An Evolutionary Connectionist Approach to Personal Information Filtering

Michael Mc Elligott, Humphrey Sorensen
Computer Science Department,
University College,
Cork,
Ireland
email {mike, sorensen}@odyssey.ucc.ie

Abstract

With the increasing availability of information in electronic form, it becomes more important to have automatic filter methods so that potential recipients are not flooded with irrelevant details. In the following paper we present a design for a system to filter news articles on the Internet. The design consists of a two-layer connectionist model in which both connections and nodes are created dynamically. This model evolves as a user's specific interests evolve, so that it effectively acts as an intelligent user agent for information filtering.

1. INTRODUCTION

The problem of information overload is certainly not a new phenomenon. Ever since the invention of the written word, the moving picture, the daily newspaper, and so on, people have striven to extract from the sea of information around them that which is relevant to themselves. In response to this need, general *filtering* methods have evolved in society in the form of clubs, newspaper sections, film categories, etc. These mechanisms all serve a common purpose - to aid the end-user in sifting through a mass of information to find what is of interest to them. Information filtering has been defined as "a name used to describe a variety of processes involving the delivery of information to people who need it" [Belkin92].

The problem, however, is reaching alarming proportions for those people utilising perhaps one of the biggest and most far reaching information sources of the current day - the Internet. It has been estimated that traffic on the Internet is growing at a rate of about 9% per month. With each new user comes another potential information source. To alleviate the problem, some techniques have already been put in place such as news groups, mailing lists and specialist information sites.

To permit people with similar interests to communicate selectively with others around the world, the Usenet newsgroup hierarchy came into being. As successful as the hierarchy model is, it is far from a complete solution. In [Fischer91], the authors describe an empirical study they carried out with a collection of users. Their findings demonstrated that reading just a few newsgroups requires perusing megabytes of information each month. They attribute the major cause of the problem to the fact that different users perform different semantic classifications of articles. Where one user may judge a certain newsgroup as the appropriate place for the posting of an article, it is quite likely that other users will have different views of where it should go. Because of this, readers of newsgroups must locate articles relevant to their interests by "browsing an information space with semantic classifications of different message senders."

In this paper, we discuss our design of a model which, in common with [Fischer91], filters incoming articles from a range of newsgroups into a *virtual newsgroup* containing only those which have been judged to be of a significantly high interest to the user. In effect, we shall build a *user profile* based on a history of those articles previously deemed to be interesting by a user. This profile will then act as an *intelligent user agent* [Maes94] for the filtering of subsequent articles, while adapting to user feedback regarding its success in filtering those articles.

2. PRINCIPLES

2.1 Information Retrieval and Information Filtering

While distinct differences exist between *information retrieval (IR)* and *information filtering (IF)* - for example, the former is generally driven by a specific query while the latter is based on a history of previously identified relevant documents - it has been noted that many techniques of IR can be applied within the domain of IF [Belkin92]. One of the primary principles of IR which is relevant here is the *word frequency model* for indexing articles, based on the observation that "the frequency of word occurrence in an article furnishes a useful measurement of word significance" [Luhn58]. The reasoning here is based on the fact that writers tend to repeat certain words as they advance and elaborate their argument on some subject. The basic strategy for indexing words (i.e. choosing them as keywords for an article) by frequency is to first count the frequency with which each word in the document occurs. Then (i) the very low frequency words can be excluded as being not much on the writer's mind and (ii) the very high frequency words can be discarded on the grounds that they are almost always insignificant [Forsyth86].

Although simplistic, this work led to the birth of a large collection of IR approaches based on applying statistical modelling techniques to textual documents. One such approach is the *vector space model,* of which the SMART system [Salton83(a)] is perhaps one of the most famous representatives. In the SMART system, each record or document is represented by a vector of terms. A given document collection may then be represented by a matrix of terms where each row represents a document and each column represents the assignment of a particular term to the documents. Queries are also represented as vectors and retrieval is carried out through a process of measuring the similarities between document and query vectors.

We believe the vector processing model has much to offer. It requires minimal user intervention and it is adaptive and robust. One big disadvantage we would see with this approach, however, is that those documents which the user terms interesting and those they don't may not (in fact, most likely will not) be linearly separable. Terms which may be relevant in one context may not be in another. This is due to the inherent lack of structure in each vector representation - terms are not recognised as having any connection to each other. Salton himself acknowledges this shortcoming and in [Salton83(b)] attempts to overcome it and allows the user to impose Boolean logic type structure on their query. Other basic modifications have also occurred.

Our work was inspired, in part, by the vector processing model in that we have adopted an approach rooted in the statistical methodology of extracting weighted index terms to build representations of documents which are then compared for similarities. Unlike the vector processing model, however, a document (or profile) representative does not consist of a vector of distinct (and, for all intents and purposes, unrelated) index terms but rather comprises a complex collection of weighted and interconnected terms based on a more stringent analysis of the text. The underlying tenet of our approach is that more consideration needs to be given to the actual relationship between index terms. This arises from the basic fact that natural language does not consist of a vast collection of scattered and unrelated words that can be woven together without regard for order. It is essentially ordered and it is in this order that semantics generally manifests itself. Each word, in part, derives its meaning from the words around it and it is through this process of phrase and sentence formation that the meaning of individual words becomes clear. It is often impossible to derive the correct meaning and intention of a word by looking at it in isolation - yet this is the basic mechanism that many of the popular IR systems of today use (the most common example would be the straightforward keyword search of an inverted file of documents). Our goal, therefore, was to integrate into a document/profile representation not only keywords but also the context in which they occurred. This, we believe, allows us to compare documents on a deeper and more accurate level.

To attempt to capture and integrate this information, we looked to the works and theories of some of the leading exponents of what have come to be known as *dynamic* or *episodic memory architectures*.

2.2 Episodic Memory

In [Kolodner84], Kolodner explains episodic memory as follows:

> *"The episodic view of memory claims that memory is organised around personal experiences or episodes, and not around abstract semantic categories. In such a memory, repeated or similar sequences of events are recognised as similar and used to build up generalised knowledge about episodes. This generalised knowledge can later be used for understanding. Thus, every person's episodic memory will be different from every other person's."*

We propose an architecture that learns from its experience (or history) of interesting news articles how to deal with future articles coming through the news server. Before we discuss this architecture, two key concepts need to be introduced - scripts and K-lines.

2.2.1 Schank's Scripts

Schank describes a *script* as a "standard event sequence" [Schank77]. He argues that each one of us possesses a multitude of scripts 'inside our heads', so to speak, which guide us in our actions in stereotypical situations. These scripts are said to form from repeated occurrences of events during a given scenario. When we are faced with a typical everyday situation we simply follow a script that has formed in our head from previous repeated dealings with this situation. When following a script, and something occurs which was not accounted for in the script, an "expectation failure" [Schank82] occurs. An index is created in the script at the point that the abnormal condition occurred so that, the next time that it occurs, it will tend to be less unexpected and will 'drift' closer towards the script - perhaps even causing the rebuilding of the script if it is found to occur with sufficient frequency.

When learning, our model treats each word in an article as an event. Each time an event is encountered, the context in which it was met is noted and compared to previous encounters. It is through this process of selection and comparison that events eventually organise themselves into script-like structures. The result of this process is the conversion of a news article into a multitude of scripts based on word co-occurrence frequencies. These scripts are then analysed for reliability and the best serve to guide the model during the filtering stage in selecting those articles which compare favourably to our experience gathered in the learning phase.

2.2.2 Minds and Minsky

In "The Society Of Mind" [Minsky87], Minsky proposed that the mind was a vast society of interacting agents which co-operated and competed with each other to produce intelligent behaviour. To co-ordinate these agents, he put forward a scheme not altogether different from the way events are organised by Schank's scripts. He proposed a theory of *K-lines* to govern memory and learning. These are described thus:

"Whenever you 'get a good idea', solve a problem, or have a memorable experience, you activate a K-line to 'represent' it. A K-line is a wirelike structure that attaches itself to whichever mental agents are active when you solve a problem or have a good idea.
When you activate that K-line later, the agents attached to it are aroused, putting you into a 'mental state' much like the one you were in when you solved that problem or got that idea. This should make it relatively easy for you to solve new, similar problems!"

We have used this notion of K-lines to facilitate the formation of a collection of what we will term *supervisors*. Recall how we said in the previous section that, in the learning phase, each article is converted into a multitude of scripts of varying reliability. We now look on these scripts as memorable experiences. The best of these experiences merit the formation of structures analogous to K-lines (supervisors). These supervisors, between them, make up the article's representation. They are a collection of phrases taken from the article which have been judged to be of potential use in the future categorisation of articles - and, hence, will migrate into the user's profile. These supervisors could be said to work in a reverse manner to K-lines. Instead of it activating a 'mental state' similar to the one we were in when we got the good idea, it is now the re-occurrence of this previous 'mental state' (or, more accurately, the previous good sequence of words) which activates the supervisor. Individual words do not now contribute to the 'interestingness' rating of an article - only suitably activated supervisors may do this.

3 THE MODEL

The system may be viewed as operating in two modes. Initially, there is a *learning* phase, where a collection of documents of interest to the user is presented to the system. Then there is the *comparison* phase, where documents arriving via the news service are filtered to the user. If these documents are considered relevant by the user, then the system again enters learning mode so as to improve its filtering ability.

The system thus consists of a two-layer network. The first (or bottom) layer's purpose is to receive an article as input and build a representation of it using script structures formed from statistical analysis of word combination frequencies. This layer is then analysed, resulting in the formation of a second (or upper) layer of supervisor nodes which monitor activity in the layer beneath when in the comparison mode. In the following sections we discuss the architectures of these layers and how they interact with each other to rate an article. For brevity purposes, we examine here the handling of two-word sequences. The extension to multi-word sequences is straightforward.

3.1 Layer 1 (The Input Layer)

Figure 1 depicts a simplified structure of an input node. Each node has a *name* (each individual word in an article is assigned a node). *Frequency* is the number of times that the word has occurred thus far in the present article. Each node has a number of *associates* (outgoing links) and *supportees* (incoming links). Each node is initially provided with S *strength* points (S is a system-wide constant currently set to 90).

In traditional connectionist systems, learning at the local level (that is to say, individual connection strengths) is a direct consequence of the *immediately previous* activation of the link's originating nodes. Learning at the global level could be said to occur through the sum of the local interactions. Although this process is responsible for one of the widely acclaimed strengths of connectionist systems - that of adaptability to changing inputs - it does so at the cost of what could be termed 'a long-term memory.' That is to say, new experiences play a more important role in the shaping of the model than do old experiences. While often this is desirable, this was not the case in our own system. One of the very corner-stones of our approach was the principle of episodic memory - of learning new lessons in the context of what has gone before. To this end, we decided to provide each node with a 'memory' which manifests itself in both a separate weighting scheme and a separate learning algorithm.

NAME		
Frequency		
Surplus_strength		
Num_associates		
Num_supportees		
Neighbour	Strength	Weight
Neighbour	Strength	Weight
Neighbour	Strength	Weight
.	.	.
.	.	.
.	.	.
.	.	.
Neighbour	Strength	Weight

Figure 1 - Structure of Input Node

A conventional weighting scheme using constrained Hebbian learning is employed to capture the preference of one node's link with another relative to its other outgoing links. This serves to provide a measure of reliability or associativeness between two nodes. In script terms, it provides a measure of the probability that given an event (in this case, a word), this other event (another word) will follow it. In our model, the weight between two nodes must be either equal to or greater than the average outgoing link weight of the source node to be considered as a reliable sequence of events (i.e. a script).

A further weighting scheme is employed on the links which again uses Hebbian learning. This time however it uses two variants of it - constrained and unconstrained. Initially, unconstrained learning is employed. Each node has S strength points which it shares out to its outgoing links as they are formed and reinforced according to their frequency of occurrence. Each time a link is used it receives s strength points from its parent. When the parent has used up its allotted share, then the links must compete with themselves for each other's strength. It is at this point that contsrained Hebbian learning comes into action. Any links that fall below the strength threshold (st) are removed and their strength returned to the parent node.

While the first weighting scheme measured link importance (or reliability) at the *local* level (from this point on, we will refer to these link values as the **weights**), the effect of this second weighting scheme is to provide a *global* measure of importance of a link relative to any other link strength in the layer (to be known from now on as the **strengths**).

3.1.1 Link Adjustment Strategies

Let us first of all consider the weights. As mentioned previously, weights are adjusted according to constrained Hebbian learning

$$W_{ij} = \frac{s}{T}(X_j Y_i)$$

- where W_{ij} is the weight on the link going from node i to node j, Xj and Y_i are the activation values of nodes j and i respectively (can be either 0 or 1), T is the total strength currently allocated by node i to its outgoing links, and finally, s (as mentioned previously) is a system defined constant denoting the increase in strength points allocated to a link that is activated in the learning phase.

We will now look at the strengths. Starting off, link strengths are updated according to simple unconstrained Hebbian learning, thus:

$$\Delta S_{ij} = +s(X_i Y_j)$$

Soon, however, the strength that the parent node is capable of donating to its outgoing links may be totally depleted - it being used up by already existing links. When this occurs, constrained Hebbian learning comes into action and links requiring extra strength drain it from the other links. Each other link is drained of an amount inversely proportional to its current weight. That is to say, those with the larger link weights get drained of less strength. This reinforces the 'survival-of-the-fittest' methodology underlying the notion of scripts.

$$\Delta S_{ij} = -\frac{s}{(1-Wij)}(X_i Y_j)$$

By imposing a link survival threshold (st), we have a simple and neat method of allowing good relations to be remembered (or strengthened) while forgetting (or weakening) bad ones. It also has a hand in conserving activity in the layer by preventing overly common words (such as prepositions) from amassing large numbers of potentially irrelevant and highly weighted links

3.2 Layer 2 (the Output Layer)

Each script formed in the input layer is said to be formed of two children - a left (A) and a right (B). For each suitable script formed in this layer (i.e. has sufficient weight and strength), there exists a supervisor node whose purpose is to monitor for the presence of this script in each article under examination. If and when the script is detected (i.e. the two words appear successively in an article), the appropriate supervisor is activated and is permitted to make a contribution (via its output value) to the interestingness rating of the article.

In Figure 2, we see the structure of a supervisor node. Each has a left and a right child which it monitors for occurring in sequence in an article. Each has an individual contribution that it makes when it is activated. A note is made when a supervisor has been activated in order to make sure that the same supervisor does not make multiple contributions during the course of rating an article. Supervisors may have links with other supervisors in the second layer. In common with strength values in the first layer, link strengths here are again allotted using a combination of both constrained and unconstrained Hebbian learning. This time, however, there is not a system wide constant strength value which is allotted on link occurrence. Here, the occurrence of a link merits a percentage of the parent's assigned contribution.

Links are formed between supervisors when it is found that two supervisors occur in sequence in the learning phase. For example, one supervisor may have been created to monitor for the term "object oriented" while another may have been created to monitor for "oriented programming." After all the supervisors have been created from the analysis of an article, a second pass is made to determine if any of these supervisors occurred in sequence. For those, if any, that did a link is formed going from the left supervisor to the right one. This link is given extra strength for each time it occurred to the detriment of the parent node who enters a process of the gradual weakening of its importance

In essence, what is occurring in the layer is a gradual imposition of structure on the supervisors. We will have the case where some supervisors contribute significantly on their own (e.g. perhaps the supervisor for "computer virus" for a person interested in articles on computer viruses in general) while others only contribute significantly when activated as part of a team (e.g. consider a person interested in articles on "natural language processing" specifically. The situation could quite plausibly arise here that the supervisors for "natural language" and for "language processing" may only contribute when the joint satisfying phrase "natural language processing" is encountered). Through a recursive process, this technique could be used to monitor for a phrase of any length to capture specific items such as organisation names, application names, etc.

Left child		
Right child		
Contribution		
Activated		
Destination	Strength	Active
Destination	Strength	Active
.	.	.
.	.	.
.	.	.
Destination	Strength	Active

Figure 2 - Structure of Output Node

4 THE MODEL IN ACTION

To get an idea of how the model builds up a representation of an article, consider the following simplified article presented as interesting by the user. Capital letters represent individual words.

A C F H Y B C D E A F H C D

Figure 3 - Simple Article

Shown in Figure 4 is the resulting network which would form from the analysis of this article. Only two supervisors have been created. This is because only two of the scripts formed succeeded in satisfying the requirements of a supervisor formation - namely, a link weight either equal to or greater than the link weight average (at the local level) and a link strength above the link strength average (at the global level).

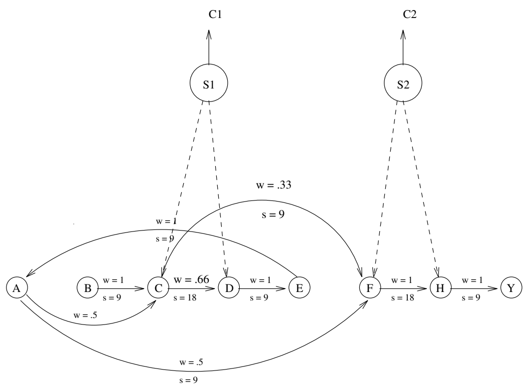

Figure 4 - Sample Model Structure

As an example of the script formation process in action here, consider node C. In figure 5 we see that the likelihood of D occurring (revealed by the link weight) after C is very high (.13 above average chance) compared to the chance of F. In other words, more often than not, if we come across C, we should expect to find D following it as the next event. If, instead, we come across F then we find that this possibility was already there as an "expectation failure" [Schank82] and are 'reminded' of this previous episode. In the reminding process the link between C and F becomes stronger (to the detriment of that between C and D). This could possibly result in C -> F becoming strong enough (equal to or greater than the average weight) to merit the formation of a second script or indeed to go on and become strong enough (by occurring sufficiently regularly) to monopolise most of the weight and thus demoting the previous script C -> D to a mere expectation failure while C -> F enjoys the privilege of being the only available script when C occurs.

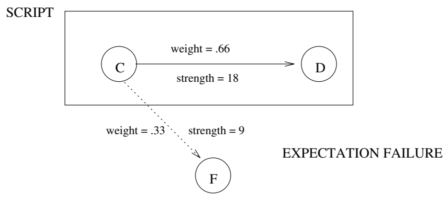

Figure 5 - Script Formation

Now that we have discussed the script formation process, let us turn our attention to the supervisors. For a supervisor to be formed two conditions must hold:
(i) must have a link weight above average and
(ii) must have a link strength above average.
When these conditions are satisfied then a node is created in the output layer to monitor for the occurrence of this link. Each such supervisor is allotted a 'contribution ration' and the total rations allotted to all the supervisors adds up to 100. Rations are assigned according to the following formula:

$$C_i = w.s.F$$

- where Ci is the contribution made by supervisor i, w and s are the weight and strength, respectively, of the script that the supervisor was formed in response to, F may take one of three values depending on the two individual frequencies of the words involved in the script:

i. F = 3 if both frequencies are between the two cut-off points [Luhn58];

ii. F = 2 if only one of the frequencies is between the limits;

iii. F = 1 if they are both outside the limits.

Finally, when all the contributions have been calculated, they are normalised so that they will all add up to 100. This means that any article presented to the model for filtering will be rated out of a 100. This enables us to now *rate* articles and allow the users to choose their own cut-off thresholds depending on the time available to them and so on.

5 TUNING BY EVOLUTION

Due to the differing needs and characteristics of the user community, it is difficult to provide a model general enough to suit them all without sacrificing accuracy. It needs to be taken into account that while one user may have a very specific interest such as, for example, all articles on the "stoned computer virus" and no other virus, there will no doubt be another user who has quite a general interest such as "any and all computer viruses." This is not a problem that can be solved solely by the presentation of more general or specific articles as the case may be. If not only for the reason that the 'perfect' training article is rarely readily available, there are other factors inherent in the model itself which could play a crucial role in its accuracy across a varying community. System constants such as minimum link and supervisor formation thresholds, the number of links that an agent or supervisor may have, the strength allocation on link occurrence, and so on, all have an input. All in all, about 5 - 10 variables exist which need to be pre-set by the programmer.

The effect of these variables manifests itself in the *recall* and *precision* (see [Malone90] for a good description of these terms) of the model - basically how specific or general it is going to be. Ideally, each user could set these parameters in accordance with their needs but, clearly, this would require expert knowledge on the part of the user. The solution we propose is both transparent to the user and easily adaptive to different needs - a genetic algorithm.

By employing a simple genetic algorithm to 'custom-tune' each model to its respective user, we aim to shift the need for expert knowledge from both the programmer and the user to the model itself.

6 FINAL COMMENTS

In the course of scanning news articles on the Internet, it is quite common to come across misspellings and bad grammer due to the large community of multinational users. Some means of dealing with these nuisances is required in order to prevent unnecessary words being allowed to consume nodes in the bottom layer. To this end, we have introduced the heuristic that any agent which no longer has any supporters (i.e. incoming links) and only has a single supportee (outgoing link) of negligible strength is to be discarded. Although this technique will be slow in resolving of the afore mentioned problems, the delay is essential to make sure that no legitimate word and word phrases are removed due to temporary inactivity in the learning phase.

REFERENCES

[Belkin92] Belkin, N.J., Croft, W.B., *Information Retrieval and Filtering: Two Sides of the Same Coin*, Comm. ACM, 35, 12, Dec 1992.

[Fischer91] Fischer, G., Stephens, C., I*nformation Access in Complex, Poorly-Structured Information Space*, Human Factors in Computing Systems, CHI'91.

[Forsyth86] Forsyth, R, Rada, R., *Machine Learning: Applications in Expert Systems and Information Retrieval*, Ellis Horwood, 1986.

[Kolodner84] Kolodner, J., *Retrieval and Organisational Strategies in Conceptual Memory: A Computer Model*, Lawrence Erlbaum, 1984.

[Luhn58] Luhn, H.P., *The Automatic Creation of Literature Abstracts*, IBM Jour. Res. Dev., 2, 2, 1958.

[Maes94] Maes, P., *Agents that Reduce Work and Information Overload*, Comm. ACM, 37, 7, July 1994,.

[Malone90] Malone, L.C., Wildman-Pepe, J., Driscoll, J.R., Evaluation of an Automated Keyword System, Microcomputers for Information Management, 7, 2, June 1990.

[Minsky87] Minsky, M. *The Society of Mind,* Simon and Schuster, 1987.

[Salton83(a)] Salton, G., McGill, M.J., *Introduction to Modern Information Retrieval*, McGraw Hill International, 1983.

[Salton83(b)] Salton, G., Fox, E.A., Wu, H., *Extended Boolean Information Retrieval*, Comm ACM, 26, 12, Dec 1983.

[Schank77] Schank, R., Abelson, H., *Scripts, Plans, Goals and Understanding*, Lawrece Erlbaum, 1977.

[Schank82] Schank, R., *Dynamic Memory*, Cambridge Univ. Press, 1982.