Applying connectionist models to information retrieval

Sally Jo Cunningham, Geoffrey Holmes, Jamie Littin, Russell Beale, and Ian H. Witten

Department of Computer Science
The University of Waikato
Hamilton, New Zealand

ABSTRACT: Adaptive information retrieval (IR) systems based on connectionist architectures have captured the attention of researchers over the past decade. This paper provides a review of connectionist IR research, including the major models for connectionist document and query representation, techniques to enhance query re-formulation, dynamic document routing (information filtering), and connectionist techniques for document clustering.

1. Introduction

Information retrieval (IR) systems provide populations of users with access to a large collection of stored information. These systems are concerned with the structure, analysis, organization, storage, and searching of such information. Collections have typically comprised text documents but are increasingly involving new data types such as pictures, audio, video, and so on. The main goal of an IR system is to extract from the files of documents those items that most closely correspond to requests from the user population. These systems are becoming increasingly important with the growing number of documents that are now available in machine readable form and the consequential advent of digital libraries.

Many of the processes involved in information retrieval and the characteristics of the data allow much of the work to be automated. Document indexing, document classification and content analysis were originally achieved manually by subject experts or trained indexers who would assign content identifiers to information items such as keywords or index terms (the content-rich portions of a document), retrieval being achieved by matching query terms with these information items and returning appropriate content identifiers.

Human involvement in the development and maintenance of such collections is costly, time-consuming and error-prone. Machine readability coupled with the processing power of modern computers has now enabled texts to be fully-indexed. Thus, the manual processes can be replaced with intelligent methods for many of the IR processes. For example, document clusters that maximise the within cluster similarity while also maximising the separation from other clusters, can be formed based on content identifiers. The same metric for forming the clusters can then be used to match queries to clusters in retrieval operations.

The data that can be automatically generated from a document has many characteristics that make it suitable for processing with a neural network. Neural networks receive input patterns comprised of a number of features represented by attributes and their associated values. In many neural network applications, however, there are relatively few features, so that the domain does not offer the richness and complexity in the input space that a neural network is particularly well-suited to represent. This is not the case in IR as each document represents an input pattern, of which there are thousands in any meaningful collection, and the number of natural language features that can be extracted from a document is very large. In fact, most applications have to limit the number of features or find ways of determining the best features for a particular application before the documents are processed by the network.

There are opportunities for neural network applications in all of the different processes of information retrieval. Query formulation is of particular interest because associative memories can make the most of queries that do not match directly to the terms that have been used to train the network. Over time, these queries help to re-train the network increasing precision.

Inference networks that rank documents based on the probability that the documents satisfy the user's information need have also been used to great effect. The architecture of the inference network can be used to rank queries across different collections in a principled manner. The network provides a framework for introducing a meta-network which combines the collections and provides a mechanism for merging retrieval results when the distribution of terms in the different collections is different. Relevance feedback, a technique for refining queries by automatically expanding queries and requesting feedback on the results has also been implemented in an inference network model.

Inference networks have been used for relevance feedback as they permit the computation of overall relevance from multiple sources of relevance such as human experts who provide relevance rankings for each document in a collection. These networks have also been used to filter unwanted documents in situations where the document collection is changing extremely quickly but the user's needs remain constant, for example, when reading information in a news group.

Document clustering is often the key to good performance in an IR system, especially when there is no established text classification scheme for a given collection. The aim is to cluster documents so that those documents that are

likely to be relevant to particular queries are in the same clusters. The hypothesis is that closely related documents are relevant to the same query. Self-Organising Maps and the Adaptive Resonance Theory (ART) model have been used for this purpose and have proven successful for not only clustering related documents but also in enhancing the browsing process by scanning the map, filtering items of interest by concentrating on regions in the map that have newly acquired nodes, and examining the distribution of topics in a body of literature. The maps do tend to associate semantically related nodes into concept areas, and afford library scientists views of emerging subjects in a discipline.

In the next section we review the various neural network architectures that have been applied to information retrieval, beginning with the most common representation format for documents. We then look at individual aspects of the IR process and their solution as neural networks; in particular we review work on query expansion (Section 3), filtering (Section 4), and document clustering (Section 5).

2. System architectures

For a retrieval system to categorize documents effectively and distinguish relevant documents during query processing, it must first extract enough information from the documents to discriminate between them. The most common representation scheme is the vector model, discussed below. This venerable document description technique is the basis for most conventional, and unconventional, IR systems—including the two connectionist retrieval architectures that have seen the widest application to retrieval problems, based on three level neural networks and inference networks.

2.1 Term vector models

The basis for the "bag of words," or *vector*, model, is the representation of a document by a set of *terms* (Salton and McGill, 1983). All documents that can be generated from a given set of terms form an n-dimensional space, where n is the number of terms. Terms are usually words, and a term vector is either a Boolean vector representing the set of words that appear in the document, or a numeric vector whose values are derived from the number of occurrences of each term in a document—the latter being based on the commonsense understanding that the more often a term is mentioned, the more likely it is to be central to the subject matter. All other information implicit in the text, such as the meaning and order of words, is lost once the document's vector representation has been computed. However, the simplicity of this model makes it one of the most popular in IR.

When document vectors reflect the frequencies with which terms appear, documents are considered similar if their term vectors are close together in the vector space. Before determining distance, the dimensions of the space should

be normalized in a way that reflects the differing importance of different words. Importance is generally measured simply on the basis of word frequency, rare words being more salient than common ones. Each term is weighted by the number of documents in which it appears, so that a single appearance of *the* counts far less than a single appearance of, say, *Jezebel*. The components of the term vector are not simply term frequencies, but term frequencies divided by the number of documents in which that term appears. This is called *term-frequency times inverse document frequency weighting*, or *tfxidf*. The standard method for determining the distance between vectors is the *cosine measure*:

$$\text{cosine(Q,Dd)} = \frac{Q \cdot D_d}{|Q||D_d|}$$

where Q is the term vector for the query and D_d is the term vector for the document *d*. This can be calculated using the formula:

$$\text{cosine(Q,Dd)} = \frac{1}{W_q W_d} \sum_{t=1}^{n} w_{q,t} \cdot w_{d,t}$$

where W_d is the Euclidean length of document *d*, and $w_{d,t}$ is the weight of term *t* in document *d*. The cosine measure calculates the cosine of the angle between two normalised vectors. A value near 1 indicates that two documents are similar, whereas a value close to 0 indicates two documents are very dissimilar, the angle between them approaching 90 degrees. The continuous nature of the function means that documents can be ranked in order of similarity to the query vector.

The effectiveness of a retrieval system is commonly measured by the *precision* and *recall* of queries run against the system, where precision measures the proportion of relevant documents present in the retrieval set for a query, and recall measures the proportion of relevant documents in the collection that were retrieved by the search (Salton and McGill, 1983).

2.2 Artificial neural network models

Associative retrieval takes advantage of the connectionist model to retrieve relevant documents that may not have many (or any!) terms in common with the user's query (Crouch *et al*., 1994). A number of prototype neural network based IR systems have been tested, with most based on the model depicted in Figure 1 (Belew, 1989; Crouch *et al*., 1994; Jennings and Higuchi, 1992; Kwok, 1989, 1991; Pannu and Sycara, 1996; Wilkinson and Hingston, 1991, 1992; Wong and Cai, 1993). In this architecture, the user's query and collection documents are given a representation similar to that of the vector model: document terms are mapped to nodes in the network, with a link between a term node and a document node indicating that the term appears in the document. Weights may

be attached to the term/document links, to represent the simple frequency of occurrence of the term in the document or the term's *tfxidf* weighting.

Activation spreads from the user's query vector, via a term layer containing nodes representing all terms contained in all document descriptions, to a document layer. At this point, the activation level of a document node can be construed as indicating the relevance of that document to the query. Activation then spreads backward to the term layer, reinforcing query terms and adding activation to new terms in relevant documents, and then forwards again to the document layer, where additional documents may be activated by the new terms. At this point the cycle is halted and the documents are ranked for retrieval according to their activation level. This stopping point is somewhat arbitrary, based on practical experimentation indicating that additional cycles through the network tend to randomize the document ranking (Wilkinson and Hingston, 1992). Tests on actual text collections indicate that associative retrieval can increase precision, but may decrease retrieval performance for queries that have few or no relevant documents in the collection. In this latter case, the query terms will be augmented by terms from irrelevant documents, and these new terms will in turn activate still more irrelevant documents.

One barrier to the creation of full scale connectionist IR systems is the sheer size of the neural network needed to represent the term/document linkages. A moderately sized document collection can easily run to hundreds of thousands of connections, and the network complexity is vastly increased in systems including lateral inhibition (for example, Wilkinson and Hingston, 1992). Efforts to reduce system complexity have focussed on minimizing the number of terms used in the document representation, primarily through term stemming eliminating non-content "stop" words, and removing terms that appear infrequently in the collection. For large collections (in the millions of documents), these techniques are inadequate. The dimensionality of an IR system can be further reduced (in a principled manner) by applying Latent Semantic Indexing (LSI). LSI uses a factor analysis-like approach to reduce the large and sparse term/document matrix into three relatively small, non-sparse matrices (Deerwester, *et al.*, 1990). Results from a trial application of LSI in a neural information retrieval system were promising (Weiner *et al.*, 1995); in the worst case the LSI representation may slightly decrease search performance, while greatly reducing the size of the connectionist architecture underlying the IR system.

2.3 Probabilistic activation spreading

Inference networks (Turtle and Croft, 1990, 1991; Tzeras and Hartmann, 1993; Haines and Croft, 1993) rank documents based on the probability that the documents satisfy the user's information need. Figure 2 shows the structure of an inference network used for information retrieval. Although inference networks were not specifically developed for IR, this structure indicates the information necessary for applying the technique to this domain. The D nodes

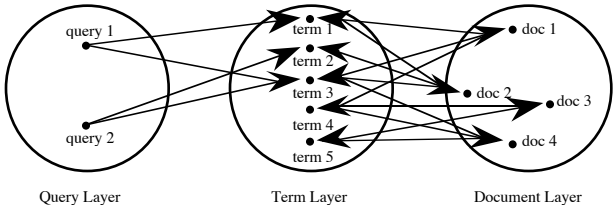


Figure 1. Model Architecture with Document-Term Links
(from Crouch, *et al*, 1994, p. 196).

represent documents in the system's database, the R nodes describe the contents of the documents—documents with similar content are linked to the same R nodes, the Q nodes represent queries, and the I node is the user's information need. The D and R nodes are constant for a set of documents reflecting relationships between the documents. The Q and I nodes are created for each query—in this case (information and retrieval) and (not files). Probabilities filter down the network from each of the document nodes at the top, and the value that arrives at the I node indicates the relevance of each document to the user's information need. The probabilities are generally based on Bayesian, rather than Dempster-Shafer, inference models.

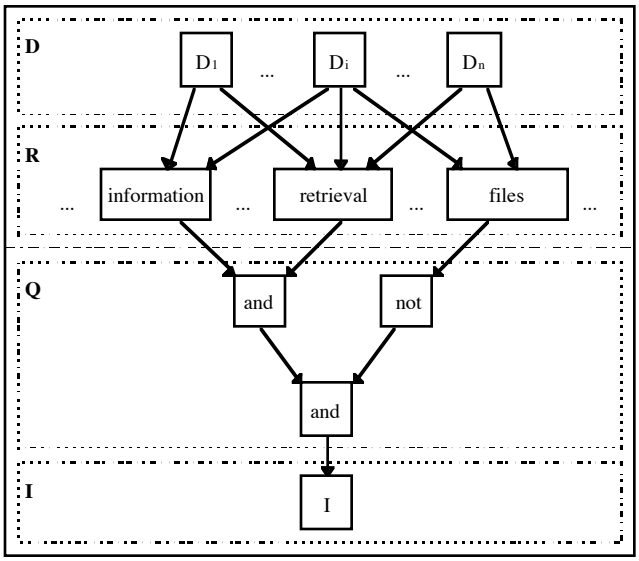


Figure 2 An example inference network created from the query *(information and retrieval) and (not files).*

The inference network architecture can also be used to address the collection fusion problem (Callan *et al.*, 1995), which arises when a single query is run against several separate collections. Problems arise in merging the retrieved documents in a valid order, since individual collections will no doubt have different term distributions—which in turn means that relevance scores from the different collections cannot be directly compared. In the inference network model, this problem is overcome by constructing a meta-network in which leaves represent entire collections, rather than individual documents. The user's query is processed against the network to retrieve a ranked list of collections, and the most highly ranked collections are then searched.

Performance evaluations of inference network-based systems have been promising. Turtle and Croft (1991) compare the precision and recall achieved by a Baysian inference network IR system to that of conventional ranked and Boolean systems over two standard testbed collections: the CACM (3204 documents and 50 queries) and CISI (1460 documents and 35 queries). The inference networks were found to achieve a statistically significant performance advantage, over both collections. Tzeras and Hartmann (1993) estimate classification quality for a Baysian inference network by measuring the consistency between index terms produced by the network and a "gold standard" of terms manually assigned by human classifiers; tests over two 1000 document test sets indicate that the inference network performs comparably, although not better than, the AIR/X system (based on a "least squares polynomial" approach to matching document terms to pre-existing descriptors occurring in the system dictionary). Given the small size of the test collections, however, these results must be interpreted cautiously. Further, efficiency remains problematic for the inference network approach, since the algorithms have an exponential time and space complexity (Tzeras and Hartmann, 1993). It is not clear how well this technique will scale up to handle real world collections, or how the architecture can be modified to ease its currently intensive computational requirements.

3. Query refinement

Rarely does a user's initial query return a precisely focussed and comprehensive set of documents matching the user's information need. Generally, locating documents of interest is an iterative process: the user issues an initial (often very general) query, then successively refines and modifies the query until the documents returned are satisfactory. The query refinement process can be difficult and frustrating, since the user may have already stated their requirements to the best of their ability. In essence, the user must try to guess the additional query terms that the system uses to express the concepts intended by the user's original query. Connectionist IR systems have explored two types of support for query refinement: relevance feedback, which bases the updated

query on a user's judgements of the relevance of documents retrieved by earlier versions of the query; and thesaurus building and consultation techniques.

3.1 Relevance Feedback

The relevance feedback technique supports the query refinement process by automatically expanding queries. Significant parts are extracted from the documents retrieved by the user's query, and the user is asked to indicate each part's relevance to their requirements (Salton and McGill, 1983). The user may be presented with whole documents, abstracts, selected passages (Allan, 1995), keywords, or other terms that the system deems representative of the results of the initial query. These items are usually ranked, and their number limited to reduce the risk of including worthless terms. The selected terms are then added to the initial query, existing terms are reweighted according to their performance in the previous search, and the query is processed again. This procedure is repeated until the user is satisfied with the documents returned by the system.

Haines and Croft (1993) describe extensions to an inference network model of information retrieval to include relevance feedback techniques. They investigated several variables pertaining to relevance feedback, such as term reweighting, additional terms, and new term weighting methods.

In the inference network model, queries are represented as links between the query nodes (*Q* nodes of Figure 2), and the information need node (*I* node). Query term weights are determined by the *weighted sum* form of the link matrix at the *I* node. To incorporate terms determined by relevance feedback, new links are added between the *I* node and the new *Q* nodes created for each new query concept. The link matrix weights are re-estimated using the sample of relevant documents. The weight associated with a query term is used to predict the probability that the information need is satisfied given that a document has that term. Relevance feedback involves the re-estimation of this probability.

The inference network differs significantly from other retrieval architectures in that the probability calculations provides a natural framework for incorporating multiple sources of evidence or information (Belkin and Croft, 1992). For example, Ribeiro and Muntz (1996) augment an inference network based on the Cystic Fibrosis test collection (1239 documents, 100 queries) with relevance rankings assigned by human experts for each document in the dataset, over each of the 100 queries in the test set. Precision and recall were improved, in comparison with both the original belief network and with a vector-based IR system. Unfortunately, the effort required to produce these relevance rankings is so great as to preclude its use in practical implementations, and it is unclear how similar information could be automatically gathered or generated.

Belew (1989) and Crouch, *et al,*. (1994) incorporate user relevance ratings into a connectionist retrieval system. Nodes in the network corresponding to documents judged by the user to be relevant are given a powerful excitatory signal, while nodes corresponding to irrelevant documents are strongly inhibited. The network then cycles again, producing a new document ranking—and

perhaps adding additional documents to the list of potentially relevant articles. Kwok (1989, 1991) suggests a similar neural network architecture for using relevance feedback to improve ranking by modifying document connections. If the modifications in the network are discarded after each query session, then the temporary link and weighting changes produced by relevance feedback can be viewed as a short-term user model (Crouch *et al.*, 1994). If the modifications are retained, then the averaging of feedback across numerous queries and users constructs a representation of the communal meaning attached to keywords by the users. In this case, the problem lies in gathering a large enough amount of feedback from a significant number of users, so that a single idiosyncratic opinion or an eccentric user does not skew the network's output.

The neural network representation of the user's interests can also be decoupled from the retrieval system. Bordogna and Pasi (1996) incrementally construct a neural network model for a query session, based on relevance feedback from a conventional IR system. In this prototype, model construction begins when the user selects at least one relevant document from the query results. The query terms and the most significant terms from these relevant documents are represented in a fully connected, Hopfield-like neural network, with node activation levels initialised as the average of the significance levels of the term in those documents. Activation signals propagate through the network until equilibrium is reached. At this point, the most active nodes are selected as candidates for query expansion, and the modified query is run against the IR system. New nodes are added to the neural network with each relevance feedback iteration.

3.2 Using a thesaurus

Another standard tool for aiding the user in expanding a query is the thesaurus (in this context, a thesaurus provides a matching between related terms describing a subject, rather than a listing of strict synonyms). Generally subject classification schemes such as the Library of Congress Subject Headings (LCSH) system contain thesaurus-like "related terms" or "use for" listings, in addition to hierarchical subject descriptions. Traditionally, the user manually browsed printed copies of the thesaurus or classification scheme, selecting additional query terms based on the user's high level understanding of the domain's semantics. These thesaurii were also hand crafted—a time-consuming and expensive process.

A collection-specific thesaurus can be constructed automatically, most commonly by detecting related terms through their co-incidence in document pairs (Salton and McGill, 1983). While these term matchings can be used effectively for automatic query expansion, the thesaurus itself is not intelligible to humans and, in general, cannot be browsed by a user to gain a deeper understanding of the semantics and structure of the domain about which the collection is focussed.

Chen et al (1994, 1995a, 1995b) report a series of experiments that evaluate the effectiveness of Hopfield-based thesaurii for concept exploration and query expansion. A Hopfield network is a single layer, weighted network—a natural architecture for representing the homogenous set of objects (terms) as nodes, together with their semantic associations (weights on node linkages). The networks they construct combine terms from both manually created and automatically generated thesaurii; with this technique, they retain a measure of the perspicuity of the manual thesaurii, while gaining the collection-specific focus of the automatically generated thesaurus. The weight propagation scheme amalgamates the multiple term sources by assigning normalized weights to the manual thesaurus links based on weights derived from the automatically constructed thesaurus.

The user "browses" the network by supplying an initial set of query terms. The nodes representing these terms are clamped on, and activation flows through the network in successive waves until a stable state is reached. The most highly active terms are then presented to the user for possible inclusion in the revised query. Initial testing indicates that manual browsing of thesaurii can lead to the identification of terms that support higher recall searches than the terms suggested by the Hopfield network; however, manual browsing was also significantly more time consuming. Automated term suggestion appears to avoid the off-track browsing behavior reported in previous hypertext exploration systems, in which users are distracted into search paths that, however interesting, are not applicable to the problem at hand.

4. Filtering systems

Conventional IR systems were designed for relatively static collections, such as a physical library; the user is fairly certain that the results produced by a given query will not vary greatly if the query is run immediately, in an hour, or in a week. While the document collection changes slowly, it is the user's interests that change from query session to query session. Since the user's information need is satisfied by a one-off request, the user is generally willing to invest the time to iteratively refine a given query.

In contrast, information filtering techniques address the situation in which a user's interests remain constant, but the document set is rapidly changing: for example, when scanning the USENET News. In this case, documents retrieved will be of short-term interest, and the emphasis is on gathering relevant documents as they appear—and before they become obsolete. Because of the mutable nature of the collection, the documents are rarely formally cataloged (marked by author, title, subject descriptor, etc.). This lack of structure makes it particularly difficult for users to construct effective queries. From an IR standpoint, the problem is to model the users' interests so as to filter the continuous stream of information and distribute incoming articles to the

appropriate audience. Since the document set is continuously changing, users will generally be unwilling to devote time and energy to query refinement. It becomes the system's responsibility to maintain an effective user model, building an initial model from an exemplar set of relevant documents and modifying it over time to reflect changes in the vocabulary introduced by new documents.

The user model in Jennings and Higuchi's (1992) USENET news filtering system is based on a neural network. An initial set of news articles are retrieved by the user and marked as relevant or irrelevant to their interests. These positive and negative examples are then used to train a neural network. The training features extracted from the documents are based on term vectors, and terms are assigned weightings based on their position in the article—for example, terms appearing in the "Subject" line are weighted more heavily than ones in the main text. Once trained, the network screens incoming articles and ranks them by predicted relevance. The network tracks changing user interests by noting the articles read or rejected during each user session, and later feeding these articles through the network as additional positive and negative examples.

Pannu and Sycara (1996) describe agent software that scans the WWW for conference announcements and requests for proposals that may be of interest to the user. The user's preferences are learned from a training set whose positive examples are papers and proposals written by the user, and whose negative examples are documents written by faculty working in other fields. *Tfxidf* and two neural networks were tested for updating the user's profile, with the former producing the best accuracy in terms of classification of new documents.

Developing a filtering system based on the belief network model is problematic, since a filtering system must essentially stand the belief network on its head (Figure 3; where D_1 is the node for an incoming document, R nodes describe content, Q nodes are queries, and P nodes represent user profiles). But the directional probabilities from an existing retrieval system cannot simply be "inverted" to create a filter; documents and profiles are not symmetric objects. Instead, the following process occurs: a term vector is created to represent the incoming document; the probability estimates $P(r|d)$ are calculated; pre-calculated probability linkages filter from active concept to profile nodes, to select relevant profiles; and finally, $P(p|d)$ estimates are calculated, and the document is associated with each profile for which this quantity exceeds a threshold.

5. Unsupervised document clustering

In the absence of an existing text classification scheme, unsupervised clustering techniques can group documents into "natural" clusters. In conventional IR systems, clustering is generally conducted by instance-based learning algorithms, using nearest-neighbour techniques to group documents according to

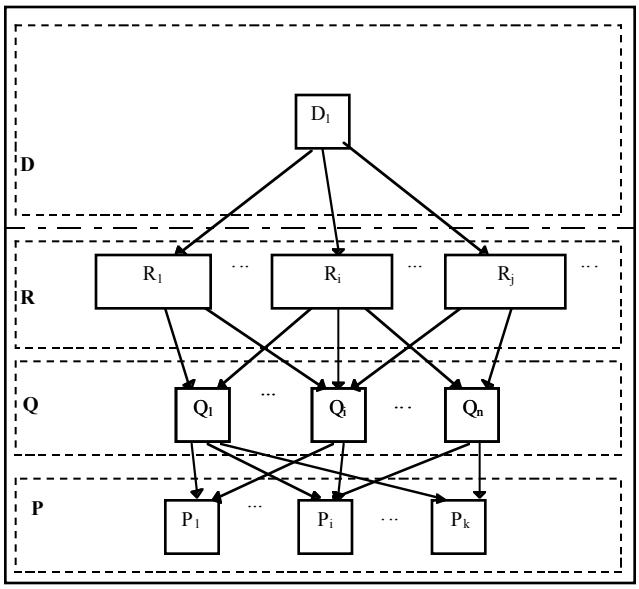


Figure 3. Belief network for information filtering

the "closeness" (generally measured by Euclidean distance) of the document vectors (see Willett (1988) for a survey of conventional clustering techniques). The clusters that are discovered might, or might not, represent semantically meaningful categories. Moreover, generally the algorithm itself does not supply a semantic label, which makes it difficult for users to effectively browse sets of clusters to find document groupings of interest.

5.1 Kohonen self-organising maps

An alternative, neural network-based approach to unsupervised clustering is the Self-Organising Map (SOM) method, based on the Kohonen feature map algorithm (Kohonen 1989, 1995). Documents are represented as n-dimensional term vectors, and are mapped onto nodes of a discrete two dimensional grid. The learning process can be thought of as the projection of the n-dimensional space into two dimensions, so as to express graphically the semantic "distance" between input documents. While some distortion will inevitably occur, the mapping attempts to preserve "neighbourhood" relationships. The map can also distinguish relative importance or significance by allocating larger sections of the map to documents/terms that occur more frequently. It is, of course, possible to create maps of a higher dimension than two, but these are difficult to effectively display.

To construct a map, each node in the grid is assigned an initial, small random weight. The weights self-organise through the following iterative learning process (Kohonen, 1989):

1) an input (document) vector is selected randomly

2) the "winning" node is located whose weights are closest to this input vector

3) the weights of the winning node are adjusted to move closer to the input vector in the n-dimensional vector space

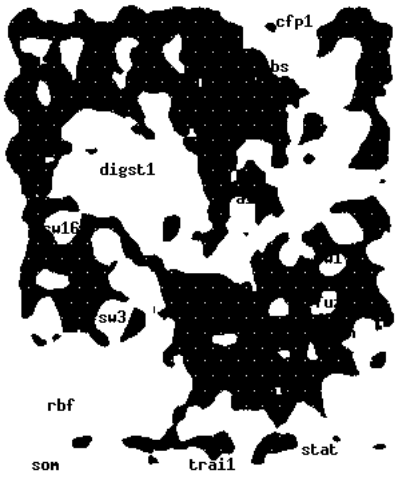
4) the weights of nodes arbitrarily "close" to the winning node are also adjusted, to bring them nearer the input vector

The learning algorithm iterates until it converges (adjustments are arbitrarily close to zero). Finally, each training document is mapped to a single node—either through a simple matching of grid vectors to document vectors (Lin, *et al.*, 1991), or by running an additional pass of the SOM algorithm to self-organise a mapping (Honkela, *et al.*, 1996). As new documents are added to the collection, they can be assigned the most closely matching node. However, eventually the growing collection should be self-organised again, to better accommodate new concepts added to the document set by the incoming articles.

The final collection maps do not typically show an even scatter of documents across the grid; instead, the documents will group themselves into clusters of varying density, separated by bare portions of the map matched by few or no documents (Figure 4). Such a map is generally touted as an aid to browsing, rather than as a search tool for locating specific topics or known items: the user's query is matched to one or more locations on the map, and the documents matched to these nodes are explored for items of interest. Several prototype systems of this sort have been implemented: Scholtes has presented an SOM-based interest map for information retrieval (Scholtes, 1991, 1992); the WEBSOM project demonstrates a WWW-accessible SOM for several collections of newsgroup postings, including a corpus of 131,500 articles drawn from twenty newsgroups (http://websom.hut.fi/websom/); Johnson and Fotouhi (1996) provide individualised maps of hypertext documents by building an SOM of a single user's link traversal history; and Merkl, *et al.*, use an SOM to semantically organise textual descriptions of program components in a software library (Merkl *et al.*, 1993, 1994a, 1994b) and to categorize legal texts (Merkl, *et al.*, 1997), as well as experimenting with learning rules that may improve the map visualization by better representing the "closeness" of documents (Merkl, 1997a).

A multi-level SOM can be constructed by dividing the base SOM (representing all documents in the collection) into a set of "neighbourhoods", or non-overlapping map sections. Each neighbourhood is then treated as an input node to the SOM at the next highest level; that higher map then self-organizes, is

divided into neighbourhoods, and provides a set of input nodes to the next map in the hierarchy (Merkl, 1997b). The final SOM set resembles a document taxonomy—although, of course, the hierarchy of clusters cannot be guaranteed to resemble a humanly intelligible, semantically based taxonomy.



Explanation of the symbols on the map
apps1 - applications: face, speech
brain1 - brain sized NN
cfp1 - conferences
digst1 - Neuron Digest, CFPs
econ - finance
fuzz1 - fuzzy logic
jobs - vacancies
rbf - Radial Basis Function networks
som - Kohonen SOMs
stat - NN vs statistics
sw1 - implementations
sw16 - software
sw2 - software
sw3 - source code
time - time series
trai1 - training, testing

Figure 4. WEBSOM map of 12088 comp.ai.neural-nets postings (from http://websom.hut.fi/websom/comp.ai.neural-nets/html/root.html). Area labels were added manually

This visualisation can greatly enhance the browsing process: by scanning the map, users can quickly gain an impression of the areas of relative paucity and plenty in the document collection. Additionally, a user can easily see which portions of the collection are just "outside" the documents retrieved by the user's query. By contrast, in conventional IR systems it can be very difficult for a user to explore other (potentially useful) sections of the collection, as this requires the user to know the terms by which those documents are indexed. In a growing document collection, the SOM can also be useful as a filter for incoming documents of interest: once an interesting are of the map is located, the user can periodically inspect those SOM nodes to trap newly acquired documents that map to that topic.

SOM descriptions of a collection can also be interesting as an analytical tool for examining the distribution of topics in a body of literature. Lin, *et al*., (1991) demonstrate this by constructing an SOM for AI documents drawn from the LISA (Library and Information Science Abstracts) database. Semantically related nodes of the map are merged into "concept areas", labelled with the best matching one to three terms in the original set of features (Figure 5). This type of analysis can provide an indication of emerging subfields in a discipline, and the relative size and "closeness" of various topics.

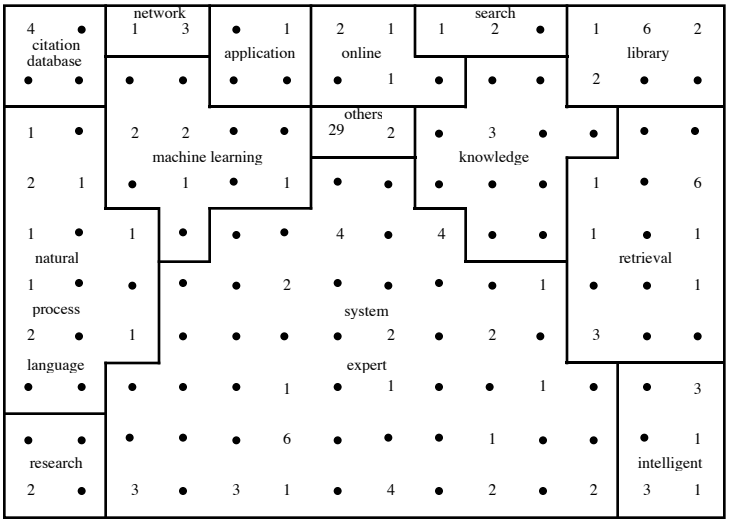


Figure 5. SOM of AI documents drawn from the LISA database (from Lin, *et al.*, 1991, p. 265).

5.2 Clustering of query results

A self-organising approach to document clustering has been applied to the results of queries on a digital library. This not only clusters documents according to their similarity to one particular query, but also provides a visual representation of the inter-relationships between documents returned by a sequence of queries. This approach uses a three dimensional visualisation tool known as HyperSpace (Hendley, *et al.*, 1995) that uses a representation of three dimensional space to generate images of data. There are two forms of basic representation within the space; nodes are spherical objects whilst links join nodes. Each of these basic types has a defined behaviour, which allows the structures produced to organise themselves into a steady minimum energy state. This self-organisation occurs within the virtual three dimensional space, the nodes and links moving around until they reach a stable arrangement. This produces a consistent visual representation for similar structural models. The physics within the space can be complex, but essentially nodes tend to repel each other, which spreads out the visualisation, whilst links act as springs pulling things together. As in other self-organising neural systems, interactions tend to be global at the beginning and are then refined to be more local. This tool has been successfully used to visualise a range of systems, particularly the World-Wide Web (Wood, *et al.*, 1995). The system allows rotation in three dimensions about an arbitrary, user-selectable point, and supports zooming in and out at will. Moreover, the interactive system uses colour; the three dimensional effect is more apparent on the screen where the user can rotate and zoom the structure in real time.

Queries are posed to a full-text index of a digital library, and each query returns a maximum of fifty documents that satisfy it. If the query is "ranked" (rather than boolean), these are the fifty documents judged most relevant to the query according to the cosine rule mentioned previously (Section 2). The documents returned are designated as HyperSpace nodes. Each query made by the user is also mapped to a node. Links are drawn between each document and the query that returned it. This structure dynamically updates itself as the user makes a series of queries. Completely separate independent queries produce a series of "dandelion heads"—unconnected clusters of nodes, each one centred on the query that generated it. More interesting patterns appear when the queries are related, because if the same document is identified by different queries it becomes linked to more than one query node. A whole series of queries on one topic will produce a more complex pattern comprising a densely connected mass of nodes in which the relationship between different queries can be discerned in terms of the degree of overlap (and hence commonality) of the documents they generate. The system therefore self-organises into a representation of the document space, metricated according to the query terms. The system is designed to be interactive and acts as an aid to navigating that space; any node in the visualisation can be selected, causing the document that it refers to, to be returned in a window. As the user moves about the space, information relating to the nearby nodes is presented.

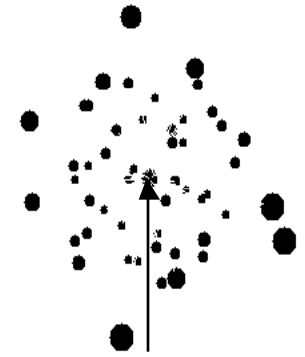


Figure 6. A three term query

Figure 6 shows the visualisation produced for the three-term query *visualisation colour graphics*, issued as a ranked query with stemming and case-folding in place (the default). Fifty documents are returned and are shown spread around the central node that represents the query, pulled in by relevance and pushed out by size. Figure 7 shows the effect of making a second query, for the three terms *3D surface graphics*: the display is automatically updated as soon as

the query is made. The new query is on the left; it has been labelled here for clarity, though the dynamic nature of the system is enough to identify the new query easily. When the user zooms in, the labels identifying the queries become apparent, but are omitted from the wider view to avoid screen clutter. It is apparent that there are two documents in common between the two queries.

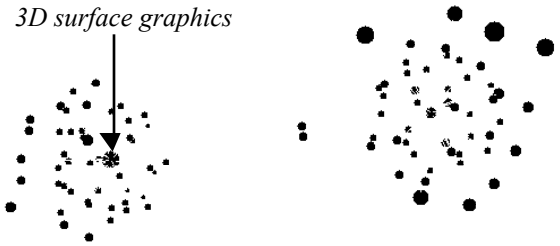


Figure 7. Adding a second query: *3D surface graphics*

The user issues a third query, this time for the single term *agents*. It is clear from the display shown in Figure 8 that the top fifty documents returned for this query have no overlap with those returned by the other queries. The other two queries have retained their structure but drifted away from this most recent