indicates disapproval of a particular clustering, the fitness of all weight vectors which support that clustering is decreased. As the population evolves it should get better and better at clustering according to the user's wishes and, as mentioned in paragraph 2.3, the population will also adapt itself to the changing needs of the user.

References

Backer, Eric (1995) “Computer-assisted reasoning in cluster analysis”, Prentice Hall

Boyce, Bert R., Meadow, Charles T., Kraft, Donald H. (1994) “Measurement in Information Science, Academic Press

Michalewicz, Zbigniew (1994) “Genetic Algorithms + Data Structures = Evolution Programs”, 2nd extended edition, Springer-Verlag

Scholtes, Johannes Cornelius (1993) “Neural Networks in Natural Language Processing and Information Retrieval”, Ph.D. thesis at the University of Amsterdam, The Netherlands

Tauritz, D.R. (1996) “Optimization of the discriminatory power of a trigram based document clustering algorithm using evolutionary computation”, Internal Report 96-5, Department of Computer Science, Leiden University, The Netherlands,
FTP: “ftp.wi.LeidenUniv.nl/pub/CS/StudentReports/ir96-05.ps.gz”
WWW: “http://www.wi.leidenuniv.nl./~dtauritz/trigram.ps”