A Network Model for Adaptive Information Retrieval

by
Fabio A. Crestani

Department of Computing Science
Faculty of Science
University of Glasgow
Glasgow

being a thesis submitted for the degree of Master of Science

March 1992

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Acknowledgements

I would like to thank my supervisor, Professor Keith van Rijsbergen, for helping me in the difficult task of doing research in Information Retrieval. His guidance during the research and writing up of my thesis were invaluable.

I must also thank Professor Maristella Agosti, of the University of Padova. It is thanks to her encouragement if I began doing research, and I would not have even started this thesis without her encouragement and help.

I would also like to thank the members of staff and fellow post-graduate students of the Department of Computing Science at the University of Glasgow whose helpful suggestions were a great asset. Thanks, in particular, to Ruben Leon and David Kerr for proof reading.

Finally I would like to thank my mother, father, family and friends for their support and patience during my career as a student at the University of Glasgow. In particular I would like to thank the community of Italian students in Glasgow for making my stay much more enjoyable than I first thought.

Abstract

This thesis presents a network model which can be used to represent Associative Information Retrieval applications at a conceptual level. The model presents interesting characteristics of adaptability and it has been used to model both traditional and knowledge based Information Retrieval applications. Moreover, three different processing frameworks which can be used to implement the conceptual model are presented. They provide three different ways of using domain knowledge to adapt the user formulated query to the characteristics of a specific application domain using the domain knowledge stored in a sub-network. The advantages and drawbacks of these three adaptive retrieval strategies are pointed out and discussed.

The thesis also reports the results of an experimental investigation into the effectiveness of the adaptive retrieval given by a processing framework based on Neural Networks. This processing framework makes use of the learning and generalisation capabilities of the Backpropagation learning procedure for Neural Networks to build up and use application domain knowledge in the form of a sub-symbolic knowledge representation. The knowledge is acquired from examples of queries and relevant documents of the collection in use. In the tests reported in this thesis the Cranfield document collection has been used. Three different learning strategies are introduced and analysed. Their results in terms of learning and generalisation of the application domain knowledge are studied from an Information Retrieval point of view. Their retrieval results are studied and compared with those obtained by a traditional retrieval approach.

The thesis concludes with a critical analysis of the results obtained in the experimental investigation and with a critical view of the operational effectiveness of such an approach.

Contents

1 Introduction **2**

1.1 The classical approach to Information Retrieval 2
1.2 Intelligent IR 6
1.2.1 IR and expert systems 7
1.2.2 IR and natural language processing 7
1.2.3 IR and knowledge representation 8
1.3 Knowledge Based IR 9
1.4 Application domain knowledge representation in IR 10
1.5 New approaches to Knowledge Representation 14
1.6 Associative and Adaptive IR 15

2 A conceptual model for Associative Information Retrieval **18**

2.1 The need of a conceptual model 18
2.2 The basic network structure 19
2.3 Conceptual modelling of IR traditional applications 23
2.4 Conceptual modelling of a knowledge based IR application 26
2.4.1 The knowledge network structure 27
2.4.2 The processing framework 31

3 Associative Information Retrieval **36**

3.1 Associative processing frameworks 36
3.2 Spreading Activation 36
3.2.1 Pure Spreading Activation 37
3.2.2 Constrained Spreading Activation 41
3.2.3 Spreading Activation with feedback 43
3.2.4 Spreading activation and IR 43
3.3 Artificial Neural Networks 46
3.3.1 Biological neurons and networks 46
3.3.2 Simulated neurons and networks 47
3.4 Local and distributed representations 50
3.5 Learning and memory 51
3.6 Neural Networks learning procedures 53
3.6.1 Backpropagation 56

3.6.2 Boltzmann Machine 59
3.7 Neural Networks and IR 61

4 Adaptive Information Retrieval **70**
4.1 Adaptation 70
4.1.1 Query adaptation 74
4.1.2 Document space adaptation 75
4.2 Use of a NN as an adaptation tool 75

5 An adaptive strategy for IR: experimental evaluation **77**
5.1 The simulation environment 77
5.1.1 The ASLIB Cranfield Test Collection 77
5.1.2 The PlaNet neural network simulator 78
5.2 The simulation system 80
5.3 The evaluation criteria 84
5.4 A Connectionist Knowledge Representation Structure 87
5.4.1 The NN model 89
5.4.2 The structure of the patterns 91
5.4.3 The internal representation 92
5.4.4 The learning parameters 93
5.5 Subsymbolic learning of domain knowledge and query adaptation 96
5.5.1 Total learning 98
5.5.2 Horizontal learning 105
5.5.3 Vertical learning 110

6 Conclusions **115**
6.1 Conclusions from the experimental results 115
6.2 Operational Adaptive Associative IR 117
6.3 Future research work 118

A Appendix **119**
A.1 Cranfield test data 119
A.2 Query Processor 120
A.3 NN simulator 122
A.4 Matcher 125
A.5 Document processor 132

Bibliography **134**

Chapter 1

Introduction

"Many of the techniques I shall discuss will not have proven themselves incontrovertibly superior to all other techniques, but they have promise and their promise will only be realized when they are understood."
[K. van Rijsbergen, 1979]

1.1 The classical approach to Information Retrieval

The Information Retrieval (IR) part of an information system is the part which is concerned with the management of a databank of unformatted data. This is usually called the *IR application*. An IR application is related to the storage, management and retrieval of weakly structured or unstructured data.

The objects handled by an IR application are usually called "documents". The term *document* is meant to represent any information bearer such as a book, a report or a letter (textual media) or an image or a drawing (visual media). The software tool which automatically manages these documents is called *Information Retrieval System* (IRS). The task of an IRS is to help a user to find, in a collection of documents, those documents which contain the information the user is looking for, that is providing help in satisfying the user's information need.

To give a clue to the size of the task, it must be noticed that usually these collections of documents contain several thousands or even millions of documents.

Frequently IR is confused with database (usually referred as DB or DBMS, database management system) technology. Figure 1.1 summarises some of the major differences between IR and DB technology; these characteristics have been identified and described by van Rijsbergen in [1]. The fundamental difference between IR and DB is that IR systems usually provide only *references* to or a description of the data they manage, while a DBMS provides the actual data. This is not just because of limitations imposed by current technology. Even when full text IR systems will become available, the task of IR will still be mainly *to point* at documents. Given this fundamental difference, an IRS usually manages only descriptions of the informative content of documents (books, reports,

	Information Retrieval	Database
Matching	Partial match	Exact match
Inference	Induction	Deduction
Model	Probabilistic	Deterministic
Classification	Polythetic	Monothetic
Query Language	Natural	Artificial
Query Specification	Incomplete	Complete
Items Wanted	Relevant	Matching
Error Response	Insensitive	Sensitive

Figure 1.1: Differences between IR and DB

etc.). The basic element of these descriptions is called *descriptor*. A schematic view of IR is presented in Figure 1.2.

Furthermore, this difference has several consequences. First, the users' queries to an IRS are usually more vague. They are usually in the form: "I want documents about ...", while users of a DB want facts, such as: "I want the price of the product abc". Second, IR systems retrieve documents in a probabilistic way, while DB retrievals do it mainly in a deterministic way. This means that an IRS retrieves documents that are likely to be considered relevant by the user; that is, they are likely to satisfy the user's information need. On the other hand, DB facts retrieved in response to a query are always considered to be a complete and true answer to the query. In IR, it is important to remember that the perceived relevance of a document varies dramatically across users, and even with one user at different times. This introduces the third consequence, which is related to the criteria used to evaluate the performance of an IRS. The evaluation of an IRS is more or less related to its "utility", that is how helpful the system is to a user. It is easy to see that this is not a well specified measure. On the other hand, DBMS are evaluated in accordance with well specified and standardised performances measures.

One of the most important problems of IR is related to the *representation of the document informative content*. In [1], p.29-30, van Rijsbergen provides a clear view of the problem:

> There are two conflicting ways of looking at the problem of characterising documents for retrieval. One is to characterise a document through a representation of its contents, regardless of the way in which other documents may be described. This might be called representation without discrimination. The other way is to insist that in characterising a document one is discriminating it from all, or potentially all, other documents in the collection. This we might call discrimination without representation (omissis). In practise one seeks some sort of optimal tradeoff between representation and discrimination (omissis). This emphasis on representation leads to what one might call a document orientation: that is a total preoccupation with modelling what the document is about (omissis). The emphasis on discrimination leads to a query orientation.

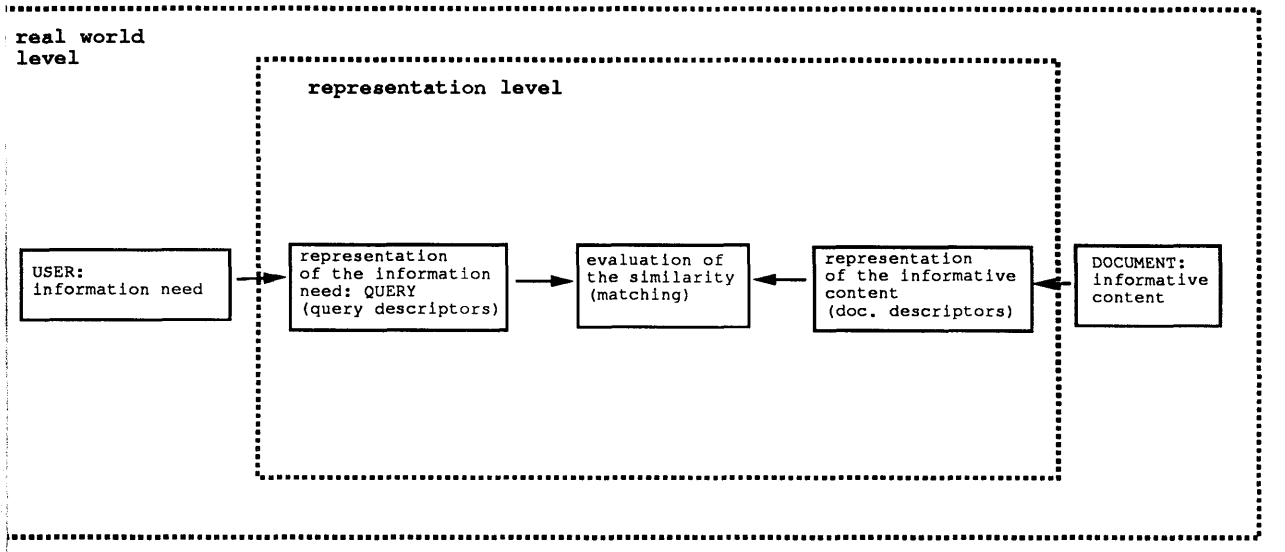

Figure 1.2: A schematic view of IR

The *Associative and Adaptive approach to IR* presented in this thesis attempts to compromise between the goals of representation and discrimination using a permanent structure representing documents and keywords and a dynamic spreading of activation generated by any one query on the structure generated. Furthermore, the representation structure can adapt itself in order to provide better discrimination power using information from the user and from a control structure on the spreading of activation. In this way, repeated use of the system will bring about changes in the representation structure of documents, allowing it to adapt itself to the user perception of the documents' contents and therefore to the user's information need. In this sense, this approach provides a good compromise between representation and discrimination. While the initial structure is based entirely on characteristics of the documents, the repeated use of the structure changes the structure itself according to characteristics of the queries submitted to the system.

In the classical approach to IR, a schematic view of which is presented in Figure 1.2, the problem of the document informative content representation is tackled assigning, in an automatic way, descriptors to a document. This process is called *automatic indexing*.

The basic assumption upon which most of the theory of automatic indexing is based, is that the frequency with which a word occurs in a document provides information as to how useful that word will be in characterising the document informative content. In [2] Luhn proposed that the ability of words to discriminate the document content follows a Gaussian distribution in relation to the rank order of the words used in the document. This is shown in Figure 1.3. Luhn called this the word's "resolving power".

In order to use only words with an high resolving power, words with very high rank order or words with a very low rank order are cut off. The first set of words is considered "noise". Words in this set, such as "and", "the", "not", "for", are used only for "function" and they do not convey a "content". On the other hand, words belonging to the other

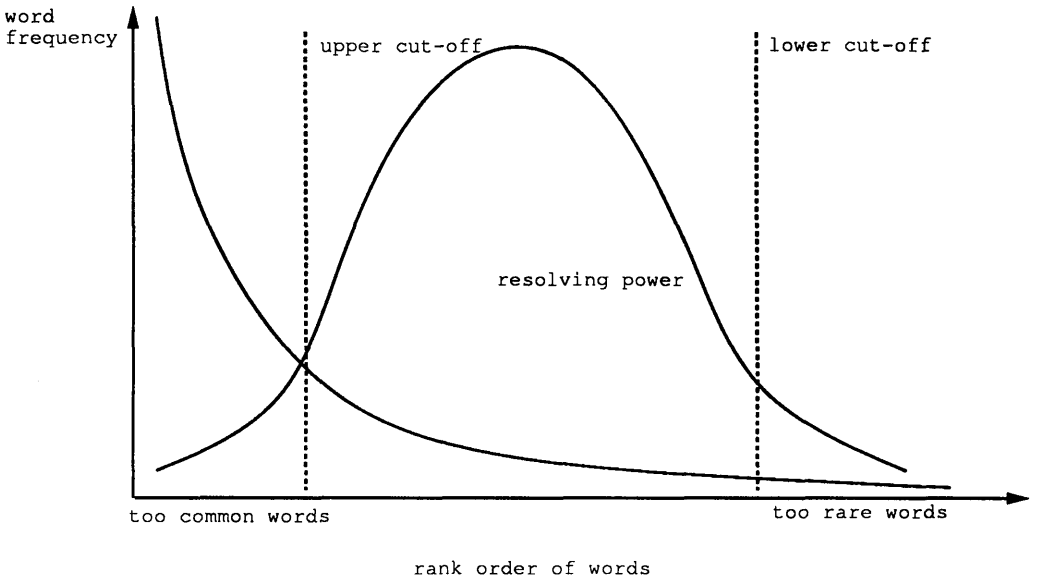

Figure 1.3: Word's resolving power

set, even if they convey content, occur too rarely to be used to help discriminating among documents.

These statistically–based techniques can be enhanced by "conflation" techniques which attempt to map individual word tokens to a single morphological form. IR has developed sophisticated techniques for this "stemming" (see for example [3]), which are used in most of the operational IR systems. Some experimental systems attempt to use phrases instead of individual words (see [4]), but the automatic identification of phrases in free texts is a problematic task.

An important result of IR research is that the indexing relation between descriptors and documents does not need to be deterministic. It has been recognised that relevance is not an absolute notion, and can be measured. A large part of the theoretical research in IR is devoted to finding an effective way of evaluating this relevance measure.

Most of the present research in IR uses weighted associations between descriptors and documents. This leads to a *theory of probabilistic indexing*, and a number of methods have been proposed for computing appropriate weights. One typical approach is Salton's *term frequency* and *inverse document frequency* weighting scheme ([5]). This approach is based on Luhn's assumption and on the assumption that the discriminating power of a word is inversely proportional to the number of documents to which that word is assigned. In particular, the inverse document frequency states that the more documents are indexed by the same term, the less important the term becomes as a descriptor to any of them.

A more formal theory of probabilistic retrieval has been proposed by van Rijsbergen in [1]. In this theory, using the Bayes' theorem as a fundamental mathematical tool, the probability of relevance of a document is evaluated from the estimated measure of probability of relevance for every document in the collection (the prior probability of relevance)

and from the estimated likelihood of relevance or non-relevance given that document.

Once a suitable representation for documents has been provided an IRS faces the problem of evaluating the similarity between the representation of the documents and the representation of the user's query. This is usually achieved by evaluating a *similarity coefficient* which uses the features of documents and query representation to evaluate an overall degree of similarity between the query and each document in the collection. However it is important to note that even the most sophisticated evaluation technique is useless if it uses poor representations. The representation issue is therefore a central point in research in IR and it is in this direction that this thesis is addressed.

Any good piece of research should take advantage of previous work on the topic. This thesis follows this line and a brief description of what has already been achieved in IR is presented in this chapter.

1.2 Intelligent IR

It has been a long time since Luhn suggested the use of statistical techniques for the representation of document informative content. There have been many changes in the field of IR since that time and there have been surprising developments in computer hardware. However some fundamental issues remain unsolved. In particular, the representation of text documents and of the user's information need remain some of the most problematic aspects of research in IR. It is true that statistical approaches to the analysis of text and retrieval of documents have significant advantages in terms of efficiency and performance compared to other techniques, but it seems recognised, at present, that statistical techniques have reached the limits of their performances. The dissatisfaction with the current state of research in statistical methods is the major factor of the recent upsurge of interest in more "intelligent" IR among the IR research community.

The area of IR research called *Intelligent Information Retrieval* (IIR) deals with the overlap of research in Artificial Intelligence (AI) and IR.

The application of AI results to IR is quite a recent phenomenon. As more and more information is stored electronically, the need for intelligent methods of accessing this information becomes increasingly critical. The basic belief underlying the research in IIR is that a truly helpful IRS must, in some sense, "understand" what the user is looking for. Furthermore, in order to understand this, the system must minimally understand the document's informative content and the domain which the documents belong to. In fact, an obvious reason for the relatively poor performances of the traditional IR technology is that the use of words, and lexical items in general, as descriptors gives a poor approximation to meaning. Even with the addition of a thesaurus or a phrasal lexicon, the approximation is not good. It is in this direction that most of the work in IIR has been done.

It is possible to identify three distinctive directions of application of AI research in IR ([6]). In the following sections a brief description of these three application areas is presented.

1.2.1 IR and expert systems

The central theme of this area of research is the development of an *expert intermediary system*. This is an expert system which assists the user query formulation, search strategy selection, and evaluation of retrieved documents. Some examples of systems corresponding to this general specification are reported in [7].

Within this common framework there are, of course, many different approaches that can be taken, and different research emphasizes different issues. The I^3R system ([8]), for example, emphasizes the following issues:

- acquisition of a request model by analysis of the query and interaction with the user;
- acquisition of a user model;
- provision of multiple search and browsing strategies;
- control of the use of system facilities during search sessions.

Although there is quite a large amount of literature on the argument, it seems that the main feature of expert systems that has been used in IR application is the rule based representation of expert knowledge. The hope is to enrich the traditional document and query representations with domain knowledge codified in a "knowledge base" and managed by the "inference engine" of the expert system. The problem of finding a better representation of the document informative contents and of the user information need has not been tackled.

Many issues remain open, like the problem of the formalisation and codification of the expert knowledge, its updating, and the use of appropriate inference rules on it. For a critical view of the results so far achieved in this area see [9].

1.2.2 IR and natural language processing

It has been recognised that natural language processing (NLP) is an essential part of the process of identifying and representing the query and the document informative content. So far, much of the traditional research in IR has been concentrated on simple language analysis techniques, such as identifying word stems and phrases. The aim of using more sophisticated NLP techniques is to produce representations that reflect more accurately the meaning of the object represented. This should result in higher performances, but there is no proof of this. However, in order to perform effective NLP it is necessary to have a large amount of domain knowledge, even larger than what is usually required for expert systems. Thus this approach is affected by some of the problems already seen in the application of expert systems to IR. It represents a step forward toward the use of complete domain knowledge.

The open issues in this area are many and include the following:

- the knowledge required to perform effective NLP is larger than what is usually available by means of a thesaurus or a general dictionary. Most of the current systems performing NLP make use only of these sources;

- the level of NLP required for effective document retrieval has not been determined yet. Many attempts have been made using techniques ranging from simple syntax-based techniques to complex conceptual analysis;

- NLP requires a large domain knowledge, but NLP can be used to build knowledge bases. In the long term, the acquisition of knowledge from document texts will be essential in the dynamic IR environment.

The main problem is that it is not clear yet how much improvement in retrieval effectiveness is obtainable by using NLP techniques, but it is intuitively attractive. So far most of the experimental systems use NLP only as a front end to traditional IR systems, without going in depth into the problem of representation. Much research is going on at present, especially in the directions of automatic abstracting, automatic indexing using syntax or semantics, and indexing by word senses.

1.2.3 IR and knowledge representation

The main issue of this application area is to represent and use the application domain knowledge and the informative content of documents and queries. The focus is, therefore, on the very problem of representation.

There are two basic approaches to this problem which reflect different views of IR. The first approach assumes that very detailed domain knowledge is available and that document contents are represented using this knowledge. Examples of IR systems following this approach are GRANT and RUBRIC. The second approach is based on conventional IR systems that can be used in more loosely defined domains. It assumes that domain knowledge is incomplete and must be incrementally acquired through interaction with users. The I^3R system is an example of this second approach.

Much research has been carried out in this area and many successful experimental systems have been developed, however, there are still many open issues that can be summarised as follow:

- the difficulties in the acquisition of domain knowledge from users or document texts depend to a large extent on the development of effective NLP techniques;

- the level of document content representation appropriate for IR applications has not been determined yet;

- the level of domain knowledge that is required for effective IR has not been determined but it has been recognised that this depends on the dimension of the application domain;

- the knowledge representation technique to be used depends largely on the use of this knowledge in the application. This problem has been studied in many AI application but not in IR.

Despite these open issues, I believe that within this area the most significant developments and improvements in the performance of IR systems in the short and medium term will come. This direction of research has been therefore chosen in this thesis.

The rest of this chapter is devoted to a review of the results achieved in knowledge representation in AI and IR.

1.3 Knowledge Based IR

The area of application to IR of knowledge representation techniques borrowed from AI is often called *Knowledge Based Information Retrieval* (KBIR). Among the three above mentioned application areas of AI to IR, this is the one which seems to me the most interesting. The problem of representation has been recognised as one of the major difficulties to the progress of IR. From my point of view it is only with tackling this problem that new significant developments in IR can be achieved. With this same perspective Fox identified in [10] some of the advantages of a KBIR approach. They are the following :

- improve the effectiveness and the overall efficiency of IR systems;
- allow heterogeneous collections of structured and unstructured texts to be searched in an optimal fashion;
- ensure that the positive contributions of search intermediaries are made available in situations where they cannot be present;
- allow integration with expert systems or other types of intelligent systems;
- ensure that the performances and usability of IR systems improve with utilisation, both in general and for individual users.

However, a review of the research in KBIR shows that these advantages are not easily achievable. A review of the attempts in this research area, presented in the following section, will show what has been achieved so far.

The main issues of KBIR are on the representation and use of:

- the informative content of documents and queries;
- the application domain knowledge.

The first issue is concerned with finding better representations of the very elements of an IR application. The basic idea is that better representation of documents and queries can provide a better ground for their comparison and matching.

The second issue is more concerned with the use of domain knowledge as a means of enriching the traditional document and query representation. The application domain knowledge is used with the purpose of "understanding" the meaning of descriptors in the particular application domain. In this way they can be related to each other and the knowledge achieved by looking at their relations can be used in the matching process to provide a better evaluation of the similarity between query and documents.

Since document and query are expressed using natural language, a representation of their informative content cannot be achieved without the use of some NLP technique. Thus, it is very difficult to use pure knowledge representation technique with the purpose of representing the document or the query content. The use of domain knowledge, on the other hand, can achieve effective results without making use of NLP techniques.

Because of the fact that the progress in the development of new NLP techniques is quite slow, and the NLP techniques available do not seem to be very reliable, an approach which makes use of application domain knowledge has been preferred in this thesis.

In every KBIR system some formalism and corresponding notation with which to represent knowledge must be used. Knowledge Representation (KR) has been, during the last decade, a fundamental part of research in AI, and a variety of different knowledge representation schemes have been studied. In the following section a brief review of some of these techniques is presented together with examples of their application in IR.

1.4 Application domain knowledge representation in IR

The purpose of KR in AI is to represent the knowledge required by the application in a way which is usable by a computer for rapid manipulation and search. As soon as AI discovered how much knowledge was required to perform a reasonable range of intelligent applications, investigators experimented with a great many ways of creating and putting a representation of the real world inside a computer. KR is, therefore, one of the most mature areas of AI.

The use of KR in IR is, however, a recent phenomenon. It has its origin in the attempts to use consolidated expert systems technology in IR, but it is going further toward the use of more sophisticated representation formalisms. Rather then enumerating various representation strategies of the KR area of AI, this section will emphasize the attempts to use KR in IR. The various approaches, exemplified by some experimental systems, will be seen in relation to the representation, use, and acquisition of the domain knowledge required by a IR application.

In AI it is common to classify the various KR formalisms in the following categories:

- languages for KR: LISP, Prolog;

- logic;
- production rules;
- semantic networks;
- frames.

From the point of view of modern IR, logic and logic programming languages, like Prolog, are considered very difficult instruments to use for the kind of knowledge required by an IR application. The retrieval process in IR is a non deterministic one, as it has been explained above. Moreover, it has been recognised that the notion of relevance is more related to a probabilistic evaluation than to a yes/no decision. Nevertheless, most of the operational IR systems are based on Boolean logic, and Prolog or LISP are used in the implementation of the other KR formalisms in many experimental systems.

The KR formalisms which belong to the production rules, semantic networks, and frames categories are more interesting than logic and logic programming languages from the KBIR point of view. They can also be adapted to handle uncertain knowledge.

The commercial system called *RUBRIC* ([11]) is based on production rules and uses a manually built rule base to assist query construction and searching. Although this system is essentially an extension of a Boolean query retrieval system, it has many new features over traditional IR systems. A query is expressed using the standard AND and OR Boolean functions and using adjacency operators. The innovative part of RUBRIC stands on its ability to recognise the presence of concepts in the query by means of the application of the rule base to the query terms. A degree of uncertainty is associated to the recognised concepts and other associated concepts can be taken into consideration according to fixed inference rules. The documents related to the activated concepts are retrieved in response to the query and ranked according to their evaluated level of uncertainty.

A recent improvement in the structure of this system has been achieved using an object oriented processing framework ([12]).

GRANT ([13]) organises its knowledge about research proposal and potential funding agencies using a semantic network. Research topics and agencies are linked using a wide variety of association links to form a dense network. A query expresses one or more research topics, or one or more funding agencies. A search is carried out by "constrained spreading activation" (see Section 3.2) on the network. The constraints, which operate as inference rules, determine which paths are to be followed through the network. Some of them are simple, such as stop after having moved over more than three links from the originally activated nodes, or stop at nodes with a high fan-out. Other constraints represent more sophisticated heuristic rules called "path endorsements". These are used to evaluate different paths on the network. They give preference (positive endorsement)

to some paths and they enable avoiding (with a negative endorsement) some misleading paths. The evaluation mechanism of the paths enable to rank the retrieved nodes.

Developing a system like GRANT involves a significant amount of knowledge engineering to construct the semantic network. This work consists of an in depth analysis of the domain in which the system will operate in order to determine the appropriate concepts and relationships.

Rau's *SCISOR* ([14]) is a KBIR experimental system which partially parses and "understands" short stories in the domain of corporate take-overs. It uses an hybrid frame and semantic net-based language called KODIAK. The domain knowledge stored is either specific (episodic), abstract (generalised episodic) or semantic. In addition to this division, another level of organisation is superimposed on the representation. Groups of related concepts in specific and abstract memory are linked together. These links are used in the retrieval phase to spread the activation through the various levels of the representation. Specific priming rules constrain the spread of activation following heuristic rules similar to those of the GRANT system. The main characteristic of this system is in the NLP component which enable the system to "read" short stories, extract certain prespecified features, and answer questions about the information derived. However, the ability of the system to extract information from texts is limited to those scenarios that it understands. For now, the attempts to give SCISOR more sophisticated forms of KR have not given good results and, at present, the knowledge is still hand coded.

The system called *CODER* was initially designed to study how to handle composite documents ([15]), which are composed of different sections, such as text, header, and citations. Later ([16]), it was developed as a testbed for studying various approaches to the application of AI techniques to IR. The architecture is focussed on a common data area, called a blackboard, and a controller which performs some maintenance and control operations on the blackboard. The blackboard serves as a communication medium for a community of expert systems, each of which implements a particular function, and can even use a different KR structure.

The CODER knowledge representation language has many interesting aspects. In particular, the frame semantics used in CODER is simple and elegant. It is characterised ([17]) by a high level data abstraction and by a sophisticated treatment of inheritance and defaults.

TOPIC and *ARGON* use frames as a representation formalism. Like CODER they incorporated concepts form earlier frame systems. The TOPIC system ([18]), though, focusses on document analysis, using semantic parsing to map text onto frame representation structures, while CODER uses its representation language for both analysis and retrieval. ARGON ([19]) uses frames to handle classification of objects, and determines matches

according to frame subsumption hierarchies.

COREL ([20]) is an experimental IR systems designed to operate in the legal domain. It uses a hierarchy of frames which represent concepts of the application domain. COREL is a menu-driven system. The user is asked to choose options from a list appearing on the screen. These menus constitute a so called "discriminant network". The menus ask questions to fill in slots of the concept frame. This process guides the user through the tree toward its leaves until a certain level of specification is reached. Only at that point, the system retrieves the documents considered relevant to the user specification. Although this system uses a sophisticated KR formalism, it lacks completely of any sort of heuristic rule and can be considered only as a navigational aid. A sophisticated representation formalism is not sufficient to develop an effective KBIR system, a heuristically sound processing framework is also necessary.

The I^3R system is designed to act as a search intermediary. It accomplishes its task using domain knowledge to refine query descriptions, initiating the appropriate search strategies, assisting the users in evaluating the output, and reformulating queries. In its initial version ([6]) the domain knowledge was represented using an AND/OR tree of concept frames, while documents were represented by means of single term descriptors. The system used the domain knowledge to infer concepts that are related to those mentioned in a query. The inference mechanism used a "propagation of certainty" on the concept frames in a way similar to the RUBRIC system, but with a more sound probabilistic base. The domain knowledge is largely acquired through interactions with the users. When a user specifies a query, the system enters a dialog with the user with the aim of identifying concepts. In this task I^3R mimics the situation where a human intermediary interviews a user to clarify a query and build up background knowledge. If a user is not prepared to provide any domain knowledge, the system can still retrieve documents using traditional statistical strategies. This was one of the first works in KBIR in which attention was paid to the acquisition of domain knowledge.

More recently, further research has been done on this system ([21, 8, 22]) and I^3R has become a testbed for investigation on the use of AI techniques in IR. In one of the latest versions, by means of a blackboard architecture, similar to the one used in CODER, a group of independently operating knowledge bases respond to changes on common data to the blackboard. In this framework, different representation formalisms can and have been used inside the same application.

The above exemplification of applications of KR formalisms borrowed from AI to IR is by no means exhaustive. Many other experimental systems have been developed in the last decade, but almost all of them make use of the same representation formalisms described above. However in the KR research area of AI new approaches have been proposed, which

break from the traditional ones. They might enable to overcome some of the difficulties, from the IR point of view, of the traditional KR formalisms.

1.5 New approaches to Knowledge Representation

In most of the books about KR (e.g. [23]) the definition of the word "knowledge" in the context of KR is related to some form of *symbolic representation*. As such, a piece of knowledge is regarded to as a symbolic model of some aspect of some universe of discourse. The emphasis is on the use of a symbolic representation form and all the KR techniques presented in the previous section belong to this class. However a symbolic representation of knowledge may be inadequate for some applications.

In recent times research in KR has been rediscovering, from a more realistic point of view, what has already been discovered in the 40s and 50s.

This first period of the history of KR is usually referred to as *cybernetics* and pre-dates what is typically considered modern AI. One reason why these early works are relevant is that a concern with learning was central. Research took examples from intelligent information processing in natural systems, where learning is a central aspect, and used this as evidence that intelligent systems could be constructed. In particular, the research on "nerve nets" provided an existence proof that apparently simple computational units could be organised in such a way as to "learn" to produce an "intelligent behaviour". One of the most famous and influential examples of this research is Rosenblatt's Perceptron ([24]). However, during this period there was little theoretical understanding of what could be considered intelligent behaviour and how it could be generated. This results in a great exaggeration of the results achievable with these techniques, damaging the credibility of what was actually achieved. This first phase in the history of KR effectively ended around the early 70s when most of the research in KR became more concerned with abstract symbols and heuristics for manipulating them.

The start of this second phase in the history of KR is often associated with the Dartmouth Conference in 1956 and with other seminal works of the same period. The main aspect of this phase, which is still lasting, is that knowledge has to be represented using some sort of *symbolic model*. The more sophisticated the model, the better the resulting representation in terms of its ability to capture the complex aspects of the universe of discourse. Many formalisms were developed, some examples of those have been presented in the previous section. Many valuable contributions came from experience with database systems. This is why most of these formalisms are mainly directed toward "structured knowledge". There are, however, other forms of knowledge, which cannot be accommodated in a structured organisation. IR is the discipline which is most concerned with managing information and knowledge of a unstructured form. Moreover, very little work was done during the 60s and 70s on learning issues. The general idea was that the issue of machine learning would have to wait until the problem of knowledge representation was betters understood. KR formalisms were therefore developed to satisfy epistemological or problem solving considerations, but they were completely useless from a machine learning

point of view.

Within the last decade there has emerged a new distinct approach to KR. This approach can be considered in some aspects as a response to perceived limitations of the symbolic representation used by most AI applications. The representation paradigm of this new phase in KR history has been called with many different names: "connectionist" , "sub-symbolic", "parallel distributed", "associative", or even "neo-cybernetic". In general, following Belew's suggestion ([25]), it seems most natural to speak of: the *connectionist* approach, *associative* representation and *sub-symbolic* elements. The best known example of this approach to KR is in the so called "Artificial Neural Networks", which will be discussed in more detail further on. One of the features of the connectionist work is that it is less concerned with replicating psychological behaviours or neuro-physiological data than the cybernetic phase of KR. However, many ideas of the connectionist approach go back to that first age, to the early works of McCullough, Pitts ([26]), and Rosemblatt.

In the symbolic representations paradigm, whose manifesto is [27], the internal structure of the abstractions is considered irrelevant and the representation and use of knowledge is obtained by manipulating abstract symbols.

The key concept of the connectionist approach to KR is the assumption that knowledge can be represented using low-level units which correspond to a level of representation underlying the symbolic level. Working on a lower level the issues of learning and of the use of knowledge (like, for example, knowledge generalisation) seem to be more related to the representation of knowledge itself than a separate issue. From this point of view, connectionist representations can be viewed more as a complement to symbolic representations than as a replacement of them, since the two approaches work on different level of modelling abstraction.

It has been recently believed that IR could benefit from this new approach to KR. The associative characteristic of this new representation paradigm seems to be congenial to IR, and it reminds some early statistical research. The following section will explain why, from the author's point of view, this new approach to KR seems preferable to the symbolic one in the KBIR research area.

1.6 Associative and Adaptive IR

An approach which makes use of symbolic knowledge representation structures has many drawbacks in its application to IR. Some of the problems previous research has pointed out are:

- it is difficult to determine the level to which the application domain knowledge should be represented; the larger the domain knowledge to represent the larger the knowledge base;
- the knowledge of an application domain, whatever specific it may be, is dynamic, that is it keeps changing, therefore it must be kept up to date;

- a symbolic knowledge representation reflects the application domain knowledge of the expert (or team of experts) who build it, it does not reflect the users' understanding of the application domain;

- a symbolic knowledge representation of an application domain knowledge is something which is attached to the IRS, it is not something which can be seen as an internal part of it.

The use of a subsymbolic representation of the application domain knowledge application may help to solve some of these drawbacks.

In particular, the fact that learning is an integral part of the representation problem in subsymbolic representations may help to solve the problem that Feigenbaum called "the bottleneck of knowledge engineering", that is the problem related to its representation and keeping up to date. Representing application domain involves a big effort, in which an expert, or a team of experts, on the application domain provides knowledge to a knowledge engineer, or to a team of knowledge engineers. Although users' needs are always kept under consideration, users only seldom take part in the construction of the knowledge base. The bringing up to date of the knowledge base is a time and effort consuming job too. The knowledge base cannot be kept continually up to date and the bringing up to date process become a discrete process, where the time interval between bringing up to date could be long sometimes.

Moreover, the decision of the appropriate level of representation of the application domain knowledge is difficult and can jeopardise the effectiveness of its use. A too in depth or a too superficial representation of the application domain knowledge could result in users' dissatisfaction. Novice users could find it too deep to be usable, while expert users could find it too superficial. Its use by the system in the process of interpreting the user information need or the document informative content could be uneffective.

What it is needed is a representation of the application domain knowledge which could adapt itself to the users' needs. An adaptive knowledge representation structure would enable the system to adapt to the user's knowledge level of the application domain required by the user. Moreover, an adaptive knowledge representation structure would be dynamic and able of keeping itself up to date. But, if a knowledge representation structure is able of keeping itself up to date it is also able to build itself up from scratch.

Subsymbolic representation structures are, often, adaptive. Whether they take as example the human nervous system (artificial neural systems) or the evolutionary system (genetic algorithms), they possess adaptive mechanisms and learning is often seen as a sort of adaptation to the environment. Thus, subsymbolic knowledge representation structures possess, in principle, the ability to help solving some of the problems related to the use of symbolic knowledge representation structures in IR. Some of these subsymbolic representation structure are also based on a associative representation paradigm. In particular, Artificial Neural Networks are adaptive systems based, as the name itself says, on a network representation.

The rest of the thesis is structured in this way:

Chapter 2 of this thesis presents a network model which could be used to represent conceptually an IIR application. The model enables the representation of the application and the application domain knowledge in a associative way.

Chapter 3 examines two possible processing frameworks which could be used on the conceptual structure. In particular, it focusses on the Artificial Neural Networks processing framework, which possesses adaptive capabilities.

Chapter 4 examines various ways to perform adaptive IR.

Chapter 5 reports some experiments aimed at investigating the possibility of using Artificial Neural Networks as tools for developing an adaptive and associative IR system prototype.

Chapter 6 examines critically the results achieved in the experimentation. It highlights the operational conclusions of the experimentation and points out some of the future directions of the work.

Chapter 2

A conceptual model for Associative Information Retrieval

> "The design of IR data has not yet been approached and studied as a complete process and in a structured way, as it has in the database area. Furthermore, suitable design tools for the design of IR data have not yet been developed. The result of this is that the designer of an IR application cannot call upon any complete and proven designed methodology or development environment with specification language and prototyping tools"
> [M. Agosti, 1990]

2.1 The need of a conceptual model

This chapter addresses the problem of conceptual modelling of an Associative IR application and seeks to produce a conceptual schema for representing IR objects and relations between them.

Previous research work by the author ([28, 29]) has already tackled the problem of conceptual modelling of IR data. A conceptual modelling paradigm necessary for the characterisation of IR data has been previously proposed, suggesting an approach based on a object oriented paradigm. The conceptual modelling paradigm has been used to produce a schema which represents objects and relations between objects in an IR application. This conceptual schema provides the user with a reference framework for the formulation of queries, and it is helpful for designing specific applications without reference to their particular implementation.

This chapter deals with the conceptual modelling of IR applications using a *network structure* as a modelling tool.

Scholars working in the field of IR have used network structures for various purposes since the 1960's. These structures have been employed to support browsing, clustering, spreading activation search, multiple search strategies, representation of user knowledge

and document or query content ([30, 13, 31]). Models using network structures differ and are often determined by the requirements of varying and diverse functionalities. So far, no general IR network model suitable for use as a conceptual modelling tool has been developed.

Generally speaking the main characteristic of a network representation is that it views an object in terms of its relations with other objects. The advantage of a network representation resides in its expressive power since the meaning of an information object can only be fully captured by considering its *semantic relationships* with other objects. In this light the complexity of IR data resides on relationships rather than on the data themselves. It is now widely recognised that further progress in the efficiency of IR systems requires a breakthrough in our ability to capture the semantics of data.

IR research is currently exploring various, and often different, techniques for improving performances of IR systems, though the importance of understanding the meaning of documents and queries is widely recognised. In this model the meaning of an IR object is realized by its relationships with higher level objects representing domain concepts. The structure arising from this approach is a network, structured on various layers, which is used as a conceptual schema of the application and as a designing and prototyping tool.

It should be noted that this work is not concerned with issues of physical storage of data. The network is seen as a conceptual structure whose actual implementation can use various data structures as required for efficient access and storage of the physical data. Moreover, one should also bear in mind that a network structure is entirely compatible with other forms of data representation such as vectors or matrices.

The rest of this chapter is structured in this way: Section 2.2 describes the basic structure of the network; the objects of an IR application are identified and modelled in terms of their relations. In Section 2.3 the network is used to develop a conceptual schema for modelling traditional IR applications. Section 2.4 describes the structure of a novel IR conceptual model using domain knowledge. The basic network structure outlined in Section 2.2 is there enhanced to account for domain knowledge, and some processing frameworks to be used on the new network structure are presented.

2.2 The basic network structure

The basic structure of the network is composed of objects and connections between objects.

An *object* is anything that has its own identity or uniqueness irrespective of whether it denotes a physical or a conceptual thing. The notions of object and of an object's identity are those of the object oriented paradigm ([32]). The use of the object oriented conceptual paradigm for modelling IR objects was introduced in [29] and here it is important to emphasize that an object can exist independently of its own characteristics or properties. This means that it is possible to establish relationships directly between objects without any reference whatsoever to the object's properties.

A *connection* expresses the existence of a relation between two objects. A connection can have an associated weight, which denotes the strength of the connection. Further,

a connection has a direction, and different directions within the same connection denote different relations between the two objects. Thus a connection can have different weights according to the different directions. Connections can be joined to build a new connection in order to relate objects not directly connected. The weight associated with this new connection is a function of the weights of the single component connections.

Using conventional terminology, a network is made of nodes that represent objects, and of links that represent connections.

The main IR objects are:

queries: a query is an expression of a user's information need. A query, however is a very subjective way of expressing an information need and different users, or the same user at different points in time, can express the same information need in varying ways. An information need can be expressed using one or more queries and queries can be more or less complex, but a complex query can always be expressed in terms of simple elements. A set of queries that express a user information need forms a query collection.

query descriptors: a query descriptor is any object used to represent a query using the query language of the system. A query can be asserted using one or more query descriptors, and the complexity of a query depends on the number and structure of query descriptors it uses. Some query languages such as the Boolean query language require the use of operators to express a query, but others do not. The query descriptors form the vocabulary which can be used to formulate a query and to express an information need, while the system provides the grammar and the syntax of the query description language. A set of query descriptors is an instance of a query representation.

documents: a document is any object carrying information which can potentially satisfy a user information need. For the purpose of this model there is no distinction between different types of documents, since a document is any object carrying information, so for example, a document can be a paper, a tape, or a picture. Usually the real documents are not part of the system; the system only contains identifiers which point to particular documents inside a collection. A set of documents form a document collection.

document descriptors: a document descriptor is every object which is used to describe the document informative content using the indexing language of the system. Usually more than one document descriptor is necessary to express the document informative content and the complexity of the informative content determines the number of document descriptors needed to represent it. Like the query descriptors, the document descriptors allow only a poor description of the real information content of a document; they form the vocabulary of the description language (indexing language), while the syntax and grammar are determined by the system. A document descriptor is an instance of a document representation.

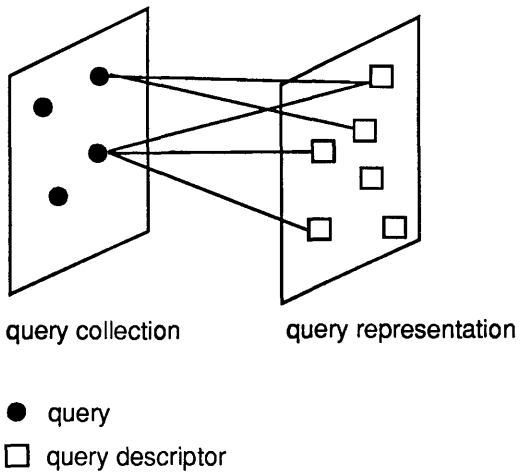

Figure 2.1: Query network.

Queries and document are usually not part of an IR system, they stand outside the system. Query descriptors and document descriptors are part of an IR system, instead, and they are derived from queries and documents by means of query processing and document indexing. Query processing and document indexing can be manually or automatically performed.

In the model similar objects are grouped in *layers*. Links can connect objects on the same layer or objects on different layers.

The network consists of two component networks: a **query network** and a **document network**. Figure 2.1 depicts a query network where a link means that either, the query is expressed using the connected query descriptors, or that a particular query descriptor is used by a query. Figure 2.2 depicts a document network where a link means that either a document is represented by some document descriptors, or that a document descriptor is used to express part of a document informative content.

Due to differences in artificial and natural query languages and in query processing methods, the same information need can be expressed using different query representations, so a query layer can be connected to different query descriptors' layers, as depicted in Figure 2.3. Likewise, due to differences in (natural) languages used in the document collection or different document indexing methods the same document collection (represented by a document layer) can be expressed using different document representations (different document descriptors layers) as is shown in Figure 2.4.

Conversely, the same document representation (document descriptors layer) can be used to represent the informative content of documents of various document collections (Figure 2.5).

Links connecting objects on the same layer represent relationships between similar objects. Examples of this type of relationships are document citations or thesaurus relationships between descriptors. These sort of connections will not be considered at this stage of the work.

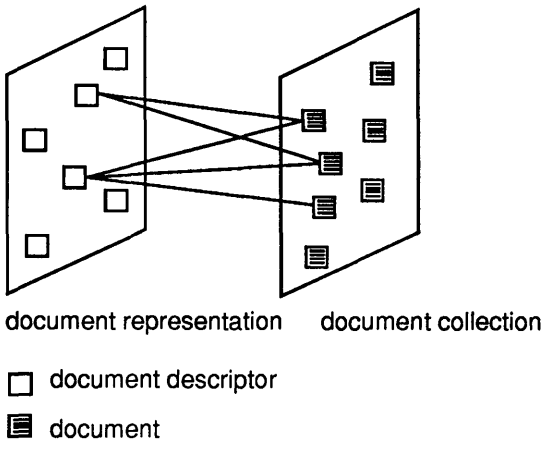

Figure 2.2: Document network.

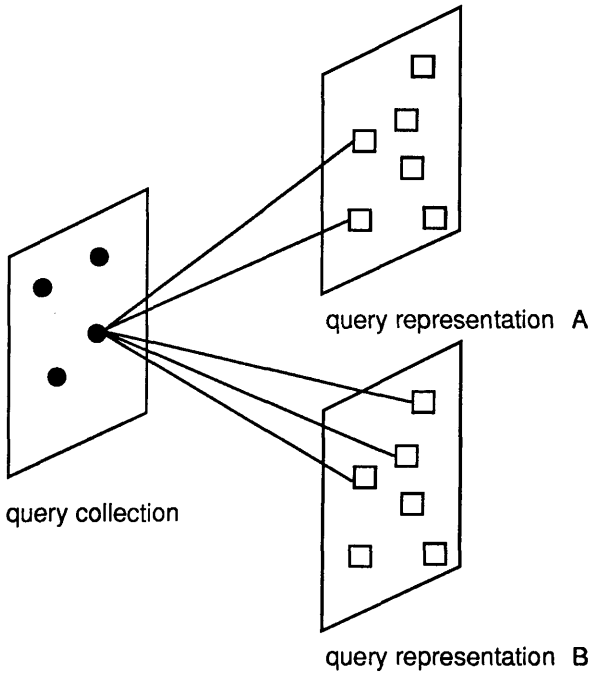

Figure 2.3: Multiple query description.

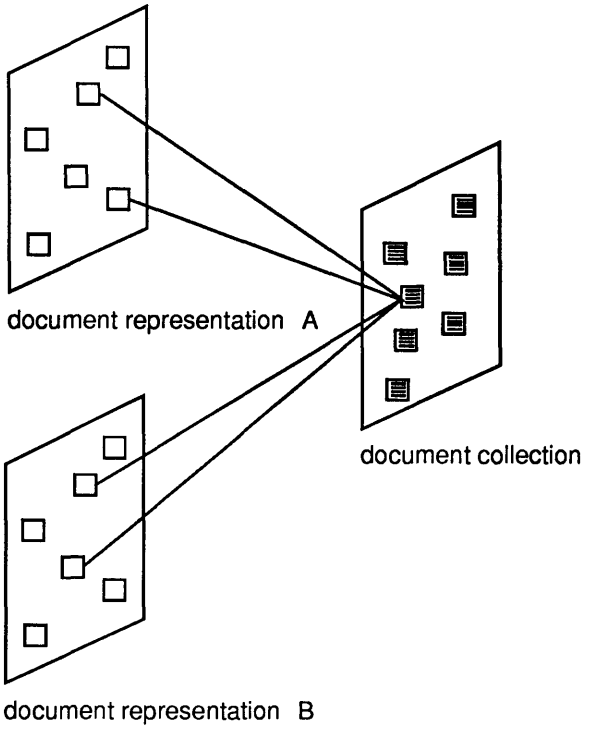

Figure 2.4: Multiple document description.

Hitherto queries and documents have been modelled as two distinct networks. At this point, however, it becomes necessary to establish a connection between these two networks to enable the retrieval of documents in response to a query. Further, the connection between the query and document networks is achieved by a **matching process**.

Generally speaking the matching process associates a value $S(q,d)$ to every couple query-document (q,d). This value expresses a *measure of similarity*, which can also be seen as a measure of association, between the query q and the particular document d and can be computed as a function of every different path $s(q,d)$ through the descriptors layers connecting the query to that document. This value can either range in a predetermined interval or be a binary value. The way the matching is performed and the way the similarity value is computed makes the distinction among different IR techniques. The model presented here allows conceptual modelling of traditional, as well as of new IR applications.

2.3 Conceptual modelling of IR traditional applications

It is possible to use the conceptual network so far presented as a tool for the conceptual modelling of IR traditional applications. In IR traditional applications ([1]) queries and documents are matched using the same set of descriptors for representing queries and

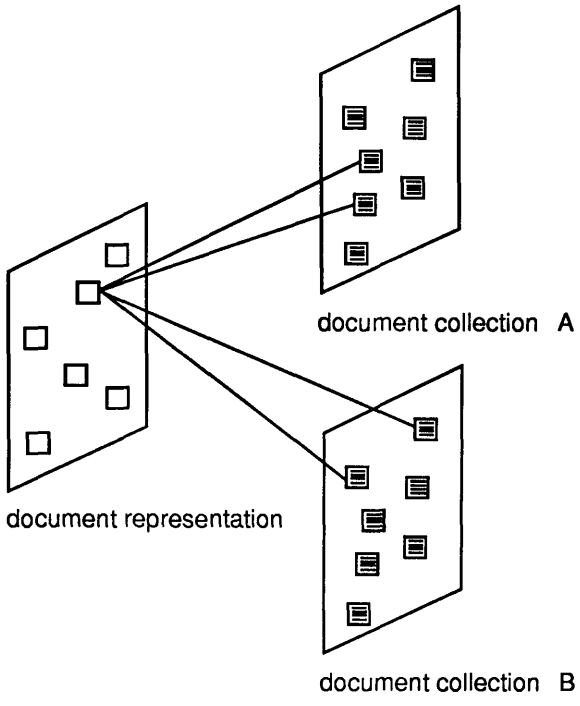

Figure 2.5: Use of various collections.

documents. This overlapping of descriptors allows the use of a single representation for queries and documents and, therefore, the model requires a single descriptors layer (Figure 2.6). The particular IR traditional technique in use determines the way the similarity value is computed.

In systems using an **exact match technique**, such as Boolean systems, the weights on the links and the values associated to paths query-document are binary values (0, 1) or logical values (T, F). Therefore the similarity value is calculated as:

$$S(q,d) = f_l(s(q,d)_1, s(q,d)_2, ...s(q,d)_n)$$

where:

$s(q,d)_i$: is a single path from the query q to the document d, and it is true if all the links it uses are true;

$i = 1, 2, ...n$: are all the different paths connecting the query q to the document d;

f_l : is a logical function which uses boolean operators.

In the previous formula $s(q,d)_i$ can be divided in two parts: $s(q,s_i)$ and $s(s_i,d)$. These are, respectively, the link connecting the query to the descriptor and the link connecting the descriptor to the document, respectively. It is assumed that $s(q,s_i)$ is true if the descriptor s_i is used by the query; and that $s(s_i,d)$ is true if the descriptor s_i is used to

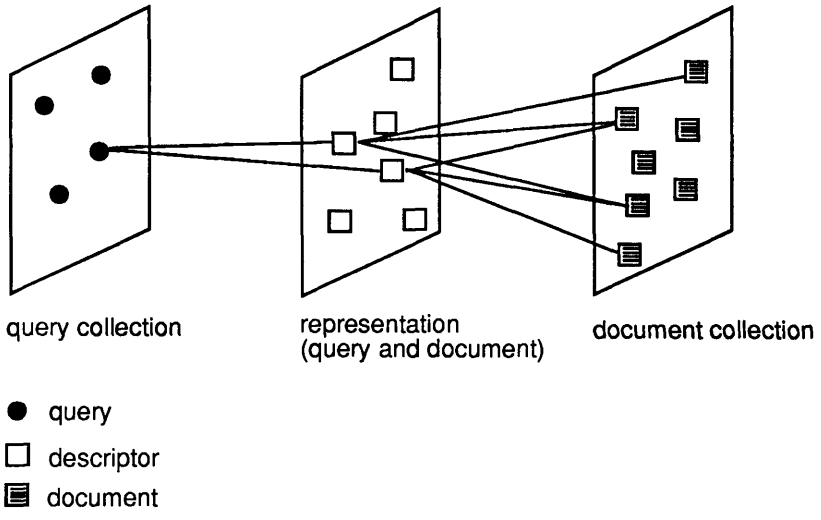

Figure 2.6: Association of the two networks.

represent part of the document informative content. The value of $s(q,d)_i$ is determined by the expression:

$$s(q,d)_i = s(q,s_i)\ and\ s(s_i,d)$$

$S(q,d)$ is determined by the logical function f_l which is the expression of the user's query. If $S(q,d)$ is true the document is considered relevant, otherwise it is considered not relevant. Hence exact match systems do not account for different degrees of relevance.

An example of conceptual modelling of a Boolean system is depicted in Figure 2.7. In this example the query q specifies a logical expression involving three descriptors, $s1, s2, s3$, thus links connecting the query to this three descriptors become true (T), while links from the query to all other descriptors are false (these links are not denoted in the figure). Two of the three above mentioned descriptors are also used to describe the informative content of the document d. In the example under consideration since the similarity value is evaluated to True the document d is considered to be relevant in respect of the query q. That is:

$$q = s1\ and\ (s2\ or\ s3)$$
$$S(q,d) = T\ and\ (T\ or\ F)$$
$$S(q,d) = T$$

In systems using a **partial match technique**, such as systems based on vector space or probabilistic models, the weights on links and the values associated to paths query-document are real numbers values belonging to a predetermined range. Therefore, the similarity value of the pair (q,d) is a real value which can be computed using the algebraic expression:

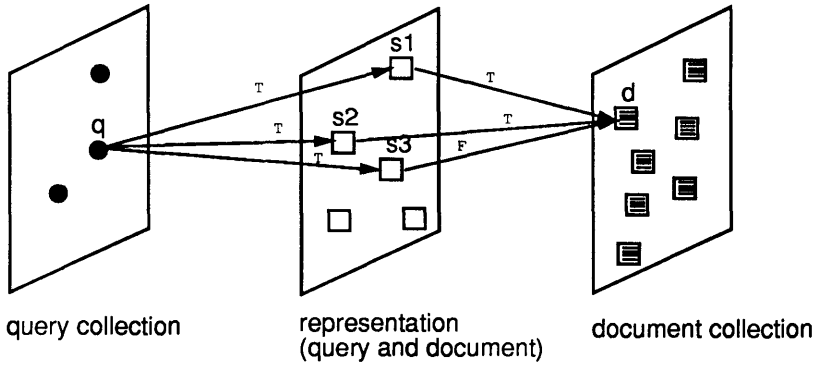

Figure 2.7: Modelling of a Boolean system.

$S(q,d) = f_a(s(q,d)_1, s(q,d)_2, ...s(q,d)_n)$

where:

$s(q,d)_i$: is a single path from the query q to the document d, and it is usually computed as the product of the weights of the two links, that is: $s(q,d)_i = s(q,s_i) * s(s_i,d)$. However, other functions can be used to compute the path value, like sum, mean, or max.

$i = 1, 2, ...n$: are all the different paths connecting the query q to the document d;

f_a: is an algebraic function.

Since similarity values are real numbers it is possible to rank them and thus organise the presentation of the relevant documents accordingly.

Figure 2.8 illustrates an example of conceptual modelling of a partial match which makes use of a a simple matching function. In this example links connecting the query to the descriptors, and links connecting the descriptors to the document are weighted (the weighting method is not important here). The similarity value is calculated as the sum of the products of the weights of the descriptors used by the query. The document is ranked with all other documents according to this similarity value, the relevance of the document being directly related to its position in the ranked list. That is:

$q = s1,\ s2,\ s3$

$S(q,d) = \sum_{s_i \in q} w_{q_i} * w_{d_i}$

2.4 Conceptual modelling of a knowledge based IR application

Some considerable effort has been devoted to the improvement in performance of IR systems, as described in the previous chapter. Recent work (see for example [6] suggests that significant improvements in retrieval performance could be achieved using techniques that

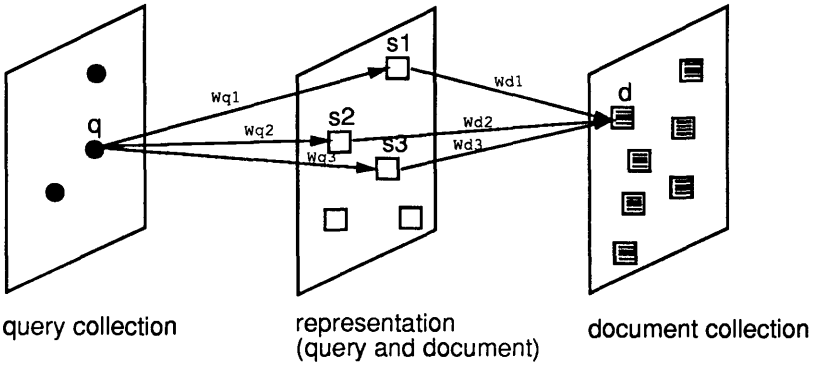

Figure 2.8: Modelling of a partial matching system.

"understand" the content of documents and queries. Techniques required for this kind of retrieval necessitate some degree of knowledge to enable the system to capture both the meaning of a document informative content and the user's information need expressed in the query. In particular there are two kinds of knowledge which any such system should have:

user knowledge: this is knowledge related to the characteristics of users and of user requirements. This type of knowledge is often captured by user models and it is used to understand queries and interact with the user;

domain knowledge: this type of knowledge is related to the subject which the documents belong to, that is to say, to the specific application domain. The wider the application domain the greater the difficulty in getting a complete and precise domain knowledge, this approach being more suitable to narrow domain applications.

In this thesis the use of a domain knowledge approach is preferred. User knowledge is assumed to be the subject of an intelligent user interface.

2.4.1 The knowledge network structure

Studies on knowledge representation in AI have developed representation techniques which have been subsequently used on several IR prototype systems. Some examples of the representation and use of domain knowledge in IR applications have been described in the previous chapter.

The idea of using networks to represent knowledge comes from an old concept in the information management system area that dates back as early as 1945 with Bush's MEMEX system ([33]). In his famous article Bush describe this system as working like the human mind, that is to say, by associations. Despite the progress made in technology and in interactive computer systems since 1945, today's systems have not yet lived up to Bush's vision, and studies on associative memory are still far from accomplishing these ideals.

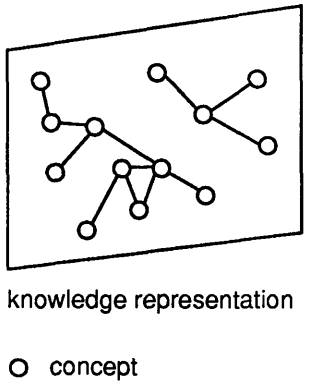

Figure 2.9: Knowledge representation structure.

At any rate, declarative network knowledge representations appear as one of the best ways of representing domain knowledge for IR applications, since such domain knowledge often capture the semantics of objects in terms of relations with other objects.

Using a network representation, the knowledge of a specific object can be enlarged or restricted by looking at its relations with other objects as a function of the knowledge level required (this is sometimes referred to as granularity). Different knowledge levels are in direct relation with the two main parameters used to evaluate the performance of an IR system: recall and precision. A deeper knowledge of an object, which comes from a precise identification of the object with relation to other objects, can enhance the precision of the system. Otherwise, a wider knowledge of an object, which extends to other objects related to it, can enhance the recall of the system, that is, it enables the system to identify objects which are different but in some way related to the specific object.

To add the domain knowledge to the network structure previously introduced, a different type of objects is necessary. The main difference between this new object type and the others lies in the fact that it carries information about what it has to "represent". Descriptors, on the other hand, carry information about what they are "describing", while queries and documents carry information about the user's information need or the author's document knowledge. Because of this distinctive characteristic these new objects are placed on a different layer and are called concepts.

A **concept** carries information about the conceptual object it represents, and in this sense a concept is similar to a frame as used in the AI field. The most important characteristic of a concept is that the information it contains is enhanced by the relations the concept has with other concepts. Figure 2.9 shows the knowledge representation structure in the form of a concept network. A concept is represented by a small circle while relationships between concepts are represented using links connecting concepts. Links are directional (the direction is not shown in the figures) and can be labelled to point out the type and/or weighted to show the strength of the relationship.

The concept network can be used to approach the modelling of a KBIR system. In a KBIR system the matching between query descriptors and document descriptors takes

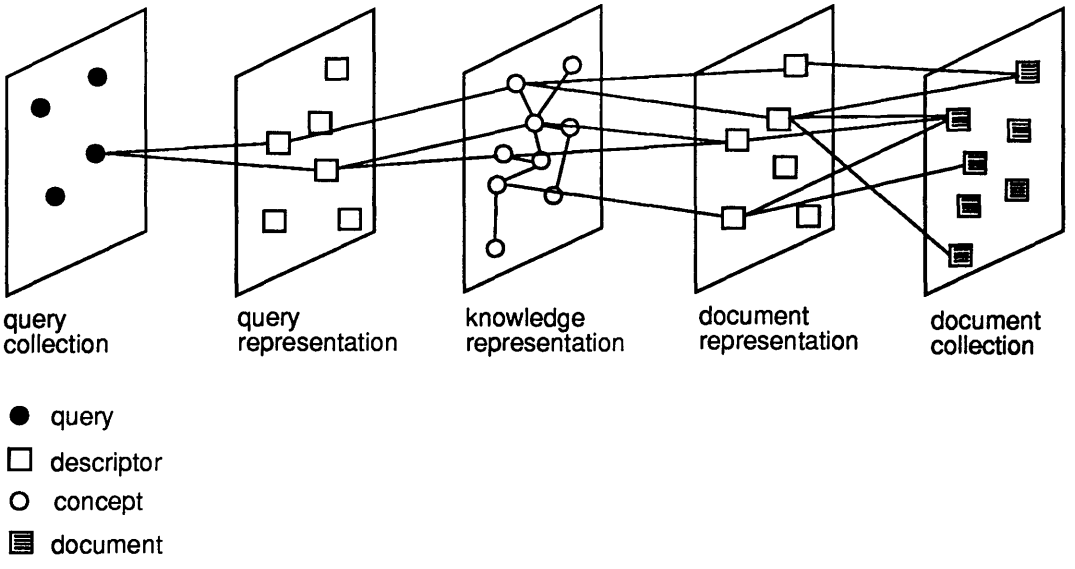

Figure 2.10: Use of the concepts layer

place on the concept layer as shown in Figure 2.10, where query descriptors and document descriptors are connected to the concepts they are describing.

In this approach to the modelling of a KBIR system some important assumptions are made:

- a descriptor is at a lower level than a concept. Descriptors depend upon the language or the symbolism used in queries and documents. Concepts are on a higher level which is independent from the language and the symbolism. If the matching is performed on a conceptual level there is no need to use the same language or the same symbolism for query descriptors and document descriptors;

- a single concept can be expressed using different descriptors and, less often, a single descriptor can express different concepts, that is to say, if descriptors are terms, a meaning can be expressed using different terms and a term can have different meanings;

- the same descriptor expresses the same concept whether it is used to describe queries or documents, so it is supposed that users and document authors use the same term with the same meaning. This assumption, however, is less strong than what it appears, as it will be explained latter on.

According to these assumptions the concept network can be used by the system to solve the major problem of an IR model: the matching between an information need and a document content which are expressed in different ways. The way the concept network is actually implemented depends on the particular knowledge representation model that has been chosen, but the previous assumptions must be present.

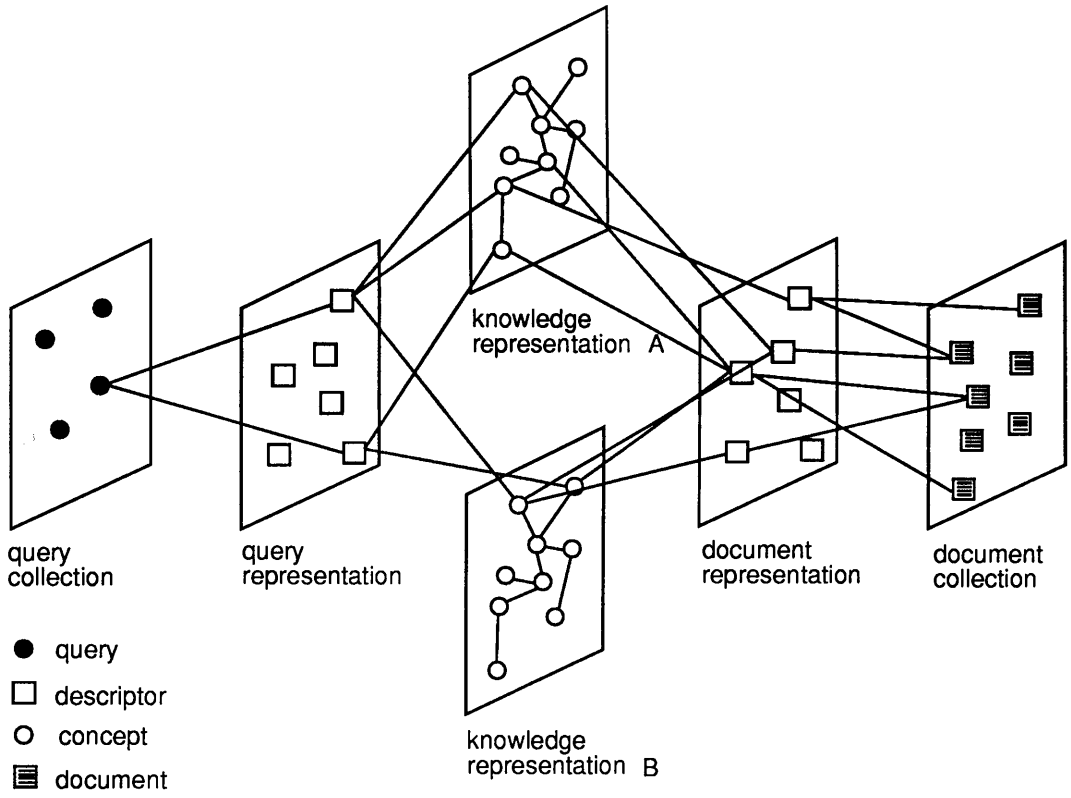

Figure 2.11: Multiple classification system.

The domain knowledge network (or concept network) is an independent part inside the main network model and the only relationship it has with the query network and the document network are the links connecting descriptors to concepts. Therefore it can be considered as a component network of the general conceptual network model.

Since the domain knowledge network is sometimes developed by more than one expert or the network, and can reflect different classification systems on the same domain, it is possible to have the situation depicted in Figure 2.11, in which the document network and the query network are connected via two or more different domain knowledge networks. Each network reflects a different encoding of the same domain knowledge.

Given the existence of different domain knowledge representations it is possible to organise them in a new layer which contains, as instances, all the knowledge representations of the same domain knowledge. This new layer, which in Figure 2.12 is simply called knowledge, provides the user or the system designer with information about the different domain knowledge representations. It is possible to think of it as a sort of *meta-knowledge*, that is, knowledge about knowledge.

The utility of having different domain knowledge representations is related to the possibility for the user (or the system itself) to choose the best domain knowledge representation or classification schema for the particular domain. Sometimes it is not easy to identify the

best application domain knowledge and it is difficult to say which classification schema is the best, even on a narrow domain. Moreover, some group of users could be familiar with a particular classification schema while other users of the same application could be familiar with another. A user might prefer to use his own classification schema and similarly a research team might prefer to build its own schema. On the other hand, if a domain knowledge representation reflects the view of a specific expert, a user could be interested to retrieve documents using views of different experts.

Another interesting possibility is depicted in Figure 2.13. In this example the same query acts on two different document collections which use two different classification schemas (or domain knowledge). In a modern environment in which resources are often distributed on complex communication networks and can be shared by different users, it will quickly become possible to interrogate different collections at the same time without any problem of different classification schemas, domain knowledge or languages.

2.4.2 The processing framework

A conceptual model is made up of two parts:

1. a data structure;
2. a processing framework.

So far the data structure has been described. An important characteristic of this structure is its flexibility. The flexibility of the conceptual structure presented, enables the use of various processing frameworks on it. The processing framework determines the way the weights on links are set and the way the similarity value is computed.

The description of the processing technique used in this thesis will be presented further on. In the rest of this section only the flavour of the various techniques will be given.

The processing framework is related to the interpretation of the network structure, that is to say, it is related to the meaning assigned to links and nodes in the network. In the following, three different interpretations are presented.

The network can be thought as a:

semantic network: if links specify semantic relationships between nodes then the network in Figure 2.10 can be seen as a semantic network. In a semantic network links are usually labelled according to their semantic meaning. The semantic meaning of links connecting queries to query descriptors or documents to document descriptors comes from their definition in Section 2.2.

The semantic meaning of links connecting descriptors to concepts is more complex because it involves the idea of the "expression" of a concept using terms. Descriptors, following the assumptions in Section 2.4.1, are expressions of concepts by means of different languages or simply by means of different terms. Further, the concept

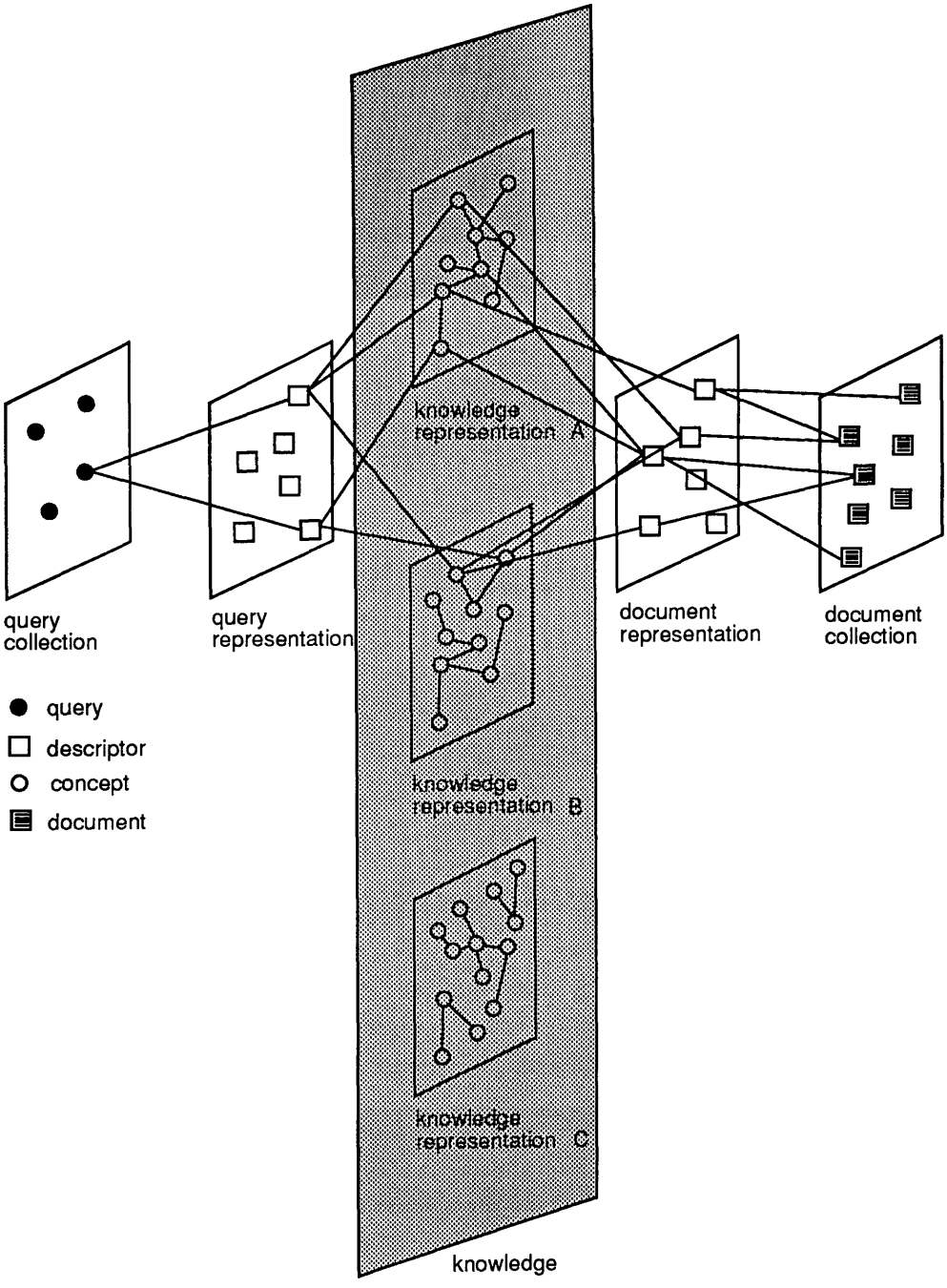

Figure 2.12: The knowledge layer

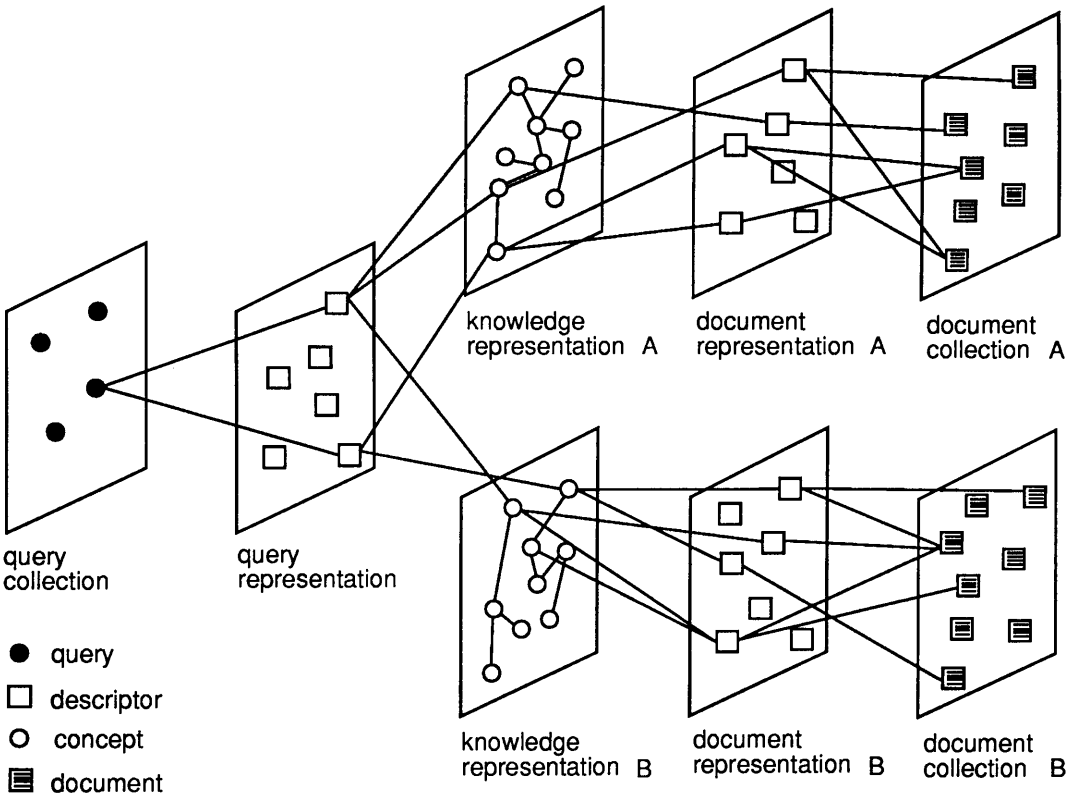

Figure 2.13: Query of different collection.

network is a real semantic network as it is often used in AI, because a concept is connected to other concepts to represent its semantic relationships. If the network represents a classification schema then the relationships between concepts can be the usual semantic relationships represented in a thesaurus and links are labelled according to this. More complex representations result from the expressions in the concept network of expert knowledge. Some examples of this are in [13, 14, 11].

There are several processing frameworks that can be applied on a semantic network. One of the most interesting is the *constrained spreading activation technique* ([13, 34, 35]) in which weights on links, assigned according to their semantic meaning, are used to spread activation from query to documents. The activation spreads from the query layer to the document layer using links between different objects. Some rules determine constraints on the spreading activation, otherwise too many nodes into the network would be activated (see Chapter 3). The spreading on the knowledge representation network adds the effect of using knowledge to the activation flow. The activation values reaching the documents determines the similarity value $S(q, d)$.

It is important to highlight how the spreading activation, working on the knowledge based network, can spread from the query descriptors to different document descriptors. These descriptors may express the same concepts indicated by the query descriptors or express concepts closely related to those pointed out by the user. This can help to solve both problems related to the matching of different expressions of the same concepts and problems related to an imprecise expression of concepts.

associative network: if links between nodes specify associations then the network can be seen as an associative network. Links do not need to be labelled because the focus is not on the semantic type of the associations. Links only need to have weights which express the strength of the associations. These weights can be calculated using different approaches.

An interesting processing framework for an associative network consists in the use of *parallel distributed processing* (PDP) models ([36, 37]). The use of PDP models enables to think to the three layers which are part of the system (query, knowledge and document representation in Figure 2.10) as a artificial neural network. An "artificial neural network" works as a pattern associator which can learn through training sessions to associate activation values on query descriptors to activation values on document descriptors (see Chapter 3). The knowledge representation layer becomes the hidden layer of the neural network and it actually represents knowledge in a non explicit way using the strength of its links. In fact, the associations the system learns in the training sessions are stored in the weights on the links.

In a query session, statistically determined weights could be associated to links connecting queries to query descriptors and to links connecting document descriptors to documents; they could be used together with the weights determined by the neural network to spread activation from queries to documents. Again, the activation values on documents determine the similarity values $S(q, d)$.

The automatic generalisation feature of some neural network models could provide good solutions to non trained query-documents associations. A user relevance feedback can be used to merge query sessions with training sessions so that the neural network can learn, modifying weights on its links, from every session.

The main advantage of this approach is that it does not required the explicit expression of knowledge in the knowledge representation network because the knowledge is automatically acquired from a user or expert feedback. Moreover there is no need to update it because the knowledge updating can be automatically performed using relevance feedback. However there are many problems related to the implementation of these ideas as it will be discussed further on.

inference network: this is another interesting processing framework proposed recently by Turtle and Croft in [31]. In this approach, links are interpreted as logic implications and they have a measure of probability associated on them. Different processing frameworks can be used on a inference network, such as methods based on Bayesian probability theory or the Dempster-Shafer theory of evidence. One of the major advantages of this proposal is the strong mathematical and probabilistic background upon which it is based. This assures the possibility of performing an in depth mathematical analysis of the behaviour of the network. For a more detailed description of this approach see the cited article.

In the following chapter an analysis of the first two processing frameworks will be presented. Examples of their use in IR will be also presented together with a comparison of advantages and disadvantages.

Chapter 3

Associative Information Retrieval

> "In our view, people are smarter than today's computers because the brain employs a basic computational architecture that is more suited to deal with a central aspect of the natural information processing task that people are good at."
> [J.L. McClelland, D.E. Rumelhart, and G.E. Hinton, 1988]

3.1 Associative processing frameworks

In this chapter two processing frameworks suitable for the use with the conceptual model of associative IR presented in the previous chapter are explained in more detail. They are: spreading activation (henceforth SA) and artificial neural networks (simply NN).

The advantages of artificial neural networks over spreading activation in the context of a processing framework for associative IR will be described further on. They explain the reasons why artificial NN were preferred in the experimental investigation reported in this thesis.

3.2 Spreading Activation

Historically speaking the use of SA preceded the use of NN, but it was not the first associative processing paradigm to be used in IR.

Studies on *associative retrieval* date as early as the 60s and had their origins in statistical studies of associations among terms and/or documents in a collection. The "associative linear retrieval model" is one of the earliest models based on the concept of associative retrieval. This model, in its basic idea ([38]), consists of expanding the original query using statistically determined term-term, term-document, and document-document associations. This technique is based on the assumption that there exists linear relations among terms, among documents, and among documents and terms. Associations between terms can be represented in a term similarity matrix. Quantitative evaluations of similarity between terms can be obtained by means of statistical analysis of term co-occurrence in documents.

Associations between documents as a quantitative evaluation of their respective similarity can be obtained by using similarities in the terms assignments to documents or by means of citations and other bibliographic indicators.

There are many heavy assumptions on this model and more recent studies ([34, 35]) have lead to the conclusion that effective term expansion methods valid for a variety of different collections are difficult to generate. IR systems based on this approach have shown a lack of consistent improvements in the effectiveness. This result can have various motivations. First, the similarities statistically derived from particular pairs of documents, or pairs of terms, may be valid only locally in the particular environment from which they are obtained. Second, most practical methods for computing the linear document associations are based on the assumption that terms or documents are originally uncorrelated, i.e., independent of each other. Such assumption is not accepted in many of the new research directions in IR.

Recently, these earlier models of associative retrieval has been revised using the so-called *spreading activation model*, which is based on supposed mechanisms of human memory operations. The basic SA model has, however, been subject to various enhancements in order to make it more suitable to IR and the way it is used in IR is quite different from its original formulation in the area of psychology (see for example [39]).

In the following three subsections the SA model will be briefly described. Section 3.2.1 will describe the "pure" SA model, which consists in the sole use of the associative nature of a network representation as a search controlling structure. In Section 3.2.2 some more search controlling structures will be added in order to give preference to particular associations. Section 3.2.3 will show how the search controlling structure can be used as an interactive process using feedback from users. Section 3.2.4 will report of some of the problems related to the use of SA in IR.

3.2.1 Pure Spreading Activation

The SA model in its "pure" form is quite simple. It is made up of a conceptually simple processing framework on a network structure.

The **network structure** (which is the same also for the following two sections) consists of nodes connected with links (Figure 3.1). Nodes model objects or features of objects of the "reality" to be represented. Links model relationships between nodes and they can be labelled and/or weighted. The connectivity pattern reflects the relations between objects and/or features of objects of the "reality" to be represented. This representation structure is very similar to a Semantic Network and indeed SA has been mainly used as a processing framework for Semantic Networks. An example of a semantic network is depicted in Figure 3.2. It is easy to note many similarities between Figure 3.1 and Figure 3.2

The **processing framework** is defined by a sequence of iterations as schematically described in Figure 3.3. Each iteration is followed by another iteration until the sequence is halted by the user or by the verification of some termination condition. An iteration

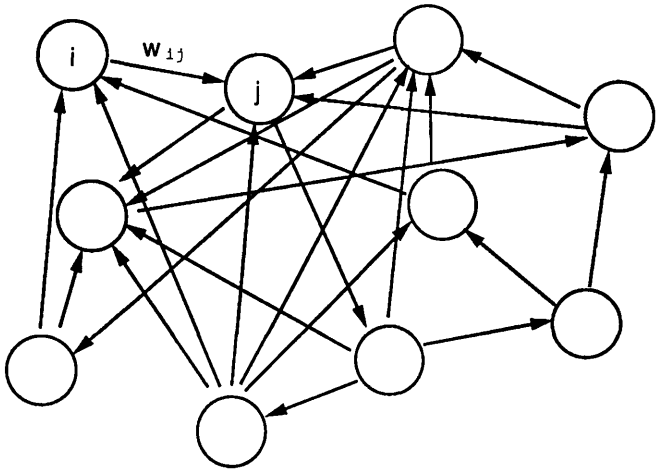

Figure 3.1: The network structure of a SA model.

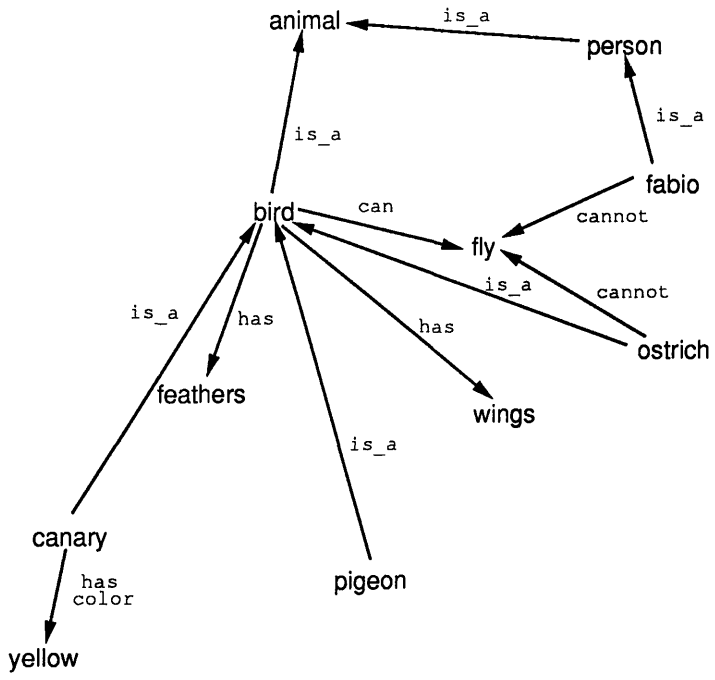

Figure 3.2: An example of the use of the network representation structure.

consists of:

1. one or more pulses;
2. termination check.

What distinguishes the pure SA model from other more complex SA models is the sequence of actions which composes the pulse. A pulse is made of three phases:

1. preadjustment;
2. spreading;
3. postadjustment.

In the preadjustment and postadjustement phases, which are optional, some form of activation decay can be applied to the active nodes. These phases are used to avoid retention of activation from previous pulses. This enable both the activation level of single nodes and the overall activation level of the network to be controlled. They implement a form of loss of interest in nodes that are not continually activated.

The spreading phase consists on a number of passages of activation weaves from one node to all the nodes connected to it. There are many ways of spreading the activation over a network (see [34]). In its more simple form and on a single unit level, it first consists in the computation of the unit's input using this formula:

$$I_j = \sum_i O_i w_{ij}$$

where:

I_j is the total input for the node j;
O_i is the output of unit i connected to node j;
w_{ij} is a weight associated to the link connecting node i to node j.

The input and the weight are usually real numbers, however their numerical type is determined by the specific requirements of the application to be modelled. In particular, they can be binary values (0 or 1), excitatory/inhibitory values (+1 or -1), or real values. Usually the first two of these options are used for network with labelled links, where the semantic value of the relation represented by the link determines, in the context of the application, the value to be associated to the link. The third option is used with weighted links to indicate the strength of the relation between two nodes.

After a node has computed its input value, its output value must be determined. The numerical type of the output of a node is also determined by the requirements of the application. The two most used cases being the binary active/non-active type (0 or 1) or a real value. In SA models there is usually no distinction between unit's "activation" or

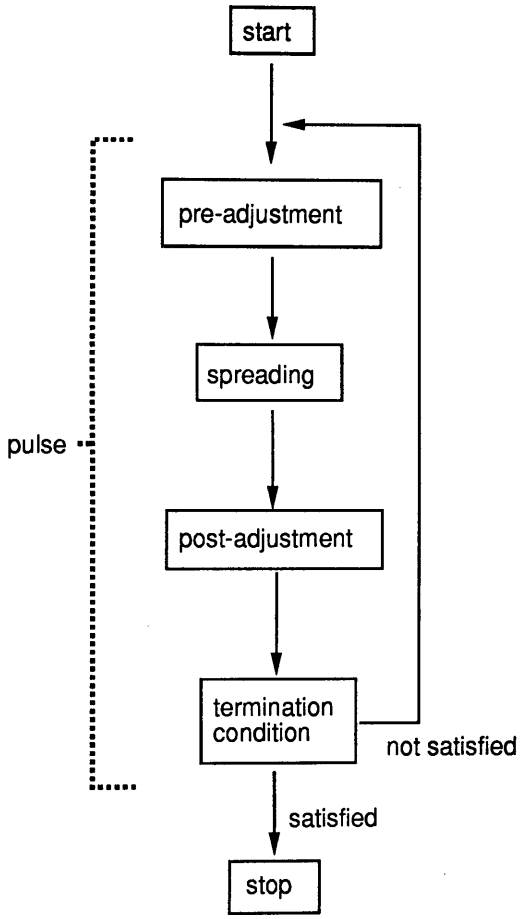

Figure 3.3: The pure SA model.

unit's "output". The activation level of a unit is its output value. This is usually computed as a function of the input value:

$$O_j = f(I_j)$$

The most common function for pure SA models is the threshold function. It is used to determine if the node j has to be considered active or not. The application of the threshold function to the above formula in the case of binary value units gives:

$$O_j = \begin{cases} 0 & I_j < k_j \\ 1 & I_j > k_j \end{cases}$$

where k_j is the value for the threshold for the unit j.

The threshold value of the activation function is application dependent and can vary from node to node. After the node as determined its output value, it fires it to all the nodes connected to itself, usually sending the same value to each of these.

Pulse after pulse, the activation spreads over the network reaching nodes that are far from the initially activated ones. After a determined number of pulses has been fired a termination condition is checked. If the condition is verified than the SA process stops, otherwise it goes on for another series of pulses. SA is therefore a iterative process , consisting of a sequence of pulses and termination checks.

The activation level of the nodes reached at termination time is the result of the SA process. The interpretation of the level of activation of each node depends on the application and, in particular, on the characteristics of the object being modelled by that node.

3.2.2 Constrained Spreading Activation

The pure SA model, however, presents some serious drawbacks:

- unless controlled carefully by means of preadjustment and postadjustment phases the activation ends up spreading all over the network;
- there is no full use of the information provided by the labels associated to the links, that is, there is no complete use of the semantics of the relationships;
- it is difficult to use some form of inference in a SA model without using in different ways links with a different semantic meaning.

These problems can find a solutions by modifying the processing framework so to consider the diverse significance of the relations among units. This can be achieved using the information provided by the labels on the links and by processing links in different ways according to their semantic meanings. It is possible in this way to implement some form

of heuristics, so that the activation on the network is spread according to some inference rules. A common way of implementing a processing framework which spreads the activation according to inference rules, is by means of constraints on the spreading. Here are some constraints commonly used in SA models:

distance constraint: the spreading of activation ceases when it reaches nodes that are far, in terms of links covered to reach them, from the initially activated ones. This corresponds to the simple rule that the strength of the relation between two nodes decreases with their semantic distance. Relations can be classified according to their distance in term of links. Relations between two nodes directly connected are called first order relations. Relations between two nodes connected by means of an intermediate node are called second order relations, and so on. It is common to consider only first, second and, at most, third order relation, however this depends on the context of the application.

fan-out constraint: the spreading of activation ceases at nodes with very high connectivity (or fan-out), that is at nodes connected to a very large number of other nodes. The purpose of this constraint is to avoid a too wide spreading of the activation which could derive from nodes with a very broad semantic meaning and therefore connected to many nodes.

path constraint: activation spreads using preferential paths, reflecting application dependent inference rules. This can be modelled using the weights on links or, if links are labelled, diverting the activation flow to particular paths while stopping it from following other non meaningful paths.

activation constraint: using the threshold function at a single node level, it is possible to control the spreading of the activation on the network. This can be achieved by changing the threshold value in relation to the total level of activation over the entire network at any single pulse. Moreover, it is possible to assign different threshold levels to each unit or set of units in relation to their meaning in the context of the application. Although the use of activation constraints causes a increase in the computations, it makes possible to implement various inference rules.

Referring to Figure 3.3 these constraints can be seen as acting during the preadjustment phase (distance, fan-out, and path constraints) or during the postadjustment phase (mainly activation level constraints). Therefore they can be considered as an enhancement of the pure SA model.

Another more practical advantage deriving from the use of constrained SA is also that the activation does not spread over the entire network because some constraints are used to cease it when it is no more meaningful. This enables a reduction of the computational effort for a system using SA because only a small portion of the network's units becomes active and send activation.

3.2.3 Spreading Activation with feedback

A further enhancement of the pure SA model can be obtained by means of feedback from an external source. In this case, an external evaluation of the activation level of some units or of the entire network provides some constraints that would be difficult or impossible to implement in the form of automatic rules. This external feedback can be due to another process or can be provided by the user of the system. The user evaluates the activation level reached by some nodes and modify it according to his requirements. This may result in a following spreading of activation over particular user selected paths which differ from those specified by predetermined path constraints. From this point of view SA adapts itself to the specific user's needs.

This model is particularly useful in application where there would be too many inference rules to be represented in the form of constraints and where it is necessary to provide an external control by means of a user's evaluation of the results. The use of feedback from user can be either be implemented in the preadjustment phase, so that the user directs the spreading of activation of the pulse, or in the postadjustment phase, enabling the user to evaluate the result of the spreading and direct the following pulse accordingly ([34]).

3.2.4 Spreading activation and IR

The SA techniques used in IR are based on the existence of maps specifying relations between terms and/or between documents. Using these relations the SA model can easily be applied to IR.

Nodes correspond to terms, documents, articles, journals, subject classifications, and authors. There is no need of homogeneity in the network. A node can represent anything. Links indicate the association of a node with another node, as, for instance, an author with a document he/she wrote or a document with a document it cites. An example of a fragment of a document collection representation is shown in Figure 3.4. Specific link types include term occurrence, document publication, term assignment by indexing, document authorship, document assignment to classification, document citation, and so forth. The set of node types and link types is determined by the data available and by the purpose of the application. The representation structure is therefore application specific and the same structure can not be applied to different applications. It is also important to note that relationships could actually be expressed by a pairs of links. Authorship, for instance, can be represented by both "authored by" and "is author of" links. Both links in such pairs connect the same two nodes, but their source and destination roles are reversed. Specific processing rules may inhibit activation in either directions, and use them in different ways, or associate different weights to different directions.

Given such a representation structure the network activation starts by placing a specified activation level at some starting term or document node. These nodes are usually identified by the initial query formulation or by documents or terms retrieved by an earlier search operation, the second option being often used in systems with relevance feedback. The activation first reaches nodes located the closest to the starting nodes, then it spreads

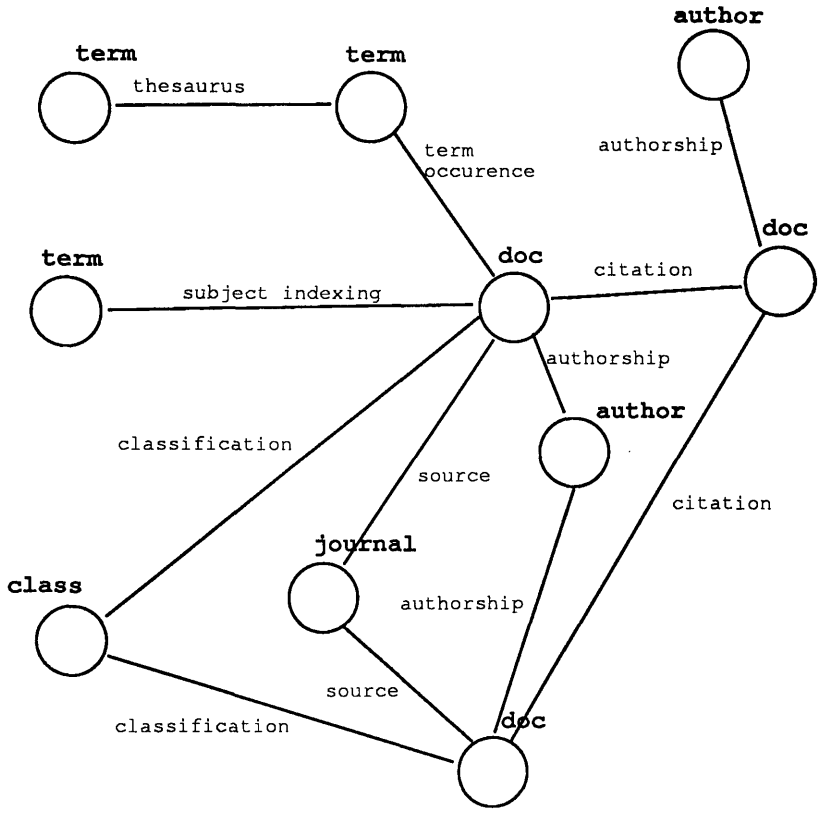

Figure 3.4: An example of document collection network representation.

through the network using links. The activation level of a node is computed using one of the functions specified above in Section 3.2.1. The process ends when some termination condition is reached. The activation level of documents at the end of the spreading process is used to compute the relevance level of each document.

Most of the SA processing frameworks used in IR systems differs from the pure SA models in several respects:

- the activation level of a node reached by the spreading of activation is determined by the starting activation level and the type of nodes and links traversed before reaching it;
- distance constraints are usually imposed by stopping or degrading the activity at some specified distance from the original node;
- nodes with a large branching ratio (fan-out) may receive special treatment in order to avoid a too large spreading of the activation on the network;
- the activation process follows specified rules, which try to mimic some sort of inference in the process of associative retrieval.

It has been demonstrated in [34] that better retrieval results can be obtained by a SA process which uses some of the above mentioned characteristics.

Much of the effectiveness of the process is, however, crucially dependent on the availability of a representative network. The problem of building a network which effectively represents the useful relations (in terms of the IR aims) between nodes representing documents and/or terms has always been the critical point of many of the attempts to use SA in IR. Most of the times these networks are very difficult to build, to maintain and keep up to date. Their construction requires a deep understanding application domain knowledge which only experts in the application domain can provide. Furthermore, their construction is a very expensive and time consuming process, which is impossible for collections consisting of a large number of documents and terms and/or spanning over a large application domain. Although it can be argue that it is possible to build up quite easily a network using statistically determined associations between terms ([40]), it must be noted that in this case the network cannot be truly considered a semantic network and it representational power much depends on the statistical methods used to build it up. Furthermore, the use of online thesauri to build up the network always requires the existence of a thesaurus covering terms belonging to the application domain, which is not always easy for very specific application domains.

At last, a complete SA system that makes use of diverse link types and of spreading rules with distance and fan-out constraints, has never been implemented with ordinary document collections. Various prototype systems are presented in the IR literature but no commercial system based on SA models actually exists.

Given these problems, a different approach to the processing of the conceptual model was preferred in this thesis. The NN approach, which will be presented in the following

part of this chapter, seems to possess the potentiality for solving some of the problems the SA approach has got.

3.3 Artificial Neural Networks

Artificial neural networks models or simply neural networks (NN) go by many names such as connectionist architectures, parallel distributed processing (PDP) models, and neuromorphic systems. Whatever the name, they consist of many simple, highly interconnected processing units which act as a whole to perform computable functions. These assemblies exhibit properties of associative memory, recognition, search, learning, and computation.

Artificial NN structures are based on our present understanding of biological nervous systems. The amount and type of biological detail retained in a given model depends on the purposes of the model. Often the abstracted features are extended beyond what is known in neurobiology, with diverse goals such as ease of simulation, suggestion of principles to seek out in biological preparations, and conceptual computational modelling. Many researchers are not interested in the biological implications, and construct networks purely as problem solving devices. Once the computational model is divorced from biology, the computing algorithm might be described as neural only for historical reasons. The back-propagation procedure (see further bellow) which is most often cited in applications papers has no known biological correlate.

Conversely our lack of detailed understanding of the working of the brain leads neurobiologists to use these models as research tools. Some representative objects of simulation are modelling of brain structures such as the visual cortex, natural language processing and understanding, optimisation problems, high density routing tables, and memory systems.

3.3.1 Biological neurons and networks

Real NN, such as the human brain, are far more complex than artificial NN. A real NN consists of many neural cells, or neurons which come in many shapes, sizes, and configurations. A representative neuron consists of a cell body or soma, a number of input fibres or dendrites, and an output axon fibre (Figure 3.5). Cells are connected to each other through synapses, which are electro-chemical junctions usually occurring between axons and dendrites. Each neuron can have from 10^3 to 10^5 connections with other neurons. Information through the connections is carried in electrical signals which flow in only one direction. Electrical activity, say from a dendrite attached to a sensory transducer, flows down the dendrite and into the cell body. This may cause the cell to generate a burst of electrical activity which is transmitted outward along its axon. The axon becomes the input to other cells. The details of these events are very complex and interrelated and are active areas of research.

Neurobiologists are trying to use artificial NN to study the characteristics of real NN. However, the research is confused by the fact that neural simulations are performed at present mostly on serial machines. This is in contrast with the architecture of real NN,

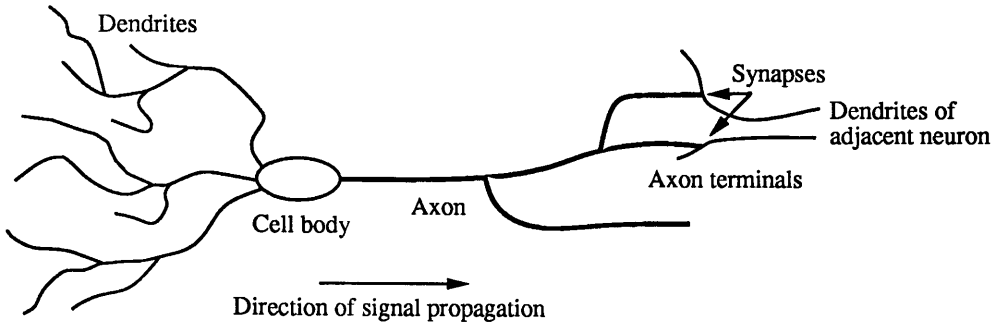

Figure 3.5: Biological neurons

which is sometimes described as "massively parallel". Even when researchers use hardware described loosely as parallel, then the implementation may involve pipelining of information through a single CPU designed for vector multiplications. Alternatively, it is possible to partition the calculation between independent coarse-grained processors such as transputers. However, each transputers carries out the work of many neurons. On the other hand, all of these implementations use a number of computing units and connections which is relatively small in comparison with the number of neurons and synapses in a real NN. Massive parallelism has not yet arrived for artificial NN hardware.

3.3.2 Simulated neurons and networks

Artificial NN (henceforth only NN) models are specified by:

1. node characteristics;
2. net topology (or net architecture);
3. training and learning rules.

A schematic diagram of a **node** is seen in Figure 3.6. Each node is connected to a number of input lines and output lines. Signals in biological neural networks are based on trains of equal valued spikes at different frequencies. Most simulations use real numbers to represent these signal frequencies at discrete time steps. Thus the activity level of a unit is a real number, as are its inputs and outputs. Input lines are attached to a unit through connections of usually variable strength, which are also represented by real numbers. The contribution of one input to the unit's total input is the product of the activity on that line with the value of the connection strength. The total *input* to the unit (I) is the sum of all the individual products. Connections may be positive or negative in sign. The former adds activity to the total, while the latter subtracts from it. Therefore the total input of a unit j is evaluated as:

$$I_j = \sum_i (O_i w_{ij})$$

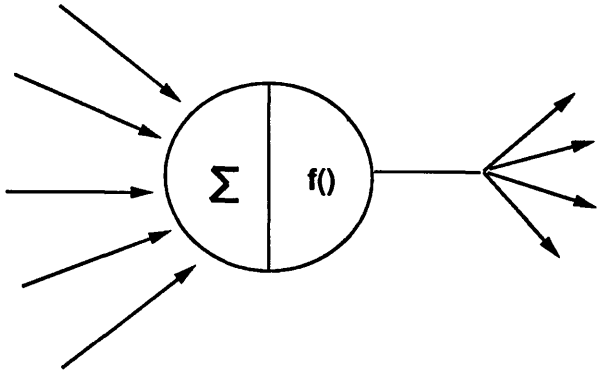

Figure 3.6: Simulated neuron

where O_j is the output of the unit i connected to unit j through the connection whose strength is expressed by the weight w_{ij}.

The *activity* value of a node (A) is typically defined to be a non linear function (f) of the total input. Some commonly used activation functions are shown in Figure 3.7. Each unit may have associated with it a "bias", which is the activity value of the unit at rest. This is similar to the resting potential of biological neurons, though it is often viewed as an additional input to the simulated neuron. The activity values of the unit j is computed as:

$$A_j = f(I_j)$$

The *output* of the unit (O) is a function of the activity (g). In many models the output value is the activity value itself, that is the output function is just the identity function. Therefore:

$$O_j = g(A_j)$$

It is worth repeating that this is a description of a representative simulated neuron. There are many NN models. They differ in the form of the input, activity, and output function and in the way they compute the bias. Variations such as temporal summation, delayed outputs, multiplicative (rather then additive) synapses, and internal memory of recent activity often occur in specialised systems.

The **architecture** or connectivity pattern of a network describes its physical (virtual, if inside a computer simulation) layout. Spatial relations between units may be specified along with connections between them. Usually, the spatial layout of a network is just a convenience for the designer, since the topology can often be changed while preserving the connectivity pattern. The connectivity pattern is therefore the specification of which unit's outputs connect to which other units as input. To facilitate this, the units in a network are often partitioned in groups. These may be seen as functional subnetworks, layers in a

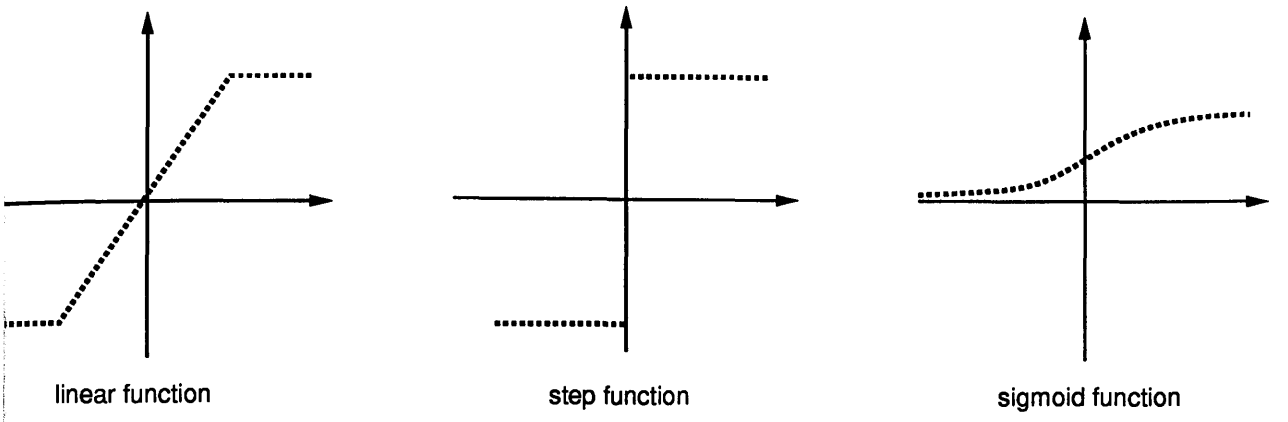

Figure 3.7: Common activation functions

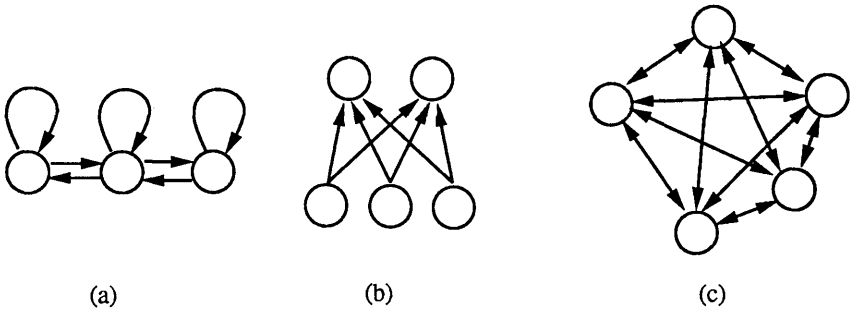

Figure 3.8: Common network architectures: a) single layer network b) two layer network c) fully interconnected network

stratified structure, or just as divisions which represent the intentions of the designer. The layers are usually distinguished by restricted connection specifications, such as allowing no connections among the units in a group, but full connectivity between groups. The direction of propagation of the signals among units is usually restricted and formalised. Some examples of net architecture are depicted in Figure 3.8. It is important to notice the similarities between the NN architectures and the architectures used by SA. Both are based on a network structure which is the most suitable for Associative IR.

The **learning rule** is concerned with the way the training and learning are performed on the already defined unit's and network's characteristics. The learning rule is usually what characterises the model the most. The general characteristics of NN learning and the description of some procedures are presented in Section 3.6.

3.4 Local and distributed representations

The structure of a network generally reflects the intended interpretations of its layers and units. The individual set of neurons in a multi-layered network might represent the notions of input, output, and processing structures. Within the set of neurons, individual units may have a local or distributed interpretation.

In a *local interpretation*, which is typical of symbolic representations, units may correspond to such high level notions as words, concepts, terms in an equation, or categories. For example, a set of input units might represent line segment detectors, with connections to another set of units which represent alphanumeric characters, with connections to another set of units which represent words, with connections to another set of units which represent categories. The term local is used in reference to the idea that these qualities are localised to specific units. Extracting meaning from this kind of network is relatively simple. The activity level of each unit represents the amount to which the indicated concept is involved in the current state of the network. A list of all the units' meanings and their current activity levels can yield a complete description of a network's current "thought". Different patterns of activity across the units represent different "thoughts".

At the other end of the spectrum are *distributed interpretations*, typical of subsymbolic representations, in which such localisation of concepts to units is not possible. Rather, units correspond to so-called micro-features of the environment. A local unit is replaced by a group of units whose activity levels as a group represent the quality in question. No single unit can be said to represent any describable entity. This makes the interpretation of activity patterns more difficult, but makes their interactions more subtle. Many concepts can be represented and thus interact in a single group of units. This type of representation also has the advantage of better fault tolerance and noise immunity, in that failure of any single unit should not greatly disrupt the system as a whole. In contrast, failure of a local unit which represents a word means that the word can no longer be expressed. It is important to notice that, while local representations define and limit the concepts available to a system, distributed representations allow flexible interpretations of what concepts are in existence or are available.

Many issues in network and problem representation in general are complex and poorly understood. Most traditional artificial intelligence research has focussed on local representation of symbols and meaning, probably influenced by the local nature of traditional computer architectures. Connectionist systems appear to alleviate this constraint to some degree, but much more work is required before any claims of superiority of one over the others could be made. A major problem with distributed representations is the difficulty in assigning meaningful activity patterns to concepts and hence to sets of units. Unless this is done carefully, systems which attempt to be distributed in nature can become nothing more than jumbled and complex versions of local instantiations.

Along with the information contained in the definition of units, there is information in the connections between them. This information is the most important characteristic in some applications. The presence of a unit (or group of units) defined as a concept means that the concept is available to the system. Similarly, a connection between two units

or groups of units implies that the two concepts may be related and thus influence each other. In the case of a more distributed representation, the connectivity pattern within the network can restrict and shape the activity patterns which are likely to occur. While this is also applicable to local systems, in the case of distributed representations it deeply affects the way concepts are realized.

A network connectivity pattern generally defines potential connections among units, but not the specific strengths. It is these strengths which embody a given system's knowledge under its framework. Connection strengths may either be part of a system's definition, or may be learned by the system through experience. The former scheme is usually restricted to local representations, where the correlations among predefined-defined conceptual units are known. The most interesting set of networks learns these correlations through experience, and this is what learning procedures are for.

3.5 Learning and memory

A widely held theory concerning the neurological basis of memory is that of long term potentiation (LTP), or Hebbian learning.

The concept of Hebbian learning, which first appeared in [41], specifically refers to the statement:

> "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased".

This was initially offered as an explanation of how information arriving in the brain could be kept during processing. Connections which quickly adapt to an activity pattern could store that pattern as long as necessary. Such structures form the basis for learning in cognitive systems. Hebb suggested, with no physiological evidence, that a change in the area of contact of synaptic knobs was responsible. Sufficient evidence now exists such that this is the predominant paradigm today. It is supposed that this increases in synaptic connection strength is related to the firing frequencies of the two cells involved.

In a simulated network, the product of the activity values of two units often determines the changes in their connection strength. This is generally what is referred to as Hebbian learning in connectionist systems (see further below). Variations commonly arise when the values used in the product are not the cell activity values themselves, but are functions of them. This is often done to include error correcting procedures. Neighbouring cells and even more global interactions may also play a role in the formulation of the learning rule.

The notion of memory in traditional computer systems differs greatly from its conception in human cognition.

"Traditional computer systems" refers to serial Von Neumann architectures with separate processing and memory systems. Information is stored in discrete locations within some storage medium. Access to that information can only be made through its 'address',

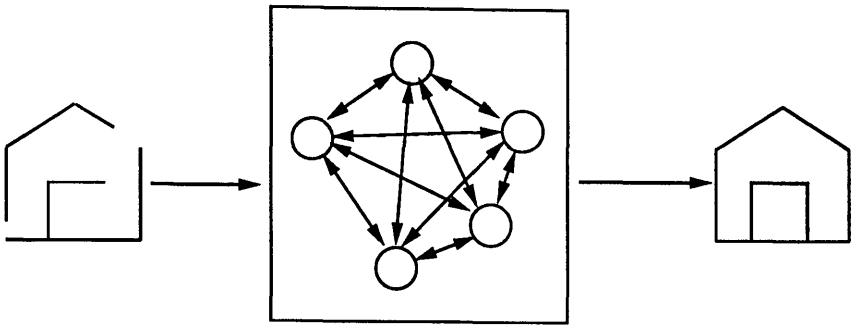

Figure 3.9: Auto-associative memory

or known physical location. When the content of the information is known but not its location, a search must be performed. In this case, the content must be known exactly, or else some pattern matching function must be used. Human memories, in contrast, are accessed only by content, with no regard to physical location. Stored information (events, names, faces) are recalled by thinking about something related, or associated with that memory. Additionally, the information used as the key need only to be partially complete or correct. Incorrectly spelled names or partial views of objects usually do not impede the acts of recognition or recall. The inability of traditional computer systems to perform such operations is undoubtedly a major obstacle in many areas of research. These most natural and prominent cognitive operations are difficult for these systems to emulate.

Two types of associative memory, auto- and hetero-associative, are often discussed. *Auto-associative memory* generally refers to the ability of a system to recognise and correctly recall information from partial or corrupted versions used as input. *Hetero-associative memory* systems 'couple' different information. If two items are stored as a pair, the presentation of one recalls the other.

Connectionist systems are well suited to perform associative memory operations. With a simple learning rule, a group of completely interconnected units (every unit connects to every unit) can perform auto-associative memory (see Figure 3.9). A pattern of activity is set up in the network (through dedicated input lines), and the synapses between the active cells are increased in strength. When a subset of that pattern is presented to the group, sufficient activity can spread through the strong connections to enable the missing units to fire, reinstating the original pattern. The information needed to recall a pattern is distributed throughout the connections in the system. Missing or inaccurate connection strength values do not greatly affect performance, as they represent only a small amount of the overall informational content. Many of these distributed representations can be stored simultaneously in the same set of connections. The fidelity of this kind of storage can depend on pattern independence, network size and structure, and learning specifications. Interference between stored patterns can be seen as a hindrance or as a benefit, depending on the aims of a particular model. Bad interference overwrites memories, while good interference allows generalisation.

Systems with two or more layers of units can perform hetero-associative memory (see

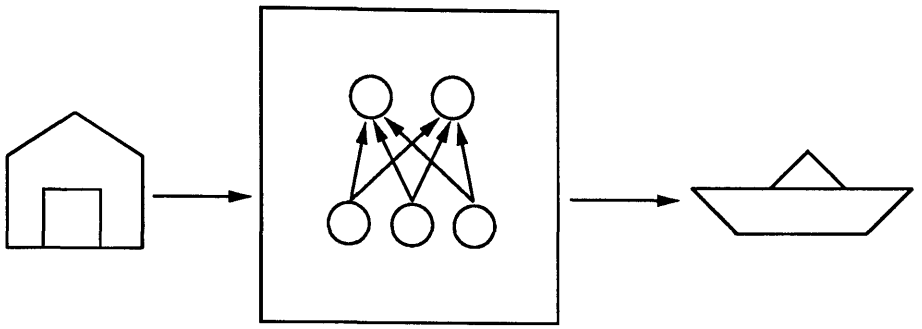

Figure 3.10: Hetero-associative memory

Figure 3.10). Patterns are presented to each group, and the connections adjust to enable the mapping from one pattern to the other. A previously learned pattern is presented to one of the groups, and activity flows to the other layer, recreating the second pattern there. Flow of activity may or may not be restricted to one direction.

The mechanism which distinguishes learning from recall in biological memory, if one exists, is not known. However, almost all simulations have distinct operations for them, and their use is under system level control.

3.6 Neural Networks learning procedures

Generally speaking a learning procedure or a learning algorithm for an artificial NN is defined as method for changing the weights of the connections between nodes during the training phase in order to improve the recall performance of the network. With regard to the word "algorithm" it has to be noted that a learning algorithm for an artificial NN is not assumed to supply, after a finite time period, the desired result. The word procedure seems therefore more appropriate to name them, and in the following this will be used.

In a more mathematical way, learning is defined as any change in the matrix W, which represents the weights of all the connections in the network. So:

$$learning = \frac{dW}{dt} \neq 0$$

where:

W is the weight matrix, so that an element w_{ij} is the weight associated with the connection between node u_i and u_j;

t is the time variable.

A learning procedure is said to be a *static* one if the NN has to settle into some kind of equilibrium state before the weight changes are carried out (e.g. a state where the units' activation values do not change any longer). On the other hand, a learning procedure is said to be *dynamic* if weight changes are allowed even before the NN reaches an equilibrium state.

The core of each learning procedure is its *adaptation rule* which specifies the amount of dW, that is the amount of change a weight w_{ij} has to perform. Examples of adaptation rules are the error correction rule, the reinforcement rule, and the Hebbian rule.

An *error correcting rule* (sometimes called a delta rule) is defined as the adaptation of the connection weights in proportion to the difference between the desired and computed values of each unit's output. So:

$$\Delta w_{ij} = \alpha o_i \delta_j$$

where:

α is the learning rate;
o_i is the computed output of unit u_i;
δ_j is the error signal of unit u_j, and it is defined as:

$$\delta_j = (d_j - o_j)$$

with:

d_j: desired output value of unit u_j;
o_j: computed output of unit u_j.

The learning rate, also called synaptic plasticity coefficient, determines the magnitude of weight changes and hence is crucial to learning performances. Usually the value of the learning rate is between zero and one.

The problem of finding the appropriate learning rate is quite difficult. In fact a small learning rate implies small changes of the weights and the net could be too stable when greater weight changes should be necessary. On the other hand, a larger learning rate could make the NN too plastic even when small changes are necessary. Some approaches to learning rate setting are in [36].

The *reinforcement rule* is similar to the error correcting rule in that weights receive a reinforcement for properly performed actions and a weakening otherwise. The difference between the two rules lies in the way the performance of the network is computed. Error correcting learning procedures the evaluation of an error value for each unit in the output layer of the network, while in reinforcement learning procedures it is sufficient to have only one value (a scalar error value) to describe the output layer's performance.

A general reinforcement learning equation is:

$\Delta w_{ij} = \alpha e_{ij} \delta_j$

where:

α is the learning rate (which is subject to the same considerations previously discussed);
e_{ij} is the canonical eligibility of the weight on the connection from unit u_i to unit u_j, which is dependent on a previously selected probability distribution which is used to determine whether the computed output equals the desired output value (see further below);
δ_j is the error signal of unit u_j, and it is defined as:

$\delta_j = (r - \theta_j)$

with:

r: scalar error value of the network's output layer;
θ_j: reinforcement threshold for unit u_j.

A problem intensively discussed within the framework of reinforcement learning is the "credit assignment problem" related with the correct assignment of credit or blame to each of the actions of each of the system components that contributed to the reinforcement received.

The *Hebbian rule*, named after Donald Hebb is the adjustment of the connection weight according to the correlation of the values of the two units it connects. There are many forms of the Hebbian learning rule, the simplest mathematical form is the "simple Hebbian correlation", which is defined by the following equation:

$\Delta w_{ij} = o_i o_j$

where[1] :

o_i is the output of the unit u_i;
o_j is the output of the unit u_j.

Other more complex forms of Hebbian learning rules can be found in [42].

A desirable feature of all learning rules is that they are *local*. This is very useful in

[1]Sometimes there is a kind of learning rate in the equation so that: $\Delta w_{ij} = \eta o_i o_j$ where η is an adaptation rate constant.

reducing the learning complexity. In fact a local learning rule has the characteristic of finding information about the weights and the input/activation/output of units locally. This means that information about how weights are to be changed can be found in the neighbourhood of the weights themselves. This request for local behaviour is expressed in the phrase: "global net order from local rules".

All learning procedures can be classified into three categories:

- supervised learning;
- reinforcement learning;
- unsupervised learning.

The classification criterion is based on the environmental learning feedback. So a *supervised learning procedure* is a process which incorporates an "external teacher". This means that the environment (i.e. the teacher) specifies the desired output of each output layer's unit.

A *reinforcement learning procedure*, on the other hand, uses an environmental learning feedback in the form of a scalar signal which indicates whether or not the actual and the desired output patterns coincide. This process, in analogy with supervised learning is said to incorporate an "external critic".

In the case of an *unsupervised learning procedure*, the network does not receive any environmental learning feedback. This procedure is also referred to as self–organisation because the process relies upon only local information and internal control to achieve the learning and capture regularities in the stream of input patterns.

For the purposes of this thesis, learning procedures belonging to the supervised learning category have been chosen. The main reason for this is that in applications such as the IR ones it is very easy to identify the "teacher". In any IR application the results obtained by the system are immediately and inevitably judged by the user who, on the other hand, is the only one who can judge of the relevance of the results obtained. The presence of such a valuable teacher has directed the research, most of the time, toward the use of supervised learning procedures or, less frequently, toward the use of reinforcement learning procedures. Moreover, only very recently there as been some attempts to use unsupervised learning in IR and the results obtained do not seem to encourage the research in this direction. The use of unsupervised learning procedures seemed no dissimilar from the use of a sophisticated clustering technique.

In the following two Sections the most commonly used supervised learning procedures are described.

3.6.1 Backpropagation

The Backpropagation (BP) learning procedure was developed by several research groups at the same time in the 1970s and 1980s.

The classical structure of a net using the BP learning procedure is a three-layers perceptron-like network with feedforward connections (Figure 3.11). The intention with this network is to map an input pattern (a pattern is a set of activities in a layer) to a corresponding output pattern. During the learning phase the teacher will assign an input pattern and provide the exact desired target output pattern. The difference between the target and the actual network output pattern guides the adjustment of the network weights. In a practical situation there will be fluctuations in the input patterns which all correspond to the same output pattern. For example, assume a network has to learn to detect a picture of an apple. The network receives binary inputs from the pixels of a digitised picture. There are apples in an infinite number of shapes, colours, and sizes, but for any apple we want the network to output 1 when an apple is presented to the input and 0 when not. After learning many various pictures of apples, the network will recognise apples not initially learned. The network will be able to generalise, or extract a rule, on how an apple looks, based on the similarity of the pictures presented during the learning phase.

Rosemblatt discovered in 1962 a supervised learning rule for adjusting the weights in a one-layer perceptron [24], called the perceptron learning rule. The learning rule has been proven to converge in a finite number of steps, providing the problem at hand can be solved with a single-layer perceptron. A learning rule for multi-layer Perceptrons was invented several times, but was forgotten only to be rediscovered around 1985 with great fame by Rumelhart, Hinton, and Williams [36]. Known as Backpropagation, the delta rule, the LMS-rule, or the Widrow-Hoff rule, it successfully solves some of the major problems the one-layer perceptron had. The major distinction with perceptron is the adding of hidden layers of units.

BP learning involves two phases for each pattern to be learned. First, there is a "forward pass", where an external input pattern is passed through the net from the input units toward the output units, leading to an external output pattern. This phase is also called "recall phase" and it is common to every NN. The output pattern produced is compared with the desired external output pattern and an error signal for each output unit is produced. Second, in the phase called "backward pass", the error signals of the output units are passed backward - "backpropagated" - toward the input units, and an error signal for each hidden and input unit is evaluated. The weight changes are proportional to the error signals. The modifications take place before a successive presentation of the same pattern in an iterative process until a general error measure of the performances of the network reaches a desired value.

More formally, given an input-output pattern p, the weight changes are determined according to the following rule:

$$\Delta w_{ij} = \alpha o_i^p \delta_j^p$$

where:

α is the learning rate;

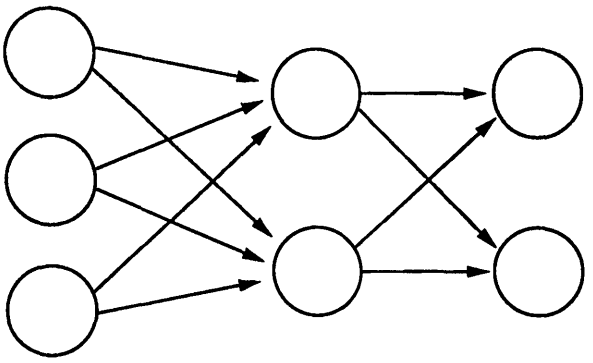

Figure 3.11: BP classical network topology

o_i^p is the computed output of unit u_i for the pattern p;
δ_j^p is the error signal of unit u_j, and it is defined as:

$$\delta_j^p = \begin{cases} (d_j^p - o_j^p)f' & \text{if } u_i \text{ is an output unit} \\ (\sum_k \delta_k^p w_{kj})f' & \text{otherwise} \end{cases}$$

with:

f': the derivative of the activation function;
d_j^p: desired output value of unit u_j for the pattern p;
o_j^p: computed output of unit u_j for the pattern p.

It can be shown that the BP learning procedure minimises the sum-squared error function:

$$Error = \frac{1}{2} \sum_p E^p$$

where:

$$E^p = \sum_j (d_j^p - o_j^p)^2$$

which is computed over all output units of each pattern to be learned. This is done by changing the weights proportionally to $\delta E^p / \delta w_{ij}$; hence BP is the least mean square procedure which implements an approximate gradient descent in the *Error* function. In weight space the error function defines a surface on which the learning phase seeks the deepest point. This procedure follows the gradient of the error function "downhill" to the global minimum of the error function by adjusting the weights. This is done by feeding an error signal backwards through the network along the connections. From the error signal each

unit calculates the changes in the converging weights. Hence the name Backpropagation. Gradient descent is one of the simplest optimisation procedures, but not a very good one. Many numerical procedures invented in the 60s and 70s perform better. Genetic algorithms have also been applied to neural network learning. The weights "evolve" towards optimum values, but genetic algorithms are generally very slow.

Ideally, no a priori knowledge of the problem has to be encoded in the solution, but in practise one has to design the network topology to suit the problem at hand. In particular choose the number of hidden units for a two-layer network, as well as adjusting several parameters.

There are many variations and extensions of the elementary version of BP learning. A good coverage of the subject is in [36]. For an algorithmic description of the BP learning procedure see [42].

Backpropagation can be used in networks with any number of hidden layers, and has been used to solve many practical problems with great success. Applications include pronouncing English text, playing backgammon, sonar target recognition, car navigation, and recognising hand-written ZIP codes.

3.6.2 Boltzmann Machine

A Boltzmann Machine (BM) is a generalisation of a Hopfield net in which the units update their states according to a stochastic decision rule. In order to understand the BM learning procedure, the essential characteristics of Hopfield nets have to be described.

Hopfield took "a step back from biology" in 1982, when he introduce a network of McCulloch-Pitts neurons with symmetric weights. A Hopfield net (HN) is, in fact, a NN whose weight matrix is symmetrical with zero diagonal elements. The synaptic strength is therefore reciprocally equal between any two neurons. This is clearly not biologically plausible. All the units are fully interconnected, and they are all both input and output units. Each unit is binary but with activities -1 and 1 for mathematical convenience. The activation function is a threshold function.

The architecture is that of a content-addressable memory, also known as auto-associative memory. Several patterns of activity (for example a digitalised picture) can be stored by changing the weights similarly to Hebb's rule [41]. When provided with a part of one of the stored patterns the network completes the rest of the pattern. Likewise, the network removes the noise from a stored pattern having noise superimposed.

However, the units are updated asynchronously, which is much more like the continuous updates present in biological systems. Stored pattern are recalled by picking units at random and assigning output values as before. When no changes in activity occurs the activities of the units are the recalled, stored pattern. HN converges to a stable state[2] if the units whose activation values have to be calculated are chosen randomly, asynchronously,

[2]It is supposed that a net state is represented by a vector whose elements are the states of the net's units.

and one at a time, though the process is very slow for large networks simulated on a digital computer.

The main characteristic of HN is the introduction of the quantity E called the *Energy function*. It can be shown that HN performs optimisation of the Energy function by converging to its minimum. The minimisation is achieved by changing the units' states, not the weights.

The HN has been several implementations of the algorithm in analog electronics and optics. The model has been extended to units with continuous activity. The analysis has gained benefit from parallels in statistical mechanics.

A BM uses the same network topology and weight matrix characteristic of HN, but a different activation function and incorporates the technique called "simulated annealing".

In a BM the activation function is a stochastic one and it allows the computation of the probability p_i for the unit u_i to be active (or to have value 1) regardless to its previous state. The probability is computed as follows:

$$p_i = \frac{1}{1+e^{-I_i/T}}$$

where:

I_i: total input of the unit u_i;
T: parameter called "temperature".

The parameter T acts like the temperature in the process of simulated annealing. A *simulated annealing* is an algorithmic simulation of the physical process called annealing, which consists of a heating–up phase followed by a cooling–down phase.

In the heating–up phase a solid is heated up to a maximum temperature, a value at which all the particles of the solid randomly arrange themselves. In the cooling-down phase the temperature is lowered, and if the cooling is done slowly enough, the solid reaches thermal equilibrium for each temperature value.

The parameter T can also be considered a measure of the noise influencing the decision whether or not a unit's state will be changed. The behaviour of the BM and the way the thermal equilibrium is achieved is influenced by this parameter. At a low temperature (T near to 0) the approach to thermal equilibrium is slow since the net responds, changing the units' states even by small energy differences, but low energy states are much more probable than high energy states. On the contrary, at a high temperature (T near to 1) the approach to thermal equilibrium is rapid since the net responds only to large energy differences, but low energy states are not more probable than high energy states. The fastest way to approach the desired low energy equilibrium is generally to start at a high temperature and to gradually reduce the temperature, like in the annealing process.

According to this, the BM learning procedure is performed in two phases: a "clamping phase" and a "running phase".

In the *clamping phase* the pattern is presented to the input and output units; the state

of these units is kept unchanged during the rest of this phase. The net runs and it is annealed until it reaches thermal equilibrium at low temperature. At equilibrium it runs for a fixed amount of time in order to measure, for each connection, the fraction of time the units connected remain both active. This is used to measure the probability p_{ij}^{+} (averaged over all cases) the units u_i and u_j are simultaneously active at thermal equilibrium when the input and the output vectors are both clamped. During the *running phase* the net runs as in the clamping phase, except that the output units are not clamped. The probability p_{ij}^{-} is determined in the same way as in the clamping phase.

The two probabilities are used to determine the weight changes using the following rule:

$$\Delta w_{ij} = \alpha(p_{ij}^{+} - p_{ij}^{-})$$

where α is the learning rate.

These two probabilities can be interpreted as follows: (a) in the clamping phase p_{ij}^{+} measures the performance of the network which is subject to environment and network structure, while (b) in the running phase only the net structure influences the performance of the net, measured by p_{ij}^{-}.

Usually BM has input, hidden, and output units, but in an arbitrary topology and has therefore no layered structure. It can perform the same tasks as Backpropagation, but proves to operate much slower. Applications include speech analysis, phoneme recognition, graph search and optimisation, hand writing digit recognition.

3.7 Neural Networks and IR

The application of NN models to IR is quite a recent phenomenon. Indeed, very few papers have appeared on this subject as yet, and many of them are just reports of work in progress. So far, there is no operational IRS based on the connectionist model, and only recently prototypes have been proposed which approach "real world"-like applications. Besides, many research papers claim to be about application of NN to IR, while they are application of spreading activation techniques (an example is [40]).

In this Section, some of the research is summarised, showing the innovative aspects and the drawbacks of each approach.

M. C. Mozer ([43]) was one of the first researcher to start working on the application to IR of the first studies about NN. Some of his ideas are still research ground for many researchers. He identified the difficulties of traditional IR systems in:

- the users' difficulty in specifying the information they are seeking, possibly because the document descriptors have different semantics than they realize, or because they fail to include relevant descriptors in the query, or because they include irrelevant descriptors;

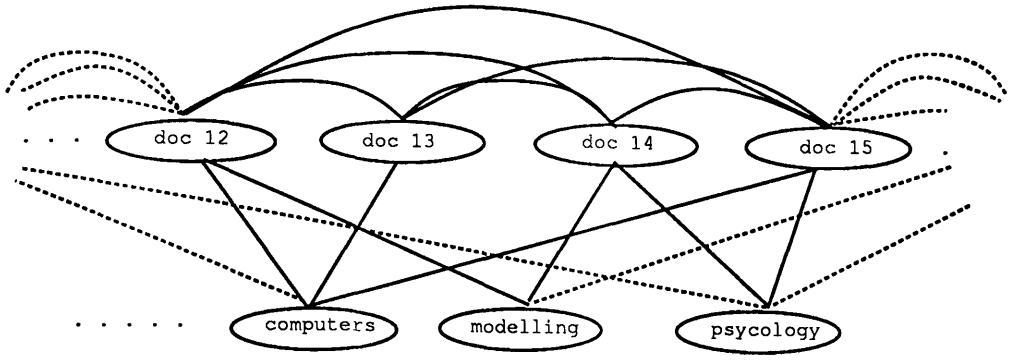

Figure 3.12: Mozer's Inductive IR model

- the indexing of the IRS is often inconsistent and incomplete.

Under these assumptions he considered the retrieval process as an "inductive process". He used what was known from the first experiments of PDP models, to develop a prototype system which, in its simplicity, resulted in being quite effective.

The dynamic of this model was based on McClelland and Rumelhart's interactive activation model of word perception ([36]. The structure of the model is depicted in Figure 3.12. In the model each document or descriptor is represented by a unit. The activation level of a unit indicates the system's belief in the relevance of the document. Excitatory and inhibitory links permit the flow of activation from document to document and from document to descriptors. As it can be noticed, an asymmetric aspect of the model is that each document is connected with inhibitory links (with a constant weight) to all other documents, whereas descriptors do not have mutually inhibitory connections. This competitive aspect among documents helps to keep their activation level under control during the retrieval phase, and helps to control the level of associativity among documents.

A query can be specified as a set of positive or negative descriptors, the positive descriptors being those which should be associated with the retrieved documents and the negative descriptors those that should not. The activation level of the specified descriptors is clamped to the maximum/minimum level allowed. The activation then flows from one layer to the other and between units of the document layer. Each unit computes the weighted sum of its incoming activations, but each unit looses a fixed percentage of its activation during each time step. This results in a exponential decay over time. The activation is allowed to flow from one layer to the other and vice versa until the system stabilises, that is, until the net increase in activation to each unit (i.e. the total of excitatory inputs) equals the net decrease (i.e. the total of inhibitory inputs and decay). The documents are then displayed in order of their activation level. It is interesting to notice that a query can be formulated also as the activation of a set of documents (query by example). This enable the identification of similar documents, where the similarity is measured in terms of common descriptors.

At an abstract level of description, the model operates as follows. The user activates a set of descriptors. These descriptors activate a set of documents, which activate a set of

new descriptors, which in turn will activate a set of new documents as well as reinforce the activation of the already active ones, and so on. This flow allows descriptors in a query to indirectly suggest other descriptors that may be useful in the documents search, and allow active documents to indirectly suggest other documents.

The evaluation of the prototype on a small set of documents and descriptors gave good and sometimes surprising results, often retrieving documents with no descriptors in common with the query yet clearly relevant. However, the model has some important drawbacks. First, weights on links are only +1 or -1 and they do not reflect the importance of the descriptor in the representation of the document's informative content. Second, weights are static, and there is no learning procedure which modifies these weights. Therefore, the system performs in the same way over the time. Lastly, there are no links among descriptors which could represent semantic relationships among them. The relationships between descriptors are induced from their relationships with documents, therefore the model requires that the documents should be indexed by a highly correlated set of descriptors. Without such an indexing scheme, the model would not be able to perform the desired induction and it would perform, as the author pointed out, exactly like a vector space model ([4]).

A few years later, *J. Bein and P. Smolensky* ([44]), understanding the importance and the potentiality of Mozer's research, rigourously tried to test whether Mozer's model of Inductive IR was useful for real world document collection. The main issues they addressed were:

- feasibility: do the properties of the model work with real size collections of documents?
- efficiency: can the model be efficiently (in terms of space and time) implemented?
- utility: is the model useful for real users in terms of improving recall and precision?

It must be recognised that these issues belong to different domains. Feasibility belongs to the domain of connectionist research, while efficiency is more an engineering problem. What is primarily related to IR is the utility of the model. Therefore, although they used a document collection of reasonable size, they could not produce recall and precision figures to be compared to standard IRS because they did not use a test document collection. This limits considerably the conclusions they reached, especially in terms of the utility of the model.

Nevertheless, the results achieved in this research gave analytical and empirical evidence to support the claim that this model is feasible and efficient enough to be implemented for real world applications. In particular, the issue of feasibility, in terms of the associative characteristics of the model, was investigated in depth. The empirical results showed a good value of induced association among descriptors that are not part of the user specified query and among documents which are not directly associated with the descriptors

from the user query. These conclusions were obtained using a very interesting approach, which consists in the evaluation of "relative" recall and precision. These measures were evaluated comparing altered queries with the original unaltered ones. Queries were altered in two ways: randomly deleting descriptors (abridged queries), or replacing descriptors with morphological variants (displaced queries). Experiments of this kind were very useful in determining the sensitivity of the induction to incorrect or incomplete queries. They showed that small queries (i.e. with a small number of descriptors involved) are more sensitive to deletions, while large queries are more sensitive to displacement.

With the CRUCS (Conceptual Retrieval Using Connectionist Style) system *R. J. Brachman and D. L. McGuinness* ([45]) opened a new direction in the research. They tried to combine knowledge representation research, IR, and NN, in what can be considered as an attempt to use the connectionist approach to support similarity-based reasoning in a frame representation.

The starting point of their work was the consideration that there exists a too deep connection between standard KR and formal logic. This results in deductive mechanisms too rigid to be useful in IR. A more flexible form of inference is needed, and they thought of finding it in a connectionist view of inference. Given a document described with one set of terms, the kind of inference they were looking for is the one which should enable determination of other ways to describe the document that are entailed by the original set of terms. While standard KR seems to rigid for such a task, a stochastic approach, like the one used by NN, could fulfill the task.

The structure of the CRUCS model is based on a frame knowledge representation system. A frame represents a concept or a superconcept. Frames and connections among them form a taxonomy, which is roughly hierarchical. In the experiments performed by the authors this representation structure is used to represent concepts and documents in the domain of programming languages. Concepts can have attached to them instances which are the documents the system should be able to retrieve. An example of this KR structure is depicted in Figure 3.13. From this structure, which is still a symbolic representation, a new structure is built in order to use a NN model. This is implemented using μKLONE, a simple frame system built on top of a BM. In this new structure a unit represents a micro-feature of a concept, that is, the smallest element (or property) that can distinguish a concept from all the others. In complete agreement with the connectionist approach to KR, a concept is now represented by a pattern of micro-feature. An instance of a concept is, instead, represented by a single unit connected to all its micro-features. The retrieval process is obtained by means of the annealing process of the BM, clamping, and keeping clamped, the micro-features (properties) of the desired document.

Experiments showed that the tuning of the weights on the connections creates several problems. The most significant negative observation is, in fact, that all the experiments were performed using a KB of a very small size. As the application approaches "real world" size, the size of KB increases greatly and correct answers depend more and more on precise tuning. Furthermore, the tuning becomes more difficult as the size of the KB increases

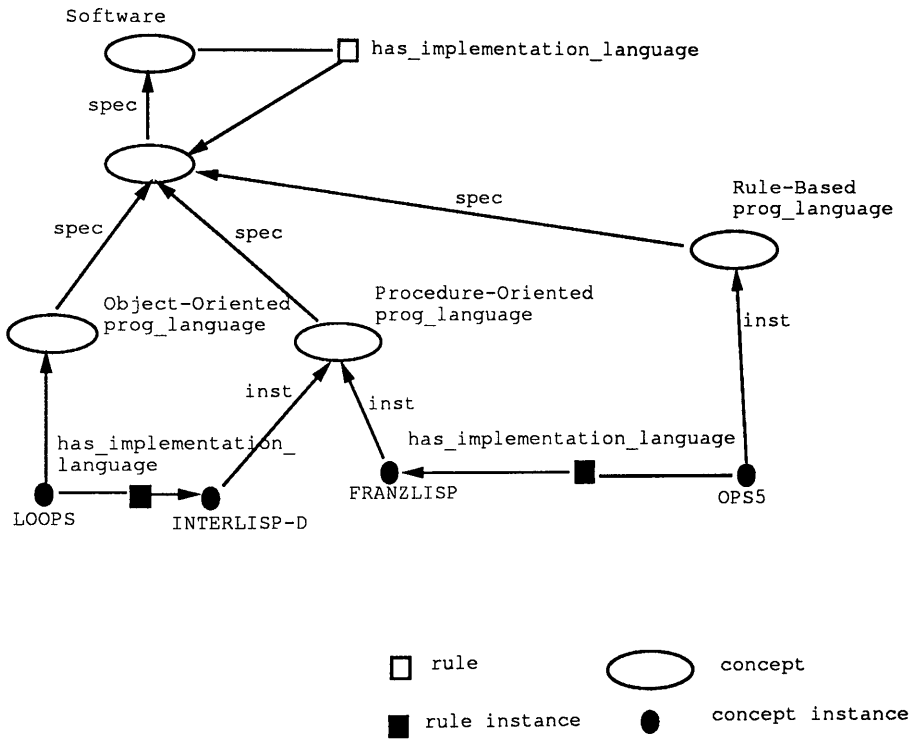

Figure 3.13: A CRUCS taxonomy

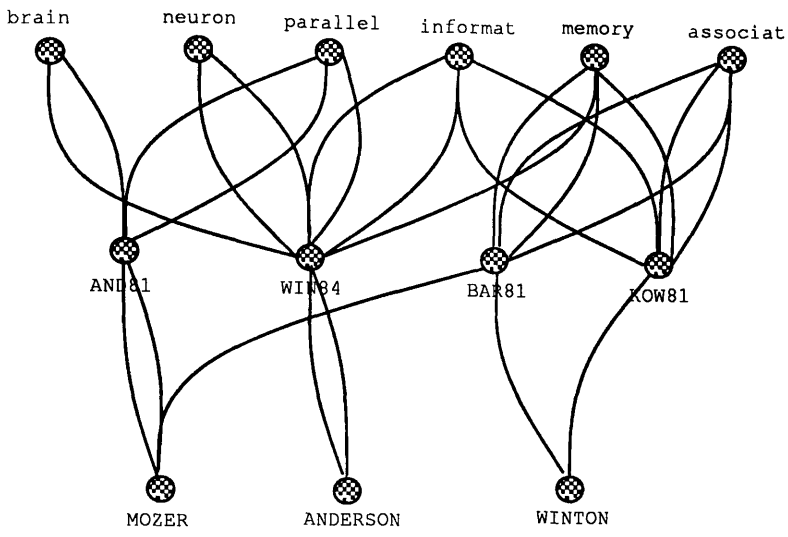

Figure 3.14: AIR's structure

because of the complex relationships among features. Attempts to use the BM learning procedure gave not very exciting results.

However, CRUCS represents an interesting approach to using both local and distributed KR models, combining the advantages of "explicity" of standard KR models, with the stochastically based inference of NN. It demonstrates that these two approaches are not mutually exclusive.

R. K. Belew ([25, 46]) investigated in depth the use of various connectionist techniques in an IRS called AIR.

As with most connectionist systems, AIR uses a weighted graph as its basic representation (Figure 3.14). The system's structure is made of three layers. Nodes on the first layer represent descriptors, where virtually each word in the document's title can be considered a descriptor. Nodes in the middle layer represent documents (in the Figure they are expressed with abbreviations). Two links (one in each direction) are created between a document and each of its descriptors. Weights are assigned to these links according to an inverse frequency weighting scheme. Similarly, two links connect a document to each of its authors, which are represented on the third layer. The sum of the weights on all links going out from a node is forced to be a constant. This weighting scheme is recognised to be simplistic, especially in the use of descriptors taken only from the document's title. However, this is only an initial weighting. Weights will be permanently modified from the first user session, by means of relevance feedback.

An initial query is composed by specifying some of the three type of features represented in the network. The query causes the activation of the nodes corresponding to the features named in the query. This activity is allowed to propagate throughout the network and the

system's response is the set of nodes that become active over a certain threshold during this propagation. Query subsequent to the first are performed using relevance feedback from the user. The user is requested to evaluate the relevance of the documents which are displayed in order of their current activation. The system constructs a new query directly from this feedback. Moreover, this relevance feedback acts as a training signal to modify the document representation by changing the weights on the links. Although Belew uses a learning rule derived from the Hebbian one, he gives to it the interpretation as a conditional probability. A weight w_{AB} is consider the conditional probability that the node B is relevant given that the node A is relevant. This probability is then extended inductively to include direct, transitive paths that AIR uses extensively for its retrieval. This method enables the system to construct a representation of the collection based on the combination of two completely different sources of evidence: the word frequency statistics of the initial indexing, and the opinions of its users.

As in most of the systems in this Section, it is possible to formulate a query by giving some example of relevant documents. In this case, an interesting aspect of this system is the ability to directly point out to the user new authors related to the subject indicated in the query. Belew's work is one of the most complete on this subject. Some of the ideas of this thesis have originated from studying Belew's work.

K. L. Kwok's work ([37, 47]) was an attempt to use the NN paradigm to reformulate the probabilistic model of IR ([1]) with single terms as document components. The main idea was to take into account what was already accomplished by IR researchers and integrate that with the new NN paradigm.

The model proposed by Kwok is represented in Figure 3.15. It is a three layer NN, with bi-directional and asymmetric connections, where no connections are allowed between units on the same layer. The structure is, therefore, very similar to a three layer BP network. Units on the layer Q, which represent queries, may receive an external input and are connected to the units in the layer T, which represent index terms. Layer T is considered a hidden layer. Units of the layer T are connected to units on the layer D, which represent documents.

The innovative aspect of Kwok's model is in the way the initial weights are evaluated. It uses a modification of probabilistic indexing to evaluate the initial strength of the connections. Without entering into the details of the actual formulas, it should be noted that the strength of the connections represents a sort of inference, determined using classical probabilistic IR measure, like the inverse document frequency. According to this, a weight on a connection is considered, depending on its direction, either as the probability of presence of that index term given a particular query (w_{ka}) or document (w_{ki}), or as the evidence that if the index term k is used it will be dealing with the contents of that particular query (w_{ak}) or document (w_{ik}).

Another original aspect of Kwok's model is in the way the retrieval is performed. In fact, the retrieval can be performed as a feed-forward of feed-backward spreading of activation. In the first case the activation flows from the query to the documents, which are then

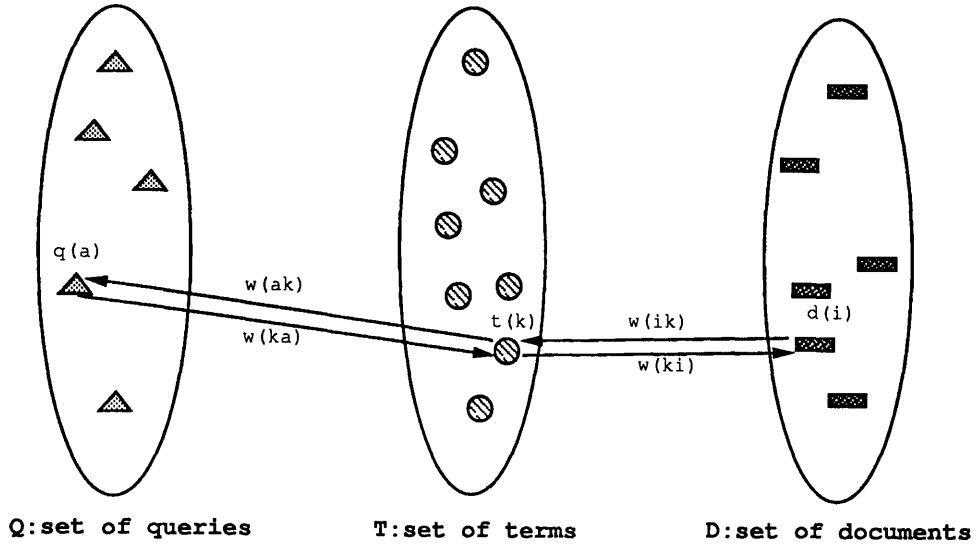

Figure 3.15: Kwok's model

ranked accordingly. In the second case, the activation spreads from the documents, whose activity values are clamped to the maximum value, to the query through the index terms' layer. Each document is then evaluated for relevance to the query based on whether the activity received at the query node exceeds a certain predetermined threshold value. This activity can also be used to rank the documents according to their relevance to the query. Experiments showed that combining these two methods together, it is possible to obtain performance measures better than the traditional probabilistic IRS.

The learning procedure used in this model may be seen as a type of Hebbian correlation learning and is of the same nature of the one used by Belew.

Although this can be considered a good attempt to combine old and new paradigms in IR, the complexity of the computations necessary for the spreading of activation and for the learning process makes this approach impracticable for real size collections.

Other attempts to combine sounded classical IR techniques with NN can be found in the work of *G. S. Jung and V. V. Raghavan* ([48]). They attempted to marry the vector space model with learning paradigms of the connectionist model.

The main contribution of their work concerns the construction of a thesaurus-like knowledge representation structure, referred as "pseudo-thesaurus". The domain knowledge contained in a pseudo-thesaurus is in numeric, rather then symbolic form and it is represented in a network structure similar to a single layer NN, where (again) terms are represented by nodes and relationships between terms are represented by links. Therefore relationships between terms are represented in the pseudo-thesaurus as real numbers (as weights on links), and they are determined by means of a learning procedure which makes use of relevance feedback from past users. The information provided by the relevance feedback

is in the form of training pairs, that is pairs of queries and document descriptions. The learning procedure is such that the pseudo-thesaurus can incrementally update itself in an adaptive way by means of continuous relevance feedback from users. Once the learning of the weights in the pseudo-thesaurus has taken place, the information contained in it is used in conjunction with the vector space model to perform the ranking and retrieval of documents in response to a query. There is a straightforward mapping between the learning procedure they use and the perceptron learning procedure.

This system is very interesting and is theoretically sounded being based on the vector space model. The major drawback of the model is in its assumption that the relationships between terms in the pseudo-thesaurus are symmetric. This is not always true. Indeed, in most cases (e.g. generalisation - specialisation relationship) the strength of the connection between a pair of terms is different according to the direction under consideration.

Recently, *P. Hingston and R. Wilkinson* ([49]) continued in the refinement of Mozer's ideas. The architecture of their model is, in fact, almost the same as the one proposed initially by Mozer. The major contribution of their work is in the proposal of incorporating relevance feedback from users and in the use of a test document collection, giving figures that could be compared with those produced by standard IRS.

The basic idea from which they started is the same as Mozer's: two terms which appear to be common to a number of documents are likely to be related to the same subject. They improved, on the same line followed by Bein and Smolensky, the way weights are evaluated, using the vector space model as a theoretical foundation. However, they went further, incorporating relevance feedback from users. When the system produces an initial ranking of documents in response to the initial query, document are examined by the user, who gives a rating. This rating is transformed into an activation level that is clamped, so that it may not change in subsequent calculations. The system is then allowed to run again. The rating influences the activation of terms connected to the documents considered relevant by the user, giving them a higher activation. The major problem they had was related to the tuning of the weights, which resulted to be too query-dependent.

The above exemplification of applications of the connectionist approach to IR can be considered almost exhaustive. Although many other attempts are under way, they are quite similar to those described above. The approach proposed in this thesis tries to move a little step further from what has already been achieved, showing that, though many problems still exist, the way is quite promising. The next chapter will illustrate the details of the proposal.

Chapter 4

Adaptive Information Retrieval

> "A program should not be a cast in stone, immutable and impossible to affect. It should, if it is to be a good model of a person, be changed by what it reads"
> [R. Schank, 1981]

4.1 Adaptation

The experiments presented in the following chapter can be better interpreted if they follow a simulation model. In the present section the simulation model which is at the foundations of these experiments will be presented. A comparison with traditional IR explains the major points of this model.

The model that will be used during the simulation is based on some assumptions. It is assumed that:

- there is a document space in which the entire collection is represented;
- there is some form of document representation formalism, that is a document is represented in some way using a certain formalism;
- the formalism used is powerful enough to enable the evaluation of some similarity measure between documents.

The query and document representation formalism used in the experiments is that of the vector space model ([4]). This formalism assumes that a document is represented in terms of some features called descriptors (usually index terms). A document, therefore, is represented in terms of a set of descriptors, having this representation in the form of a vector. Each element in the vector represents a descriptor and it is assumed that the same set of descriptors is used to represent all the documents in the collection. The descriptors are represented in the vector in a binary form (using 0 or 1). The presence of the descriptor in the description of a document is represented by a 1 in the respective position in the vector, otherwise 0 is used. Using this formalism, the entire collection is

	t_1	t_2	...	t_k
d_1	1	0	...	1
d_2	0	1	...	1
d_3	1	1	...	0
...	...	...	...	...
d_n	1	0	...	0

Figure 4.1: Vector representation of a documents' space

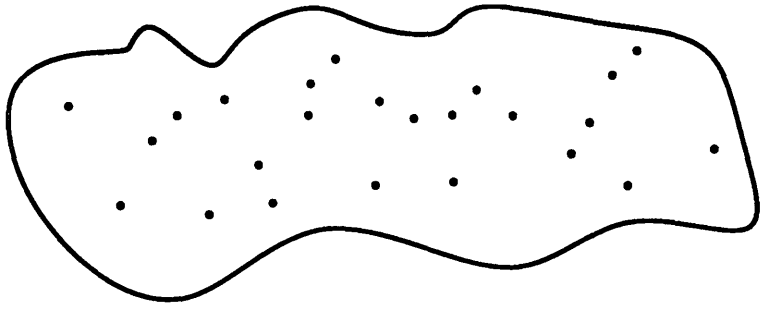

Figure 4.2: A simplified representation of the documents' space

represented in the form of a matrix in which a row represents a document and a column represents a descriptor (see Figure 4.1).

Once a formalism has been chosen, the collection can be represented in a abstract way as in Figure 4.2, where each dot represents a distinct document. This representation oversimplifies the document space, which should have as many dimensions as the number of descriptors. However, a two dimensional representation is sufficient for the purpose of this section, which is to give a simple but clear idea of what the experiments are about.

Using this simple model, it is possible to compare the traditional IR approach with the an adaptive one.

The **traditional approach to IR** assumes that:

1. The query and the documents have the same structure, and they are represented using the same formalism. This means that the user has to express in the query a set of descriptors that are considered of interest and that can be useful for finding relevant documents. In other words, the user expresses his information need by specifying some features of the documents he is looking for.

2. The user, using the formalism provided, has the possibility to express his information need in almost a perfect way.

3. The relevance of a document to the user information need is evaluated as similarity between the document and the query using the formalism provided.

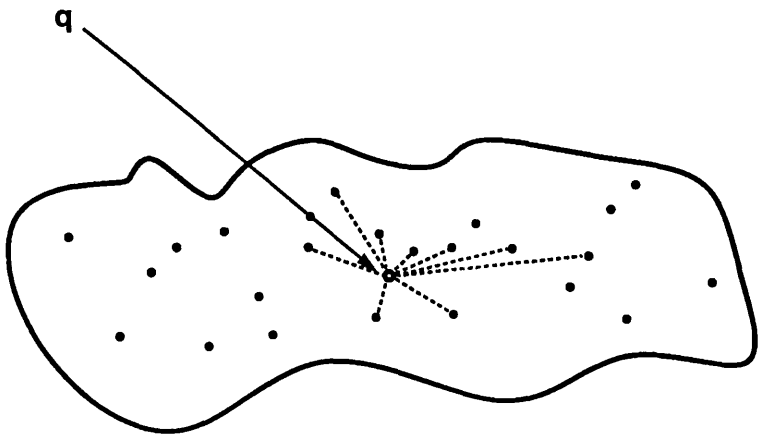

Figure 4.3: Query/documents similarity in the documents' space

Given these three assumptions, a traditional IRS can be thought to work in the following way.

A query is formulated by the user. The query, which has got the same structure of a document, is "dropped" into the document space. Inside the document space the query represents the very document the user is looking for. It is, now, just another element in the document space, and there is no distinction between the query and anyone of the documents. The task of the IRS is to provide the user with a set of documents ranked according to their similarity with the element representing the query: the query representative. This is achieved by evaluating some sort of distance between document and the query representative, using the document space metric (see Figure 4.3).

It is possible to depict this retrieval process with a simple metaphor. We can think at the user in terms of a fisherman who wants to catch a particular kind of fish (relevant documents) in a lake (a document collection). The fisherman throws his line (formulates a query) where he thinks the fishes could be, in terms of a certain position in the lake and a certain depth. If he knows exactly where the fishes are, we may expect the hook to drop exactly over them which will swallow it (that is the query would retrieve exactly that document) and the fisherman will have his desired prey. However, this is quite difficult even to imagine and only very expert fishermen could even think of doing that. Usually we are expecting the line to fall close to the fishes but, as the lake is quite large and well populated, it can fall close to other kind of fishes too. It is sensible to assume that the closest fish will swallow the bait first. If the fisherman keeps throwing the line in the same position he will get all the fishes close to that particular position ordered according to their closeness to it, that is, out of the metaphor, the documents will be ranked in order of their similarity measure, or distance to the query.

The traditional IR approach presents some problems:

1. The user must express his information need using a language determined by the

representation formalism of documents in the document space. The query languages based on this formalism are usually not very user-friendly, and they use a restricted vocabulary compared with a human or pseudo-human language.

2. Sometimes the user is not able to express his information need in an effective way. This is a quite sensible concept considering that the user must express his ideas using a language which is quite poor in terms of representation power.

3. The relevance of a document to the user information need is evaluated using some measure of similarity between the document representation and the query. The query representation describes the user information need at the moment in which it is formulated, but during the query session the user information need can be subject to modification according to the information the user receives from the system. On the other hand, the document representation describes the document informative content at the moment in which the indexing of the documents was performed. It is recognised that the meaning of words (and descriptors are words) changes with time. Therefore, the document space is not a static space, but a space where documents are likely to move their respective positions, and so changing their relative distance to other documents. The evaluation of the similarity between a query which expresses the dynamically changing information need and documents represented in a static (and often obsolete) space is not to be considered an optimal situation.

In terms of the previous metaphor, this means that the fisherman should be: a master in the art of fishing, and should know perfectly the lake and the position of his prey. Moreover, the fishes in the lake should occupy a static position, that is, they should not absolutely move their position in the lake. In terms of the above metaphor, the assumptions upon which the traditional IR is based really seems too strong.

An **adaptive approach** can remove or, at least, modify some of the previous assumptions (see Figure 4.4).

In particular:

- The query does not need to have the same structure and be expressed using the same formalism of the documents. A transformation T will adapt the original query or the documents in a way that they can be compared each other and their similarity evaluated. This will enable the user to express his information need in a way which is more similar to his usual way of expressing his ideas.

- The user is not assumed to express his information need in a perfect way. The transformation T, using domain knowledge, will modify the original specification of the query, directing the query accordingly.

- It is possible to use an adaptive evaluation of the similarity between query representative and documents representations using information about their effective relevance provided by the user with a feedback process. The document position in the document space will, therefore, be modified according to user's judgements.

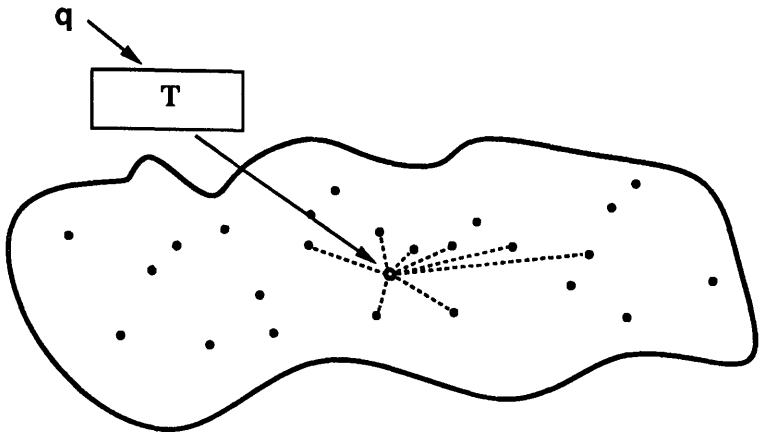

Figure 4.4: New query/documents evaluation of similarity

This modification of the assumptions of the traditional approach to IR makes the example of the fisherman to appear more real. The fisherman uses some knowledge of the lake, acquired from experts or personal experience, to choose where to throw his line and he can modify the position of the line according to the catches he is getting and to his knowledge of the movements of the fishes in the lake.

In other words, the transformation T uses some domain knowledge to modify the original specification of the query and/or of the documents.

Two questions about T obviously arise now:

1. how does T work?

2. how is the domain knowledge used by T acquired?

There are various adaptive strategies and therefore various ways to make T working. Furthermore, there are various ways to acquire the necessary domain knowledge. In the following some hints about two different adaptation strategies are reported.

4.1.1 Query adaptation

This is the strategy used by this thesis. This thesis tries to answer the above questions taking the query as the focus point. The transformation of the query works like an expert intermediary who, after having received information about what the user is looking for, gives help in terms of: "if this is what you are looking for, than this is the best way to look for it". A modification of the query is the easiest adaptive strategy because it involves the modification of only a single element, the user's query, and not the entire collection of document representations.

Regarding the procedure of query adaptation and the two questions arisen in the previous section, an answer to the first question is conceptually explained in Chapter 2. The

conceptual model presented there explains how an original query can be modified taking into account knowledge represented in the form of a network. In Chapter 3 two different processing frameworks for that model have been presented. The NN processing framework has been chosen and Chapter 5 try to provide an answer to the second question, that is how to acquire the necessary knowledge. Some experiments of domain knowledge acquisition, using the NN processing framework are reported.

4.1.2 Document space adaptation

Another different adaptive strategy concerns the modification of the way the document are represented by the indexing process. A document space adaptation strategy involves the use of domain knowledge in order to adapt the document representations to the modification in the application domain knowledge. Furthermore, the domain knowledge should be used to adapt the representations of the document contents produced by the indexing process to make them more similar to a more or less natural language formulated query.

4.2 Use of a NN as an adaptation tool

In this thesis the transformation T uses domain knowledge to modify the user's query to make it more suitable to the particular document collection and to the particular document representation technique used. The model presented in Chapter 2 explains how the domain knowledge can be used by a system based on an associative paradigm. But, this domain knowledge must be acquired somewhere and represented in some way.

The approach followed by this research is to use a sub-symbolic knowledge representation which enables the domain knowledge to be acquired from examples provided by domain experts.

The motivation for the choice of a sub-symbolic representation structure has already been explained in Chapter 3. The generalisation capabilities of such representation techniques are very appealing and they have not been completely explored yet. The basic idea is to learn the correct answer (in term of a set of relevant documents) of some training queries and use what has been learned to handle new queries. Figure 4.5 provides an exemplification of the basic concepts of learning by examples. A person, namely a tourist, has to learn to ask for something in a foreign language studying some example phrases in both languages. The tourist has to generalise the structure of the phrases in order to be able of asking for something which does not appear in the examples he has been taught. This activity is quite simple for humans and it is performed very often in our daily life, but it is extremely difficult for a computer.

In Chapter 3 it has been reported that the approach to knowledge representation called "artificial neural networks" seems to possess some abilities to perform this kind of generalisation.

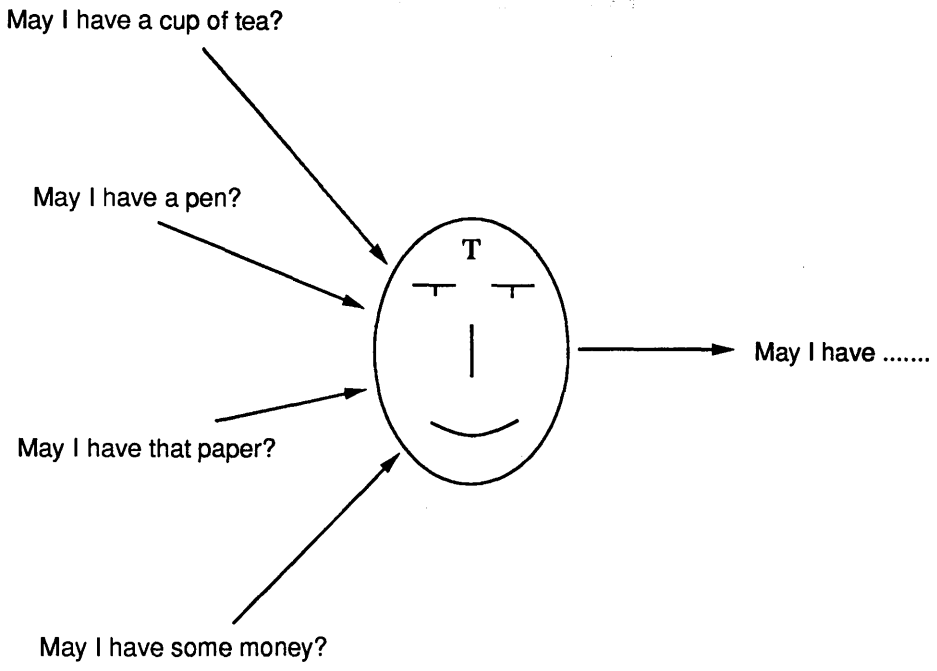

Figure 4.5: Learning by examples and generalisation

Chapter 5

An adaptive strategy for IR: experimental evaluation

> "The original Aslib-Cranfield investigation ...did not, as some had anticipated, demonstrate that one system was "better" than another, either generally, or in any given situation. ...It stimulated a considerable amount of discussion which has helped to clarify the problems of Information retrieval, and created an interest in the methodology of evaluation."
> [C. Claverdon, J. Mills, and M. Keen, 1966]

5.1 The simulation environment

The experiments in this chapter try to analyse the possibilities of the use of the learning and generalisation capabilities of NN in the IR environment. In order to perform this experimental analysis, two tools, at least, are necessary:

1. a document collection with relevance judgements;
2. a neural network or a neural network simulator.

In the following two subsections the two tools used in the experiments are presented.

5.1.1 The ASLIB Cranfield Test Collection

The document collection chosen for the investigation is the ASLIB Cranfield test collection. This collection was built up with considerable effort in the first years of the 60s as the testbed for the ASLIB-Cranfield research project. This project studied the "factors determining the performance of indexing systems" and produced two collections of documents about aeronautics. They were comprehensive of documents and relative requests and relevance judgements developed in order to perform various retrieval experiments. The

two main collections have different sizes. The biggest is made up of 1400 documents with 279 requests and relevance judgements. The second one, which is the most used among the various subsets obtained from the main collection, is made of 200 documents with 42 requests and relevance judgements. These collections were subsequently used in many other project, like, for example, the Salton's SMART project [50]. For a full description of the methods used in the building of the collection and the details of the project, see [51].

Because of operative limits of the simulation environment, which are mainly due to the use of a conventional computer, only the 200 document collection has been used in the investigation. It is of course understood that this limits the generality of the results obtained. However, the main purpose of this investigation is to demonstrate the feasibility of the proposed approach. There are still many open issues in the NN area, and the problem of scaling the results is one of the major ones. The availability of faster computers and of special purpose hardware will enable the investigation of the extension of the results so far achieved. Never the less, it is worth noticing that the dimension of the data set used in the following experiments is larger than those used in most of the other applications of NN techniques to IR, to the present author's knowledge. The fact that the results achieved, as it will be reported further below, confirm some of the previous conclusions obtained on smaller data sets by other researchers, makes the possibility of scaling up some of the conclusions more credible.

For more technical details about the 200 document collection see the cited book. The Appendix reports the table with the relevance assessments used in the experiments.

5.1.2 The PlaNet neural network simulator

For the investigations reported in this thesis a NN simulator running on a fast conventional computer has been used. The choice to use a simulator on a conventional computer, instead of using a proper NN on a parallel computer was dictated by the need to prove the real possibilities of the proposed approach on theoretical basis before attempting its practical use. The practical implementation of a system, which makes use of these ideas on more complex hardware, would have required a longer time and the availability of more powerful tools. Given the various problems found by other studies on application of NN to IR, as reported in Chapter 3, this thesis concentrates on the soundness of the approach more than on its practical present use in an IRS.

Following this decision, a survey on the various NN simulators available in the academic environment was conducted, and a departmental report was produced to describe and compare the characteristics of the various tools ([52]. Among these, the PlaNet System was chosen.

The *PlaNet System* was first developed by Yoshiro Miyata at the University of Colorado at Boulder (USA) in the 1989, and at first it was called SunNet. The version used in this thesis is the 5.6 (December 1990), where the name was changed to PlaNet ([53]). PlaNet is a tool for constructing, running and examining NN. The most significant characteristic of PlaNet is that it allows one to deal with NN at a fairly high level of conceptualisation, and yet provides the flexibility to construct networks of almost arbitrary structure and

size, and to run the network in many different ways. The network is defined by specifying layers of units and connection between layers. The network can be programmed by means of "procedures" that specify the way it should be activated, the activation evaluated, the connections modified, and other important features. The procedures are written using a network specification language, fairly similar to the C programming language, which is general enough to allow many different types of networks to be constructed. Another important characteristic of PlaNet is that it allows the analysis of the NN through a graphic display of various network states, such as activation patterns, weight matrices, or by plotting the learning curve (that is a graph depicting the errors or other networks states as function of learning cycles) in a graph. It is also possible to modify, in an interactive way, parameters affecting the nature of the NN or changing the user interface of the system.

Although some other NN simulators available were even more flexible, like the Rochester Connectionist Simulator, or like Sfinx, PlaNet was chosen because of its particular suitability to the BP learning algorithm, which was chosen as the main algorithm for the present investigation. In fact, especially in its previous versions, PlaNet was developed with particular attention to that learning algorithm, and the PlaNet package includes high-level routines for NN based on the BP learning algorithm.

In the experiments reported in this chapter a slightly different version of the BP learning algorithm has been used. This is the one actually used by PlaNet. It uses the following formulas:

$$\Delta w_{ij}(t) = \eta o_{pi} \delta_{pj} + \alpha \Delta w_{ij}(t-1)$$

where:

the indexes t and $t-1$ refer to the time variable;
η is the learning rate as already described in Section 3.6.1;
α is the momentum;
o_{pi} is the computed output of unit u_i for the pattern p;
δ_{pj} is the error signal of unit u_j, and it is defined as:

$$\delta_{pj} = \begin{cases} (d_{pj} - o_{pj}) f' & \text{if } u_i \text{ is an output unit} \\ (\sum_k \delta_{pk} w_{kj}) f' & \text{otherwise} \end{cases}$$

with:

f': the derivative of the activation function, in this case, using a logistic activation function the derivative is: $o_{pj}(1 - o_{pj})$
d_{pj}: desired output value of unit u_j for the pattern p;
o_{pj}: computed output of unit u_j for the pattern p.

The major modification to the procedure reported in Section 3.6.1 regards the adding of

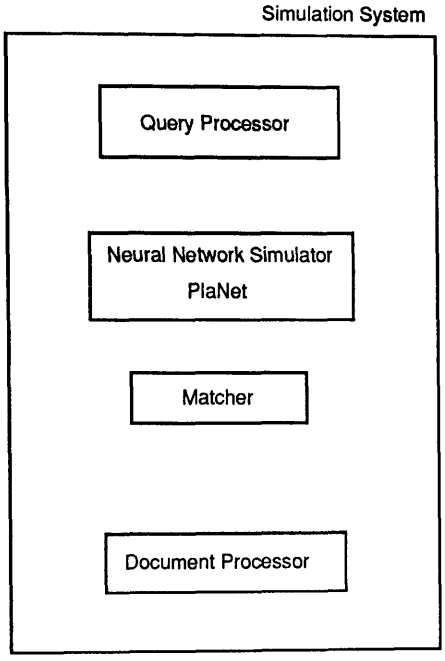

Figure 5.1: Schematic view of the simulation system

the *momentum*, that is a parameter which expresses how much the previous modification of the weights influences the present modification. With the present investigation being mainly experimental, it was decided to keep this new parameter, instead of going back to the original BP procedure, because it introduces another useful variable on which to experiment.

The Appendix reports the source codes of the NN definition and of the learning procedures used in the experiments.

5.2 The simulation system

A simple simulation system (henceforth simply called SS) was developed to operate in the simulation environment described in Section 5.1. This cannot be considered a proper IR system and not even a prototype of an IR system. It is just a set of programs that operate together, though requiring some external intervention, allowing the processing of a query and the retrieval and ranking of relevant documents from the test collection. A schematic view of the structure of the simulation system is described in Figure 5.1.

The simulation system is composed of the following modules:

Query Processor : a C program which transforms a query expressed using terms into a vector of 0s and 1s, where 1 indicates the presence of the term in the query and 0

its absence. The dimension of the vector must enable the representation of all the possible queries a user might formulate.

Neural Network Simulator : a BP model working on the top of the NN simulator described in Section 5.1.2. Although in this thesis a simulator has been used, in a operational implementation of this system a real NN could be used, or, otherwise, a NN simulator working on a parallel machine. As the dimensions of the application grow, the necessity of using a real NN or a parallel implementation of the simulator become more and more pressing.

Matcher : a C program. It evaluates the similarity between two binary vectors using Dice's coefficient (see Section 5.3) and produces a value (in a range between 0 and 1) indicating that similarity. This value is attached to the document identification number. After all the documents are processed they are ranked according to their similarity with the query and displayed to the user.

Document Processor : another C program. This works in a way analogous to that of the Query Processor. Its task is to transform documents which are usually represented using terms into a binary vector representation. This is done once for all the experiments, transforming the entire collection into a large matrix.

Each and every experiment which will be reported further below is composed of two phases: a training phase and a retrieval phase.

Before using the SS for retrieval purposes, it must be trained. The structure of the system during the **training phase** is depicted in Figure 5.2. During the training phase there is no use of the Matcher. On one side, the Query Processor gets the query in the form of of a set of terms. It transforms the query into a binary vector whose dimension is that of the input layer of the Neural Network Simulator. The SS simulator is fed according to various teaching strategies (see Section 5.5) using queries and sets of documents that are known to be relevant to those queries. The input and the output layer of the BP model used by the NN simulator are kept unchanged to represent a query and one or more relevant documents. Then, one of the training algorithms presented in Chapter 3 is used with the purpose of altering the values of the weights on the links connecting nodes in the NN structure. The learning is monitored by the NN simulator control structure and when some predetermined conditions are met the learning phase is halted. Two matrices are produced, representing the application domain knowledge acquired, they are stored for their further use in the retrieval phase.

During a **retrieval phase** the modules interact with each other in the following way (see Figure 5.3). After the query processor has transformed the query into a binary representation, the NN is activated. The activation spreads from the input layer to the output layer using the weight matrices produced during the training phase. The vector representing the query is therefore modified or, better, adapted according to the application

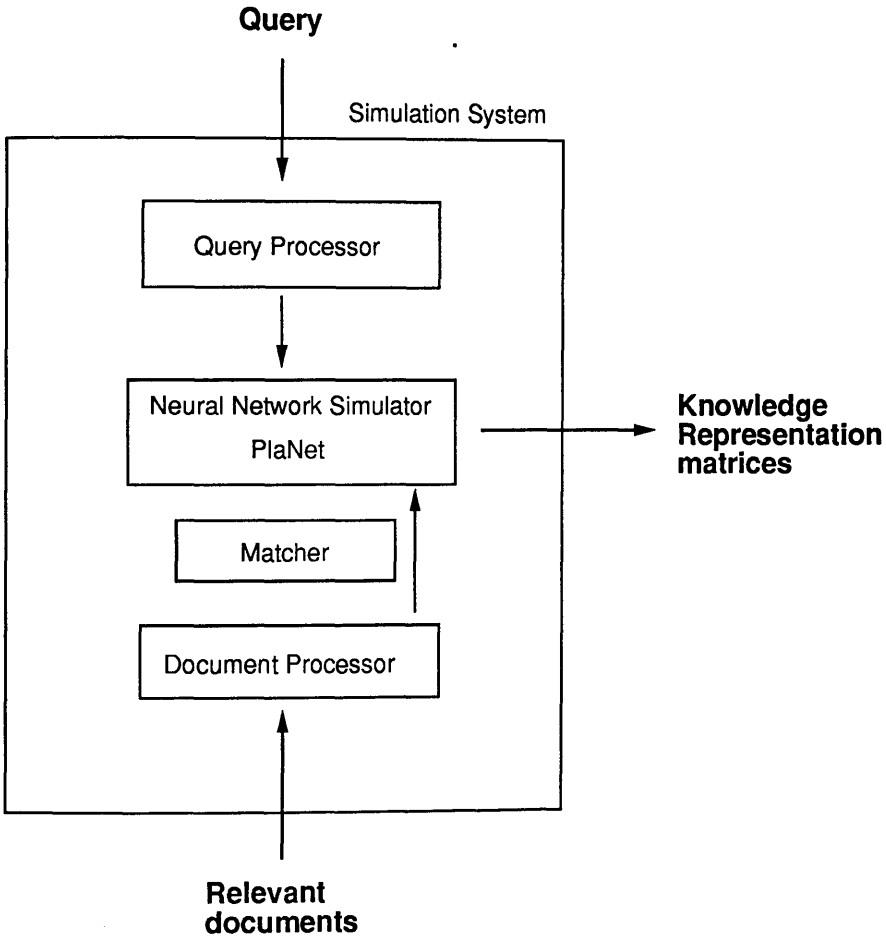

Figure 5.2: Schematic view of the simulation system during the training phase

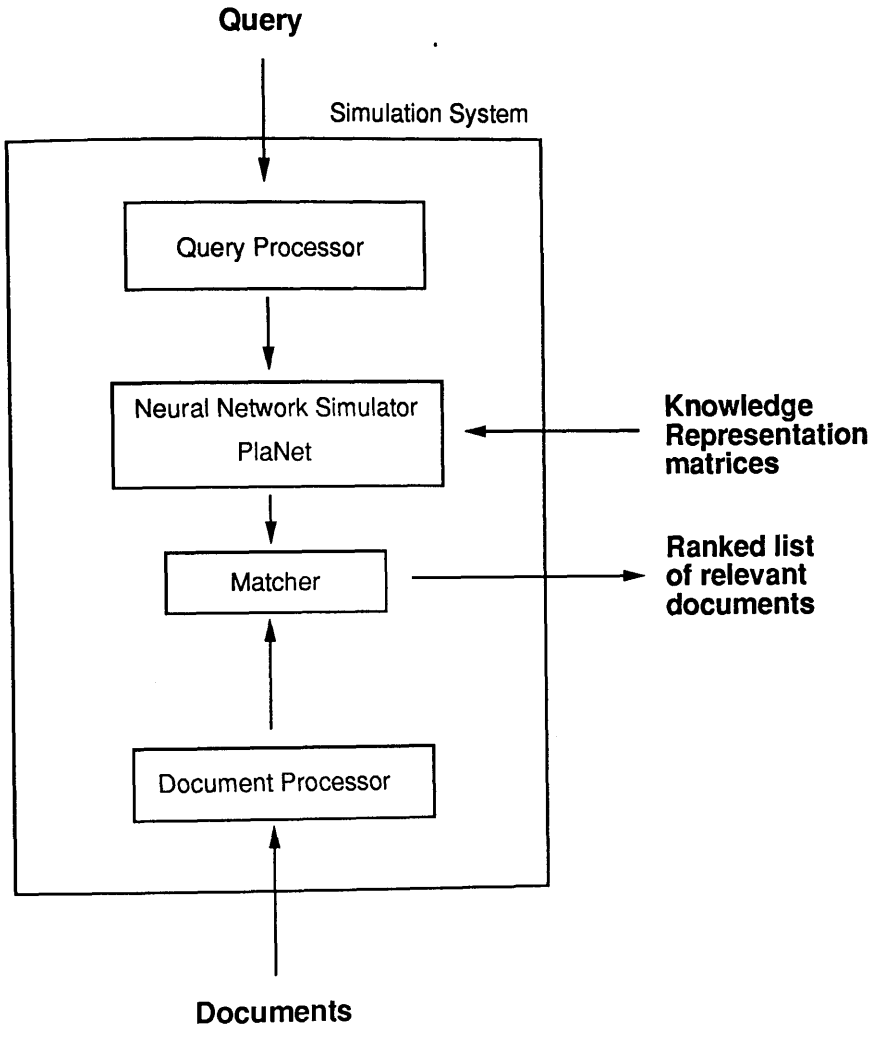

Figure 5.3: Schematic view of the simulation system during the retrieval phase

domain knowledge and a new query representation vector is produced on the NN simulator's output layer. On the other side, the entire collection of documents is transformed into a large representation matrix by the Document Processor. This big matrix is then fed, together with the result of the query adaptation into the Matcher. The task of the Matcher is to produce a ranked list of document identification numbers. The ranking reflects the evaluated relevance of the documents to the query.

This thesis is not concerned with the efficiency of the Query and Document Processor, or of the representation structure they use, or with the efficiency of the Matcher. Only the characteristics of the NN model used by the NN Simulator and the performances of the query adaptations will be analysed in the following Sections. The ability of the NN model to store and use application domain knowledge is the main object of the experimental analysis.

5.3 The evaluation criteria

Much effort and research has gone into studying the problem of evaluation in IR. However, most of the people active in this field still feel that the problem is far from solved. Here the problem is to evaluate a new approach to IIR, and therefore it is only partially possible to use already developed evaluation techniques.

Never the less, in the present investigation, using the approach followed by van Rijsbergen in [1], at least these two questions must find an answer:

- what to evaluate?
- how to evaluate?

The answer to the first question is related to the main purpose of this investigation. The aim of this research is to show the possibility of learning and use of domain knowledge for an IIR application by means of a connectionist approach. The object is to obtain a subsymbolic knowledge representation of an IR application domain knowledge which should be used by an hypothetical operational IRS to adapt an original user formulated query interpreting the user information need in the light of particular characteristics of the application domain. The purpose of this investigation is, therefore, not particularly concerned with the demonstration that an IIRS using a subsymbolic knowledge representation structure, acquired through connectionist learning, can perform better than a traditional IRS or IIRS, but to give support to the idea that this could be possible. In fact, before attempting to prove the superiority of a subsymbolic approach to the knowledge representation of an IR application domain knowledge, much more effort and research should be put in experimentation. This thesis together with other related works points to some directions for this research.

For the above mentioned reasons, with the purpose of this investigation far from being that of proposing an operational system, the objects of evaluation were chosen accordingly. Among the various main measurable quantities proposed, as early as 1966, by Claverdon in [51], the *time lag*, the *presentation*, and the *effort*, have been completely ignored, because they are related to an operational implementation of the system. The main features which were considered for evaluation were:

1. ability of the system to acquire domain knowledge;
2. ability of the system to use domain knowledge in the retrieval phase;
3. retrieval performance of the system when compared with an operational IRS.

The ability of the system to acquire application domain knowledge is fundamental to the performing of the experimentation, therefore this has been tested first. However, it would be useless to acquire domain knowledge without having the ability to use it later. In particular, the knowledge acquired should be used to deal with new queries,

documents:	*relevant*	*not relevant*	
retrieved	$A \cap B$	$\neg A \cap B$	B
not retrieved	$A \cap \neg B$	$\neg A \cap \neg B$	$\neg B$
	A	$\neg A$	

Figure 5.4: Determination of precision and recall values

generalising the information acquired during the training phase and applying it as a mean of query adaptation. Furthermore, the retrieval performance of such a system must be comparable to that obtained by operational systems. Such a new system would be useless if its performance were much worse than those obtained by an operational IRS.

The question of how to evaluate requires more technical answers. How is it possible to determine if some knowledge about the application domain has been acquired by the NN? How is it possible to determine if a subsymbolic knowledge representation has captured some features of the application domain through the examples it has been taught? The approach to evaluation used in this thesis is a simple one.

First the *learning results* are evaluated. This is done using an approach which is classical in NN research. It consists of the evaluation of the "mean error" between the the training (target) results and the obtained results, as it has already been explained in Section 3.6.1.

Then the *generalisation results*, in terms of recall and precision of the prototype, are determined. They have been evaluated at different stages of learning and with different sets of training examples. If some learning has taken place, and if the knowledge representation structure can generalise what it has acquired, then an improvement of the performance obtained in the retrieval of new queries has to be expected. This improvement in the performance has to be related to the number of learning cycles and/or the dimension of the training set.

Finally, in order to evaluate the *retrieval results* of the system, a comparison with the performance of a classical keyword matching retrieval has been performed. This enables the evaluation of the query adaptation strategy versus the use of the original query. The results of this comparison are presented using recall and precision tables.

The two well known measures of effectiveness: *recall* and *precision*, have been used to evaluate to which extent the knowledge acquired is used in the retrieval phase either in absolute terms or comparing the SS with an operational one. In order to have clear the meaning of the recall and precision measures, their definition, as described in [1], is here reported. It is helpful to refer to Figure 5.4, from which recall and precision can easily be derived. They are defined as:

$$Precision = \frac{|A \cap B|}{|B|}$$

$$Recall = \frac{|A \cap B|}{|A|}$$

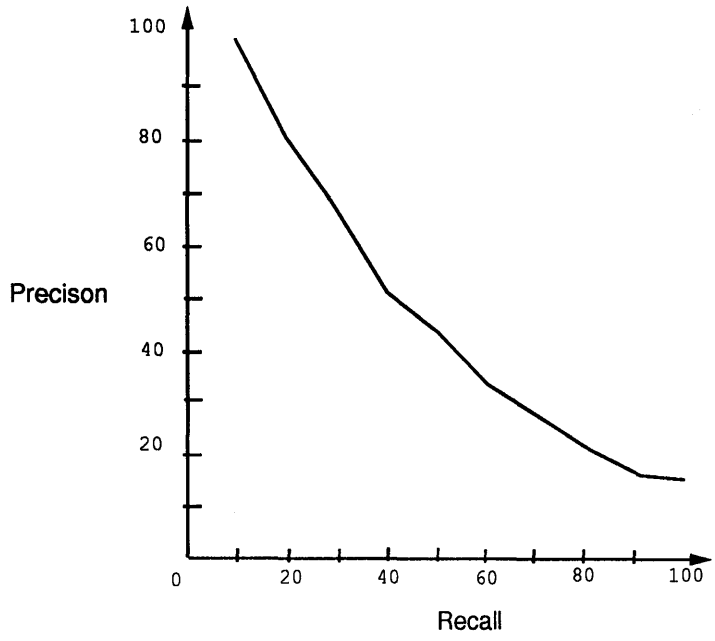

Figure 5.5: Precision vs recall graph

where:
$| A \cap B |$ is the number of relevant and retrieved documents;
$| B |$ is the number of retrieved documents;
$| A |$ is the number of relevant documents.

For each request submitted to the system, these values have to be evaluated and given as percentage values. However they depend on the cut-off point (the co-ordination level) in the ranked list of documents retrieved in response to the query. Therefore, a better way of displaying these measures is through a *precision-recall graph*. An example of such a graph is depicted in Figure 5.5, where precision values are reported corresponding to standard recall values.

To measure the overall performance of the system on a set of queries, it is necessary to combine in some way the set of graphs, one for every query, in order to produce an average graph. This has been done using the "macro-evaluation" approach, which consists in averaging over all queries the individual precision values corresponding to the standard recall values. All the graphs reported in this thesis are obtained in this way. For a more in depth explanation of evaluation techniques see [1].

Every evaluation technique is based on the ranking of the documents retrieved by the system in response to a query. The ranking is supposed to reflect the relevance of the documents by evaluating their similarity with the query. There are several similarity

functions that can be used in the matching phase to produce a ranked list of documents. In the experiments reported in this thesis, *Dice's coefficient* has been used. It evaluates the similarity between a document D and a query Q both represented by a set of descriptors. Dice's coefficient is defined as:

$$M = \frac{2 \mid D \cap Q \mid}{\mid D \mid + \mid Q \mid}$$

where:

$\mid D \cap Q \mid$ is the number of descriptors the document and query representations have in common;

$\mid D \mid, \mid Q \mid$ are the numbers of descriptors in the document and query representation; it is used as a normalisation factor.

The evaluation of the adaptive associative approach is, therefore, obtained in this way: after the query has been modified and adapted according to the application domain knowledge stored in the NN, the similarity between the query and each document is evaluated and a ranked list of documents is produced. By evaluating precision and recall with different NN structures and with different values of learning parameters, it is possible to derive some considerations about the amount of knowledge acquired and about the ability of the NN to generalise this knowledge. It is also possible to compare the traditional IR approach with this adaptive and associative approach by means of a comparison of the two ranked lists in terms of recall and precision.

5.4 A Connectionist Knowledge Representation Structure

A preliminary investigation was devoted to the determination of the best structure for the subsymbolic knowledge representation. The mathematical and complexity theory underlying the determination of the optimal NN topology has been only partially used and more empirical considerations are at the basis of the chosen structure.

Several experiments have been conducted investigating the optimal architecture of the NN model to fit in the SS. Of course there are various different criteria for optimal. Given the hardware restrictions, there may be quite a complicated cost function for the architecture, being necessary to consider elements like: learning time, generalisation capabilities, number of units and so on. In particular, from an IR point of view, the choice of the architecture was influenced mainly by the following elements:

1. number of query descriptors used for the representation of the test queries;

2. number of document descriptors used for the representation of the documents;

3. number of hidden layers and hidden units necessary to enable generalisation to take place;

4. processing time, especially during the retrieval phase.

In particular, the processing time is an element which is in a trade off relation with the other three elements. The larger the number of query descriptors, document descriptors, or hidden units, the longer will be the processing time required during the spreading of activation in the retrieval phase and during the training phase. The processing time necessary for the training phase, though, is not so crucial, it being possible to run the process when the system is not used. Since the main purpose of this investigation is to demonstrate the feasibility of the approach, all the experiments are performed on a conventional sequential computer (a SUN Sparc 1 with the UNIX operating system) and using a NN simulator. This results in a very slow training and in a relatively slow retrieval. However, this important factor has not been underestimated. The time a user has to wait before having a response to his query is a measure of the efficiency of an IRS which is too often forgotten in experimental studies. Even if this thesis does not deal with efficiency problems, it should be remembered that if this time becomes considerably long the effectiveness of the system itself, in term of its ability to satisfy a user need, drops considerably. In the present feasibility research this element was not forgotten, but only put temporarily aside.

Regarding the optimal setting of the first three elements, the approach proposed in [54] has been used. It consists of constructing or modifying the NN architecture proceeding incrementally. Given that the main required feature was the generalisation ability, the *optimal architecture* is the one which can perform the best (or an acceptable level) of the generalisation with the minimum amount of units. So, starting with a large number of units, more and more units are taken away until the generalisation performance drops to an unacceptable level.

Hence, in order to determine the best network architecture for the associative knowledge representation structure some experiments were performed. These were devoted to choosing among the many alternative structures and the many different settings of the main parameters. In particular:

1. A first choice had to be taken regarding the *NN model* to use. Chapter 3 reported a brief overview on some of the many NN models. Given the particular requirements of IR applications, two models were considered suitable to the object of this research: the Backpropagation (BP) model and the Boltzmann Machine (BM) model. The following subsection 5.4.1 gives the motivations regarding the choice of BP as the more suitable model.

2. A second choice concerns the *structure of the patterns* the NN has to learn. The structure of the patterns determines the structure of the input and output layers, as

well as the number, characteristics and meaning of input and output units. Most of these issues have already been discussed in Chapter 2, where the conceptual model underlying the experiments has been presented. A more quantitative approach has been used in the experiments reported in 5.4.2.

3. The structure of the *internal representation* of the patterns is another important choice. The appropriate numbers of hidden layers and hidden units have to be chosen. The internal structure of the patterns is strictly related to the learning and generalisation capabilities of the network, therefore particular care had been taken in the experiments reported in 5.4.3.

4. Finally, after all the previous choices have been taken, there are still several other parameters, mainly *learning parameters*, to be set in order to improve the speed and capability of the learning. The setting of these parameters is a complex technical issue, more related to the mathematical theory of NN than to IR. Some experimental results devoted to the optimal determination of these parameters for the chosen architecture are reported in 5.4.4.

5.4.1 The NN model

In Chapter 3 some NN models were presented. As has already been explained, a NN model is made of at least three components: a network structure, a set of propagation rules, and a learning rule.

The *network structure*, or network topology, refers to the spatial arrangement of the units in relation to each other. The topology is usually network like and therefore it is possible to define it in terms of a connectivity matrix. Moreover, it is common to group sets of units according to their functions in the propagation and learning rules. The connectivity usually reflects this grouping, for example allowing connections only between units belonging to different sets and not between units of the same set.

The *set of propagation rules* refers to the use of particular input, activation, and output functions for the propagation of the activity of the unit. It refers, also, to the particular way of evaluating and applying the bias.

The *learning rule* is the main focus of a NN model. It distinguishes NN models from simple SA models. Moreover, a learning rule is what mainly distinguishes one NN model from another. Some learning rules have been described in Chapter 3.

It must be stressed that some NN models are particularly good for specific applications and not so good for others. In [42] a classification of NN models according to their applications has been attempted by Simpson. Therefore, it has been considered useless to attempt to use odd models for very specific applications like the IR ones. Only NN models previously applied to areas which share some similarity with IR were considered. Under the applications: "database retrieval", "knowledge processing", "language processing", and "text processing" Simpson cites the following models:

- Discrete autocorrelator associative memory;

- Backpropagation (BP);
- Binary adaptive resonance theory;
- Boltzmann Machine (BM);
- Brain-State-in-a-Box;
- Fuzzy Associative Memory;
- Fuzzy Cognitive Map;
- Perceptrons.

A complete description of these models can be found in [54, 42, 55]. Since even a brief description of these models could cover several pages of this thesis, only BP and BM have been described in Chapter 3. They have been chosen among the others for the following reasons:

1. NN simulators have only been implemented for some of the above models. Although it could be possible to write software to simulate all of these models, most of the available NN simulators deal only with BP, BM, and Perceptrons. The reason for this stands on the fact that they are more widely used than others, being more theoretically sound and more general purpose. In this thesis the choice of using a NN simulator instead of developing ad hoc software was taken for time and efficiency reasons. It was considered useless to develop another NN simulator with the sole purpose of experimenting its usability in IR. The application of NN to IR is such a new topic that much can still be done using already developed NN tools and theories.

2. There are various levels of complexity among these models. Some are relatively simple, like Perceptrons or Discrete autocorrelator associative memory, while some others are more complex. The complexity, of course, affects the efficiency of the simulator and the effectiveness of the retrieval process. Furthermore, some of these models are more theoretically sound than others. Being that the purpose of this experimental investigation was to analyse the feasibility of the application of NN to IR, it was considered inappropriate to use models whose theories are not consolidated yet.

3. Some previous research work investigated the use of associative memory and Perceptrons in IR. In order to avoid a useless duplication of work, it was decided to use mainly BP, whose potentialities in IR have not been investigated yet. BP has been successfully applied to many areas and is among the most consolidated NN models. The likelihood of obtaining good results from the application of BP to IR seems to be higher than with other NN models.

4. BP fits into the structure of the conceptual model presented in Chapter 2. The distinction among three or more sets of elements in the knowledge representation structure, which has a parallel in the distinction among input, hidden, and output layers, is an important feature of the conceptual model, which BP (and not BM) allows one to preserve.

Therefore, for the reasons listed above it was decided to use the BP model. The topology used is the classic three layers feedforward NN with the query descriptors on the first layer (input layer) and the document descriptors on the third layer (the output layer). Between these two layers there are one or more hidden layers with a certain number of hidden units.

5.4.2 The structure of the patterns

The first task of the experimentation was to set a suitable representation structure of the patterns to be learned.

The input of the pattern had to represent every possible user query. Therefore it has to be possible to use, at least, all terms and combinations of terms used by the queries in test collection. After some analyses on the collection were performed, the number of terms used by the documents and by the test queries was determined. The number of single terms used to represent the document informative contents was 1142. These were all the single terms used in the abstracts of the 200 documents, excluding terms appearing in a stoplist. The number of single terms used to represent the user information needs in the 42 queries was only 195. That is in the 42 queries only 195 terms out of the 1142 were used.

Being that this was the first attempt to use such a structure in IR, it was decided to organise the experimentation in a way that every possible structure of the model could be experimented. In the following, two different structures used for representing documents and queries are reported. These structures will be referred to using the labels (see Figure 5.6). They are:

S1 : the number of query descriptors used in the input layer is equal to the number of single query descriptors used in representing the test queries (i.e. 195); while the number of document descriptors in the output layer is equal to the number of single document descriptors used in the entire 200 document collection (i.e. 1142). This structure can only be used during the training phase, because it does not enable the formulation of new queries.

S2 : the number of query descriptors in the input layer is equal to the number of single document descriptors used in the 200 document collection (i.e. 1142); while the number of document descriptors in the output layer is the same (i.e. 1142).

Both these two structures enable the representation of the test queries. The second structure, S2, however is more general than S1 and its has some advantages over it.

label	*n. input units*	*n. output units*
S1	195	1142
S2	1142	1142

Figure 5.6: Number of input and output units used in the experimentation

In particular, it enables the spreading of activation during the training phase to input units never used by any user in the query formulation. This is certainly a positive feature, but the number of units and links that this adds to the structure makes its practical use very difficult (see Figure 5.7). The structure S1, on the other hand, operates a reduction of the terms the user can actually use in the query formulation from those used in the indexing of the documents. This reduction could seem unreasonable because it may seem to limit the possibility of the user to express his information need. This problem could be solved by designing a training phase covering the entire domain knowledge so that, even using a number of terms inferior to the one used in the indexing, it is possible to be sure to have covered the entire possible spectrum of user queries. This can also be seen as an attempt to use two different representation languages on the same structure. On the input side of the NN structure a controlled language is used. Only terms chosen by experts as particularly representative are used. On the output side of the NN a free language is used. This allows more freedom in the indexing processing of documents, and the use of free terms from abstracts to be used in the indexing. The NN, during the training phase will identify associations between the controlled language and the free language. There are no problems related to the manual specification of these associations. The NN will detect them from examples of working associations used in test queries whose relevance assessment is known. The association of controlled terms to free terms is a dynamic process, being constantly updated by means of relevance feedback from users. The meaning of these associations is therefore neither static nor determined only by few experts, but is a process of dynamic evolution determined initially by experts and driven by a users' consensous usage of the associations.

5.4.3 The internal representation

Another very important characteristic of the NN model regards it internal representation of the patterns. This aspect is of extreme importance to the generalisation capabilities of the NN. It has been demonstrated that it is necessary for the NN to develop an internal representation of the patterns in order to possess the ability of generalising what it has learned. An internal structure enables the NN to store its internal representation of the patterns. In this internal representation a sort of synthesis of the patterns takes place. The better the synthesis, the better the generalisation capability of the NN.

The NN model used in the following experiments is the BP. The internal structure of this model is determined by the number of hidden layers and hidden units the NN structure possesses and their connectivity pattern. In order to determine the best internal structure to be used, several experiments were conducted using different internal structures:

label	*hidden layers*	*hidden units*	*connections (with S1)*
H1	0	0	222,690
H2	1	100	22,269,000
H3	1	200	44,538,000
H4	1	300	66,807,000
H5	2	100+100	2,226,900,000

Figure 5.7: Number of hidden layers and hidden units used in the experimentation

H1 : no use of hidden layers or hidden units.

H2 : one hidden layer with 100 hidden units.

H3 : one hidden layer with 200 hidden units.

H4 : one hidden layer with 300 hidden units.

H5 : two hidden layers, each one with 100 hidden units.

It has been demonstrated for BP that taking two structures with the same learning capabilities, the one with the smallest number of hidden layers and hidden units performs better generalisations. On the other hand, taking two structures with the same generalisation capabilities, the one with the largest number of hidden units is capable of storing more patterns. The best internal structure is therefore the one which is capable of storing the largest number of patterns and at the same time capable of good generalisations.

A point which should not be underestimated is that a large number of hidden layers and units in a BP model makes the number of connections extremely large with the necessity of extremely long training and retrieval phases. Figure 5.7 reports the number of connections for each of the experimented structures. The time necessary to complete a training or retrieval phase depends on too many factors to be evaluated objectively. Moreover, the use of faster machines, like parallel machines, more suitable to the kind of computations necessary in NNs, could shorten considerably this time. In the following experiments the time spent in the training phase spans from 1 to 8 hours, while the time required for the retrieval phase spans from 8 seconds to 1 minute.

5.4.4 The learning parameters

There are several parameters which influence the learning behaviour of a NN based on the BP model. The main two are:

- the learning rate, here indicated with η, and the momentum, indicated with α (see Section 5.1.2);
- number of learning cycles.

label	α	η
P1	0.7	0.4
P2	0.9	0.2
P3	1.0	0.1

Figure 5.8: Values of the learning parameters used in the experimentation

The roles of the *learning rate* and of the *momentum* have already been explained in Chapter 3. The approach used here for the setting of their values is as follows. A first analysis was conducted in order to identify the range inside which the optimal values should be found. The fact that there exists a trade-off between learning rate and momentum makes it necessary to consider different combinations of these parameters. This first analysis identified the following three combinations (see also Figure 5.8):

P1 : η equal to 0.4 and α to 0.7: to stress the importance of the learning rate over the momentum.

P2 : η and α at their default values for the NN simulator PlaNet: η equal to 0.2 and α to 0.9.

P3 :η equal to 1.0 and α to 0.1: to stress the importance of the momentum over the learning rate.

As it can be noticed the combinations are centred on the default values of the simulator used. The default values are those that have been proved to be considerably better for most cases.

The *number of learning cycles* is a parameter which does not influence qualitative the learning but only quantitatively. The more learning cycles the NN is subject to, the more it learns. Although this is always true, it must be noticed that the learning is very fast for the first few learning cycles and it gets quite slow as the number of learning cycles gets higher. This is an implicit characteristic of the BP learning algorithm. BP is an error correcting algorithm, that is based on the correction of errors between the effective output of the NN and its target output. The correction to the weights the algorithm determines is large when the error is large and small when the error is small. The learning is therefore fast when the NN produces large errors, like at the beginning of a training phase, and slow when it produces very small errors, like after a large number of training cycles. The following three numbers of learning cycles were used in the training phase (see also Figure 5.9).

C1 : 300 learning cycles.

C2 : 600 learning cycles

label	*learning cycles*
C1	300
C2	600
C3	900

Figure 5.9: Number of learning cycles used in the experimentation

C3 : 900 learning cycles

These numbers were chosen after having performed an analysis of the minimum and maximum number of learning cycles necessary for the particular characteristics of the patterns to be learned.

5.5 Subsymbolic learning of domain knowledge and query adaptation

The following two sections report on the methodology used and the results obtained from some experiments regarding learning and using application domain knowledge for an IR application. The knowledge used is expressed in the form of application dependent associations among query descriptors and document descriptors and it relates to the conceptual model presented in Chapter 2.

The purpose of the following experiments is to prove that the SS is able to learn and use domain knowledge by learning associations between query descriptors and document descriptors from examples used during a training phase. The SS is based upon the conceptual structure presented in Chapter 2 and uses a NN processing framework based on the BP learning algorithm. The word "learning", in the context of the SS, is related to the ability of the NN to store and recall the associations among query and document descriptors used in the training phase. Moreover, these experiments will investigate the ability of the NN part of the SS to make use of the knowledge acquired. The use of the knowledge makes use of the ability of the SS to generalise what it has learned and use this information to adapt a new query to the characteristic of the specific application domain. In particular, the NN associates a set of document descriptors to the new query by recognising previous queries using the same query descriptors and by recalling already learned associations with document descriptors. This process can be seen as an adaptation of the original query to the application domain. Assuming that the set of training examples was properly set, the SS should have already been trained with some query similar to the new one. Therefore, it should be able to find an appropriate set of document descriptors which adapts the original query specification to the application domain in order to identify the proper set of relevant documents.

This process shares many similarities with the process used by the human memory to generate new information by recalling and processing information previously acquired from experience. When, for example, the human mind is in a problem solving context it tries to find and recall in the memory information about already solved problems analogous to the problem under examination. Then, the solution to the present problem is found by modifying good previous solutions to problems sharing important characteristics with the present one. This process is called "associative" memory and the way it can be implemented on computers has already been discussed in Section 3.5. The problem and its solution are then stored and they can be used later to solve new problems. This storing of information can be considered as a form of acquisition by "experience".

In the following experiments the "experience" is provided by the relevance assessments, which are part of the Cranfield test collection. The experience is provided by a set of exemplifying queries and relative sets of relevant documents. For the SS they are just examples of problems and their relative solutions. They can also be seen as solutions to previously arisen problems. Furthermore, these solutions can be considered as good ones, being found by experts, so they have a very high value. What the SS does is, therefore, to

use expert provided experience for solving retrieval problems posed by less expert users. Then, if the user can be considered an expert and if the solution SS provided is considered a good solution by the user, it can be incorporated into the SS's "experience", as another example of an "exemplary" solution, and it can be used later.

The typical form of the following experiments consists of training the system using a subset of the examples provided by the relevance assessment. The knowledge acquired by means of the training phase, is used to adapt the original user query formulation of the problem to take into account the experience acquired by solving similar problems in the training phase. So, after the training has been performed and its effectiveness evaluated, the SS is tested to see if it is able to generalise the associations learned and to respond correctly for the remaining part of the examples. The SS, being asked to solve problems which are part of the remaining set of examples, must find a solution very similar to the one provided by domain experts. This can be seen as a form of objective evaluation. The solutions provided by the SS are compared to the solutions provided by domain experts. In other words, the effectiveness of the responses of the SS is evaluated against the maximum level of effectiveness that any system could possibly obtain with the test collection used. After this has been done, the effectiveness of the SS is tested against the effectiveness of another system, which is based on the classical evaluation of similarity between the original user formulated query and the documents. This in order to assure that the solutions provided by the SS are not only in tune with those provided by application domain experts but also better than those provided by methods which make no use of application domain knowledge. Methods not applying domain knowledge are considered as using only the original user query formulation. Dice's coefficient (see Section 5.3) is used in both cases to evaluate the similarity between query (original or adapted) and documents. Recall and precision tables are produced for the two systems. In this way the query adaptation response is tested against the response obtained for the original query and both solutions to the IR problem are compared with the known right solution.

The following three sections report the methodology and evaluations of three different forms of learning, resulting from three different ways of teaching the SS using the examples provided by the relevance assessment. Unusual names are used here to indicate them. They have their origin in a way of considering knowledge which belongs to the Italian Renaissance.

During the Renaissance there were essentially two ways of considering knowledge. The most common way of seeing knowledge among scientists such as Leonardo Da Vinci, or Michelangelo, was to see it as a unique corpus, whose branches were deeply interconnected each other. It was not possible to study engineering without studying art or studying the anatomy of the human body without studying the laws of the physics which apply to it. The vision of the work of the scientist was one of someone building a palace, the taller the better, whose problem was first to build a good base for it, over which to develop the rest of the building. The scientist had to learn everything and there were no distinctions between branches of science. In analogy of this, the first and second kinds of learning used by

the following experiments are called *Total Learning* (TL) and *Horizontal Learning* (HL), which mean exactly the same concept. This is because the application domain knowledge is learned by training the SS using examples which relate to all sorts of IR problems the SS should solve. The learning is done on all the single illustrative problems at the same time.

Some time later, another vision of knowledge and science arose. This was conceived from the consideration that it is impossible to learn and know about every aspect of every branch of science. This idea found his way especially among British philosophers and "empirical" scientists. Scientists became more keen to deepen their knowledge on a specific subject than enlarge their knowledge on many different subjects. They did not bother about other subjects unless there were some relations to their investigations. This vision of science and of the acquisition of knowledge is still en vogue at present and it is going further with scientists becoming more and more specialised on subjects that tend to become narrower and narrower. Using the previous analogy of building a palace, this kind of learning can be considered as *Vertical Learning* (VL), because they tend to build going straight on a vertical dimension, without bothering to build a large base. In the experiments reported under the VL Section, solutions are learned for a problem at a time without concern to solutions to other problems.

5.5.1 Total learning

The purpose of this set of experiments was to investigate the ability of the system to learn application domain knowledge from a set of training examples of the form:

$$(q_i, d_1^i), (q_i, d_2^i), (q_i, d_3^i), ...(q_i, d_k^i), (q_l, d_1^l), (q_l, d_2^l), (q_l, d_3^l), ...(q_l, d_m^l),$$

where (q_i, d_s^i) is a single training example made of a query and a document which is known, from the relevance assessment, to be relevant to that query. The set of training examples is made of all the documents known to be relevant to a set of queries (see Figure 5.10).

Various experiments were conducted training the SS over a set of training examples corresponding to both a single query, and alternatively to more queries at the same time. Each training example is considered by the NN module of the SS as a pattern to be learned. Different ways of presenting the patterns (or presentation strategies) were used, such as submitting the training examples to the NN in a fixed order, or submitting them in a random order.

This set of experiments and all the experiments relative to the following subsections were performed and evaluated according to the methodology presented in Section 5.3.

Learning results

Several experiments were conducted to determine the ability of the SS to learn the associations between query descriptors and document descriptors described in the training

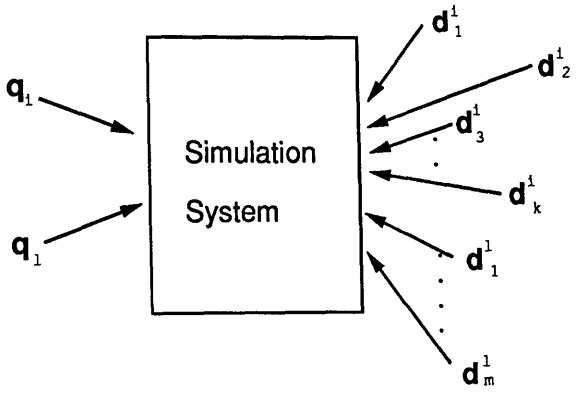

Figure 5.10: Training phase for Total Learning

ref.	*experiments*	*mean error*
1	S1 H2 P1 C1	0.007501
2	S1 H2 P2 C1	0.007123
3	S1 H2 P3 C1	0.007273
4	S1 H3 P2 C1	0.007421
5	S1 H4 P2 C1	0.007934
6	S1 H2 P2 C2	0.007103
7	S1 H2 P2 C3	0.007073
8	S1 H1 P2 C1	0.009003
9	S2 H1 P2 C1	0.008841
10	S2 H5 P2 C1	0.008392

Figure 5.11: Learning results for Total Learning with one query

examples. Experiments were conducted first using a single query and the set of its relevant documents. Then more than one query was used at once, incrementing sequentially the number of training examples.

Figure 5.11 refers to some illustrative mean errors reported while learning the association relative to the training examples of a single query. This was done with different queries and the data reported refer to the worst case of learning which occurred with a query whose set of relevant documents was considerably large. It was observed that the results were good for queries with a small set of relevant documents and they gradually got worse as the size of the set of relevant documents increased. This is in complete accordance with the NN theory. The larger the set of associations (or patterns) to be learned the bigger the error. The mean error is a measure of the difference between the training set and the retrieved set. In the present context this can be seen as the difference between two sets of documents. The first set is composed of documents the SS should associate with the query and it is considered the target of the learning. The second set of documents is the one the SS actually gives as a response when that query is submitted. This is the

real response of the SS. The higher the error, the bigger the difference between these two sets, and therefore the difference between the desired and real response of the SS. With the target being determined by the relevance assessments, this evaluation can be considered rather "objective", because it evaluates the SS versus the best results it could possibly give. Therefore it gives a measure which is independent from the user. This same interpretation applies to all the experiments related to the learning performance of the SS.

The first three results reported in Figure 5.11 display different errors corresponding to the use of the same NN topology and number of learning cycles but with different values of the two main learning parameters α and η. As can be seen the combination of parameters known as P2 (see Section 5.4.4) appears to be the best. Moreover, it shows a better stability in the learning compared with the other combinations. This can give better assurances that the learning would not be trapped in a local minimum of the error function whatever the pattern to be learned. P2 has therefore been chosen for all the other experiments.

A comparison of the figures obtained for the experiments 2, 4, 5 and 10 shows, ceteris paribus, the effect on the learning of changing the number of hidden units and/or hidden layers. Although the number of hidden units has important implications on the generalisation capability of the NN, the learning performance was considered of higher importance. Generalisation needs to be based a good learning in the training phase. In fact, it must be considered that in an operational environment the training phase has to be performed with a number of patterns which is very low compared with the number of real patterns a system has to deal with in the retrieval phase. Therefore, it is important for the system to have a good learning of the patterns used in the training phase, because it is on them that the successive generalisations are based. The possibility of several other successive phases of training, especially using relevance feedback from the users, could make this situation better, after a while. However, when the system is first used, it bases its generalisations only on the learning achieved during the training phase. This must therefore be considered of primarily importance. For this reason, the NN internal representation structure H2 was chosen. Another reason for the choice of H2 is related to the computational work necessary during the training phase. Figure 5.7 shows the number of connections relative to the various NN structures. They represent weights whose value has to be evaluated and modified several times during the training phase, and that are used to determine the activation of the document descriptors in the retrieval phase. A large number of connections means a large number of evaluations to be performed at each learning cycle and therefore a longer time for both the training and retrieval phases. The structure H2 is the minimum structure, regarding the number of connections necessary to have some sort of generalisation of the training examples.

Another interesting comparison, again ceteris paribus, concerns the number of learning cycles to be performed during the training of a single pattern. Figures of experiments 2, 6, and 7 of Figure 5.11 and 1,2 and 3 of Figure 5.13 show that the improvement gained by performing more learning cycles is not worth the time and computational resources used in obtaining it. Sometimes there was no gain in learning by performing a larger number of training cycles. Figure 5.12 shows an example of the learning graph. This graph shows the

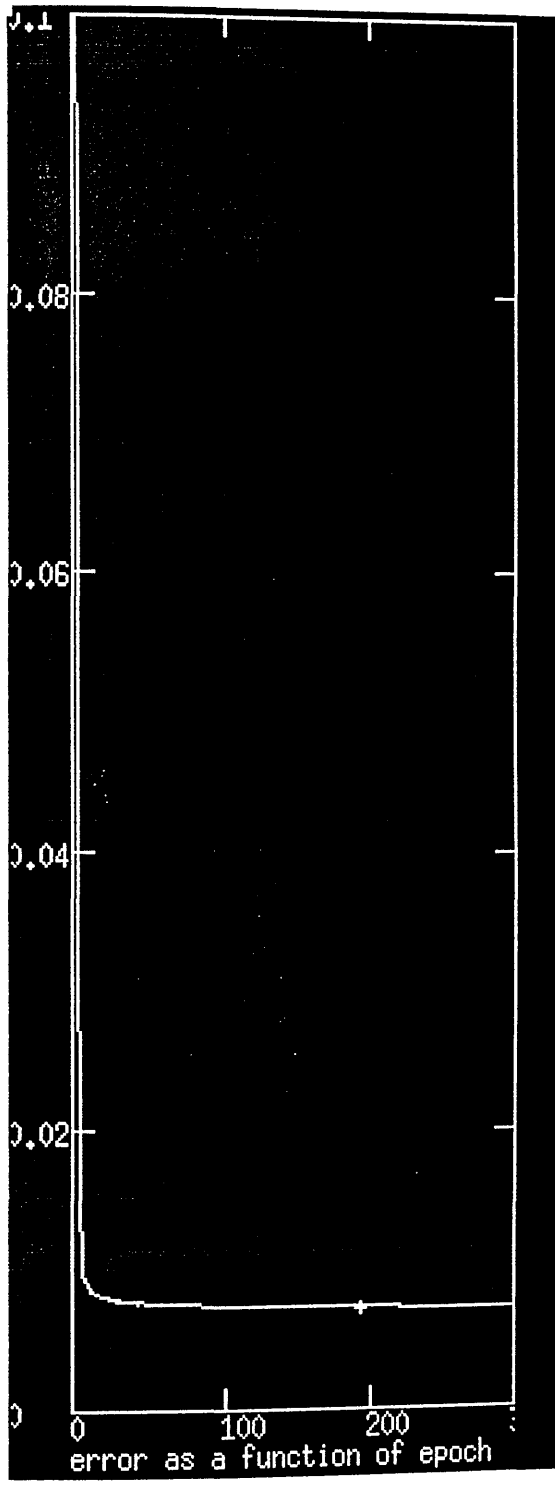

Figure 5.12: Example of a learning graph

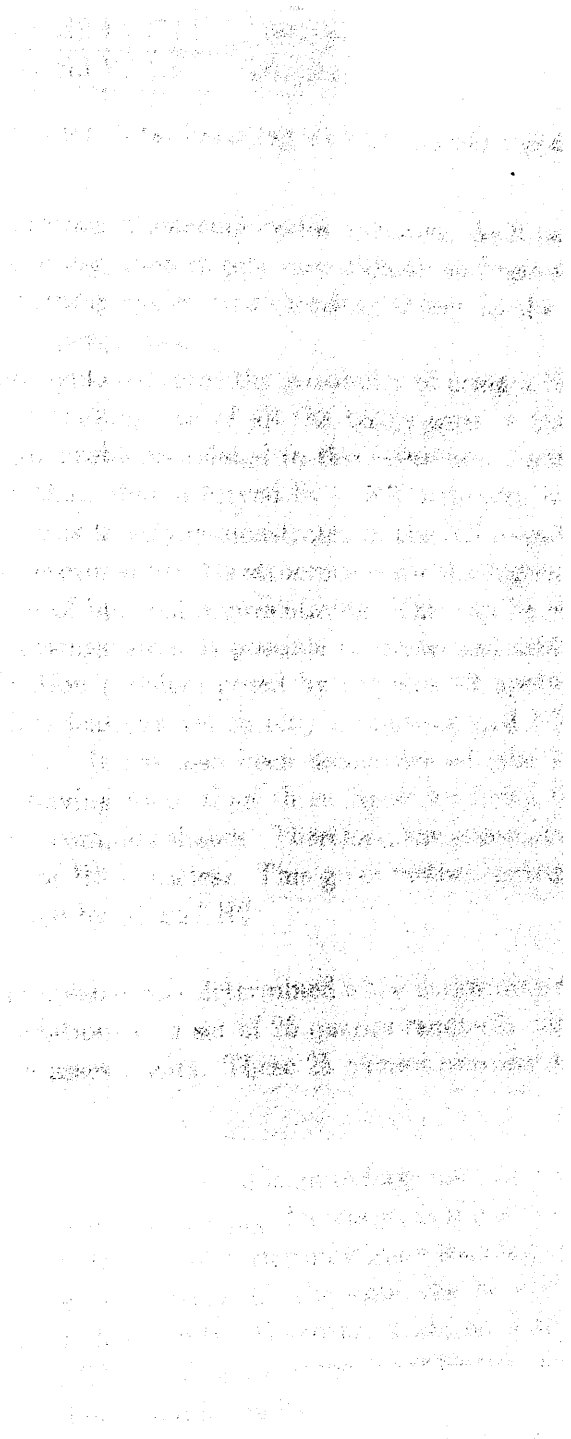

Figure 5.12: Example of a learning graph

ref.	*experiments*	*mean error*
1	S1 H2 P2 C1	0.009102
2	S1 H2 P2 C2	0.009079
3	S1 H2 P2 C3	0.009136
4	S1 H3 P2 C1	0.009006

Figure 5.13: Learning results for Total Learning with 25 queries together

value of the error function as the number of learning cycles increases. As it can be noted, the learning is very fast at the beginning, then it gets very difficult to improve the error. The option C1, that is using 300 learning cycles, was therefore chosen as the best option for the generalisation and retrieval experiments.

The last interesting set of experiments concerns the possibility of using a NN structure with no internal representation but making use of all the terms used in the document representation. There are two major problems related to this structure. First, its ability to generalise is considerably lower than that achieved by a NN structure with internal representation of the patterns. This was largely demonstrated in the NN literature (see for example [54]). Second, the error determined for this structure is usually higher than those determined for structure making use of internal representation. This can be explained by the fact that using an internal representation it is possible to model any arbitrary shape in the pattern space. The classification problem posed by complex IR applications, like the one under investigation, requires features which only a multi-layered NN structure, like H2, H3, H4 and H5 can provide. It has also been demonstrated (the Kolmogorow theorem) that there is no need of having more than three layers for being theoretically able to separate classes of arbitrarily complex shapes. Therefore, the enormous amount of computation introduced by the use of H5 is useless. This gives further motivation to the choice of the NN structure determined by S1 and H2.

Figure 5.13 reports exemplifying mean errors determined while performing some experiments relative to learning the associations of a set of 25 queries randomly chosen among the 42 making part of the relevance assessments. These 25 queries provides 114 training association patterns.

In this case the number of single patterns to learn is quite large and the results are, of course, worse than in the case of a single query learning. Moreover, as it has been explained above, the improvement in the error obtained by performing more learning cycles is not enough to justify the computational efforts involved. The same can be said about the improvement gained by adding more hidden units. Moreover, it has been demonstrated that the generalisation capabilities of a NN based on the three layers feedforward structure generally improve using a smaller number of hidden units.

The experiments of the following two sections are devoted to investigating the generalisation and the retrieval capabilities of the SS. The model identified by the labels $S1H2P2C1$ has been used.

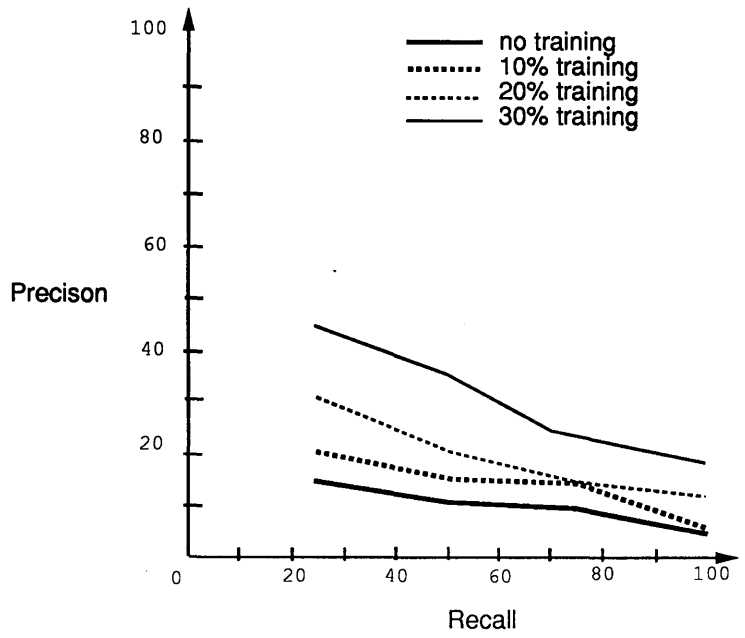

Figure 5.14: Precision/recall graphs and generalisation for Total Learning

Generalisation results

This set of experiments aimed at investigating the capabilities of the SS to perform generalisation by induction on trained associations between query terms and document terms.

The SS is supposed to have been trained trained using a set of the queries whose relevance assessments was known. The ability of NNs to perform generalisation by induction, as explained in Section 3.5, should enable the SS to generalise the associations it was trained on. This generalisation should enable the SS to deal with a new query. The new query is supposed to share some similarities with queries the SS has already been trained on. This is quite a sensible hypothesis if the SS has received proper training and the application domain is quite narrow. Recognising the similarities between the training set and the new query is the first step. The second is the use of the recognised similarities. Recognising similarities between the queries belonging to the training set and the new query is equivalent to recognising associations between the query terms used in the new query and document terms on the basis of those learned in the training phase.With this, the SS associates an appropriate set of document descriptors with the new query. This is equivalent to a modification or adaptation to the application domain of the original user formulated query. The two steps are performed by the NN structure of the SS in a way which does not allow distinction between them, in the same way as there is no distinction between recall and processing of information in the human brain.

The principle at the base of the query adaptation is that similar queries should have similar sets of relevant documents. Of course, the larger the training set the easier for the SS to find a query similar to the new one among those used in the training set. However,

as has been seen in the previous set of experiments, the larger the set of training examples used in the training phase the higher the error and therefore the lower the performance of the learning. It is therefore interesting to see how the generalisation varies with different numbers of queries used in the training phase. The results obtained for various dimensions of the training set are reported in Figure 5.14. They refer to precision and recall values reported for the entire set of queries after the SS was trained for only a portion of them.

As can be noticed from the graph, the precision and recall values are higher when the SS receives a larger training. This demonstrates that the generalisation capabilities of the NN perform better when there is more ground on which to base the generalisation. The NN acts as a features detector, detecting and encoding in the numeric values associated with the connections among nodes the associations between query descriptors and document descriptors present in the training examples. The generalisation is then performed on the basis of the features the NN was able to detect. The larger the set of training examples the better the generalisation. No tests were performed on this, but it seems reasonable to think that the precision and recall figures obtained for the case of absence of training are just those that it would be possible to obtain if document descriptors were chosen randomly.

Retrieval results

After the ability of the SS to perform the kind of generalisation required for dealing with new queries was tested, the performance of the SS was compared to that achieved by a conventional operational IRS. The performance of the SS was evaluated at different stages of training. A subset of 10 queries was chosen randomly among those not used in the training phase and used to evaluate the precision and recall of the two systems. The results are depicted in Figure 5.15. There are three recall/precision graphs reported in that figure are. The first is obtained from a conventional IRS based on the use of Dice's coefficient of similarity between the original query and the documents in the entire collection. The second and third are obtained by evaluating the same similarity measure between the set of document descriptors obtained for the SS in response to the submission of the query. They are queries that have been adapted to the application domain knowledge acquired by the SS during the training phase. Two different stages of TL are displayed. A comparison between the three curves enables an indicative performance comparison between the two systems.

The graph shows that at reasonable levels of training such as, approximatively 10% (4 queries used) and 20% (8 queries used) the SS still performs very poorly when compared to an operational IRS. The retrieval performance when measured in terms of recall and precision of the SS is not an acceptable level. The approach needs to be modified.

The most probable explanation for these results is in the structure of the training examples. The training examples represent patterns to be learned. These patterns are designed so that the input part, that is the set of query descriptors associated with the query, is constant for the entire set of patterns belonging to the same query. In this case

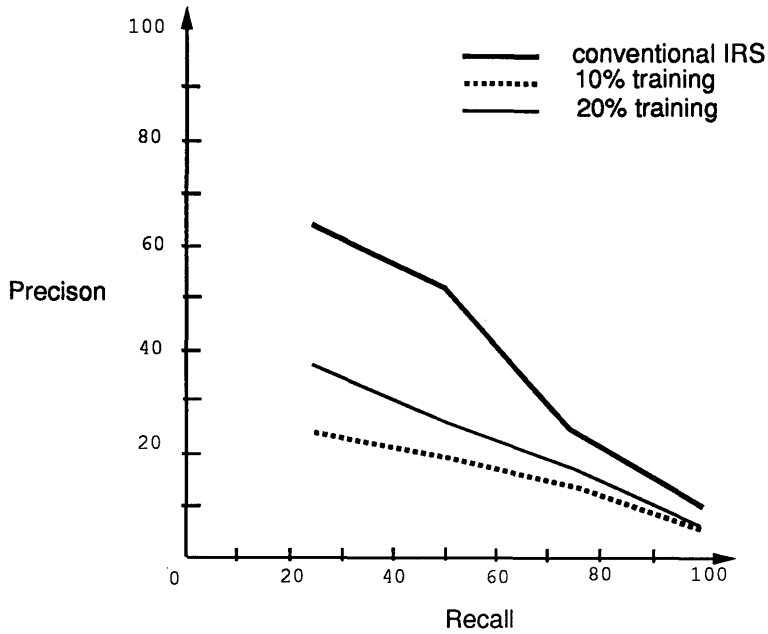

Figure 5.15: Comparison of precision/recall graphs for Total Learning

the NN has to learn to associate different outputs with the same input. This is not an ideal situation for a pattern matcher. The association of different outputs to the same input generates noise in the encoding of patterns. This makes it quite difficult for the NN to detect similarities among the patterns and it makes very difficult to perform a generalisation of these similarities among associations.

5.5.2 Horizontal learning

The purpose of this set of experiments was to see if it was possible for the SS to learn application domain knowledge, to be used in the query adaptation process, from a set of training examples of the form:

$$(q_i, c_j^i)$$

where (q_i, c_j^i) is a single training example made of a query and the cluster representative of the set of documents known to be relevant to that query.

The motivation for performing such an experimentation came from an analysis of the results obtained in the TL (see Section 5.5.1). As it has already been explained, the use of different training examples where the input is constant while the output varies sometimes quite considerably, causes the NN to be subject to too much noise to be able to generalise what it has learned. A possible way of avoiding this problem could be in using some kind of synthesis of the characteristics of the set of documents relevant to a query. This is

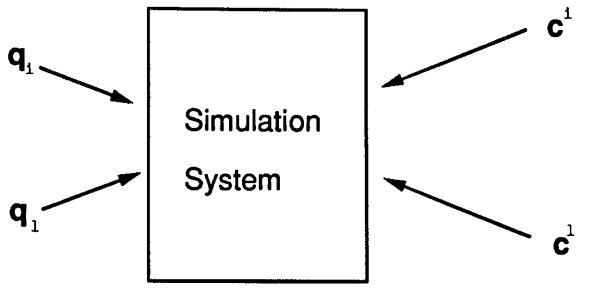

Figure 5.16: Training phase for Horizontal Learning

equivalent to using a single document representation for each query in the training phase, thus having a single different output for each input of the patterns. However, this unique document representation should characterise all the relevant documents for a query. The most common way of obtaining such a representation is by clustering the set of documents in order to produce a cluster representative.

A *cluster representative* is simply an object which summarises and represents the objects in the cluster. There are many different procedures for obtaining a cluster representative but not all of them are useful in IR. A survey of some of the procedures used in IR is reported in [1]. The procedure used here is referred to as "centroid evaluation". However, being that the documents were represented using a binary form, a variant of this procedure was used. This consists in determining a vector $c^i = (c^i_1, c^i_2, ...c^i_n$ whose dimension is the same of that of the documents. A generic j element of this vector is determined as follows:

$$c^i_j = \begin{cases} 1 & \text{if } \sum_{l=1}^{n} d^i_{l,j} > 1 \\ 0 & \text{otherwise} \end{cases}$$

where $d^i_{l,j}$ is the jth element of the binary representation of the lth document relevant to the query i. The intuition is that terms occurring more than once in the cluster should be taken into consideration as representative of the cluster.

Consequently, the following experiments were performed using patterns which were different to one another. The experiments were performed using a single query or more queries together and using different presentation strategies, such as submitting the training pairs to the NN in a fixed order, or submitting them in a random order. No relevant difference was detected regarding the order of presentation of the training pairs. The random presentation order was therefore arbitrarily chosen.

Learning results

Several experiments were conducted in order to investigate the ability of the SS to learn the associations between query descriptors and the document representative descriptors. This set of experiments was conducted using more than one query at once, incrementing sequentially the training set. The case of using a single query is trivial, being similar to

ref.	*experiments*	*mean error*
1	S1 H2 P2 C1	0.008503
2	S1 H2 P2 C2	0.008394
3	S1 H2 P2 C3	0.008345
4	S1 H2 P1 C1	0.008409
5	S2 H2 P3 C1	0.009001

Figure 5.17: Learning results for Horizontal Learning

the case of the training of a single query with a single relevant document, though its effects on generalisation and retrieval may be very different.

Figure 5.17 refers to the mean errors reported while learning the association of a set of queries. In order to make the results comparable, 25 queries, corresponding to 25 patterns, were used. They are exactly those used in the TL, where they were chosen randomly among the original set of 42 queries. Several other different sets were tested.

The results summarised in Figure 5.17 show that the learning performance have increased compared to those obtained for TL. The examples 1,2, and 3 refer to the same model used in TL with different numbers of learning cycles. The best result appears for C3. This however does not seem to justify the time required for the training. The results obtained with C1 number of learning cycles appear to be good enough. It must also be considered that the time required to train the SS is now considerably lower, due to the fact the number of patterns used in the training is 25 instead of 114. The time required for the training is now, in this case, only around 22% of the time previously necessary.

The results were again gradually getting worse as the size of the set of queries used in the training increased. This is again in complete accordance with the NN theory. The larger the set of associations to be learned the larger the recall error. However, in this case the situation was better than in TL. The adding of a new query to the set of training examples adds only one pattern, while in TL it adds as many patterns as documents relevant to that query. In HL the increase of the error is linear to the increase of the number of queries used in the training, while in TL it increases more than linearly.

It is also interesting to note that the results of example number 4, which uses a different combination of learning parameters, are better than those obtained in example number 1. It seems that with this pattern structure the combination P2 gives more effective learning than the combination P1, which appeared more successful with the pattern structure used in TL.

Generalisation results

This set of experiments aimed at investigating the capabilities of the SS to perform generalisation by induction on associations between query terms and document terms which were used in the training phase. The principle at the base of this investigation has already been explained in Section 5.5.1. The only difference here is that the SS is using "second

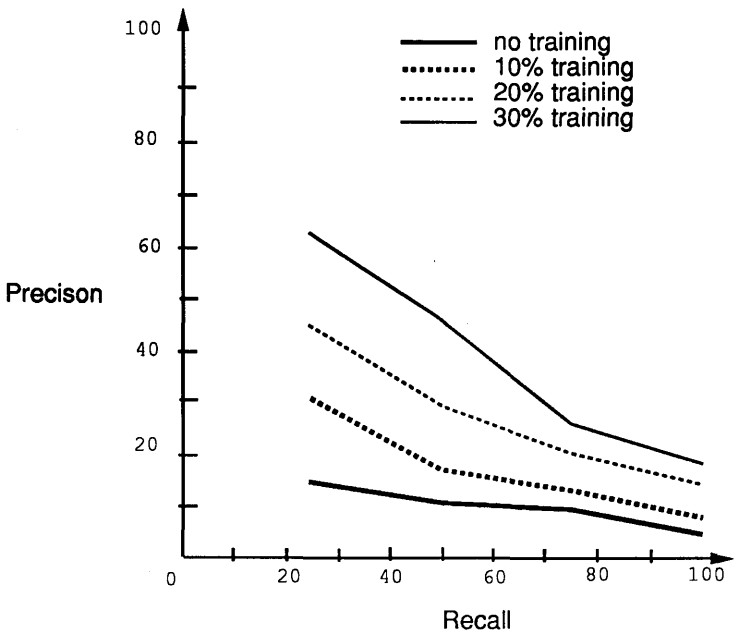

Figure 5.18: Precision/recall graphs and generalisation for Horizontal Learning

hand" information, in the sense that it does not use directly the terms used in the representation of the relevant documents, but terms used by the representative of the entire set of relevant documents. This implies a loss of information, but the number of training examples decreases enormously, easing the computational problems of the NN learning.

Results obtained evaluating the entire set of queries for various dimensions of the training set are reported in Figure 5.18.

As it has been noticed before, the SS performs better in terms of generalisation when it can base its generalisations on a higher number of training examples. The generalisation performance of the SS appear considerably improved. Comparing the generalisation performance of HL with those of the TL, the SS shows now higher values of recall and precision for any one of the different levels of training. This demonstrates that the simpler the patterns used to train the NN, the easier it is for the NN to encode them and detect similarities among them. Although the actual number of different patterns was reduced, the fact that the patterns are simpler and clearer facilitate not only their encoding but also the feature detection the NN has to perform to be able to generalise the learned associations to new queries.

Retrieval results

Here the performance of the SS is compared to that of a conventional operational IRS. The results are depicted in Figure 5.19. Three recall/precision curves reported in that figure. The first is obtained with a conventional IRS based on the use of Dice's coefficient

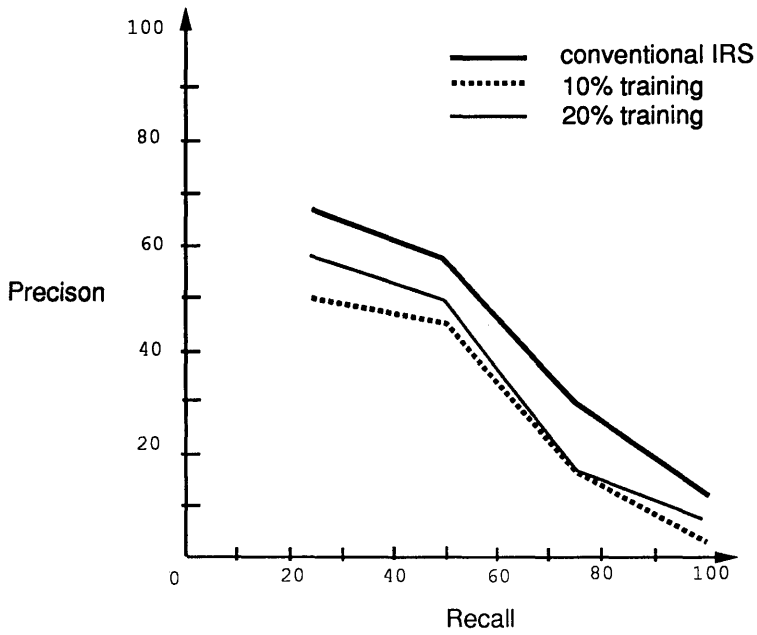

Figure 5.19: Comparison of precision/recall graphs for Horizontal Learning

of similarity between the original query and the documents in the entire collection. The second and third are obtained by evaluating the same similarity measure between the adapted query and the documents. The query adaptation process is based on training performed with 10% and 20% of the queries whose sets of relevant documents were known. A comparison between the curves enables a performance comparison between the two systems. The results are based on the precision and recall measures obtained from querying the SS with 10 queries randomly chosen among those not used in the training.

The graph shows that there is an improvement in the retrieval performance of the SS due to the new type of training. However, the performance of the SS is still lower than that achieved by a classical IRS based on the evaluation of the similarity between the original query and the document representations. It is easy to predict that the performance of the SS would still improve with further training, although performed experiments show that the improvement tends to decrease. Further, it is not reasonable to assume one will perform a lot of training, covering large sections of the application domain knowledge. In a hypothetical operational IRS based on this approach, training comparable in scale to the one performed in these experiments would involve thousands of queries, which should be produced by experts, and which should be carefully designed in order to cover as completely and uniformly as possible the application domain.

5.5.3 Vertical learning

The purpose of this set of experiments was to see if it was possible for the SS to learn application domain knowledge from being shown a set of training examples of the form:

$$(q_i, d_1^i), (q_i, d_2^i), (q_i, d_3^i), ...(q_i, d_l^i)$$

where (q_i, d_j^i) is a training example made of a query representation and a document representation. The document is known, from the relevance assessment, to be relevant to that query. The set is made of only a subset (l documents) of all the documents (k documents, with $k > l$) known to be relevant to that particular query (see Figure 5.20). Different dimensions of the learning subset were used. Experiments are identified as the ratio of the cardinality of the learning set over the entire set of documents known to be relevant. The parameter k is determined as follows:

$$k = \frac{card(learning\ example)}{card(relevance\ assessment)}$$

The values of k is therefore in the interval $[O, 1]$. For a value of k equal to 0 there is no training, while for k equal to 1 there is complete training. Three illustrative values of k were used: 1/3, 1/2, and 2/3.

There are various differences between VL and TL. The main one is that VL concerns a single query and uses information about some documents known to be relevant to finding other relevant documents. In this way it is similar to classical relevance feedback. In TL, on the other hand, the information the SS receives is complete but it is about more subjects at the same time. Moreover, a bigger difference exists between the two different retrieval tasks that have to be accomplished in TL and VL. In TL the SS has to use application domain knowledge to adapt the original query formulation to the application. In particular it has to generalise the information about queries and their relative relevant documents acquired during the training phase to find the proper query adaptation of the original user formulated query. This new query will enable the retrieval of documents which appear to be relevant to the query according to the application domain knowledge the SS possesses. The SS uses this heuristic rule: "if these sets of documents were relevant to these queries, than this set of document must be relevant to this query". On the other hand, in VL the SS has to use information about the kind of documents that are considered relevant to a query in order to find other documents which on the basis of this information appear to be relevant too. Therefore, it has to adapt the query in order to add the information provided by the fact that it is known that some specific documents are relevant to it. The heuristic rule the SS has to answer in this case is: "if these documents are relevant to this query, then these other documents must be relevant too". These two tasks are not mutually exclusive, but they can be thought to work together at different stages of a query session. The SS could first point the user to a set of documents which, according to its knowledge of the

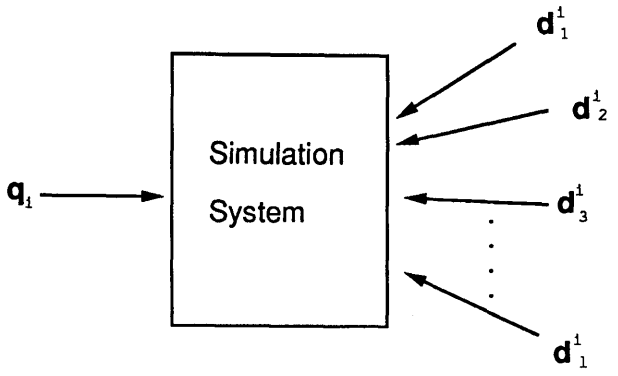

Figure 5.20: Training phase for Vertical Learning

ref	*experiments*	k	*mean error*
1	S1 H2 P2 C1	1/3	0.007601
2	S1 H2 P2 C1	1/2	0.007806
3	S1 H2 P2 C1	2/3	0.007879
4	S1 H2 P2 C2	1/3	0.007587
5	S1 H2 P2 C3	1/3	0.007529

Figure 5.21: Learning results for Vertical Learning

application domain, appears to be relevant. Then, using relevance feedback from the user on the previous set of documents, the SS could point out other relevant documents which did not appear in the previous set. In this case the general application domain knowledge will be first used to locate a set of document considered relevant, while a more specific knowledge related to the specific query and acquired through interaction with the user will be used later to identify more precisely the relevant documents.

The following experiments were performed using a single query, and with different presentation strategies, such as submitting the training couples to the NN in a fixed order or submitting them in a random order. The best results were obtained with the random order. The results reported here refer to this particular presentation strategy.

Learning results

In order to investigate the ability of the SS to learn the associations between query descriptors and the document descriptors, experiments were conducted using only one query at once, incrementing sequentially the size of the training set. The training set of patterns is composed of only one query, that is only one input pattern, and a set of documents, that is a set of output patterns.

Figure 5.21 refers to the mean errors reported while learning the association of a set of documents to a single query. The data reported referred to the worst case of learning

reported, which was obtained with a query whose set of relevant document was quite large. They refer to three cases of VL depending on 3 different values of the parameter k.

Again, the results were gradually getting worse as the size of the set of relevant documents, or the value of k were increasing. However, it is easy to note the big increase in the learning performance. This is due partially to the relatively small set of patterns involved in the learning and also to the the very specific application domain context in which the training is performed. The fact that all the documents used in the learning are relevant to the unique query used in training makes the set of document descriptors involved in the training quite small. This makes it easier for the NN to associate different output patterns with the same input pattern.

It is interesting to note that the increase in the error is quite considerable passing from indicating as relevant one third of the known relevant documents to one half. The number of patterns involved gets larger. The increase in the error obtained by passing from one half to two thirds is smaller because the number of patterns used is already large enough to cause a considerable error to occur.

Generalisation results

The focus of these experiments is to test the ability of the SS to generalise the information obtained in the very specialised training session and, after that, retrieve documents sharing similarities, in terms of their representation, with the documents used in the training set.

The principle at the basis of query adaptation is that documents which are relevant to the same query must have some similarities among them. The SS should detect these similarities and use this information to retrieve other documents which are similar to those indicated as relevant. This is done by means of an adaptation of the query which should take into account the similarities among relevant documents detected in the training phase. If some terms appear to be quite common among the set of relevant documents, they will be added to the original query and used to rank with a higher position in the retrieved list of documents those which share these terms with the documents used in the training phase. Of course, the larger the training set of documents the easier for the SS to detect similarities among them. However, as has been seen in the previous sections, the larger the set of patterns to be learned the higher the error and therefore the lower the learning results. It is therefore interesting to see how generalisation varies with different numbers of documents in the training set.

Figure 5.22 reports the results obtained for various dimensions of the parameter k, that is with different dimensions of the set of documents used in the training session. As can be noticed, the generalisations get better when the SS possesses larger amounts of information on which to base them. It is important to note that it is not necessary to provide the SS with such a large amount of relevance information all at once. It is possible to provide this information gradually in an interactive process. The user could point to a small set of relevant documents and let the SS reorder the relevance evaluation of the entire collection

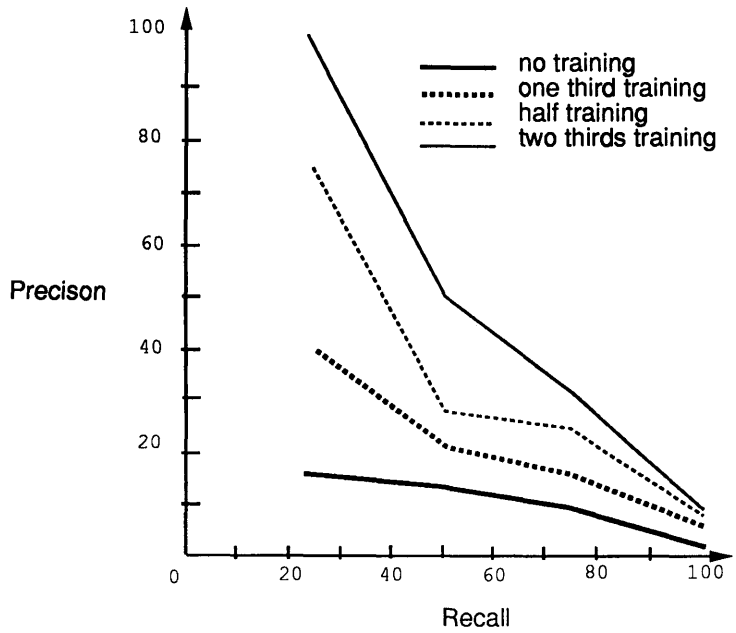

Figure 5.22: Precision/recall graphs and generalisation for Vertical Learning

according to this information, by means of a training session. Then, the user could look again through the documents and identify some other relevant documents which will be used, together with those provided before, for another training session. In this way the generalisation performance of the SS will improve and the SS will identify new relevant documents more and more precisely.

Retrieval results

In this section the performance of the SS is compared to that of a conventional IRS. The results are depicted in Figure 5.23. Three recall/precision curves are reported in that figure. The first is obtained with a conventional IRS based on the use of Dice's coefficient of similarity between the original query and the documents in the entire collection. The second and third are obtained by the evaluating the same similarity measure between the adapted query and the documents. A comparison between the curves enable a performance comparison between the two systems. Again the results refer to a test performed using 10 queries chosen among those not used during the training phase.

The graph can be interpreted as a comparison between results obtained using a conventional system without relevance feedback and those obtained using various dimension of relevance feedback on a system employing a subsymbolic knowledge representation structure to store the user relevance feedback. It is interesting to note that the results obtained by VL are always better in term of precision than those obtained by the traditional IRS. Moreover, the precision obtained by the two thirds training is better that the one obtained

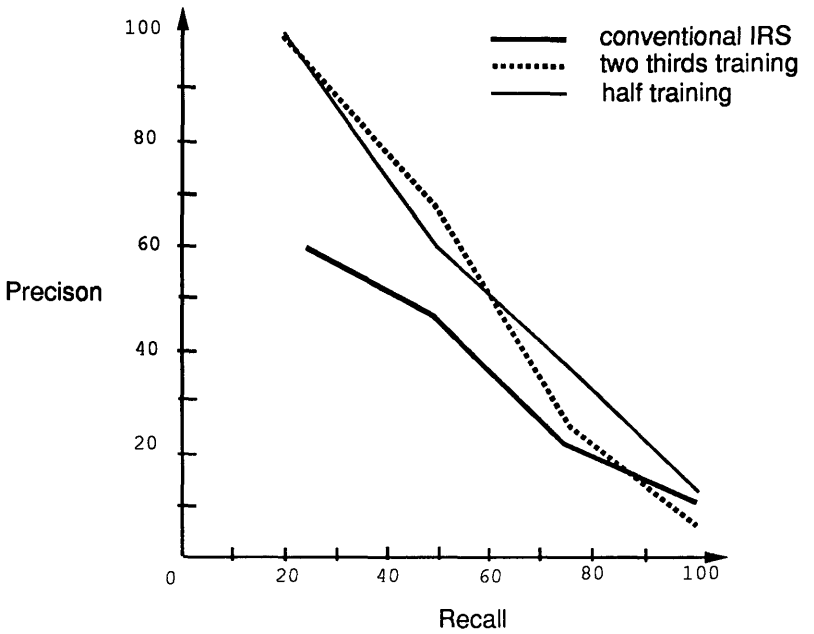

Figure 5.23: Comparison of precision/recall graphs for Vertical Learning

by the half training for low values of recall, while the situation is the opposite for high levels of recall. A possible explanation for this can be that very specific training, obtained by the use of a large number of relevant documents does not favour high recall but high precision and vice versa. The behaviour of the SS is therefore optimal because when the information to be used is small it favours a larger recall, while when the information gets more specific the SS favours precision.

Chapter 6

Conclusions

6.1 Conclusions from the experimental results

The experimental results reported in the previous chapter can be summarised in the following three points:

- query adaptation resulting from TL gave disappointing results;
- query adaptation resulting from HL gave results which, with a bit more tuning of the learning process, can be considered comparable to those given by using the original query in the similarity evaluation process;
- the query adaptation resulting from the VL gave good results, better than those given by using the original query, but are they good enough to justify the efforts involved?

However, these results show some interesting facts which ought to be pointed out.

Query adaptation produced after a TL training session does not give good results. The NN is not able to learn and generalise the characteristics of the application domain knowledge. The amount of information submitted to the system seems to be too much and in the end the system shows a form of "confusion", which could be compare to the one a student could show after a disorganised session of study. Someone has to filter and prearrange what the student is supposed to learn so that the learning can be more effective. After all, this is what lectures, tutorials or introductory books are for.

Query adaptation produced by the system after having been trained in the way described as HL gives results which are similar to those provided by the use of the original query in the evaluation of the similarity between query and documents. However, the adapted query is most of the time quite different from its original formulation. The fact that the original query and the adapted one give the same level of performance in term of recall and precision is quite interesting. The two sets of documents resulting from the use of the two different formulations of the same information need, the original query and the

adapted one, are sometimes quite different. The adapted query is often able to retrieve relevant documents which the original query is not able to identify. In fact, the adaptation process enables the determination of useful descriptors which are not specified in the original query, but that are considered by the system to be useful for the determination of the set of the document relevant to the information need.

Practically speaking, the system acts in this way: it uses the domain knowledge stored in the NN to adapt the query to what it already knows producing a new query, which can be quite different from the original one, to be used in the matching phase. What happens is that the system gives too much importance to the domain knowledge stored in the NN, modifying the query accordingly and doing so it looses some of the information contained in the original query. Some descriptors specified in the original formulation of the query are, in fact, not used by the adapted version.

There are three possible ways to make use of this lost information.

The first is to tune the adaptation process so that it combines in a better way the information provided by the domain knowledge base with that provided by the user in the original formulation of the query. This involves a study of user typology. It is necessary, in fact, to determine which of the two formulations of the query (the original or the adapted) is to be trusted the most. This can be done only by knowing the user and the level of the user's knowledge of the application domain.

A simpler way would be to use an hybrid approach and combine the results of the retrieval process of both the adaptive retrieval process and the traditional one. The user could express the level of confidence in his query formulation and the system could rank the documents in order of their evaluated relevance merging the results of the two retrieved lists accordingly. If the user is very confident in his representation of his information need into the query then the results of the traditional retrieval would become more important, because it does not modify the query at all. Moreover, if the retrieval results do satisfy the user, the query and the set of relevant documents obtained can be stored and considered for use in some training session. On the other hand if the user is not so confident in the expression of his information need, the adapted query, expression of the system's perception of the user information need, would become more important. In both cases all the information contained in the original query formulation will not be lost. This can be seen as an application of the "principle of multiple evidence", which was first used by Croft et al. in [21]. Both the results of the retrieval using the original and the adapted query provide evidence of the relevance of documents. The main problem lies in the combination of this multiple evidence which can be tackled in the above described way.

Another simple way would involve the evaluation of the similarity between the new query and those the system has been trained with. However this would involve keeping record of all the queries used in the training phase and this may not be easy as the life of the system gets longer and it is subject to more and more training.

The results produced by the adaptation process after a VL training session has been performed show that it is easier to learn about a very narrow topic when there is no other knowledge which can interfere with learning. The advantage of such a result is that it is

possible to distinguish two different kind of query:

1. a generic query, in which the user expresses his non well defined information need;
2. a very specific query in which the user is able to point out some document he knows to be relevant.

In the first case the query adaptation resulting from HL or a combination of traditional and adaptive retrieval can be used. The result provided by that retrieval should be good enough to enable the user to point out some relevant documents among those retrieved. In this case he will be able to produce a more specific query as in the second case.

In the second case the user, either in a process of relevance feedback or in a formulation of the query by giving example of relevant documents (known as "query by example"), can provide a more specific formulation of his information need. In this case the system can use this information to retrieve other relevant documents without taking into account its knowledge of the entire application domain, but focussing only on that particular topic.

The two situations are typical of an interactive query session and they can be combined together as is done in many systems using relevance feedback devices.

Again the result of a good query session, that is a query specification and a set of relevant documents, can be stored and used in some training sessions. This process should keep improving the performance of the system's response to the first kind of query. However, this has not been proved yet and some other experimental investigations should be devoted to the analysis of the improvement of the domain knowledge base.

6.2 Operational Adaptive Associative IR

There are many issues related to the operational use of the proposed approach to IR. Most of these issues need further investigation.

In particular, the first issue is related to the analysis of the advantages in terms of retrieval effectiveness of this approach compared with its costs in terms of allocation space and computation time. These problems have not been tackled in this thesis, but it seems necessary to investigate them before attempting the operational use of the ideas presented here. This is, of course, a software and data engineering problem which is far beyond the intended purposes of this MSc. thesis.

Moreover, many theoretical considerations should be investigated regarding the minimum number of training examples to be used in the training phase to enable the system to have an acceptable starting level of generalisation capabilities. The problem is quite difficult to solve in the case of IR applications where the number of possible queries, and therefore of possible patterns, is very large compared to those it would be possible to use during the training phase. A possible solution is to use a hybrid process of query adaptation. At the beginning of the life of an operational IRS based on this approach the process of query adaptation could be used in a softer way. The modification made by the query adaptation process could be made minimal. The system could use the original query

formulation with practically no modifications. Then, using relevance feedback from the users as a means of training the system, the process of query adaptation could become more effective and take a more substantial part in the retrieval process. However, these considerations do not change the fact that it is always possible to use VL as a useful device for the relevance feedback process.

6.3 Future research work

A direction of further research which seems very interesting is concerned with the use of different representation elements in the query representation language and in the document representation language. It would be very interesting, for example, to see if it would be possible to train the SS to associate descriptors (or, more generically, words) in different languages. This could be very useful for querying a document base whose documents are indexed using a language A with queries expressed using language B. A and B could be two different natural or artificial languages like, for example, English and German, or Italian and Chinese and so on. In the training process of association between descriptors used in a query and descriptors used in its relative relevant documents there is no symbolic use of the semantics of the descriptors. Only information regarding their co-occurrence with other descriptors is used and stored in a sub-symbolic form by the NN. There is no reason why it should not be possible to use in the training phases queries represented using descriptors from the vocabulary of a language A (let say for example Italian), and relevant documents represented using descriptors from the vocabulary of another language B (let say for example English). It would be very interesting to see if the experimental results obtained here are still valid. If so, there would be many possible applications of such a result, enabling the interrogation of various document bases with the same query at the same time, regardless of the language used for the document indexing. No more language barriers would exist in the querying of different document bases around the world.

However, the most important next step is certainly the testing of the scaling up of the generalisation performance to larger sized collections. This is a very important point, which it is necessary to assess before exploiting the results described in this thesis to other more challenging directions.

Appendix A

Appendix

A.1 Cranfield test data

This section reports some information about the Cranfield test document collection that were used in the experimentation.

teach-tab

The following is a table reporting the queries used in the experimentation and the documents which are evaluated to be relevant to them. These data were extracted from the Cranfield 1200 documents collections provided by Prof. Karen Spark-Jones, from Cambridge University, using information reported in [51].

TEACHING DATA FOR NN LEARNING:

Query number:	Relevant document numbers:					
1	1	21	22			
2	104	105	106	107		
3	4	30	31	32	33	57
4	8	13	58	59	60	
5	8	13	58	59	60	199
6	34	35	36			
7	41	42	72	89	94	
8	39	40	41	42		
9	62	115				
10	66	67	68	69		
11	16	44	45	91		

```
12      46   47   48   49   50   51
13      20  109  110  111  112  113
14      73
15      79   80   81   82   83   84   85   86   87  101  108  192
16      67   79   80   81   82   83   84  101  192
17      89   90   92   93   94
18      95   96  154  198
19     121  122  123  124
20      11   43
21     128  129
22     126  127  129  130
23     142  147
24     141  142  143  144  145  146  147
25     157  158
26      71  157  158  159  160
27      28   29   56   71  159  160
28     161  162  163  164  165  166  167
29     168  169
30     140  170  171  172  174  175
31       2    3    9   18   19  114  196  197
32       5    6    7   26
33      12   24
34      10  126  127  128  129
35      34   35   36   38  100
36      34   35   36   37
37      66   67   68   69
38      49   50   52   53   54   55  177
39      17  134  135  153  156
40     189  190
41     119  191  192  193  194
42      57  181  183  184  185  186  187  188
```

A.2 Query Processor

In this section the parts which compose the Query Processor module of the simulation system are described.

repr-q.c

The following is the listing of the C program which composes the Query Processor module of the Simulation System.

```
/* representation building program for queries

rows are queries and columns are terms

the fact that an "a" is positioned on a certain line-column states
that in the queries identified with the number of that line uses the
terms identified by the number of the column to represent
its informative content */

#include<stdio.h>
#include<stdlib.h>

#define MAXTERMS (195+1)        /* because the index 0 is never used */
#define MAXQS (42+1)            /* same reason as above */
#define IDENT 'a'               /* presence of term */
#define NOTHING '0'             /* absence of term */

main(argc, argv)
     int argc;
     char *argv[];

{
  FILE *f;
  char rep[MAXQS][MAXTERMS];
  int docnum, ternum, i, j;


  if (argc != 2)
    {
      printf("Usage: repr <file-name>\n");
      exit(-1);
    }

  if ((f = fopen(argv[1], "r")) == NULL)
    {
      printf("File %s not found\n", argv[1]);
      exit(-1);
    }

  while (fscanf(f, "%d", &docnum) != EOF)
    {
      fscanf(f, "%d", &ternum);
```

```
      while (ternum != 0)
        {
          rep[docnum][ternum] = IDENT;
          fscanf(f, "%d", &ternum);
        }
    }
  fclose(f);

  for (i = 1; i < MAXQS; i++)
    {
      for (j = 1; j < MAXTERMS; j++)
        if (rep[i][j] == IDENT)
          putchar(IDENT);
        else
          putchar(NOTHING);
      printf("\n");
    }
}
```

A.3 NN simulator

In this section the parts which compose the Neural Network simulator module of the simulation system are described.

ir

The following is the listing of the source code which composes the Neural Network Simulator module of the Simulation System. The language used is the PlaNet's specification language.

```
#!/bin/csh -f
source /users/staff/fabio/NN/PlaNet/RunNet
set NIN = 195
set NHID = 100
set NOUT = 1142
set NET = n.3layer
set PAT = p-odd
set WTS = save
network Nin=$NIN Nout=$NOUT Nhid=$NHID $NET
pattern $PAT
whatis alpha eta
```

```
names layer
nset print 50
date
cycle 300
date
savefile ww-ir-300-5
save
savefile close
date
cycle 300
savefile ww-ir-600-5
save
savefile close
cycle 300
savefile ww-ir-900-5
save
date
savefile close
quit
netclear
```

n.3layer

This is the source code describing the internal structure of the three layers feed forward neural network and of the BP learning procedure adopted.

```
## n.3layer: generic definition of a 3 layer back-prop network.
## define Nin, Nhid, and Nout as desired.

define ErrMsg \n\tread\swith\s'network\sNin=<no-of-input>\sNhid=<no-of-hidden>\s
Nout=<no-of-output>\sn.3layer'\n

IFNDEF Nin;  printf ErrMsg; exit; ENDIF
IFNDEF Nhid; printf ErrMsg; exit; ENDIF
IFNDEF Nout; printf ErrMsg; exit; ENDIF

                ## DEFINITIONS OF LAYERS

layer Input Nin
layer Hidden Nhid
layer Output Nout
```

```
                ## DEFINITIONS OF INPUT/TARGET BUFFERS

target Nout
input Nin

                ## DEFINITIONS OF CONNECTIONS

connect InputHidden Input to Hidden
connect HiddenOutput Hidden to Output

        ## PROCEDURE FOR ACTIVATING NETWORK FORWARD

procedure activate
        input           Input
        forward         InputHidden
        activation      Hidden
        forward         HiddenOutput
        activation      Output
        target          Output
end

        ## PROCEDURE FOR TRAINING NETWORK

procedure learn
        call            activate
        backward        HiddenOutput
        delta           Hidden
        learn           InputHidden
        learn           HiddenOutput
        learnbias       Output
        learnbias       Hidden
end
```

do-out

This is a C-shell script used to produce and store in files the output patterns for different input patterns.

```
#!/bin/csh -f
```

```
foreach pat (pat16 pat17 pat18 pat19 pat20 pat21 pat22 pat23 pat24 pat25)
        pattern $pat
        present 0
        print Output > out-300-$pat
end
```

A.4 Matcher

In this section the parts which compose the Matcher module of the simulation system are described.

patterns.c

This C program builds the input and output patterns to be used in the learning and in the retrieval phase. It takes the document and query representation matrices and, given a series of pairs conposed of a query number and a document number, it builds up the training example and stores it in a file.

```
/* building patterns program

the program takes the data from doc-rep-matrix e q-rep-matrix and builds
the patterns according to the query and documents specification given
interactively */

#include<stdio.h>
#include<stdlib.h>

#define MAXT_Q 195
#define MAXT_D 1142
#define MAXQS 42
#define MAXDOCS 200



main(argc, argv)
     int argc;
     char *argv[];

{
  FILE *f, *g, *p;
```

```
char repq[MAXQS][MAXT_Q], repd[MAXDOCS][MAXT_D];
int docnum, qnum, i, j;


if (argc != 3)
  {
    printf("Usage: patterns <doc-rep file-name> <q-rep file-name>\n");
    exit(-1);
  }

if ((f = fopen(argv[1], "r")) == NULL)
  {
    printf("File %s not found\n", argv[1]);
    exit(-1);
  }

if ((g = fopen(argv[2], "r")) == NULL)
  {
    printf("File %s not found\n", argv[2]);
    exit(-1);
  }

for (i = 0; i < MAXDOCS; i++) /* read the docs representation matrix */
  {
    for (j = 0 ; j < MAXT_D; j++)
      repd[i][j] = getc(f);
    getc(f);
  }

for (i = 0; i < MAXQS; i++)   /* read the queries representation matrix */
  {
    for (j = 0 ; j < MAXT_Q; j++)
      repq[i][j] = getc(g);
    getc(g);
  }

fclose(f);
fclose(g);

p = fopen("pat-new", "w");

printf("\v");
printf("Program for setting patterns:\n");
```

```
  printf("The file <pat-new> will be created\n\n");

  printf("Insert the query number (0 to end):\t");
  scanf("%d", &qnum);
  while (qnum != 0)
    {
      for (i = 0; i < MAXT_Q; i++)
        putc(repq[qnum - 1][i], p);
      putc(' ', p);
      printf("Insert the document number:\t");
      scanf("%d", &docnum);
      for (i = 0; i < MAXT_D; i++)
        putc(repd[docnum - 1][i], p);
      putc('\n', p);
      printf("Insert the query number (0 to end):\t");
      scanf("%d", &qnum);
    }
  fclose(p);
}
```

ir.c

This C program evaluates the similarity between a query, or the adapted query produced by the NN simulator, and the documents of the entire document collection which are represented in a matrix. The output is a list of all the documents whose similarity value is over a certain threshold. The list can then be sorted to have the documents ranked according to their similarity to the query and, therefore, to thei expected relevance.

```
/*************************

retrieval program

*************************/

#include<stdio.h>
#include<stdlib.h>

#define N_DOC 200
#define D_TERMS 1142
#define THRESHOLD 0.35     /* usually 0.35 */
#define D_THRES 0.3        /* threshold value for showing documents (0.3)*/
```

```
#define N_RESP 10          /* number of documents to be shown at the end */

main(argc, argv)           /* the argument is the output file of a pattern  */
     int argc;
     char *argv[];


{

  char doc_rep[N_DOC][D_TERMS]; /* representation of all docs */
  char response[D_TERMS];       /* response of the NN at the sel. query */
  float doc_resp_sim[N_DOC];    /* vector of similarities response-docs*/
  float output[D_TERMS];        /* from PlaNet's file nn-ww */


  printf("\n\n");
  printf("EVALUATION PROGRAM FOR ADAPTIVE INFORMATION RETRIEVAL\n");
  printf("Dice's Coefficient of similarity\n");
  printf("\n\n");
  printf("Evaluation of output file: %s\n\n", argv[1]);

  load_documents_representation(doc_rep);
  load_output(argv[1], output);
  threshold(output, response);

  dice_coefficient(response, doc_rep, doc_resp_sim);
  show_doc(doc_resp_sim);

}


/*************************************************************/
/****** read in document representations ********************/
/*************************************************************/

load_documents_representation(doc_rep)
     char doc_rep[N_DOC][D_TERMS]; /* return the doc_rep matrix */

{
  FILE *f;
  int i, j;
```

```
  if ((f = fopen("data/doc-rep-matrix", "r")) == NULL)
    {
      printf("Document representation matrix not found\n");
      exit(-1);
    }

  /* notice that doc 1 will have index 0 */

  for (i = 0; i < N_DOC; i++)
    {
      for (j = 0 ; j < D_TERMS; j++)
        doc_rep[i][j] = getc(f);
      getc(f);
    }
  fclose(f);
  return;
}


/*********************************************************/
/* reading the Output of the NN ************************/
/*********************************************************/

load_output(arg, output)
     char arg[15];
     float output[D_TERMS];

{
  FILE *f;
  int i, j;


  if ((f = fopen(arg, "r")) == NULL)
    {
      printf("Output vector not found\n");
      exit(-1);
    }

  for (i = 0; i < D_TERMS; i++)
    fscanf(f, "%f", &output[i]);

  fclose(f);
  return;
```

```
}


/*************************************************************/
/* apply threshold on the response ***************************/
/*************************************************************/

threshold(output, resp)
     float output[D_TERMS];
     char resp[D_TERMS];
{
  int i;

  for (i = 0; i < D_TERMS; i++)
    if (output[i] < THRESHOLD)
      resp[i] = '0';
    else
      resp[i] = 'a';        /* to allow a better comparison with doc_rep */
 return;
}


/*************************************************************************/
/* evaluation of the similarity between response and each document       */
/*              Dice coefficient                                         */
/*************************************************************************/

dice_coefficient(resp, doc_rep, d_r_sim)
     char resp[D_TERMS], doc_rep[N_DOC][D_TERMS];
     float d_r_sim[N_DOC];
{
  int i, j;
  float cont, cont_d, cont_q;

  cont_q = 0;
  for (i = 0; i < D_TERMS; i++)
    if (resp[i] == 'a')
      cont_q++;

  for (i = 0; i < N_DOC; i++)
    {
      cont = 0;
```

```
      cont_d = 0;
      for (j = 0; j < D_TERMS; j++)
        {
          if (doc_rep[i][j] == 'a' && resp[j] == 'a')
            cont++;
          if (doc_rep[i][j] == 'a')
            cont_d++;
        }
      d_r_sim[i] = 2.0 * cont/(cont_d + cont_q);
    }

  return;
}


/*****************************************************************/
/* show the documents in order of their evaluated relevance *****/
/*****************************************************************/

show_doc(d_r_sim)
     float d_r_sim[N_DOC];

{
  int i, j, cont, temp;
  float ord_resp[N_DOC][2], max;



  /* show the relevant documents */

  printf("\v");
  printf("The document the system found to be relevant to your query are:\n");
  printf("Document number\t\tRelevance assessment\n");


  for (i = 0; i < N_DOC; i++)
   if (d_r_sim[i] > D_THRES)
      printf("%d\t\t\t %f\n", i, d_r_sim[i]);

  return;
}
```

evaluation

The following is the shell script used to produce evaluation files.

```
#! /bin/csh

date > eval35.2on3

foreach file ( outs/out* )
        ir2-35 $file | sort -r >> eval35.2on3
end
```

A.5 Document processor

In this section the parts which compose the Document processor module of the simulation system are described.

repr-d.c

The following is the C program used to produce a matrix representation of the entire document collection. The formalism used is the vector space one.

```
/* representation building program for 200 document collection:

rows are documents and columns are terms

the fact that an "a" is positioned on a certain line-column states
that in the document identified with the number of that line the
terms identified by the number of the column is used to represent
the informative content */

#include<stdio.h>
#include<stdlib.h>

#define MAXTERMS (1142+1)       /* because the index 0 is never used */
#define MAXDOCS (200+1)         /* same reason as above */
#define IDENT 'a'               /* presence of term */
#define NOTHING '0'             /* absence of term */

main(argc, argv)
     int argc;
     char *argv[];
```

```
{
  FILE *f;
  char rep[MAXDOCS][MAXTERMS];
  int docnum, ternum, i, j;


  if (argc != 2)
    {
      printf("Usage: repr <file-name>\n");
      exit(-1);
    }

  if ((f = fopen(argv[1], "r")) == NULL)
    {
      printf("File %s not found\n", argv[1]);
      exit(-1);
    }

  while (fscanf(f, "%d", &docnum) != EOF)
    {
      fscanf(f, "%d", &ternum);
      while (ternum != 0)
        {
          rep[docnum][ternum] = IDENT;
          fscanf(f, "%d", &ternum);
        }
    }
  fclose(f);

  for (i = 1; i < MAXDOCS; i++)
    {
      for (j = 1; j < MAXTERMS; j++)
        if (rep[i][j] == IDENT)
          putchar(IDENT);
        else
          putchar(NOTHING);
      printf("\n");
    }

}
```

Bibliography

[1] C.J. van Rijsbergen. *Information Retrieval.* Second edition, Butterworths, London, 1979.

[2] H.P. Luhn. A statistical approach to mechanized encoding and searching of library Information. *IBM Journal of Research and Development*, 1:309:317, 1957.

[3] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[4] G. Salton and M.J. McGill. *Introduction to modern Information Retrieval.* McGraw-Hill, New York, 1983.

[5] G. Salton. *Automatic Text Processing.* Addison-Wesley, 1989.

[6] W.B. Croft. Approaches to Intelligent Information Retrieval. *Information Processing & Management*, 23(4):249:254, 1987.

[7] R.H. Thompson. The design and implementation of an intelligent interface for Information Retrieval. Technical report, Computer and Information Science Department, University of Massachusetts, 1989.

[8] W.B. Croft and R.H. Thompson. I^3R: a new approach to the design of document Retrieval systems. *Journal of the American Society for Information Science*, 38(6):389–404, 1987.

[9] H.M. Brooks. Expert systems and Intelligent Information Retrieval. *Information Processing & Management*, 23(4):367–382, 1987.

[10] E.A. Fox. European Summer Schcol in Information Retrieval. Presentation, Bressanone. Italy, July 1990.

[11] R.M. Tong, L.A. Appelbaum, V.N. Askman, and J.F. Cunningham. Conceptual Information Retrieval using RUBRIC. In *Proceedings of ACM SIGIR*, New Orleans, Luisiana, June 1987.

[12] R.M. Tong, L.A. Appelbaum, and V.N. Askman. A knowledge representation for conceptual Information Retrieval. *International Journal of Intelligent Systems*, 4(3):259–284, 1989.

[13] P.R. Cohen and R. Kjeldsen. Information Retrieval by constrained spreading activation on Sematic Networks. *Information Processing & Management*, 23(4):255–268, 1987.

[14] L.F. Rau. Knowledge organization and access in a conceptual information system. *Information Processing & Management*, 23(4):269–283, 1987.

[15] E.A. Fox. A design for intelligent Retrieval: the CODER system. In *Proceeding of the Second Conference on Computer Interfaces and Intermediaries for Information Retrieval*, Boston, Massachusetts, USA, 1986.

[16] E.A. Fox. Development of the CODER system: a testbed for artificial intelligence methods in Information Retrieval. *Information Processing & Management*, 23(4):341–366, 1987.

[17] M.T. Weaver, R.K. France, Q.F. Chen, and E.A. Fox. Using a frame based language for Information Retrieval. *International Journal of Intelligent Systems*, 4(3):223–258, 1989.

[18] U. Hahn. Making understanders out of parsers: semantically driven parsing as a key concept for realistic text understanding. *International Journal of Intelligent Systems*, 4(3):345–394, 1989.

[19] P.F. Patel-Schneider, R.J. Brachman, and H.J. Levesque. Argon: knowledge represenation meets Information Retrieval. In *Proceedings of The First Conference on Artificial Intelligence Applications*, Boston, CO, USA, 1984. IEEE.

[20] M.K. Di Benigno, G.R. Cross, and C.G. de Bessonet. Corel. a conceptual retrieval system. In *Proceedings of ACM SIGIR*, Pisa, Italy, September 1986.

[21] W.B. Croft, T.J. Lucia, and P.R. Cohen. Retrieving documents by plausible inference: a preliminary study. In *Proceedings of ACM SIGIR*, Grenoble, France, June 1988.

[22] W.B. Croft, T.J. Lucia, J. Crigean, and P. Willet. Retrieving documents by plausible inference: an experimental study. *Information Processing & Management*, 25(6):599–614, 1989.

[23] R.A. Frost. *Introduction to Knowledge Based Systems.* Collins, 1987.

[24] F. Rosemblatt. *Principles of Neurodynamics.* Spartan Books, New York, 1962.

[25] R.K. Belew. *Adaptive Information Retrieval: machine learning in associative networks.* Phd thesis, University of Michigan, USA, 1986.

[26] W.S. Mac Cullough and W.H. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[27] A. Newell. The knowledge level. *AI Magazine*, pages 1–20, 1981.

[28] M. Agosti, F. Crestani, and G. Gradenigo. Towards data modelling in Information Retrieval. *Journal of Information Science*, 25(6):307–319, 1989.

[29] M .Agosti, F. Crestani, G. Gradenigo, and P. Mattiello. An approach to conceptual modelling of ir auxiliary data. In *Proceedings of IEEE International Conference on Computer and Comunications*, Scottsdale, Arizona, USA, 1990.

[30] W.B. Croft and T.J. Parenty. A comparison of a network structure and a database system used for document retrieval. *Information Systems*, 10(4):377–390, 1985.

[31] H. Turtle and W.B. Croft. Inference networks for document Retrieval. In *Proceedings of ACM SIGIR*, Brussels, Belgium, September 1990.

[32] M.E.S. Loomis, A.V. Shah, and J.E. Rumbaugh. An object modelling technique for conceptual design. In *Proceedings of the European Conference on Object-Oriented Programming*, pages 325–335, Paris, 1987. AFCET.

[33] V. Bush. As we may think. *Atlantic Monthly*, 176(1):101–108, 1945.

[34] S.E. Preece. *A spreading activation model for Information Retrieval.* Phd thesis, University of Illinois, Urbana-Champaign, USA, 1981.

[35] G. Salton and C. Buckley. On the use of spreading activation methods in automatic Information Retrieval. In *Proceedings of ACM SIGIR*, Grenoble, France, June 1988.

[36] D.E. Rumelhart, J.L. McClelland, and PDP Research Group. *Parallel Distribut Processing: exploration in the microstructure of cognition.* MIT Press, Cambridge, 1986.

[37] K.L. Kwok. A Neural Network for probabilistic Information Retrieval. In *Proceedings of ACM SIGIR*, Cambridge, MA, USA, June 1989.

[38] G. Salton. *Automatic information organization and retrieval.* Mc Graw Hill, New York, 1968.

[39] D.E. Rumelhart and D.A. Norman. Representation in memory. Technical report, Department of Psychology and Institute of Cognitive Science, UCSD La Jolla, USA, 1983.

[40] H. Kimoto and T. Iwadera. Construction of a dynamic thesaurus and its use for Associative Information Retrieval. In *Proceedings of ACM SIGIR*, Brussels, Belgium, September 1990.

[41] D. Hebb. *Organization of behavior.* John Wiley & Son, New York, 1949.

[42] P.K. Simpson. *Artificial Neural Systems: foundations, paradigms, applications and implementation.* Pergamon Press, New York, 1990.

[43] M.C. Mozer. Inductive Information Retrieval using parallel distributed computation. Technical report, Institute for Cognitive Science, University of California, San Diego, June 1984.

[44] J. Bein and P. Smolensky. Application of the interactive activation model to document retrieval. Technical report, Dept. of Computer Science, University of Colorado, Boulder, 1988.

[45] R.J. Brachman and D.L. McGuinness. Knowledge representation, connectionism, and conceptual Retrieval. In *Proceedings of ACM SIGIR*, Grenoble, France, June 1988.

[46] R.K. Belew. Adaptive Information Retrieval: using a connectionist representation to retrieve and learn about documents. In *Proceedings of ACM SIGIR*, Cambridge, USA, June 1989.

[47] K.L. Kwok. Application of Neural Network to Information Retrieval. In *Proceedings of the International Joint Conference on Neural Networks*, Washington D.C., USA, January 1990.

[48] G.S. Jung and V.V. Raghavan. Connectionist learning in costructing thesaurus-like knowledge structure. AAAI Spring Symposium on Text-Based Intelligent Systems, Working Notes, March 1990.

[49] P. Hingston and R. Wilkinson. Document retrieval using a Neural Network. Technical report, Dept. of Computer Science, Royal Melbourne Institute of Technology, 1990.

[50] G. Salton. *The SMART Retrieval System. Experiments in automatic document processing.* Prentice-Hall, New Jersey, 1971.

[51] C. Claverdon, J. Mills, and M. Keen. *ASLIB Cranfield Research Project: factors determining the performance of indexing systems.* ASLIB, 1966.

[52] F. Crestani and Ø. Olsen. Introduction to Neural Network simulation. Technical report, Department of Computing Science, University of Glasgow, UK, December 1991.

[53] Y. Miyata. *A user's guide to PlaNet version 5.6: a tool for constructing, running and looking into a PDP network.* Computer Science Department,, University of Colorado, Boulder, USA, December 1990.

[54] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the theory of Neural Computation.* Addison-Wesley, New York, 1991.

[55] R. Beale and T. Jackson. *Neural Computing: an introduction.* Adam Higler, Bristol, UK, 1990.