2003

Formal concept matching and reinforcement learning in adaptive information retrieval

Rajapakse, Rohana Kithsiri

Formal Concept Matching and Reinforcement Learning in Adaptive Information Retrieval

by

Rohana Kithsiri Rajapakse

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Computing, Communications and Electronics
Faculty of Technology

November 2003

Formal Concept Matching and Reinforcement Learning in Adaptive Information Retrieval

by

Rohana Kithsiri Rajapakse

ABSTRACT

The superiority of the human brain in information retrieval (IR) tasks seems to come firstly from its ability to read and understand the concepts, ideas or meanings central to documents, in order to reason out the usefulness of documents to information needs, and secondly from its ability to learn from experience and be adaptive to the environment. In this work we attempt to incorporate these properties into the development of an IR model to improve document retrieval. We investigate the applicability of concept lattices, which are based on the theory of Formal Concept Analysis (FCA), to the representation of documents. This allows the use of more elegant representation units, as opposed to keywords, in order to better capture concepts/ideas expressed in natural language text. We also investigate the use of a reinforcement learning strategy to learn and improve document representations, based on the information present in query statements and user relevance feedback. Features or concepts of each document/query, formulated using FCA, are weighted separately with respect to the documents they are in, and organised into separate concept lattices according to a subsumption relation. Furthermore, each concept lattice is encoded in a two-layer neural network structure known as a Bidirectional Associative Memory (BAM), for efficient manipulation of the concepts in the lattice representation. This avoids implementation drawbacks faced by other FCA-based approaches. Retrieval of a document for an information need is based on concept matching between concept lattice representations of a document and a query. The learning strategy works by making the similarity of relevant documents stronger and non-relevant documents weaker for each query, depending on the relevance judgements of the users on retrieved documents. Our approach is radically different to existing FCA-based approaches in the following respects: concept formulation; weight assignment to object-attribute pairs; the representation of each document in a separate concept lattice; and encoding concept lattices in BAM structures. Furthermore, in contrast to the traditional relevance feedback mechanism, our learning strategy makes use of relevance feedback information to enhance document representations, thus making the document representations dynamic and adaptive to the user interactions. The results obtained on the CISI, CACM and ASLIB Cranfield collections are presented and compared with published results. In particular, the performance of the system is shown to improve significantly as the system learns from experience.

List of Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

First of all, I would like to thank my director of studies Prof. Michael Denham for his great support, encouragement and guidance throughout the period of this research. In particular, I thank him for encouraging me to study intelligent information retrieval, directing me to the interesting approach that this thesis takes, and for his invaluable contributions to make this a success.

My sincere thanks also go to the University of Plymouth, in particular to the School of Computing, for offering me financial assistance through a studentship grant to realise my studies. In this regard, Dr. Sue Denham deserves special thank for encouraging me to take this oppertunity. This gave me a great opportunity to work with excellent people at the Centre for Neural and Adaptive Systems. In particular, I would like to mention the names of Prof. Roman Borisyuk, Dr. Guido Bugmann and Dr. Angelo Cangelosi for their willing support and fruitful research discussions. My appreciation also goes to the University of Colombo, Sri Lanka for granting me leave from the lecturer position I hold at its School of Computing, to pursue my higher studies abroad.

I would also like to extend my grateful acknowledgments to prominent experts in IR with whom I had valuable verbal and email conversations. Prof. C. J. van Rijsbergen, Prof. Fabio Crestani, Dr. Claudio Carpineto and Dr. David Grossman (of IIT) are a few to mention. Moreover, thanks to other IR related researchers not named here, but who supported me in numerous ways, in particular for locating valuable resources such as research papers that were out of my reach. Also, those publishers and authors who granted permission to reproduce figures from their publications are acknowledged.

My sincere appreciation also goes to the organising committees and sponsors of the ESSIR'2000 (Fabio Crestani, Gabriella Pasi and Maristella Agosti), ECIR'02 (van Rijsbergen, Fabio Crestani and Mark Girolami) and UM'03 (Peter Brusilovsky) for supporting me with student grants to attend the summer school, the conference and the workshop, respectively.

Finally, my parents and my family deserve special appreciation for the encouragement, understanding and tolerance they showed during this period of study.

DECLARATION

At no time during the registration for the degree of doctor of Philosophy has the author been registered for any other University award.

This study was financed with the aid of a studentship from the School of Computing, University of Plymouth, UK.

The author had undertaken two advanced masters study programmes prior to this study, one in Computer Science at the University of Colombo, Sri Lanka and the other in Computational Intelligence at the University of Plymouth, UK. Also, he served as an instructor for two advanced course modules on Neural Computation (undergraduate and postgraduate level), and conducted a series of seminars for another undergraduate course module on Information Retrieval at the University of Plymouth.

Relevant workshops and conferences were attended during this study and the work was presented at those workshops and conferences. A list of Conferences/Workshops/Summer Schools attended is given below, and the publications made on this work are listed at the end of the thesis on page 279.

1. The 24th BCS-IRSG European Colloquium on IR Research (ECIR'02), March 2002 Glasgow, UK.
2. The 15th European Conference on Artificial Intelligence (ECAI'02), and its associated FCA KDD workshop, July, 21-26 2002, Lyon, France.
3. The Third European Summer School in Information Retrieval (ESSIR2000), September 11-15, 2000, Varenna Italy.

Signed .....

Date 24th September 2004

CHAPTER 1 - INTRODUCTION

1.1 WHY IR ?

"Information" plays an inevitably vital role in today's information society. The invention of electronic media for storing huge amounts of information in tiny electronic media and the invention of the computer for processing enormous amounts of information stored in such electronic media are the major contributors for today's electronic information society. The amount of information being handled by various organisations in today's world is huge and management of them without a computer is unimaginable. These facts have raised major challenges for Information Management tasks such as efficient storage of information for easy access, efficient transmission of information and efficient retrieval of information.

Seeking for desired information for individual needs has ever been a challenging task for mankind. The challenge has never been fully met even with the latest developments and inventions of electronic and communication technologies that support storage of vast amounts of information, data communication and faster information processing. Instead aspects of the problem seem to have slowly transferred from the unavailability of information (sources) for access to the difficulty of extracting desired information from available sources. The latest advancements of electronic and communication technologies has only solved, in part if not fully, the primary problem faced in the past "*the unavailability of information for access*". Today we are privileged to live in a world rich of information in which most, if not all, information we need is available at our fingertips and ready to be used. The consequences of this information flood have led to the return of irrelevant, often distracting, data in response to our information requests. The user is confused with where to begin his search, when to end, whether he has got the correct and latest information during his search and also what he has got is all that is available or

whether there is more useful information. In other words, the difficulty of accessing desired information has grown with the growth of the availability of information and as a result today we are suffering from lack of sufficient tools to help maximum use of available information [Amati & Crestani 1999, Lagus et al. 1996, Fuhr & Buckley 1991].

Historically, the growth of textual material such as books and articles in libraries along the centuries has demanded efficient mechanisms to locate and refer to them. The early techniques such as abstracting, indexing and use of subject classifications have marked the birth of the "Information Retrieval" research discipline. There have been tremendous efforts ever since, as evident in the literature, for developing ways and means to find out desired information effectively and efficiently from large collections of textual material.

"*History of IR is long and fraught*", van Rijsbergen 1979.

The latest advancements of electronic media for document storage and communication have made the task of searching for information more challenging and demanding than ever before. Even though the continual efforts of IR researchers have endowed the field with a rich set of sophisticated tools, the sophistication of the tools for creation and transmission of information far outstrips the sophistication of tools for automating and managing information.

1.2 Core processes of Information Retrieval

The core processes of searching for information are well described by R.K. Belew in his book "*FOA- Finding Out About*" [Belew 2000]. He summarised the entire process under three key processes: "Asking a Question", "Constructing an Answer" and "Assessing the Answer". The first of these has much to do with human cognition in which he defines an information need as a desire to fill a gap in the user's knowledge. Forming a clearly posed question corresponding to an information need that arises in the user's mind is known to be the hardest part of answering it. The state of mind of the user may be such that he either knows exactly what he needs or he only has a vague thought about what he needs. A user

may or may not be able to define fully the characteristics of the "answer" (the information need) he seeks even if he knows without any doubt what information he needs. This ill-defined internal cognitive state is then turned into an external expression in some language and is termed as the "Query".

The job of "Constructing an Answer" is the responsibility of the answerer or in case of a computer the Information Retrieval System (IRS). The problems inherent to machines make this task much more difficult to an IRS than to a human answerer. Some of these inherent problems include the lack of intelligence for understanding the problem and searching for solutions, lack of background knowledge to understand the problem and also to provide the answer with adequate detail for the user to understand it and lack of intelligence to cope with ambiguities of natural language text. A detailed discussion of these problems and issues of IR is given in Chapter 2.

The last phase "Assessing the answer" involves the user assessing the answer(s) (in his mind) to decide how relevant they are for his information need. The process may end here ideally if the user is fully satisfied with the answer or else the user assessment(s) can lead to re-thinking and re-defining of the information need by the user himself, a process that occurs outside of the IRS. Within the IRS, user assessments can be used as user feedback to re-formulate the query or/and to initiate a learning process. The IR researcher has little control over such user-oriented issues that are external to the IRS. Nevertheless it is essential that he is aware of the external issues as he is expected and is responsible for designing and creating sensitive and flexible IR systems that are tolerant to those external issues.

1.3 TEXT RETRIEVAL

The concept "Information Retrieval", as described above (by Belew) applies to seeking a broad spectrum of information bearers such as books, reports, letters, images, drawings,

movies and sounds from any form of an information source (collection). Salton and McGill [Salton & McGill 1983] defined the concept of "Information Retrieval" as: "*one concerned with the representation, storage, organisation, and accessing of information items*". There is not much difference between these definitions as far as the major components are concerned. In practice, these definitions of IR have almost been synonyms for keyword-based querying of textual databases.

Despite the presence of sources with different forms of information expression or presentation formats, retrieval of which is covered under this same definition(s), the main emphasis in the field over the years has been on text retrieval. This is mainly due to the fact that majority of digital information resources contain more textual information than anything else. Due to the success of the Internet and its services such as the WWW, email, News groups, Bulletin boards etc., and the widespread PC users who create and access these resources, a large portion of the information being made available for use today is in the form of text. Interestingly, researchers in IR have long recognised the importance of text, and have focused primarily on development of techniques for representing and retrieving text documents. The high popularity and high emphasis given to the retrieval of textual material have made it distinctly recognised as "Text Retrieval" or "Document Retrieval" within IR community. A Text Retrieval system is distinguished from a traditional information storage and retrieval system (ISAR) mainly by the unstructured nature of data items (textual material) to be searched and the unstructured nature of query statements. A text retrieval system is required to have a component for acquisition of the needs of specific users approaching the system with unstructured and imprecise query statements [Tyrväinen 1984, pp13]. Only a ranking list of documents that appear to contain some relevant information is produced by a text retrieval system as opposed to an exact answer produced by an ISAR for a given information need.

A primary problem of any automated application that deals with textual inputs is processing text and creating a useful form of representation of the content to support machine understanding of the content. Enabling computers to understand text documents by their content allows the automation of information management tasks such as document retrieval, document routing, web information discovery, story understanding and email sorting etc. This problem, which involves substantial text analysis effort, is an unsolved problem [Kohle & Merkl 1996] in Natural Language Understanding (NLU) research. However, it is generally conceded that many of the tasks do not need complete understanding of text in a natural language sense, but it is only necessary that text be understood well enough to be correctly manipulated.

> *"... complete understanding of the text may not be necessary for IR ... it may suffice to have a shallow and partial representation of the content of documents"*
>
> [Evans & Zhai 1996]

For instance, majority of the successful Text Retrieval models only manage with meanings at keyword (individual terms or phrases) level rather than meanings at the levels of sentence, paragraph or entire document. The common practice of document retrieval is to create document surrogates by extracting useful information (Knowledge Acquisition) from text items (documents) and representing them in a form that supports the underlying retrieval mechanism. Thus, it is the combination of Knowledge Acquisition, Knowledge Representation (KR) and Retrieval Mechanism (operating on the underlying KR) that makes up an IR system. From this perspective, the problem of text retrieval can be seen as a problem of defining the relations of the representations and the central concepts of reasoning out "aboutness" of text to information needs.

However, despite the success of keyword-based representations, such as term frequency-inverse document frequency (*tf-idf*) based representations, the increased volumes of information items, diversity of writing styles, usage of different vocabularies, ambiguity of natural languages and various other parameters such as length differences

[Singhal et al. 1996] of individual text documents etc have made it increasingly difficult to reach higher accuracy levels using poorly represented and thus poorly understood text material.

Currently, computers have little ability to manipulate text, based on content and are limited instead to manipulations based on format tags or other explicit labels such as file names, file formats, keyword labels, web meta-tags or mark-up codes and email subject lines. The small number of labels they typically use and the difficulty of creating such labels, either manually or automatically, impose limitations on these approaches. Moreover, standards of using such labels are not well defined and even the defined standards are not strictly followed. Hence, they do not provide a comfortable platform for a generic retrieval system to base upon. On the other hand, systems that can read and understand natural language text in unconstrained contexts do not exist. Even in restricted narrow domains, written documents are difficult due to the complexity of natural language.

Nevertheless, there have been many attempts to use more comprehensive meanings (concepts) instead of simple keywords, and a variety of techniques and theories have been tried to deal with the problems associated with extracting, representing and learning correct meanings of textual material. Some of these techniques such as Semantic Networks are far from manageable for real world applications. Despite these efforts, text-based information retrieval remains challenging owing to the unstructured nature of natural language text, the subtle nature of conveying meaning by sentences, the ambiguity of understanding meaning of words and phrases, the large problem space, and implicit contextual information. These challenges have directed the IR researcher to structure the information carefully into knowledge structures (e.g. ontologies) with manageable level of complexity and use this knowledge for IR. Without representing the natural ontological knowledge in a machine understandable way, computers have little chance of understanding correctly what is stated explicitly and implied or left unsaid.

The reasons outlined above motivated us to investigate into techniques for representation and learning meanings of textual material at a manageable level that can operate effectively in information management tasks (IR in particular).

We have identified the problems of text representation (Chapter 2), reasons for problems and in particular the need for understanding the contents of documents adequately enough to create meaningful document representations in order to improve the IR task. Thus, we paid substantial attention to feature extraction from textual material for the representation of document contents during this research.

1.4 THE TEXT RETRIEVAL EXPERIMENTS: TRADITION AND PRACTICE

Traditionally, text retrieval experiments were conducted on free text documents as opposed to structured /formatted databases of records. A typical text retrieval system goes through a text-processing phase before any text can be retrieved. This includes extraction of text structures, generation of content identifiers (indexing), or classification of a text etc. requiring a substantial text analysis effort. The result of this is an internal representation of documents and queries in the IR system. In case of full text indexing, all the words in the content of a document except noise words (i.e. words that say little about the document's content) are used in the representation.

Two basic approaches used for concept extraction are: (1) Statistical approaches, and (2) Language Analysis approaches. Statistical approaches are based on frequency counts of term or phrase occurrences. Automating such operations as finding occurrences of individual words or phrases is relatively easy compared to language analysis approaches which needs an in-depth investigation into the syntactic structures and semantic meanings of natural language. Different schemes and approaches used in IR for text representation are detailed in Chapter 3.

The representations, thus created at the text processing phase, are compared between a query and a document and a Retrieval Status Value (RSV) is computed. Based on RSV values, documents are ranked according to a Ranking Principle and the ones in the top beyond a pre-decided threshold value are returned to the user as retrieved documents. The retrieval process may not end here. As described above, it is very difficult and impossible in most cases to transform the information needs that occur in our brains into language symbols, to describe them using the particular query language used by the IR system. Therefore, it is very unlikely for the initial query formulation to retrieve only desirable documents. This is where the relevance feedback techniques have been useful. In relevance feedback, the user decisions are used for reformulating the query or enhancing the query representations (see [Salton & McGill 1983] for more detail of relevance feedback). The user has the opportunity to re-think his information need and/or reformulate the query by himself, or in case of systems with automatic query reformulation by relevance feedback, he can simply give his feedback by indicating which are useful to him and which are not (binary feedback) or by ranking each document according to its usefulness in a pre-decided scale of values. The first case involves a cognitive process in the user's brain in which he has to re-think what exactly his information need is and how he can reformulate his query accordingly. Results of the last retrieval give guidance as to whether he is in the correct direction and provide appropriate terminology to be used. In the second case, it is the IRS that reformulates the query based on the user feedback.

Figure 1.1 shows a schematic diagram of the traditional IR setup. The three processes shown in the figure, namely the Representation, Comparison and Evaluation model the essence of the three core processes of IR mentioned above (described by Belew (2000)). The effectiveness of an IR system depends on all three processes and the relationship between them.

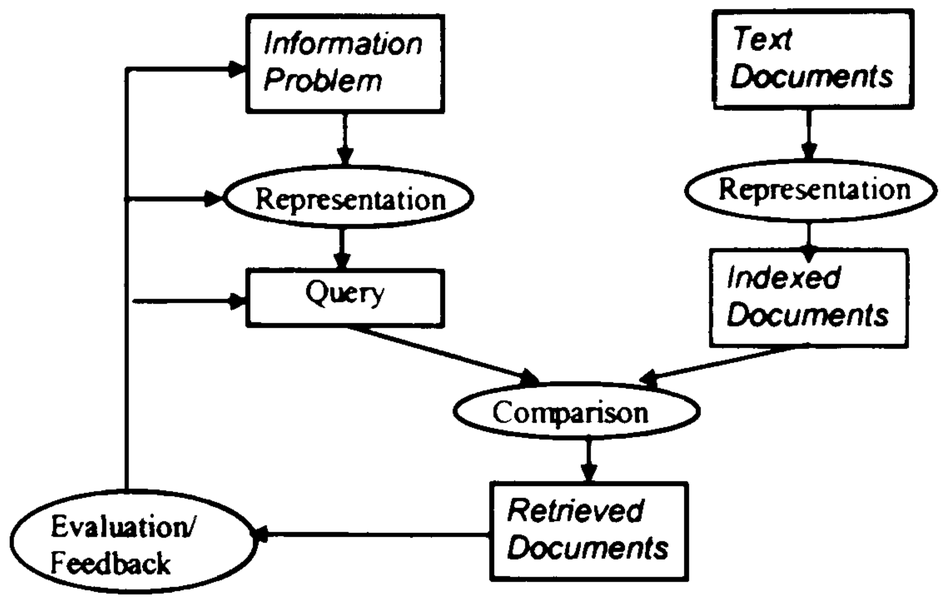


Figure 1.1 : The Conventional IR Processes

Experimental Evaluation

Evaluation of the effectiveness of IR systems is typically conducted experimentally rather than analytically. This is in accord with the in-deterministic nature of the IR problem in which, though the problem is well understood in general terms, it is hard to give a formal specification for the problem. In addition the traditional experimental evaluation is defined and developed for measuring effectiveness rather than efficiency (time and space requirements) of IR systems. This may be partly due to the difficulty of measuring efficiency in a machine independent way. Nevertheless, the efficiency requirement remains important especially in evaluating interactive IR systems, and therefore it is desirable that it be measured in conjunction with effectiveness.

The traditional evaluation model is based on the notion of *Relevance*, a notion of how useful a retrieved item to the user, and its measurement by two figures, *Recall* and *Precision*. Precision calculates the proportion of retrieved documents that are relevant and recall calculates the proportion of relevant texts that are retrieved. In an experimental setting in which the sets of queries and documents are known and the relevance judgements are defined for each query-document pair, an aggregated measure of relevance is reported by (usually non-interpolated) average precision and recall. Precision-Recall curves (P-R curves) are also commonly used to compare the performances of IR systems. However, comparison between two IR systems can only be made when evaluated on the same document collection using the same performance measures under the same

operational conditions. For further detail, reader is referred to the Crainfied model [Cleverdon et al. 1966, Cleverdon 1991], the best exemplar around for experimental evaluation of IR.

The appropriateness of recall and precision for the measurement of relevance has been questioned and criticised by a number of researchers [Schamber et al. 1990] mainly for its ignorance of the user involvement. Nevertheless, precision-recall based evaluation model has been widely accepted by a large majority of IR researchers and has been the primary evaluation model in IR research, in particular, due to its simplicity and practicality.

1.5 Research In IR

Research in the IR field is varied. Each stage of the IR process leads to one or more sub-disciplines of research. Content extraction, indexing/representation, comparison/matching, evaluation and query re-formulation/relevance feedback are a few examples. Van Rijsbergen [Rijsbergen 1979] subdivided IR research into: (1) Content Analysis, (2) Information Structures, and (3) Evaluation. Content Analysis is concerned with describing the contents of documents in a form suitable for computer processing (Representation). Information Structures are concerned with exploiting relationships between documents in the view of improving effectiveness of retrieval and Evaluation is concerned with the measurement of the effectiveness of retrieval. We can choose one or more of these sub-disciplines to investigate and make use of existing well-established theories, techniques and mechanisms for other components. For instance, one can make use of the Vector Space Model and work on a Query Reformulation strategy. In our case, however, since we start from scratch by defining the information unit (concept) rather differently, we cannot re-use any existing implementations of IR components. Instead we have developed our own method of content extraction to support the representation mechanism we have chosen, our own weighting scheme for concept weighting, our own strategy to compare documents

with queries for computing RSV values and our own learning strategy based on user feedback for learning document representations.

1.6 Research Direction (Aims And Objectives)

In this work, we restricted ourselves to the domain of text. The underlined primary hypothesis of our research is to investigate whether a more comprehensive meaning/concept matching would help in improving Text Retrieval. We took an experimental approach in which we investigated a concept lattice based document representation scheme that is capable of representing the content of a document in terms of formal concepts. For concept matching between queries and documents, a matching strategy was developed utilising the relationships between concepts in hierarchical representations of concepts in concept lattices.

We define a concept, a meaning or a thought according to the theory of FCA as having two sets - a set of *objects* and a set of *properties* common to all the objects in the first set. Obviously, such a structure (a formal concept) carries a more comprehensive meaning than individual keywords or key phrases, due to its use of two related components. In addition, the chances of misinterpretation of such a formal concept is less, as it involves more than one term/word and a relationship between them, i.e. some of them are interpreted as objects and others as attributes or properties of objects. We organise the concepts extracted from each information item (text document) into a separate conceptual lattice structure according to the subsumption order relation defined in FCA. Query concepts are also structured in the same manner. Our main goal is to match Query and Document concepts between such conceptual structures and rank the documents according to the degree of similarity between them. More precisely, we are looking for the presence of query concepts in the document concept structures and the similarity of a document to a query is computed based on the common or matching concepts/features.

A secondary objective of our work is to investigate the impact of continuous learning of concepts in document representations, based on retrieval performance. Due to the inherent ambiguities of the meanings of words in natural language and the difficulty of extracting relations between words, it is always difficult to create complete representations to carry the correct and complete meaning of the overall content of a document [Croft & Turtle 1992] in one pass, as normally done by the indexing processes of conventional IR systems. Therefore, such a static representation of text is unlikely to help perfect retrieval. Instead, an adaptive representation that is continuously updated or that learns through experience is desirable. We use a learning strategy based on "re-inforcement learning" to make documents learn their representations as they are retrieved for user queries. The goal of our learning strategy is two fold, (1) to learn a sort of "complete" document representation by including any concepts that would have been there, yet have not been extracted and included at the time of first creating the representation, (2) to tune the significance weights of concepts in document representations in such a way that the concepts that are likely to help retrieval of desired documents to the users have higher significance values, and those concepts that are common to many documents and hence do not help identifying desired documents distinctly have lower values.

An operational simulation of the model was implemented and tested on three public domain document collections according to the well-established traditions and practices of IR research. The document collections used for evaluation were CISI, CACM and Cranfield (available to download at *http://ir.dcs.gla.ac.uk/resources/test collections/* and *ftp://ftp.cs.cornell.edu/pub/smart*). Details of these collections are given in Table 9.1 (Chapter 9). Precision averages at different retrieval points were computed, and compared against results published by Carpineto and Romano [Carpineto & Romano 2000]. Also, average precisions and P-R curves were used to demonstrate the effectiveness of the

learning strategy over time, as the system gains more experience as it comes across more queries. Effectiveness of the two components of the learning strategy - weight learning and concept addition- were also examined over training iterations. The improvements shown by the system as it learns were quite impressive. Also, the system shows near perfect results for the case of known or already seen queries. One of the requirements for effective learning (by our strategy) is to have sufficient overlaps of queries and documents judged as relevant to those queries. The results degrade to the level of keyword-based models or even below when this requirement is not met by the document collection. Also, sufficiently long natural language query expressions, rather than of short keyword type queries are desirable to help creating sufficiently informative representations. These requirements are further described in Section 9.1.1. Their impact on performance and possible directions for improvements are discussed further in Chapter 10.

1.7 Our Contribution and Novelty

The primary objective of our research is to investigate for a possible improvement of the effectiveness of IR by utilising more comprehensive and informative units/concepts than simple keywords, for representation and comparison between queries and documents. The comparison unit we used to achieve this objective is the simplest unit of a formal concept: an *object- attribute* pair (Section 7.1.1). Each of the object-attribute pairs is assigned a weight with respect to the document it appears in (Section 7.2.1.1), allowing the same object-attribute pair to have different significances in different documents. This is a completely different approach to that of existing IR models. The novelty here is two fold:

1. the use of object-attribute pairs (*unit-concepts*) instead of keywords or keyphrases, and
2. the assignment of significances for such pairs with respect to the documents they appear.

Although attempts have been made in the past by some researchers to employ FCA in information retrieval, the way a concept is defined in those models is quite different to ours (Section 5.4 and Section 5.5). For instance, objects in those approaches are documents

(document identification numbers) rather than objects (physical or conceptual) that the documents are talking about (in their content).

In addition, our method of encoding concept lattices in Bidirectional Associative Memory (BAM) structures, and also the way we use concepts according to their specificity/generality relationships are new. The capability of a BAM to learn concept lattices, proposed and proved by Radim Bělohlávek [Bělohlávek 2000] has not been used in IR research before. The reason for using BAMs for encoding concept lattices in our approach is mainly to reduce lattice building and node traversing overheads.

Finally, the way we update document representations with concepts, in user-formulated queries, based on relevance judgements is novel (Section 7.2.2 and Section 8.5). There are models that update document representations in terms of weight modifications/estimations through relevance feedback (e.g. Maron & Kuhns's work [Maron & Kuhns 1960]). In contrast to them, what is new in our model is the addition of query concepts that are judged as relevant to the query by the user, to document representations. This idea which adds additional information to the original document representation may lead to a controversy as it leads to modification of the original content of the document. Our objective, however, is to fine-tune the document representation according to what users think the document is about, rather than what the writer originally intended. By this, we implicitly retain the important user decisions and allow the document representations learn from experience. The contribution to RSV values by original concepts and the ones that were added later to the document representation through reinforcement learning can be controlled at the implementation level by differently weighting the two sources of concepts.

The most common approach of using user feedback has been to reformulate the query with adding terms from relevant documents and re-running the enhanced query rather than using it for learning document representations. This approach neither retains the important user

decisions nor learns anything from them for future use. The important decisions of the users are only available for a single query session and are lost thereafter. As a result, the system needs to go through the same query reformulation process each time the same query is encountered. In contrast, by allowing the document representations to learn, we not only make use of the previous user interactions to influence retrieval of documents for the subsequent queries, but also avoid having to repeat the retrieval process for the information needs that the system has encountered before.

1.8 Defined Terms, Abbreviations and Conventions

In this work, the term "text" is used alternatively to refer to any piece of natural language text and also to separately identifiable piece of textual entity, for example a text document or a query description. The terms "information retrieval", "text retrieval" and "document retrieval" are alternatively used as synonyms throughout this report to refer to retrieval of textual documents for given query descriptions in text form. Also, the terms "information need", "user request", "user query" and "query" are alternatively used to refer to what the user wants to find. The terms "concept" and "formal concept" are alternatively used mainly to refer to a formal concept as defined in FCA. We use the convention {<*objects*>}→ {<*attributes*>} to write a formal concept, where there can be any (finite) number of objects and attributes (respectively) in the two components.

Following are the abbreviations used in this text.

Object - a label with one or more terms that refers to a physical or conceptual object found in text (usually a noun)

Attribute - a label with one or more terms that describes a property of an object

Unit-concept - a pair of an object and a related attribute

Keyword - a single-term keyword or a keyphrase with more than one adjoining terms

IR - Information Retrieval

TR - Text Retrieval

KR - Knowledge Representation

RSV - Retrieval Status Value

IRS - Information Retrieval System

NN - Neural Network

NLP - Natural Language Processing

NLU - Natural Language Understanding

FCA - Formal Concept Analysis

BAM - Bidirectional Associative Memory

1.9 OVERVIEW

The dissertation starts by giving a brief overview of IR, Text Retrieval (TR) and the traditional practice of TR in the early part, and the aims and objectives and our contribution/novelty at the later part of this Chapter (Chapter 1). The problem domain, problems, issues and needs of IR are detailed in the next chapter (Chapter 2). One of our major design strategies was to use a more informative representation scheme for IR. Hence it was thought that it would be useful to include a brief review of document characterisation techniques (indexing) in Chapter 3. A review of well-known IR models and approaches follows next in Chapter 4. This includes reviews on classic IR models such as VSM and probabilistic IR, as well as other more recent models and approaches that make use of diverse techniques including connectionist and fuzzy logic based approaches. The main theory on which our research is grounded upon is the Formal Concepts Analysis (FCA). Chapter 5 is devoted to presenting FCA theory and its analogy to certain properties of the way the human brain might structure information. Chapter 6 presents the specific NN architecture called the BAM and details of how it can be used to learn concept lattices. In Chapter 7 we describe the design of our model at a conceptual level, and implementation detail follows in Chapter 8. Experiments conducted and the results of these experiments are reported in Chapter 9 to give a formal evaluation of the model. Finally, Chapter 10 concludes the thesis with final conclusions and proposals for future work.

CHAPTER 2 - THE PROBLEM DOMAIN, PROBLEMS, ISSUES AND NEEDS

The problem domain of text retrieval is essentially natural language text. The richness of natural language causes many text retrieval problems. In addition to natural language oriented problems, each individual approach made to solve the IR problem has its own set of problems, limitations and issues to address due to the particular theories, techniques and strategies used. Solutions to the key problems require certain needs (such as the need for better representations, the need for learning for adaptivity etc.) to be satisfied. In this chapter, we look at the core problems, limitations and issues that hamper text retrieval. We start with problems caused by natural language, and then move onto the other key problems inherent to IR processes, followed by a discussion of the requirements a solution might need to possess.

2.1 THE PROBLEM DOMAIN - UNSTRUCTURED TEXT

An IR researcher can restrict his work to a particular domain in which case he will be working in a "controlled" environment, where usage of each term would have a specific meaning with respect to the context. Alternatively, he can leave the domain unrestricted. In the domain of unrestricted text there are no context boundaries. The vocabulary or the search space is unrestricted. In addition, if the structures of the source documents are ignored, any textual material can be taken into account. Examples of such unrestricted search spaces include collections of books in libraries, collections of news stories, e-mail collections and the World Wide Web. However, as soon as restrictions to the vocabulary are lifted, we run into a dimensionality explosion, which increases the ambiguity of word meaning. It requires capturing the conceptual knowledge of all possible contexts/domains in the world.

> *The amount of background knowledge necessary to resolve the word ambiguity problem is immense* - Croft & Thompson 1987.

Nevertheless, if we are to develop generic search engines or retrieval systems for large collections of documents that come from diverse domains such as collection of books in digital libraries or news wire collections etc. we cannot restrict ourselves to work on a selected domain.

2.2 Natural Language and IR

Natural languages provide the means for us to interact with the environment. It is the medium we use to describe things in the environment, exchange ideas between us and document them. A "word" is the most basic linguistic building block of natural language [Allen 1988], at least for the IR researcher. A set of words (vocabulary), together with a set of syntax rules, makes up a language. Expressing an idea or defining a concept usually involves arranging one or more words according to a defined or agreed syntax. Such arrangements create semantic relationships among words and among ideas.

Computer processing natural languages requires the text to be in the form of full, grammatical sentences [Soderland 1997]. Therefore, in principal, understanding written text is simpler for machines (also for humans) compared to understanding a spoken conversation, owing to the fact that the defined syntaxes of languages are usually not strictly followed during speaking. Yet, the flexibility of syntax rules in natural language, the size of the vocabulary and the lexical ambiguities etc. have caused machine understanding of natural language impossible at the present time.

The rules of natural languages are so flexible to the extent that they allow us to express the same thought in different word assemblies, possibly using the same or different words. Also, the contextual interpretation of a given expression may result in a different meaning in different contexts. Even though we have evolved a wide range of strategies and expertise for understanding natural language, it is often difficult, even for us, to read and understand certain expressions correctly, especially in unfamiliar contexts. For a machine, no context may be familiar. Therefore, machine understanding of natural language is far

from a reality for computer manipulation [Croft & Turtle 1992, Kohle & Merkl 1996], especially in unrestricted domains in which the dimensionality or the size of vocabulary is high, the diversity of domain knowledge is great and the ambiguities of words are large.

2.2.1 Natural Language Processing (NLP)

Natural Language Processing is the title given to the study of language manipulation by computers. Two major wings of NLP are Natural Language Generation (NLG), which studies the creation of grammatically correct and meaningful statements; and Natural Language Understanding (NLU), which explores the issues of interpreting meaning of natural language statements. IR researchers are mainly interested in Natural Language Understanding: we never attempt to create natural language statements in text retrieval, but to understand the concepts and meanings of textual material to an extent that allows us to search whether the concepts or meanings of a query are present in a corpus. However, there are certain sub-disciplines within IR, such as question answering, that require forms of NLG.

2.2.1.1 Language Analysis

Languages are analysed by linguists under a number of interconnected subdivisions namely: *Phonology*, *Morphology*, *Syntax*, *Semantics* and *Pragmatics*. In NLP, these subdivisions are treated as a set of independent layers, i.e. they are processed one level after another [Allen 1988, Tyrväinen 1984]. Although this approach is rather simplified and deeply criticised [Fauconner 1990], it could still be used as an outline of NLP.

Phonology considers the formation and combination of phonemes - how words are realised as sounds. This is of less interest to us as we are not concerned of sound or speech. Morphology deals with the forms of individual words - how words are constructed out of more basic units (base-forms) called morphemes. The word form determines, to some extent, the type and the function of the individual words. A word can be decomposed into word stems and affixes. One word stem may have several inflected forms. In English there

are some 75 prefixes and 250 suffixes. Typically, word stems are stored in a dictionary or a lexicon. An entry in a lexicon used for morphological analysis contains a word stem or base-form and additional morphological information, such as the lexical word category or categories, for each interpretation of the stem. Each entry may also contain additional information, such as syntactic and semantic descriptions for each word sense and references to other entries. Also, some lexicons include multi-word phrases [Allen 1988, Karlsson et al. 1990, 1991 (as cited in Tyrväinen 1984)].

Syntaxes deal with the structural properties of natural language - formation of sentences from phrases and individual word forms. Various techniques have been employed to describe the syntax of well-formed text statements, e.g. grammars based on theory of formal languages, automata, constraints and statistical methods. The most popular method to represent the structure of an individual sentence has been a tree structure, with a sentence label *S* in the *root* and the individual words in the *leaves*. If the syntax is described by a set of syntactical rules, each of the rules describes the expansion of a leaf in the tree to a new level of nodes [Allen 1988, Karlsson et al. 1990, 1991(as cited in Tyrväinen 1984)]

Semantics concern the meaning of expressions - what words mean and how their meanings combine in the meaning of sentences. Pragmatic concerns the use of expressions or sentences in different contexts, and their interpretation within those contexts. This needs use of world knowledge: the general knowledge about the structure of the world that language users must have in order to communicate with other human beings [Allen 1988, Karlsson et al. 1990, 1991]

2.2.2 Natural Language Understanding (NLU)

There are three main approaches for understanding text: (1) structural, (2) reasoning-based, and (3) statistical. The level of language analysis required varies depending on the particular approach taken. Structural methods deal with representing and analysing

structural components of sentences to help in understanding meaning. A few examples of the structural approach are part-of-speech tagging for determining the category or type of words as nouns, verbs or preposition; grouping words in sentences into noun or verb phrases; and parse trees to represent and analyse the grammatical structure of sentences. Reasoning-based approaches deal with encoding schemes for capturing the knowledge of language and making inferences on them. They are used to bring knowledge to language, to resolve ambiguities of the natural language. Statistical approaches use numerical tools to examine the relationships between features in text in order to help resolve ambiguities. They have also been used to learn numerical irregularities in text. In particular, they have been successful in creating parsers and part-of-speech taggers at the sentence level.

Statistical approaches have long been used in IR to classify and retrieve documents. Most IR models use weighted term or word frequencies as features for representing documents, ignoring the rich grammatical structures and inter-word relationships. Structural methods and reasoning-based approaches are rarely used for text representation and understanding, perhaps due to the dominance of statistical and probabilistic approaches in the field, difficulty of structural analysis of natural language text, and the common understanding that complete understanding of the meanings of textual material is not required for text retrieval. More details of these approaches are given in Chapter 3.

2.3 Key Problems and Complexities in NL that hamper IR

Recent research in IR suggests that significant improvements in retrieval performance require techniques that, in some sense, "understand" the content of documents and queries [Monarch & Carbonell 1987]. The most fundamental and central problem of natural language text is that its semantics is not well represented by surface features, such as individual words [Croft 1993]. This hampers the performance of retrieval systems that rely on matching surface features between the query and the documents. For instance, the user is generally not interested in retrieving documents with exactly the same words, but with

the concepts that those words represent. The problems related to pragmatics and world knowledge, for instance, are inherently extremely difficult or even impossible to solve.

> *"From the point of view of hermeneutics, the possibility of formalising mental processes behind understanding and interpreting text is still an open issue"*
>
> [Tyrväinen 1984]

- **Mismatch Problem**

The statistical examination of text documents has utilised "keywords" as the basic unit of the problem space for the characterisation of textual material in IR. The use of words as features, in particular, may result in features that are redundant, irrelevant, or even conflicting [Boons 2000]. For instance, suffixes and prefixes of terms and use of synonyms cause mismatches between queries and documents. In addition, the difficulty of using the vocabulary in expressing information needs and characterizing documents leads to word mismatches. Selecting the right words to formulate a query is a difficult task to achieve, since an average user has no idea of how the documents are indexed and what keywords have been used. This "word mismatch" problem results in relevant documents being missed out from retrieval, and thus leads to poor recall. On the other hand, existence of terms in different meanings (polysemy) and the differences between the term relationships within and between queries and documents may cause irrelevant texts to be retrieved. The use of acronyms and anaphors, typically pronouns, also causes misses when certain words are searched. Extra hits are caused by use of analogy or metaphorical expressions and negations. The inability to recognise structures containing many words causes problems when trying to find texts containing several words, such as parts of a compound word or an expression. The use of more general and broader expressions leads to the retrieval of a great deal of unnecessary text, while the use of more specific expressions leads to a failure to hit relevant documents.

- **Dimensionality**

The curse of dimensionality of the problem space is another factor of text domains that limits the abilities of machine understanding, manipulation and learning of text. Each individual word in the vocabulary represents one dimension of the problem space. For instance, the dimension of the vector representation of a document is the number of features (words) in the vocabulary or the domain concerned. The number of words in a natural language is immense and depends on the language. The following statistics and numerical analysis (extracted from [Boons 2000, pp. 18-19]) illustrate the severity of the dimensionality problem in the English language. The number of words in unrestricted text exceeds 150,000 for the English language. In addition to this, there are technical terms specific to various technical, scientific and other domains that should be counted. However, it is conceded that only a small set of words is sufficient to understand the majority of spoken and written terms. A vocabulary of only 750 words will be sufficient to recognise 75% of the words in speech, and 1000 words will recognise 80% of the words in written language [Fishler & Firschein 1987, Boons 2000]. However, even with these reduced vocabularies, the space of possible sentences remains too large. For instance, with only 1000 words, 10^{22} possible 20-word strings can be formed. Although syntactic structures make further constraints on the possible combinations, there are still too many sentences to be tractable. It has been estimated that, on average, there are 10 possible word choices at any position in a grammatically correct and sensible sentence [Pinker 1997]. This only reduces the possible 20-word sentences to 10^{20}. The problem is even more severe in machine learning of text. For instance, with typical vocabularies containing thousands of words, learning spaces based on words can have extremely high dimensionality, in the order of 10^4 to 10^7 dimensions [Lewis 1992]. These factors undoubtedly make perfect retrieval impossible. A number of dimensionality reduction techniques has been tried to solve the problem of high dimensionality of the text domain. A good example is the Latent Semantic Indexing (LSI) approach (Section 4.1.5) by [Deerwester et al. 1990]. However,

reduction of dimensionality is usually achieved at the expense of losing information. In addifion, dimensionality reduction is an expensive task, and has its own drawbacks such as the need for re-computation in the event of new additions etc.

2.4 OTHER INTRINSIC PROBLEMS IN IR

The non-deterministic nature of the goal causes a major problem in IR. Uncertainty, imprecision and vagueness are present in the entire retrieval process. These are partly caused by the involvement of human cognition at different stages of retrieval, augmented by the ambiguities of natural language. This imprecise nature of the IR process makes the quantification of parameters and reasoning difficult and imprecise. These, together with the high dimensionality of the problem space, have made text retrieval to be a computationally expensive and in some cases intractable task.

2.4.1 Cognitive Aspects

As defined by Belew (2000), the need for information is a "thought"; a vague notion that arises in the user's mind. Often the user is uncertain of what exactly he needs. At the next step, he needs to express this ill-posed question either in natural language or in a specific query language used by the system. The process of the transformation of the original vague notion of the information need to a formal language causes a loss of information, making the query expression imprecise. The degree of imprecision depends on the complexity of the information need, the degree of vagueness about the information need in the user's mind, and the ability of the user to transform it into the query language. Even if the user is relatively sure of the kind of documents he wants, it is still difficult if not impossible, to inform the retrieval system so that it understands exactly what documents are desired. The information retrieval system works on this uncertain imprecise expression of the query.

Furthermore, the judgments of the usefulness of retrieved documents for the user's need affect systems that use those judgements for learning and query reformulation. Most often, no document will contain all the information the user is looking for, i.e. no document will

be 100% relevant. Certain documents will contain more useful information while certain others contain less useful information. Deciding which are useful, which are not, and reporting the level of usefulness are uncertain cognitive processes that take place in the user's mind.

User judgements depend also on the expectations of the individual users. The expectations of different users for the same query statement vary depending on the depth of the knowledge they possess about the subject/topic of the search and what they really mean by the query expression. As a result of different expectations, different users may decide the relevancy of retrieved documents differently. Even the same user might respond differently to the retrieved set of documents for the same query expression on two different occasions. This may be caused by the slightly different aspects of the information he might be looking for at the second time, or by the inherent uncertainty of human cognition that he is uncertain of the relevance of a retrieved document to his information need. This result not only affects the fulfilment of the user need, but also the subsequent processing that might rely on the user feedback. This is severe particularly in the systems that require the user to give a binary relevance feedback, as the user has only two options in this case.

In any case, no machine would be able to cater for different expectations of different users given an identically expressed information need. One approach to solve this problem would be to let the system learn from the relevance judgements given by the users in the past for the retrieved documents, and make use of this learnt knowledge to decide which documents are retrieved or ranked the highest based on a majority function. Another approach is to use individual user profiles to enhance queries according to the preferences of the users defined in their profiles. We take the first approach in this research.

2.4.2 Computational Aspects

- **Representation**

IR systems typically operate on document representations rather than on the original documents. Therefore, the richness of the representation certainly affects the effectiveness of retrieval. However, losing information is unavoidable during the process of creating document representations from their source documents. As a result, no document representation is as rich as the original document. For instance, the use of keywords to represent documents loses the underlying semantics and hence hampers the understanding of the message a document expresses. No current representation technique completely captures the meaning of a piece of textual material (document or information need) [Croft & Turtle 1992]. A fundamental problem therefore is to create an appropriate representation that represents reasonably well what the document is about.

- **Acquisition**

Acquisition of information from natural language source documents is a secondary problem related to representation. This is easier with simple keyword-based representations. However, more complex representation schemes, such as semantic networks, need sophisticated tools to acquire underlying syntactic and semantic relationships in natural language expressions. Currently, there are limitations of acquisition of information due to the inherent ambiguities of NL and the high dimensionality of the problem space detailed above. In addition, hardware limitations such as processing power and memory limitations may demand us to operate on simple representations, rather than on more complex and comprehensive representations. The IR researcher is bounded by these limitations to find efficient structures to represent the contents of documents of high dimension with a lot of ambiguities in a rather limited operational environment, using limited tools for information extraction from natural language documents.

- **Background Knowledge**

In addition to the information present in the content of documents, external sources such as knowledge bases, dictionaries and thesauri have been used as additional knowledge sources to help create effective document representations. The use of knowledge bases, in particular, has been primarily to give application domain knowledge to the IR system. They help in disambiguation of concept understanding in order to create correct representations. However, creating domain-specific knowledge bases is a human intense task that requires gathering expert knowledge in the corresponding domain, which in many cases has found to be impractical. Attempts have been made to develop mechanisms for the automatic creation of knowledge bases, but they suffer from the same difficulties that natural language processing and understanding suffer. On the other hand, the use of thesauri in information retrieval has been primarily to obtain synonyms to help word mismatch problems, and to enhance queries with broader/narrower terms. See [Chen et al. 1993] for a detailed discussion of using thesauri, and the problems of developing and using them.

- **Search Strategy**

The central process of an IR system is to match concepts in query representations against concepts in document representations and reason out the usefulness or relevance of documents to user queries. This task needs some criteria to compare queries with documents and quantify how similar they are or how well the document satisfies the user need. The complexity of the matching criteria depends on the complexity of the representation scheme and the reasoning mechanism used. The early best-match search models adopted a simple rule in which a concept was a keyword or a term and the documents having most matching keywords were considered the most useful ones. This approach had a number of drawbacks in that each keyword was treated as equally important, and that it did not help in ranking documents. Also, related concepts may not match if they were described in semantically related terms. A solution for the problem of

different importance was to assign significance values to keywords and use the significances of matching keywords to compute a retrieval status value (RSV). The RSV values decide the position of the document in the rank list. A range of search strategies used by different models is indicated in Chapter 4 under the corresponding model.

- **Concept Weighting**

IR systems typically weight the importance of search terms/concepts according to document and collection statistics. For example, the *tf-idf* scheme [Section 3.3.1.3] uses frequency counts of words within the document (*tf* factors) and presence counts of words across the documents (*idf* factors) in the collection to compute significances of keywords. It increases the weight of words that occur frequently in a document but infrequently in the document collection. The effect is to emphasise the words that are the most suitable for unique identification of the document, while reducing common words that are likely to mislead the retrieval. Although this works moderately well, it can easily go wrong in case of unusual or deliberate repetition of keywords in documents. For instance, a well-known trick to get search engines to rank a web page highly is to deliberately repeat keywords in it.

The significance of a concept in a document is a relative measurement. It depends on the complexity of the concept itself; how much it is related to the overall context of the document; how good it is in communicating or representing the main topic of the document; how useful it is in retrieving the document; and, more importantly, the likelihood of it being used by the end user for formulating queries targeting that document. For keyword concepts, a single document contains a number of concepts, but for an arbitrary user, only a few of them may be important. Moreover, there may be a lot of overlaps of these concepts between documents. These make the significance measurements (concept weighting) difficult and imprecise regardless of how you compute them. The reasoning process has to rely on this imprecise and vague measurement, thus resulting in imprecise outcomes. Various techniques have been tried to model the uncertainty and

impreciseness of the significance of concepts. The use of probability theory, fuzzy logic and machine learning are a few examples for such techniques. A particular approach that interests us is the one based on the assumption made by Maron and Kuhns [Maron & Kuhns 1960] that "term weights for a document can be estimated on the basis of relevance information from a number of queries with respect to the specific document". Coincidently, our reinforcement learning strategy is also based on a similar idea in which the significance of concepts is determined solely based on the relevance of retrieved documents to user queries.

- **Relevance Feedback**

Later IR systems viewed the retrieval process (of a single user need) as a continuous process that goes through a number of passes of retrieval and query reformulation stages. The user retries the system by modifying his query statement based on the information he has received from his previous attempt until he is satisfied with the retrieved documents. A complete set of passes from the original query to a point at which the user is satisfied with the retrieved documents is termed a "query session". It allows the user to understand his information need better and thereby express it better using appropriate terminology. The documents he receives at each pass give some sort of direction with regard to which terms to use and which terms to avoid. This is a form of user learning process that helps to alleviate the gap of knowledge and the differences of the vocabularies between the user and the authors. The well-known automatic relevance feedback technique is simply an automated version of this process. An attempt to automate relevance feedback requires two problems to be addressed: (1) how to obtain the user feedback and in what form, and (2) how to use it effectively to support IR.

In the simplest form of automated relevance feedback method, the terms of the first few best-ranked documents are added into the query statement and the query is re-run. In this case, it is the system's matching and ranking algorithms that decide the best documents for obtaining concepts from, for query reformulation. A better method is to get the user to rank

the retrieved documents according to their usefulness to the user, and then use the concepts of those documents that are most useful, for query reformulation.

The main drawback of query reformulation with user feedback information is that it does not retain the past user feedback information. The user decisions obtained are only available during that query session. For instance, the same user for the same information need would have to go through the same process again if he does not remember the way he formulated the query at his last successful attempt. Instead, if the user decisions were retained, it would not only help the same user but also be of help to other users with similar information needs. This indeed was one of the primary goals in our reinforcement learning strategy used in this work (Chapter 7).

- **Query Language and Global Search Space**

Two other practical problems faced by the user are the complexity of the query language and lack of knowledge about the global search space. In practice, conventional IR systems typically have different query languages and notations for operations. The facilities provided for the user vary from system to system and in some cases require more learning than is usually expected. Also, the users have difficulties in realizing the size and contents of the database, which is usually invisible to the user. The users are not able to estimate the recall based on the responses of the system and are thus uncertain whether they have already received the most useful documents or whether they should redefine the query and try again. For instance, if you know for sure the presence of a particular book in a library, you would continue searching until you found it. If not, you would probably decide the book is not available in the library having not found it after a few attempts.

2.5 Needs For Better Performance

The word sense ambiguity and variability problems caused by synonymy and polysemy in particular, should be tackled by developing better **representations** with sufficient knowledge by capturing the central concepts from text material. For this, retrieval based on

concepts and relations between them would be needed [Croft & Turtle 1992, Appel et al. 1988, Rada & Hafedh 1989], including both hierarchical relations to support decisions about conceptual similarity and non-hierarchical relations for describing other relations. Such representations with the central concepts of the subject domain should be efficient to build and use for text indexing and retrieval.

Capturing the user need as precisely as possible despite the vagueness in the user's mind about his information need and the imprecision involved in expressing it in a query language are two other major needs in IR. Attempts have been made to solve these problems by enhancing the query in various ways. Relevance feedback is the first to mention. In addition, external knowledge bases have been used to provide background domain knowledge to help clear and correct understanding and formulation of information needs, as well as thesauri to help reduce word mismatches. Systems with more natural language origin have used more complex NLP to help word disambiguation. Anaphor resolution can be cited as an example of such complex processing.

The comprehensiveness of the **basic unit of the problem space**, the "concept", and the level of knowledge included in the representation of the information items (documents) within the computer provide the basic infrastructure for effective retrieval. This is described in detail in Section 3.5 with a more formal definition of the notion of the "concept". The word-mismatch problem (of keyword matching) and most other problems mentioned above demand a better information unit that is able to capture underlying semantic relationships between words for the creation of well-representative document surrogates. This task requires tools that support the acquisition of such information units or concepts from natural language text. The importance of developing such tools does not seem to have been well recognised and researched by the IR community.

However, there is a trade-off between the complexity of the representation and the efficiency of its building, manipulation and maintenance by computers. Finding a rich representation that operates on low computational requirements is a fundamental challenge to the IR researcher.

Learning, Adaptation and Background Knowledge

Adaptation achieved through learning can help alleviate most of the problems caused by the ambiguities of natural language text and insufficient background knowledge. Various strategies have been tried for making systems learn and adapt. Most of these are query adaptation/reformulation mechanisms that are based on relevance feedback. Ontologies and knowledge bases have been employed for incorporating domain knowledge in order to help understand the content of documents and queries.

Conventional relevance feedback mechanisms, as used in query reformulation mechanisms, provide a form of user adaptation to system responses in addition to adapting the query representation with terms/concepts picked up from relevant documents. System adaptation in this case is temporal, as the important feedback information given by the user is used only within the current retrieval session. In some automatic relevance feedback mechanisms (blind relevance feedback), the top ranked documents are considered as relevant and terms/concepts obtained from those documents are used for query reformulation. In contrast to system adaptation, the user also gets feedback from the sets of documents the system retrieves as he interacts with the system. This lets the user learn from the retrieved documents, for example, which terms to use and which terms to avoid in order to directing his search session successfully. This user adaptation though may not be permanent, stays longer as the user may use the knowledge he has gained in the past in subsequent search attempts. However, this learning is local to individual users. Instead, in our work, we are interested in a long term system adaptation. This requires systems to learn from past experiences and retain the learnt knowledge for future use. A conventional approach for system adaptation is to learn information filtering rules or knowledge

representation rules from a given set of (training) data set as in traditional NN learning. Crestani's model [Crestani 1994, Crestani & van Rijsbergen 1997] is an example for learning information filtering rules through a set of training samples in which a neural network learns rules from training data to respond to queries (Section 4.5.3.2). In this approach, learning is done prior to the deployment of the system, and therefore the knowledge learnt is fixed thereafter. It does not make a system continuously adaptive to the environment. In addition, the result of learning depends on how good and how representative are the training samples and their relevance assessments (user judgments). As a result, changing user interests and changing users are not dealt with by this approach.

Note that, a surprisingly little overlap has been found between relevant document sets that different users have indicated relevant for the same information need [McGill et al. (1979) as cited in Lee 1998]. In addition, the meanings of descriptors are subject to change with time. Therefore, NNs trained prior to deployment require retraining when new documents are added. Instead, the use of a learning strategy that keeps learning interactively as users interact with the system would be a preferable approach, especially, given the dynamic nature of IR environments. The reinforcement learning strategy that we use in this work learns interactively from user feedback, and retains the learnt knowledge for future use. In addition, its adaptive feature lets the system forget (in time) the importance of documents that users found useful in the past but no longer do so.

Despite the advantages indicated above, learning knowledge through experience, or, to be precise, through user feedback, is bound to have certain limitations and in most cases takes time to build. Also, the knowledge learnt might not be sufficient, accurate or consistent as it depends on the knowledge, views and expectations of its past users. Alternatively, rather formal knowledge can be incorporated to an IRS through the use of domain specific knowledge bases or ontologies. However the lack of knowledge bases or ontologies in certain domains makes such an approach unsuitable in such domains. On a larger scale, the

unavailability of generic ontologies that cover all domains (in general) makes the use of knowledge bases impractical for a generic IR system.

2.6 SUMMARY

The primary problem in IR is to deal with the uncertainty, imprecision and vagueness that exist in all components of the IR task. In this chapter we first discussed the problems caused by various complexities and ambiguities of natural language and then the cognitive and computational aspects of the causes that make the problem uncertain, imprecise and vague. The key issues and needs for improving the effectiveness of IR were identified.

The role of an IR researcher is to create a computational model to deal with these issues. In our work we paid considerable attention to the problems, issues and needs discussed in this chapter to find a solution(s) for the IR problem within the framework of FCA.

It should be mentioned that in addition to the core problems and issues of IR mentioned in this chapter, IR researchers are also faced with an additional set of problems and issues with regard to the evaluation of their models. These include the lack of evaluation strategies and supporting test collections, especially for evaluating adaptive IR models; the difficulty of creating such test collections with a sufficient amount of queries and user assessments; the impracticality of experimenting on huge collections such as TREC (**T**ext **RE**trieval **C**onference) collections due to computational overheads; and the suitability of performance measurements such as precision and recall etc. These are not discussed in detail in this thesis.

CHAPTER 3 - TEXT/KNOWLEDGE REPRESENTATION

Content-based document retrieval systems require the contents of documents to be extracted and represented in an efficient way that helps their retrieval for user queries. Despite the limitations of "keywords" in representing contextual "concepts", the majority of today's operational IR systems use keyword-based representation schemes. Instead, in our work, considerable effort was devoted to creating a text representation embedding a form of conceptual knowledge extracted from textual material to help concept matching. In this chapter, we give a review of well-known text/knowledge representation methods used in IR. The aim is to understand what representation languages are being used, how they are implemented, what are the problems associated with them in terms of both the level of representation power and practical implementation and tractability. The representation language used in our work is not discussed in detail in this chapter as Chapter 5 is dedicated to this purpose containing details of the theoretical foundation of the representation methodology and the rationale of using it.

3.1 KNOWLEDGE REPRESENTATION IN IR

Davis, Schrobe and Szolovits [Davis et al. 1993] describe each knowledge representation technology as a trade-off between the following five basic roles they play: (1) a surrogate; (2) a set of ontological commitments; (3) a fragmentary theory of intelligent reasoning; (4) a medium for efficient computation; and (5) a medium of human expression. All 5 roles are equally important in the context of knowledge representation in IR. Document or query representations inside IRS are indeed surrogates for their real partners that exist in the physical world. When we select a particular representation technology to use (for instance a bag of keywords), we make commitments about what the world (problem space) of the system looks like and what to look for in it (keywords in this case). The reason for creating/using a representation in IR is mainly for efficient computation to help reasoning

which documents are useful to user queries. Deciding the relevancy of documents to the user involves making intelligent decisions based on the content of the document. This requires a mechanism to match concepts between the query and the documents and quantify the usefulness of them to the user. Matching concepts between textual materials in general is not possible if we do not know what to match. Attempting to match all terms (words) of a query with all terms of a document is not a sensible approach as a "word" in natural language does not necessarily represent a "concept" and is prone to word mismatch. Therefore, a way of representing documents in terms of more meaningful concepts is required. The result of such an attempt is a document surrogate with a conceptual representation of knowledge in the document.

In IR, the user's information needs, which exist in their mind merely as ideas rather than in textual form, need to be mapped to the document representations that fulfil the user's need. The mapping processes utilise explicit representations of the information needs and representation of world or domain knowledge, as well as representations of the documents. The information needs may be mapped against taxonomies of the system represented using abbreviations and knowledge representations, such as domain classifications and index terms. A major design constraint for IR systems therefore is the choice of an effective representation language (scheme) for both describing information problems and characterisation of text. A representation language provides a vocabulary based on a particular view of the world. Inadequate capabilities to represent and exchange common conceptual knowledge of the domain between the retrieval system and users are a primary problem in IR. Most IR systems describe documents and queries using the same view of the world (i.e. using the same representation language) in order to help direct exchange of knowledge between the user and the system, and also to help direct comparison of query representation units with document representation units.

3.2 Two Levels of Representation in IR(An Overview)

Tyrväinen [Tyrväinen 1984] describes two levels of representation used in IR: (1) knowledge representation; and (2) text representation. Knowledge representation contains representations at the idea level (less string-dependent structures than terms or phrases); their properties and organisation serve as a collection of common knowledge describing the structure and concepts of the world or domain from a commonly accepted point of view. This provides a common grounding for the users to unify their information needs with the ideas of the authors by mapping them to the common concepts at the knowledge representation level. As a result of this, the gap between reader's and authors' mental models of the subject information domain is reduced to the mapping between readers' mental models and the representations of the system's explicit model.

The text representation level contains representations extracted from the texts with formal (syntactic) methods such as content representatives extracted from natural language text using NLP and statistical methods, document structures, and formal properties extracted utilising mark-up information. The effectiveness and the complexity of content representatives extracted from text with statistical or NLP methods vary a lot depending on the level of processing used. At the beginning of the NLP scale there are strings limited by space characters, i.e. the technique of traditional inverted files with the use of all words of a text as its content representatives. The next level includes using truncated words and word base-forms, compound words and phrases etc. At the other end of the scale there are normalised knowledge representations extracted from the text using morphological, syntactical and semantic processing. These representatives could reach the level of knowledge representations when pragmatic issues about the context are also taken into account. Typically the text representatives extracted automatically from text do not reach this level [Tyrväinen 1984].

The scope of our work with respect to representation is to extract concepts/features from local documents and use them to create abstract representatives of their content. We use some NLP techniques for extracting features and relations between them to create text representations, but do not attempt explicitly to create representations of common background knowledge of the world or domain. According to the definitions given above, our representation strategy can be categorised as a form of text representation. However, our attempts to use a conceptual structure and in particular the relationships between concepts gives it a flavour of knowledge representation properties as well, thus leaving it in between text and knowledge representation.

Two conflicting ways of characterizing documents have been identified for retrieval.

1. Using local information, i.e. each document is represented by using its own content independently from other documents in the collection (locality).
2. Discriminating each document from others by taking into account the contents of all the documents (globality).

Different representation approaches have different levels of locality and globality. In reality, there is a trade-off between the two and therefore a balance between the two is practised. For instance, the *tf-idf* approach (Section 3.3.1.3) can be considered as an approach that attempts to control the balance between these two extremes via its *tf* (term frequency) and *idf* (inverse document frequency) components.

3.3 INDEXING - A REVIEW

Indexing is the process of extracting important concepts from textual material and creating a representation to support subsequent processing. Representation of a document should enable it to be retrieved in response to requests of information needs (queries) if the document contains useful information to help the user. It was in this manner that documents were traditionally characterised by human indexers when index terms were assigned to documents. It is the task of the indexer to anticipate index terms that a user would be likely to use in his query expression to retrieve each document. Van Rijsbergen

viewed this as constructing a set of potential queries for which the document is relevant [Rijsbergen 1979].

Two important factors that govern the effectiveness of an indexing language are the exhaustivity of the indexing and the specificity of the indexing language. Indexing exhaustivity is defined as the number of different topics indexed and the index language specificity is the ability of the index language to describe topics precisely [Rijsbergen 1979]. These two factors are assumed to be vaguely related to the distribution of index terms in the collection. Exhaustivity is assumed to be related to the number of index terms assigned to a given document and specificity to the number of documents to which a given term is assigned in a given collection [Rijsbergen 1979]. It has been recognised that a high level of exhaustivity of indexing leads to high recall and low precision and vice versa.

The unit of representation for indexing could simply be single terms or more complicated constructs with multiple terms. In fact, the choice of terms in the indexing units (single terms, phrases or terms/phrases with relations etc.) to represent documents depends on the context in which they are going to be interpreted. The target audience (users) who are going to access the document is also a major factor.

3.3.1 Bag-of-Keywords Representation

3.3.1.1 Binary Indexing

Binary indexing is one of the oldest indexing schemes pioneered by Luhn [Luhn 1958], within which each document and request is represented by a set of keywords without weights. Similarity measure between a document and a request was given by the number of terms they have in common (known as ***Naive Keyword Hypothesis***), i.e. we assume that "*if a query and document have a keyword in common, then the document is about the query to some extent*".

3.3.1.2 Term Weighting

Binary logical restrictions may often be too restrictive for document and query indexing. It is not always clear whether a given document should be indexed by a given term. In addition, the binary indexing representation method ignores the variability of the importance of different keywords in a given document and also the variability of a given keyword in different documents. It treats all the keywords as equally important. This obviously is an incorrect assumption as certain keywords in a document are more important than others in representing the document with respect to the contextual relevance. An improved version of binary indexing, known as term weighting, is to use the frequency of occurrence (term frequency *tf*) of these words in the body of the text to indicate the degree of significance of each keyword. This provides a simple weighting scheme for the "keywords" in each bag making a document representative in the form of a "weighted keyword description" [Rijsbergen 1979]. It creates a distinction among terms and increases indexing flexibility. Similarity between a document and a query is decided based on the number of terms they have in common, weighted by the component *tf*.

3.3.1.3 TF-IDF

Salton [Salton & McGill 1983] pointed out that those terms occurring very frequently in the collection do not help to discriminate between relevant and non-relevant items. He took into account a term's frequency in the collection (inverse document frequency *idf*) in order to find the significance of a token in the document. The *idf* gives a large weight to more sparsely used words and a smaller weight to more frequently used ones. This gives terms appearing often in a given document and rarely in other documents in the collection a higher weight.

A "good" weighting formula obviously should take into account the document lengths as well, so that shorter documents are not penalised against longer ones. In this case, a match on a short document will be treated as more valuable than a match on a longer document.

This is achieved by document size normalisation which prevents ranking a document either too high or too low simply because of the number of terms in the document. Thus, in *tf-idf*, *tf* and *idf* information on terms is used to compute a weight for each term in each document normalised for the size of the document. The *tf-idf* weight computation is illustrated below.

Let N be the total number of documents in a collection, n_i be the number of documents in which the keyword k_i appears, and $freq_{ij}$ be the raw frequency of the keyword k_i in the document d_j. The factor $freq_{ij}$ quantifies the importance of the keyword k_i to the document d_j (i.e. the term frequency *tf*) and the factor $\boldsymbol{log}(N/n_i)$ quantifies the importance of the keyword k_i as a discriminating factor for the whole document collection (the inverse document frequency *idf*). The term-document weight w_{ij} is computed as $w_{ij} = freq_{ij} \times \boldsymbol{log}(N/n_i)$ and the term-query weight w_{iq} as $w_{iq} = freq_{iq} \times \boldsymbol{log}(N/n_i)$, where $freq_{iq}$ is the raw frequency of the keyword k_i in the text associated with the query q. Given the sets of weights w_{ij} and w_{iq}, the weighted query and document vectors q and d_j are represented by $q = (w_{1q}, w_{2q}, \ldots, w_{tq})$ and $d_j = (w_{1j}, w_{2j}, \ldots w_{tj})$, where t is the total number of keywords in the system.

3.3.1.4 Drawbacks of Keyword-based Representations

Keyword-based representation schemes fail to capture the syntactic and semantic information present in natural language text and also suffer from a number of problems originating from "*language variation*". The following are reported in [Arampatzis et al. 1998] as drawbacks of keyword-based representation schemes:

1. They do not handle cases where different words are used to represent the same meaning or concept in queries and documents (lexical variation or *synonymy* problem).
2. They do not distinguish cases where single words have multiple meanings due to "*semantical variation*" (*polysemy* problem).

3. They do not deal sufficiently with the problem of "*syntactical variation*", e.g. a document saying "*near the the river, air pollution is a major problem*" is not about "river pollution".
4. To make matters worse, keywords can, due to "*morphological variation*", appear in different numbers, for instance "*woman*" and "*women*", or different cases, like "*man*" and "*man's*".

3.4 Organised File Structures

Early experiments with document retrieval systems used serial file organisation for keeping document representations. This was sufficient for early batch processing systems, but was proved to be inadequate in real time processing. Later on, the need for logically structured files to keep document representations, referred to as an *information structure*, was recognised to be important. One of the most popular such organisations for keyword-based representations is the *inverted index* file structure. Another organisation demonstrated as superior for on-line retrieval is the *clustered files* produced by automatic classification methods.

3.4.1 Inverted Index

Typically, IR systems that are based on keywords build an *inverted index* to store and access terms in a document collection efficiently. An inverted index consists of two components: a list of distinct terms referred to as the *index* and a set of lists referred to as *posting lists* (see Figure 3.1). The posting list is simply a linked list that holds information about the documents with which that index term is associated. The structure of a posting list entry does vary from implementation to implementation. It always includes the document number but can also include entries for *term frequency*, *term weight*, and possibly position data, such as *the location of the term* in the document (e.g. word, sentence, paragraph) to facilitate a proximity search.

Building an inverted index is an expensive task that involves parsing each document in the collection and computing the required statistics (*tf* etc.). Usually, inverted index files are built only once for a given document collection before the system is put into operation.

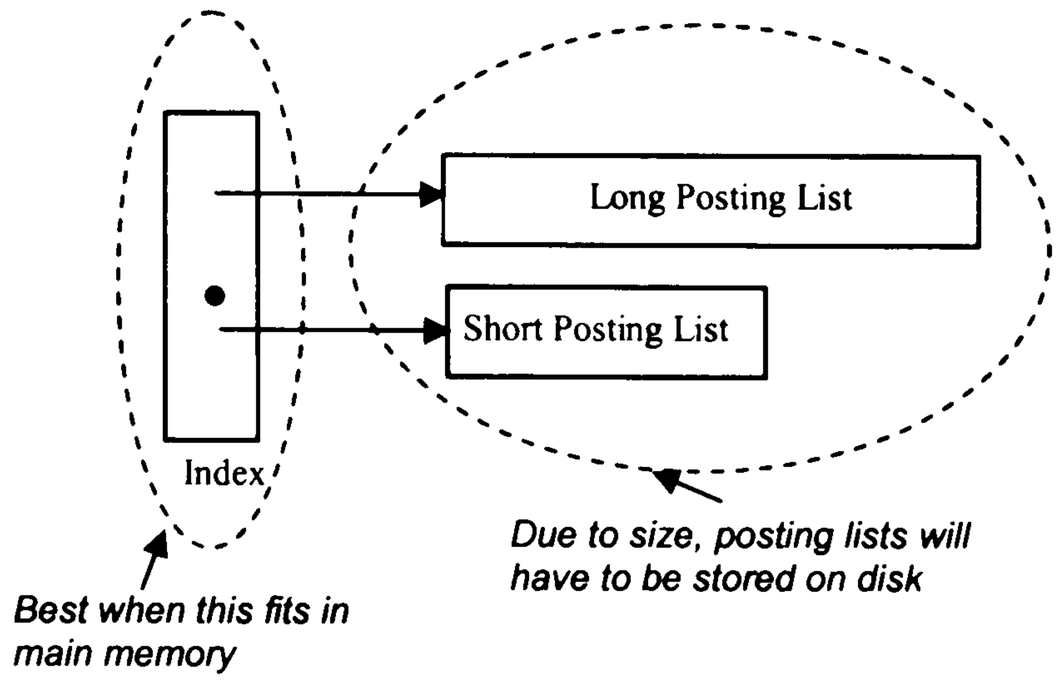


Figure 3.1 : Structure of an Inverted Index

For example, if one document (say Doc#1) in a given document collection contains five occurrences of the term *computer* and two occurrences of the term *apple* and another document (say Doc#2) contains three occurrences of *computer* and one occurrence of *apple*, then the part of the inverted index (for these two documents) would be:

computer → (1,5),(2,3)
apple → (1,2),(2,1)

Note that only document numbers and term frequencies are shown in posting lists for simplicity.

This information structure supports any representation scheme that makes use of term frequencies and document frequencies. Term frequency (*tf*) of a given term in a given document can be obtained directly from the corresponding list entry. Inverse document frequencies (*idf*) for each term can be calculated by scanning the entire list of unique terms.

A study of efficiently generating an inverted index (including parallel processing) and compressing an inverted index for efficient storage etc. is reported in [Frieder et al. 1999].

3.5 Hierarchical Organisations and Conceptual Indexing

The notion of a "concept" is central to all methods and mechanisms that attempt to represent knowledge. In keyword-based approaches, a "keyword" is considered as a "concept" that represents an idea. A major problem with keyword-based representation schemes is that they implicitly assume that the keywords or concepts are independent of each other. They do not attempt to capture the semantic relations between them. Organising concepts according to some form of a structure that captures the underlying semantic relationships between individual concepts is extremely useful in understanding the meaning of a concept with respect to the context, as well as reducing ambiguities in IR [Sanderson & Croft 1999]. This indeed is one of the main goals of recent information retrieval research.

Techniques developed for creating hierarchical organisations of concepts attempt to capture forms of hierarchical relationships or categorisations of keywords/concepts. The standard way of organising information in libraries, encyclopaedias and in the indexes at the backs of books by manually organising topics into a fixed hierarchy or listing topic phrases in alphabetical order can be regarded as a basic form of hierarchical organisation. The simple alphabetical ordering of lists of topics has serious limitations due to the difficulty of guessing the exact sequence of words used by the scheme to describe a topic. Guessing incorrectly can lead to looking for the desired information in the wrong place. Although hierarchical topic trees are useful for outlines and tables of contents of books, they start to break down when applied to whole library collections (such as the Dewey decimal classification system or the Library of Congress (LC) classification system) or topic lists in online information services when they contain more than a few levels of depth in thousands of topics [Woods 1997].

> *"Strict hierarchical systems like Dewey and LC are unable to fully capture all of the desired relationships because of the necessity to place each concept in exactly one place in the hierarchy "* - Woods 1997.

There have been many other attempts to develop more elaborate techniques for effective representation of relationships between concepts in different structural forms (hierarchical or other). Among them semantic networks and its descendents are the most important.

3.5.1 Semantic Networks

Since their introduction by Quillian [Quillian 1968], semantic networks have played a significant role in knowledge representation research. They express knowledge in terms of concepts, their properties, and the hierarchical sub-super class relationships between concepts. Concepts and their properties are represented by nodes in the network. The hierarchical relationships between concepts (and properties) are depicted by connecting appropriate concept nodes via relationship links, such as "*is-a*" or "*instance-of*". Nodes at the lowest level denote individuals whilst nodes at the higher levels denote classes or categories of individuals. Concepts get more abstract as one moves up the "*is-a*" hierarchy.

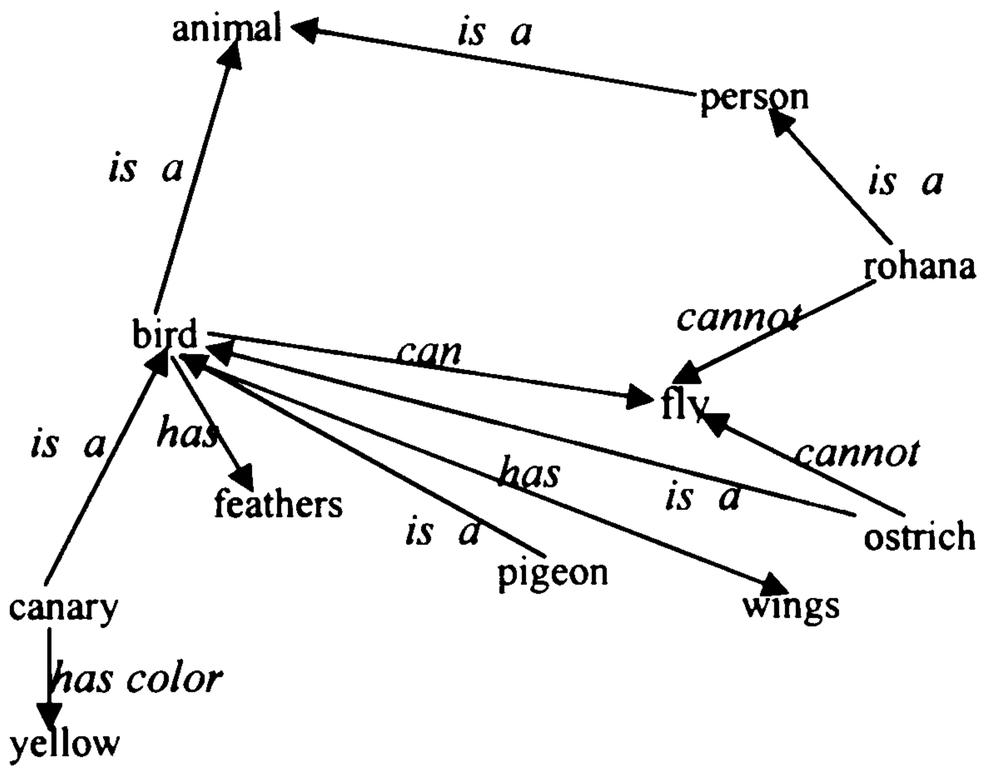


Figure 3.2 : A Semantic Network Representation
(Reproduced with modifications from [Crestani 1997] with permission © 1997 Kluwer)

Typically, a property is attached at the highest concept in the conceptual hierarchy which possesses it. If a property is attached to a node, it is assumed that it applies to all nodes that are descendants of that node. An example is given in Figure 3.2.

Although this formalism is highly expressive and has a great deal of potential to represent almost any kind of knowledge, it is too unconstrained and places the burden of constructing appropriate sets of facts and rules on the programmer [Luger 2002].

Semantic Nets in IR

The form of semantic networks used in IR, however, has often been a far more general form of "Associative Network" than that described above [Crestani 1997]. This is a generic network of information items in which nodes represent information items, and links express associative relations among them. These links are sometimes undefined and unlabelled. In recent applications, however, the links that represent relationships among information items are assigned weights to express the strength of associations. These weights are calculated based on statistical techniques during the indexing phase.

Almost all the networked representations, including Bayesian networks, self-organising maps and other neural network methods, conceptual graphs and concept lattices can be regarded as simplified semantic networks. Despite the attempts made using different network representations with different levels of representation capabilities, the difficulty of automatic network creation and the difficulty of developing efficient mechanisms for concept matching between semantic network representations of documents and queries restrict the use of such more elaborate network representations in IR.

3.5.2 Scripts and Frames

Natural language understanding needs a large amount of background knowledge to understand even the simplest conversation. In particular, any ambiguities in NL expressions are resolved in a way consistent with the contextual knowledge. For instance, if the subject of a story changes abruptly, there is evidence that people pause briefly in their reading, presumably to change knowledge structures. Also, when the subject of a conversation changes abruptly, people seem to get confused over which context to use in resolving pronoun references and other ambiguities in the conversation [Luger 2002].

It is evident that humans organise this knowledge into structures corresponding to typical situations [Luger 2002]. The "script" representation technique is based on this idea. It is a structured representation describing a stereotyped sequence of events in a particular

context, designed as a means of organising "conceptual dependency" structures into descriptions of typical situations. The elements of the script, the basic "pieces" of semantic meaning, are represented using conceptual dependency relationships. Placed together in a frame-like structure, they represent a sequence of meanings or an event sequence.

Frames are similar to scripts in many ways. They support the organisation of knowledge into more complex units that reflect the organisation of objects in the domain. A frame is viewed as a static data structure used to represent well-understood stereotyped situations. It is a remembered framework with default values that can be adapted to fit reality by changing details as necessary. Frames allow the organisation of our own knowledge of the world. We adjust to every new situation by calling up the information structures built in our memory by past experiences and revising the detail according to the new situation. We could represent these high-level structures directly in a semantic network by organising it as a collection of separate networks, each of which represents some stereotypic situation. However, frames and scripts have not been used for representing documents/queries in mainstream IR research, possibly due to the difficulty of automatically generating them and the difficulty of applying them in unrestricted domains.

3.5.3 Conceptual Graphs

Conceptual graphs (CGs) [Sowa 84] are a system of logic, based on the existential graphs of Charles Sanders Peirce and the semantic networks of artificial intelligence [Luger 2002]. Their purpose is to express meaning in a form that is logically precise, humanly readable and computationally tractable. Conceptual graphs are formally defined as a graph or network of two kinds of nodes: Concepts and Relations. The nodes have directed (unlabelled) arcs between them. Concept nodes represent either concrete or abstract objects in the world of discourse. Concrete concepts are concepts which can form an image of in our minds (mainly physical objects, such as a *cat*, *telephone*, or *restaurant*), while abstract concepts include ideas or feelings, such as *love*, *beauty* and *loyalty* that do

not correspond to images in our minds (conceptual objects). Conceptual relation nodes indicate a relation involving one or more concepts. Because conceptual graphs are bipartite, concepts only have arcs to relations, and vice versa. In conceptual graphs, every concept/node is a unique individual of a particular type. Each concept box is labelled with a type label, which indicates the class or type of individual represented by that node. Types are organised into a hierarchy. Each concept box is labelled with the name of the type and the the name of the individual concept. The type hierarchy is a partial ordering of the set of types; thus a type may have one or more supertypes as well as one or more subtypes. In fact, the type hierarchy of a conceptual graph representation is a lattice, a common form of multiple inheritance system. In Concept Lattices (see Chapter 5), types in a type hierarchy may have multiple parents and children. However, to be a true lattice, the type hierarchy of a conceptual graph representation requires a "universal" supertype of all types and an "absurd" subtype of all types [Luger 2002].

Since each conceptual graph represents a single proposition, a typical knowledge representation will contain a number of conceptual graphs. For instance, a conceptual graph representation of the contents of a document requires a collection of conceptual graphs, one for each proposition in the document, as opposed to the single network of concepts usually used in a semantic network representation.

3.5.4 Concept Lattices

In our research we use concept lattices as defined in the theory of *Formal Concept Analysis* (FCA) [Ganter & Wille 1999] to represent documents/queries in the form of hierarchical concept structures (*concept lattices*) which are built by the concepts extracted from the text. Since a detailed treatment of Concept Lattices is given in Chapter 5, they are not discussed here to avoid duplication.

3.6 Citation Indexing and Hyperlinks

The main purpose of having hyperlinks in a web page (at least for the user) is to let the user navigate the search space. For IR researchers and system developers, hyperlinks are additional informative entities that give relationships between web pages. This is the same concept as citation indexing used in IR to identify relationships between scientific papers by using bibliographic reference information. Citation indexing is based on the assumption that bibliographic references give credit to related work. However, the practice of citation has been based on more complex motives than citing other pertinent documents [Liu 1993]. Liu's studies have shown that, on average, about half the references in a paper are not connected with the main problem of the paper. Work on citation/hyperlink schemes for improving searching can be found in [Kleinberg 1998, Bharat & Henzinger 1998, Brin & Page 1998, Dean & Henzinger 1999].

3.7 Problems with Knowledge/Text Representation in IR

In addition to the drawbacks of keyword-based representations discussed in Section 3.3.1.4, two major problems of text representation are identified with respect to conceptual indexing. Firstly, the source documents, in general, lack sufficient contextual or domain knowledge for a computer system to resolve ambiguities in word meanings based on the content of the document alone. A solution for this would be to use external knowledge sources to assist the IR system with world knowledge. Ontologies, thesauri, dictionaries and other domain-specific knowledge bases have been tried for this purpose. This approach has limitations due to the domain specificity of external knowledge sources and is therefore suitable only for IR systems that operate in restricted domains. It is not appropriate for domain independent information retrieval. In addition, more exhaustive knowledge representation techniques, such as semantic networks, suffer from the difficulty of building and handling them efficiently in a computer system. Information retrieval is a time-critical application that calls for efficient access to the concepts in document

surrogates in real time. For this reason, semantic network representations have not been recognised as a practical solution for text representation in IR. A more practical solution would therefore be to incorporate a learning mechanism on a computationally lighter representation scheme and allow the system to learn document representations through experience.

3.8 SUMMARY

In this chapter a review of text/knowledge representation techniques used mainly in IR were described and the distinction between knowledge representation and text representation was emphasised. The traditional bag-of-keywords and *tf-idf* based representation mechanisms are mainly text representation mechanisms that depend only on the presence of words and phrases in the text. Despite their drawbacks (Section 3.3.1.4), they have been very popular in the development of practical IR systems due to their simplicity and efficiency. More elaborate conceptual knowledge representation mechanisms, such as conceptual graphs and semantic networks, on the other hand, encode contextual meaning in a given context or domain by means of objects and the relationships between them. Typically, a hierarchy of objects as well as a hierarchy of relations are assumed. Acquisition of such knowledge requires consultation of domain experts and extraction of contextual domain knowledge from the entire collection. Not only is the acquisition of such knowledge expensive and sometimes impossible, but it is also expensive in terms of both storage and access, and thus is inefficient to implement in a computer. However, given the inadequate semantics of keyword-based indexing, ways of creating efficient means for text representation are required. Such a representation should be enriched with sufficient semantic knowledge. In this regard, we find the concept lattice a suitable candidate for text/knowledge representation for IR tasks, given the fact that it lies in between the spectrum of keyword and conceptual indexing. Although concept lattices may not be as powerful, flexible or representative as semantic nets or conceptual graph representations, they share some of the important properties of these two techniques

and are computationally more efficient and tractable. Chapter 5 presents a detailed treatment of concept lattices and the underlying theory of Formal Concept Analysis. Before that, a review of IR approaches, their underlying theories and techniques is presented in the next chapter (Chapter 4) for the sake of the completeness of this thesis.

CHAPTER 4 - A REVIEW OF IR APPROACHES

This chapter reviews various IR models and their underlying techniques. A vast number of research attempts in IR over the past half a century have resulted in an incredible amount of publications available within IR. Although each of these publications reports something new, it is impossible, and is not intended to review them all here. We restrict this review to the most popular text retrieval approaches, leaving numerous variations of the key approaches and their application to other sub disciplines of IR research, such as indexing, clustering, text filtering, IR in the Web etc. out of our discussion.

The chapter begins with short reviews of the more conventional and well known approaches such as Boolean, Vector (VSM) and Probabilistic models followed by more detailed reviews on more relevant connectionist approaches. Advantages and shortcomings of each approach are indicated. Research most related to that described in this thesis, however, is based on conceptual structures, in particular, the application of formal concept analysis (FCA) and concept lattices into IR. Since the understanding of these models requires some knowledge of the basics of the theory of FCA and concept lattices, we only briefly mention the work on concept lattice-based IR in this chapter. A detailed review is given on the next chapter (Chapter 5) after the theory of FCA and concept lattices are presented.

4.1 Conventional Approaches

4.1.1 Boolean approach

The traditional "Boolean Search Strategy" is the oldest of all the conventional IR models. Old library systems are classic examples that have a long history of Boolean retrieval. It allows the user to formulate a structured Boolean query according to the formalisms of Boolean algebra using index terms (keywords) and logical Boolean connectives AND,OR, and NOT. It retrieves only the documents that exactly satisfy the Boolean conditions of the

query. As a result it is too selective. In addition to the word mismatch and other problems common to the models based on keywords (discussed in Section 2.3), it suffers from another two major inadequacies : (1) it treats each keyword as equally important, and (2) the basic Boolean model is incapable of ranking documents. Furthermore, the complex query language used by the Boolean model makes it difficult for an inexperienced user to formulate his information need. However, Boolean search model is recognised for its strength to make very restrictive searches to obtain exact and specific information for an experienced user.

4.1.2 Extended Boolean Approach(s)

As a remedy to the two major problems of the naïve Boolean model pointed out above, it has later been extended with various term weighting schemes. This has enabled it to compute an RSV value for each document based on the weights of the terms and thereby rank documents according to their RSV values. These improved weighted boolean models are discussed under the term "Extended Boolean Model" in the IR literature. Most of these extensions take a probabilistic nature (e.g. the *p-norm* model) due to the way weights are computed and RSV values are computed [Salton et al. 1983]. Losee (1998) has shown, using the ranking provided by individual boolean operators, that the extended boolean model is a special case of probabilistic retrieval.

A form of extended boolean model, reported by van Rigsbergen, uses a structured hierarchy of keywords to allow the user to narrow or broaden the search [Rigsbergen 1979]. Another modification reported in [Rigsbergen 1979] takes into account the actual number of terms the query has in common with a document, which is referred to as the *co-ordination level*, and uses partial ranking by the coordination levels based on a simple matching strategy. However, based on experimental evaluations, the *P-norm* model has been recognised as the best performing extended boolean model [Lee 1995 as cited in de Vries 2000]. Because of its success, features of the *p-norm* model have later been

incorporated in other models, such as in the inference network architecture [Greiff et al. 1997, Losada & Barreiro 1999]. Analytical comparisons of the performance of boolean and term weighting systems can be found in [Losee 1998 & Yang 2000].

4.1.3 The Probabilistic Model

The probabilistic model is based on the principle that given a user query q and a document d_j, the model tries to estimate the probability that the user find the document d_j interesting (i.e. relevant). It assumes that this probability of relevance depends on the query and the document representations only, and that there is an optimal set R_q of documents that maximises the overall probability of relevance to the user. Documents in the set R_q are considered relevant to the query q and others are considered not relevant. The model works by iteratively guessing/estimating the set of relevant documents R_q and thereby improving the guess at each trial (retrieval) [Rijsbergen 1979, Ribeiro et al. 2000].

The probabilistic model represents queries and documents as sets of keywords with binary weights. The probability that the keyword k_i is present in a document randomly chosen from the set R_q is $P_{i,R_q} = P(k_i = 1 \mid R_q)$, and the probability that the keyword k_i is not present in R_q (i.e. it is in $\overline{R_q}$) is $P_{i,\overline{R_q}} = P(k_i = 1 \mid \overline{R_q})$. The similarity of a document d_j to a query q is defined as:

$$sim(q, d_j) = \frac{P(R_q \mid d_j)}{P(\overline{R}_q \mid d_j)}$$

where $P(R_q|d_j)$ is the probability that document d_j is relevant to query q, and $P(\overline{R_q} \mid d_j)$ is the probability that d_j is not-relevant to q. Taking this ratio as the degree of relevance of the document d_j with regard to the query q, the average probabilistic error is minimised [Ribeiro et al. 2000].

By applying *Bayes* rule, assuming the independence of keywords and other order preserving transformations such as *logarithms*, the estimation of similarity measure is simplified to the following classic expression for ranking in the probabilistic model.

$$sim(q,d_j) = \sum_i w_{iq} \times w_{ij} \times \delta_{i/R}, \quad where \quad \delta_{i/R} = \ln\left(\frac{p_{i.R}}{1-p_{i.R}}\right) + \ln\left(\frac{p_{i.\bar{R}}}{1-p_{i.\bar{R}}}\right) \text{ and } w_{ij}, w_{iq} \in \{0,1\}$$

[Rijsbergen 1979, Chapter 6][Ribeiro et al. 2000, Fuhr 1992]

Advantages and Disadvantages

Probabilistic models are efficient to implement, more effective than boolean queries (exact matching), have a sound theoretical basis [Croft & Turtle 1992] and independent of domain [Croft & Thompson 1987]. One major obstacle with probabilistic IR models is that of finding methods for estimating the probabilities used to evaluate the probability of relevance that are both theoretically sound and computationally efficient [Crestani et al. 1998]. For the simplicity, term independence assumption is made in practice.

4.1.4 Vector Space Model (VSM)

The VSM model is based on a spatial representation of both documents and queries in a multidimensional space defined by the entire set of terms used in the collection. Each term has its own dimension in this space (i.e. distinct keyword vectors k_i of the keywords define the space) in which queries and documents are represented as points or vectors [Salton 1971]. These keyword vectors are assumed to be pair-wise orthogonal, i.e. $i=j \Rightarrow k_i.k_j = 0$, implying that they occur independently within documents and queries. A weight associated with each keyword (computed according to the *tf-idf* weighting scheme) expresses the significance that the keyword has in synthesising the information content of the document. Given the sets of weights w_{ij} and w_{iq}, the weighted query and document vectors q and d_j are represented by $q = (w_{1q},w_{2q},\ldots,w_{tq})$ and $d_j = (w_{1j},w_{2j},\ldots w_{tj})$, where t is the total number of keywords in the system, and each w_{ij} (w_{iq}) weight is associated with a k_i vector. Similarity between a document and a query representation (vectors) depends on how close the two vectors are in the t-dimensional space. A document that is very similar to the query tends to have its document vector at a small angle to the

query vector, while documents less similar to have larger angles. The similarity, therefore, is usually computed as the *Cosine* angle between the document and query vectors or variants of the *Cosine* angle such as *Dice* and *Jaccard* functions.

The most well-known and well-studied implementation of the VSM model is the SMART system developed by Salton [Salton 1971]. Later work on the VSM model has been in the direction of finding better similarity computation, better weight computations, and efficient data structures and algorithms for using VSM model in large-scale document collections. A more elaborate adoption to the vector retrieval model has been to assume term dependency [Losee 1998].

Advantages and Disadvantages

Vector space model is similar to the probabilistic model in many ways, except that it lacks a sounder theoretical base [Croft & Turtle 1992]. Also, VSM ranks documents according to a measure of similarity with the query, instead of probability of relevance to the user's information need as does by probabilistic models. Although the vector model has been criticised as an "ad hoc" model, it is one of the few most influential classical IR models which has been well-studied and well-accepted.

4.1.5 Latent Semantic Indexing Model (LSI)

The Latent Semantic Indexing (LSI) [Deewester et. al. 1990] model uncovers truly orthogonal basis axes or factors for indexing. It attempts to explicitly model the interrelationships between terms using the truncated Singular Value Decomposition (SVD) and exploit this to improve retrieval. SVD is a mathematical technique that computes the singular vectors so that the vector corresponding to the largest singular value accounts for the direction of maximum variation in the data.

In the LSI model, the *term frequency* by *document* matrix (X) is decomposed by using SVD into three components, USV^T, where U and V are orthogonal matrices containing the

left and right singular vectors of X, and S is a diagonal matrix of singular values of X. These matrices reflect a breakdown of the original relationships into linearly independent vectors of factor values. The rows of U and V are considered the *term* and *document* vectors respectively [Dumais et al. 1991]. These matrices are then truncated to the columns containing the k largest singular values, i.e. to use only k factors, allowing both terms and documents to be represented in k-space.

Retrieval proceeds by using the terms in a query to identify a point (Vector) in the k-space. One way to achieve this is by representing the query q as a vector in k-dimensional space by $\hat{q} = q^T U_k S_k^{-1}$; here q is simply the (column) vector of words in the users query multiplied by the appropriate term weights. $q^T U_k$ represents the sum of the k-dimensional term vectors, and the right multiplication by S_k^{-1} differentially weights the separate dimensions. Thus the query vector is located at the weighted sum of its constituent term vectors [Deerwester et al. 1990]. The similarity between the query and document vectors is computed by the *cosine* of the angle between the two vectors as in VSM.

One advantage of using SVD in the LSI model is that the representational power can be controlled by choosing the number k of dimensions. In addition, it has the preferred property (in IR) of representing both documents and terms simultaneously in the same space. Moreover, the truncated SVD carries out dimensionality reduction. The LSI model partially overcomes the problem of variability in human word choices by automatically organising objects into a "semantic" structure [Dumais et al. 1991]. The truncated SVD, in one sense, captures most of the important underlying structure in the association of terms and documents, yet at the same time removes noise or variability in word usage that plagues word-based retrieval methods. Intuitively, since the number of dimensions k, is chosen to be much smaller than the number of unique terms m, minor differences in terminology will be ignored. Terms, which occur in similar documents, for example, will be near each other in the k-dimensional factor space even if they never co-occur in the

same document. It has been shown [Berry et al. 1995] that these statistically derived conceptual indices (factors/singular vectors) are more robust indicators of meaning than individual terms.

4.2 BAYESIAN NETWORK MODELS

A Bayesian network is a directed acyclic graph that compactly represents a probability distribution [Sahani et al. 1998]. In such a graph, each random variable X_i is denoted by a node. A directed edge between two nodes indicates a probabilistic influence (dependency) from the variable denoted by the parent node to that of the child. Consequently, the structure of the network denotes the assumption that each node X_i in the network is conditionally independent of its non-descendants, given its parents (naïve Bayesian approach).

Document classification and filtering have been the major areas in IR to which the Bayesian network model has been successfully applied [Ribeiro et al. 2000, Sahani et al. 1998, Yang et al. 1998]. The Bayesian classifier is simply a Bayesian network with a node C representing the class variable and a node X_i for each of the features (keywords). Given an instance $\mathbf{x}$ (i.e. given values $x_1, x_2, \ldots x_n$ of the feature variables), the probability $P(C=c_k \mid X=x)$ for each possible class c_k (i.e. how much similar a class is to a given set of terms) can be calculated according to the Bayes theorem as:

$$P(C = c_k \mid \mathbf{X} = \mathbf{x}) = \frac{P(\mathbf{X} = \mathbf{x} \mid C = c_k)P(C = c_k)}{P(\mathbf{X} = \mathbf{x})}$$

The assumption of the conditional independence of features given the class variable simplifies this to $P(\mathbf{X} = \mathbf{x} \mid C = c_k) = \prod_i P(X_i = x_i \mid C = c_k)$, the one used in the classic Naive Bayesian Classifier of Good [Good 1965].

More recently, there has been a great deal of work on learning much more expressive Bayesian networks from data. These later approaches relax the restrictive feature

independence assumptions to some extent allowing a limited form of dependence between them. Out of all the varied Bayesian network approaches, "*the inference network model*" and the "*belief network model*" are of important to us due to their application into IR.

4.2.1 Inference Network Model

The inference network model introduced by Turtle and Croft [Croft & Turtle 1992] associates random variables to the keywords, documents and to the user queries, and represents them as nodes of a Bayesian network. A random variable associated with a document d_j represents the event of observing that document. The observation of the document d_j induces a belief upon the keywords of that document. Edges directed from a document node to its keyword nodes indicate that the observation of the document yields improved belief on its keyword nodes. A random variable associated with the user query models the event that the information request specified by the query has been met. The belief in this (query) node is a function of the beliefs in the nodes associated with the query terms. Thus, edges are directed from the keyword nodes to the query node. The following figure illustrates an inference network for a user query q.

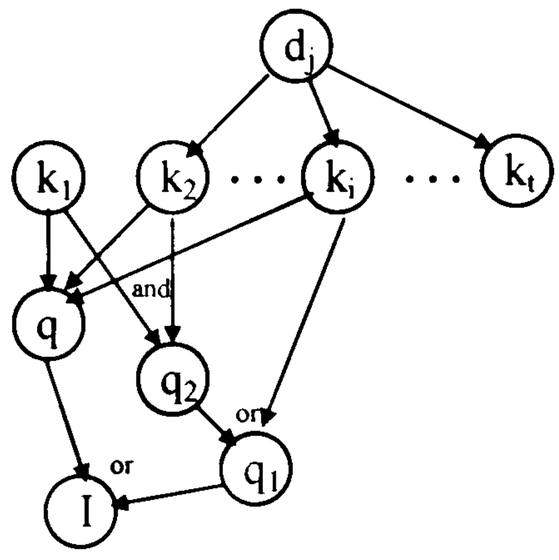


Figure 4.1 : Basic Inference Network Model
(Reproduced from [Ribeiro et al. 2000] with permission © Springer-Verlag 2000)

The additional query related nodes q_1 and q_2 are used to model (alternative) Boolean formulations if any additional information (e.g. in the form $q_1 = ((k_1 \wedge k_2) \vee k_i)$) is available in the query. In this case the new user information need "*I*" is supported by these additional nodes.

In the inference network, all random variables (i.e., d_j, k_i, and q) are binary. The ranking of a document d_j with respect to a query q is a measure of how much evidential support (belief) the observation of d_j provides to the query q. Such ranking is computed as $P(q \wedge d_j)$, where q and d_j are short representations for $q = 1$ and $d_j = 1$, respectively. In general, the ranking is obtained by basic conditioning and application of Bayes' rule as given below.

$$P(q \wedge d_j) = \sum_{\forall k} P(q \mid k) \times P(k \mid d_j) \times P(d_j)$$

$$P(\overline{q \wedge d_j}) = 1 - P(q \wedge d_j)$$

The variable k is a short representation for the state of the k_i variables. Notice that, $P(q|d_j \wedge k) = P(q|k)$, because the k_i nodes separate the query node q from the document node d_j. Also, the notation $\overline{q \wedge d_j}$ is a short representation for $\neg(q \wedge dj)$.

The instantiation of a document node d_j (i.e., the observation of the document) separates its children keyword nodes making them mutually independent (see Bayesian theory for details). Thus, the degree of belief asserted upon each keyword node k_i by instantiating the document node d_j can be computed separately. This implies that the probability $P(k|d_j)$ can be computed in product form as:

$$P(k \mid d_j) = \prod_{\forall i \mid g_i(k)=1} P(k_i \mid d_j) \times \prod_{\forall i \mid g_i(k)=0} P(\overline{k_i} \mid d_j) \; ;$$

where $g_i(k)$ returns the state (0 or 1) of the k_i node according to k, and $P(\overline{k}_i \mid d_j) = 1 - P(k_i \mid d_j)$.

The inference network can cover a wide range of useful information retrieval ranking strategies through proper specification of the probabilities $P(q|k)$, $P(k_i|d_j)$, and $P(d_j)$. Turtle and Croft [Croft & Turtle 1992] show how the inference network model can be used to subsume both the probabilistic and the boolean models, and also how it can be used to represent *tf-idf* based ranking strategies. Their evaluation results have shown that given equivalent document representations and query forms, the inference network model performs better than conventional probabilistic models. Tzeras and Hartmann

[Tzeras & Hartmann 1993] show that the network can be applied to automatic indexing in large subject fields with encouraging results. The best operational example for the inference model is the INQUERY retrieval system of Callan et al. [Callan et al. 1992].

4.2.2 Belief Network Model

The belief network model is derived from a probabilistic argument based on a clearly defined sample space (universe of discourse) U. It is the set of all the keywords that makes up U in the context of IR. A random variable is defined for each keyword k_i (also denoted by k_i). A subset u of U ($u \in U$) is a concept in U. Queries and documents are represented as subsets of keywords in U (i.e. concepts in U). The probability P(c) associated with a generic concept c in the space U is defined as $P(c) = \sum_{u} P(c \mid u)P(u)$, where P($u$) is a prior probability associated with each concept u in U. P($c|u$) defines a coverage relationship between the concepts c and u in the space U and thus interpreted as the degree of coverage of the space u by c. The ranking computation is based on interpreting the similarity between a document d_j and the query q as a coverage relationship between the concepts d_j and q. The degree of coverage of the concept d_j, given the concept q, is given by the probability P($d_j|q$) = P($d_j \wedge q$)/P(q) (by Bayes Law). Here P(q) is a constant for all documents and P($d_j \wedge q$) can be computed by the expression $P(d_j \wedge q) = \sum_{u} P(d_j, q \mid u)P(u)$. Instantiation of the keywords k_i (the root nodes) d-separates q and d_j making them mutually independent, i.e. P(d_j,$q|u$)=P(d_j $|u$) P($q|u$) [Ribeiro et al. 2000].

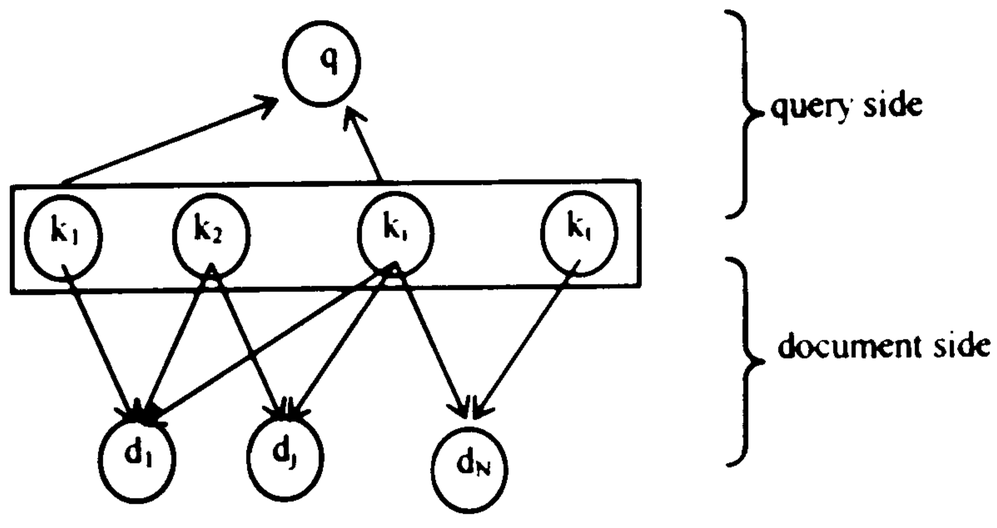


Figure 4.2 : Belief Network for a Query q given by the Keywords k_1 and k_i
(Reproduced from [Ribeiro et al. 2000] with permission © Springer-Verlag 2000)

Ribeiro [Ribeiro et al. 2000] used the generic expression $P(d_j \mid q) = \eta \sum_u P(d_j \mid u)P(q \mid u)P(u)$ for calculating the rank of a document d_j with regards to a query q, and represented the three classic IR models (Boolean, VSM and Probabilistic) in the framework of the Bayesian network. η, in this expression is a normalisation constant.

4.3 KNOWLEDGE-BASED TECHNIQUES

Knowledge-based systems attempt to incorporate semantic domain knowledge into IR systems to better represent and understand the meaning of concepts that the user is searching for. Some systems make use of external knowledge sources (commonly available sources or specially tailored for the task) such as *Thesauri* or *Dictionaries*. Some others make use of *Rule Bases* within the system to represent the knowledge. Yet another class of systems learns knowledge on the fly and represents it in different forms of hierarchical or/and semantic network structures [Ginsberg 1993].

[Chen et al. 1993] gives a good review of the attempts made to capture expert domain knowledge for information retrieval. CoalSORT [Monarch & Carbonell 1987], a knowledge-based interface, facilitates the use of bibliographic databases in coal technology. A frame-based semantic network, representing an expert's domain knowledge (in its cognitive organisation), embodies the system's intelligence. The GRANT system, developed by Cohen and Kjeldsen [Cohen & Kjeldsen 1987], is an expert system for finding sources of funding for given research proposals. Its search method, constrained spreading activation in a semantic network, makes inferences about the goals of the user and thus finds not only the information that the user explicitly requests but also the information that is likely to be useful. Shoval's expert system [Shoval 1981] for suggesting search terms is composed of two components: (1) a *knowledge base*, represented as a semantic network in which the nodes are words, concepts, or phrases, and links express the

semantic relationships between the nodes; and (2) a set of rules or procedures which operate upon the knowledge-base analogous to the decision rules of work patterns of the information specialist. Fox's CODER system [Fox 1987] consists of a thesaurus that was generated from the "Handbook of Artificial Intelligence" and Collin's Dictionary. CANSEARCH [Pollitt 1987] is a thesaurus presented as a menu for selecting terms for formulating queries by browing the through the menu. The "Intelligent Intermediary for Information Retrieval" (I^3R) developed by Croft and Thompson [Croft & Thompson 1987, Croft et al. 1989], consists of a group of "*experts*" that communicates via a common data structure, called the *blackboard*. The system consists of a user model builder and a query model builder, a thesaurus expert, a search expert (suggesting statistics-based search strategies), a browser expert and an explainer. Chen and Dhar [Chen & Dhar 1991] incorporated a portion of the Library of Congress subject headings into the design of an intelligent retrieval system. Their system adopted a branch-and-bound spreading activation algorithm to assist users in articulating their queries.

NLDB [Jacobs 1993], WorldViews [Ginsberg 1993] and JUSTICE [Osborn & Sterling 1999] are a few other knowledge-based implementations in IR. NLDB automatically assigns categories to news stories for dissemination, retrieval and browsing. WorldViews was developed for processing electronic news articles, as well as abstracts of technical reports from Bell Labs and other organisations. JUSTICE is a legal knowledge-based system that can identify heterogeneous representations of concepts across all major Australian Jurisdictions, and some concepts within US and UK cases.

The bottleneck in the design of knowledge-based systems is the (manual) knowledge acquisition process, which demands extensive effort on the part of knowledge engineers, who need to interact with subject experts in order to extract their knowledge and expertise in detail and completeness. In addition, a knowledge-base needs to represent the complete knowledge in the document collection and keep up-to-date with its underlying database.

However, the manual knowledge acquisition approach is not practical mostly due to the lack of human experts in various areas and the difficulty of cooperating with them. A complementary approach to manual knowledge creation is the automatic thesaurus generation. The attempts made in the past for automatic thesaurus generation systems are based mainly on the statistical co-occurrence of word types in text [Chen et al. 1993]. Machine learning is a newer approach that has been recognised as one of the promising techniques for automatically acquiring knowledge from examples.

4.4 Logical Approach to IR

The logical approach to IR is based on defining and using non-classical logic to provide a representation of information and its semantics. The model assumes that the queries and documents can be represented effectively by logical formulas. In order to retrieve a document, the IR system has to infer the formula that represents the query from formulae that represent the document. This inference process works upon the information present in the document itself and external information such as user knowledge.

The logical model introduces logic to cope with the intrinsic uncertainty present in IR. The logical uncertainty principle proposed by van Rijsbergen was the first attempt to make an explicit connection between non-classical logics and IR uncertainty modelling. It says:

> *"Given any two sentences x and y, a measure of the uncertainty of y→x related to a given data set is determined by the minimal extent to which we have to add information to the data set, to establish the truth of y → x"* - van Rijsbergen 1979.

Application of this principle requires the combination of a logic formalism and uncertainty theory. The generality of the principle has made the choice of the appropriate logic and uncertainty mechanisms a main research theme in logical IR modelling. Various Logical approaches can be described under two broad classes: (1) approaches that make use of non-classical logic combined with a theory of uncertainty (Logical Models); and (2) approaches that use a theory of uncertainty defined in terms of a non-classical logic (Logical

Uncertainty Model). An extensive review of logical models and various uncertainty theories used in logical approaches can be found in [Crestani & Lalmas 2000].

4.5 Soft Information Retrieval

Soft Information Retrieval refers to a class of approaches that aim at applying techniques for dealing with vagueness, uncertainty and imprecision of the IR process. The principle methodologies and techniques of Soft Computing include fuzzy logics, neural networks, probabilistic reasoning, evolutionary computing, chaotic computing and other machine learning theories. The major contributions (in IR), however, according to the literature, have been from connectionism (spreading activation (SA) and neural networks (NN)). In the following we give a review of a selected set of models that we thought significant, due to the space limitations. They cover the use of genetic algorithms, fuzzy logic and connectionism (SAs and NNs) in IR.

4.5.1 Genetic Algorithms in IR

Genetic algorithms (GAs) were developed by John Holland [Holland 1975] based on the principle of genetic evolution. It belongs to the class of stochastic optimisation methods, and provides self-adaptability properties. The GA works within a space of possible solutions (individuals) to a given problem. An individual or a potential solution is represented by a set of genes. In a GA, a set of starting potential solutions (initial population of individuals) undergoes an evolution through a sequence of operations of reproduction, crossover and mutation [Goldberg 1989]. During this process, these individuals strive for survival through the selection scheme that is biased towards selecting fitter individuals and producing the individuals for the next generation. After some number of generations, the program converges to the best individual representing the optimum solution.

In IR, the GA is mainly used for feature subset selection for finding either an optimal query or optimal document representations. An initial population of documents or query

representations is allowed to compete with each other based on a fitness function to obtain a better collection of concepts (keywords) for indexing (representation) [Chen 1995, Gordon 1988, Vrajitoru 2000, Boughanem et al. 2000]. In the design of document representation optimisations, a gene represents a keyword, a document's list of keywords represents an individual (chromosome) and a collection of documents initially judged relevant by a user represents the initial population. Based on Jaccard's score matching function (fitness measure), the initial population evolves through generations and eventually converges to an optimal population - a set of keywords that best describes the documents. These "optimised" keywords can then be used by an IR system to suggest relevant documents to the user. This process can be repeated as a relevance feedback mechanism by using the user decisions on the suggested documents to create an initial population, and re-apply the GA processes until the search is completed or user decides to stop. In contrast, Boughanem et al. [Boughanem et al. 2000] presents a query formulation technique in which the GA generates several queries that explore different areas of the document space. The above-mentioned designs make use of only binary term weights, but designs that make use of non-binary term weightings do exist. For instance, [Kraft et al. 1994, Petry et al. 1993] applied genetic programming to a weighted Boolean query formulation to improve search performance. [Yang & Korfhage 1993] applied the GA to query optimisation by re-weighting the document term indexing without expanding the query.

Moreover, Yu and Liddy [Yu & Liddy 1999], and Chen and Kim [Chen & Kim 1994] report hybrid approaches by combining the GA with neural networks (NNs). Yu and Liddy's model uses the GA as a mechanism to select the best (or optimal) feature set for NNs. They used the *Baldwin effect* [Baldwin 1896] to guide and improve the GA-based evolution of the feature subsets. NNs constructed dynamically corresponding to optimal feature sets obtained by the GA evaluate the classification performance of the feature sets. In Chen and Kim's model (GANNET) [Chen & Kim 1994], concepts optimised by the GA

are used to perform concept exploration in a large network of related concepts through the Hopfield Network's parallel relaxation procedure. During concept exploration, the Hopfield network produces other relevant concepts (new genes suggested by nature). These are then included in the GA for the next concept optimisation and the process repeated until there is no further improvement.

The work described above provides evidence of the viability of the GA for deriving good feature sets for document or query representations. Specific results of these experiments include: (1) the choice of the fitness function has a significant effect on performance [Petry et al. 1993], (2) larger populations, in general, tend to improve the effectiveness of retrieval [Vrajitoru 2000] and (3) queries converge to their relevant document representations within a few (~six) generations [Boughanem et al. 2000].

4.5.2 Fuzzy Information Retrieval Models

A fuzzy set allows the characterisation of its elements by means of the concept of 'graduality', which lets a class of elements be described in more precise terms when the nature of the elements themselves do not support sharp boundaries of memberships [Koczy & Gedeon 2000]. This key feature has influenced IR researchers to employ fuzzy set theory to model vagueness and imprecision that exists at different levels in the IR processes, mainly to reduce the incompleteness and deal with the imprecision of the indexing process, to manage the user's vagueness in queries, and to deal with discriminated answers by allowing partial relevance of the documents. The main applications of the theory of fuzzy logic found in the IR literature include: extending the boolean model by flexible indexing and representation of documents and query language (fuzzy indexing); defining associative mechanisms such as fuzzy thesauri and fuzzy clustering; defining knowledge-based models of IR; and defining fuzzy measures for evaluating the effectiveness of IRS in terms of recall and precision. An extensive review of the application of fuzzy set theory to IR can be found in [Bordogna & Pasi 2000].

4.5.3 Spreading Activation and Connectionism

Spreading activations of nodes through links in a network of nodes is the basis for connectionism or in particular neural networks (NNs). A NN is considered as an improved form of the basic SA model due to its use of a non-linear activation function and a learning procedure to modify the weights on links so that the spreading of the activations over the network reflects some desired patterns [Crestani 1997]. This section reviews applications of both the basic SA model and NNs in IR. A more comprehensive survey of the literature, from which some of the present material is drawn, may be found in [Crestani 1997].

4.5.3.1 Spreading Activation (SA)

The SA model is inspired by the mechanisms of human memory. In the "original" SA model, nodes in the network model objects or features of the real world and links model relationships between nodes as in a semantic network. A link usually has a direction, a label and a weight assigned according to a specific direction. The connectivity pattern between nodes reflects the relationships between objects or between objects and their features. Processing is done by iterating a sequence of actions (a "pulse"). A pulse in the basic SA model consists of an optional preadjustment phase, a spreading phase and an optional postadjustment phase. The preadjustment and postadjustment phases are used to avoid retention of activation from previous pulses by applying some form of activation decay in the active nodes. The spreading phase consists of a number of passages of activation waves from one node to all other nodes connected to it. A simplest and frequently used form of the spreading activation formula to compute the activation level of a node is the sum of weighted outputs (activities) from incoming (connected) nodes ($I_j = \sum_i O_i w_{ij}$). The output of the node is determined by a threshold function such as a step function, a linear function or a sigmoid function. It is this new output that is fired (spread) to the connected nodes at the next iteration.

Since the spreading of the activations in this pure SA model is not controlled, they tend to spread all over the network, and the model does not use the rich semantics (labels) of the associations to help making inferences. The Constrained Spreading Activation (CSA) model, an improved form of the SA model, uses a range of constraints (rules) to restrict the spreading of activations and also to define heuristics to process links differently according to their semantics, distance from initially activated node and connectivity density of the node(s).

Spreading Activation in IR

SA techniques adopted into IR systems differ from the pure SA models mainly by the way activation weights are assigned, and the specific rules (constraints) are used to control spreading of activation over the network and also to embed some form of inference in the process of association retrieval. The SA network in IR is essentially a map of relations between concepts, information items (documents) or any other entities of the problem domain (Figure 4.3). For instance, a node in the network can represent terms (concepts), documents, articles, journals, subject classifications or authors. Links between nodes indicate the associations between them. This representation structure is dependent on the purpose and the domain of the application, as the node and link types are determined based on the data in the target domain.

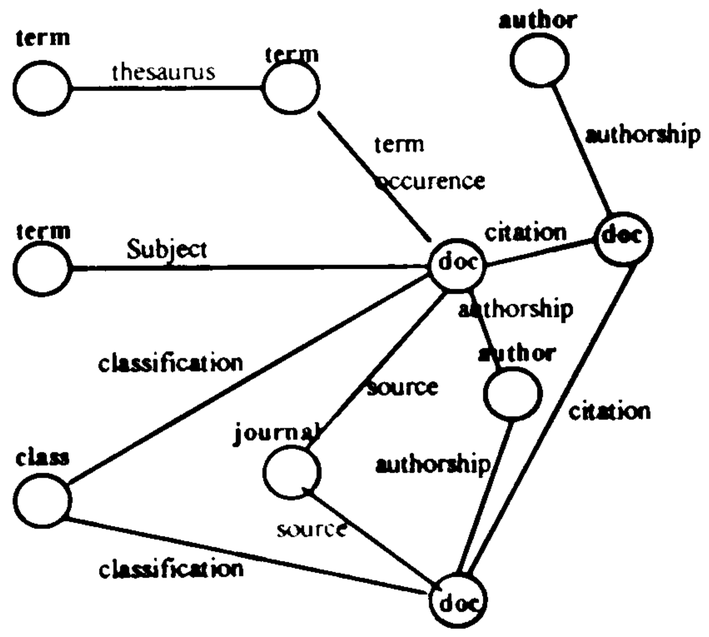


Figure 4.3: A Document Collection Representation in a SA Network
(Reproduced from [Crestani 1997] with permission © 1997 Kluwer)

Nodes (terms, documents etc.) corresponding to a query formulation or result of an earlier search operation are activated by placing a specific activation level on them and the

activations are allowed to spread through the network according to defined constraints (rules). Activation levels of documents at the end of the spreading process (triggered by a given termination condition) are used to compute the relevance levels of each document.

SA has been employed in IR mainly to expand the search vocabulary and to complement the retrieved document sets [Salton & Buckley 1988]. Most designs use hand-made semantic network structures of nodes and links to represent the objects (documents and concepts) and associations between them, while few others make use of static or dynamic thesauri to automatically construct the network representation structures. For instance, the I^3R [Croft & Thompson 1987], GRANT [Kjeldsen & Cohen 1987] and Preece's [Preece 1981] models use hand-made semantic networks and, the AIRS [Kimoto & Iwadera 1990] and Shoval's model [Shoval 1981] use thesauri for automatic network constructions. Some of these models incorporate feedback mechanisms into SA. Feedback can arrive from another process or can be user evaluations of the activation levels of some nodes. It is also possible to enable the user to modify the activation of some nodes according to his requirements or to indicate some particular spreading path so that activation can follow based on directions given by the user. A brief summary of these models is given below, and the interested reader is referred to [Crestani 1997] for an extensive review.

One of the pioneers to use associative search by SA is Preece (see his PhD thesis [Preece 1981]). He showed how the classical IR approaches such as boolean and vector models can be explained in terms of different SA processing techniques on a network representation of the document collection. In addition, using relevance feedback, he has showed how SA can be used for automatic classification, indexing and for concept building.

Shoval [Shoval 1981] reports an implementation of an interactive query expansion system using a thesaurus-based semantic network (a knowledgebase). In addition to the common

types of relations used by thesauri (such as hierarchical relationships and synonymous relationships), he used two special types of links to combine source/generic words to multi-words (as of "information" and "systems" to "information systems") and to link a word with a component of its meaning (as of "business" to "organisational area"). The model works by expanding user (query) terms along the semantic network links in order to suggest better representative set of terms for the query. The direction of search (for terms) is guided by interactive feedback coming from the user about the relevance of suggested terms. The most advantageous feature of this model is its automatic construction of the network representation structure (given a thesaurus). The model is recognized to have the strength to operate independently of specific domain knowledge if based on a generic thesaurus. A disadvantage is the excessive amount user interventions that it rquares.

The GRANT system of Kjeldsen and Cohen [Kjeldsen & Cohen 1987] may be the best working example for the application of constrained SA in IR. It operates on a semantic network of knowledge about research proposals and funding agencies to suggest funding agencies for research topics. A specific feature of GRANT is that it uses a large number of path constraints in the form of "path endorsements" in addition to other commonly used constraints. These are essentially inference rules created for the particular application for which GRANT was designed. The model works by first activating nodes correspond to a query (one or more research topics or funding agencies) and then spreading them according to defined constraints. Experimental results (in its own domain) have shown better performance of GRANT over simple keyword-based search systems. One of the limitations of GRANT however, is that it needs the parameters of the path endorsements to be well tuned to the domain in which the system will operate. This is a difficult task which requires an in-depth analysis of the domain determining the appropriate concepts and relationships to build in the network, and preferences to give paths of activation spreading over them.

The I^3R system of Croft and colleagues [Croft & Thompson 1987, Croft et al. 1989] is an implementation of a document retrieval model (by plausible inference) as a form of constrained SA. The system uses the domain knowledge represented in a semantic network to infer concepts that are related to those mentioned in the query. A particular processing technique of interest used in I^3R (out of several others) uses top-ranked documents from a probabilistic search as its starting point, and spreads the activations of the corresponding nodes initially through strongest plausible relationships between documents (nearest neighbours and citations) and then through nearest neighbours only. Weights on the links (specified as "credibility" values associated to inference rules representing the existence relationships between nodes) are used to evaluate the activation levels of the nodes. This implementation is considered to be of the type "multi source of evidence" in which the relevancy of a document is supported by many different clues.

Kimoto and Iwadera's AIRS system (Associative Information Retrieval System) [Kimoto & Iwadera 1990] incorporates SA in a dynamic thesaurus. It produces "term information" based on the user's interest in his sample of relevant documents and they (term information) are used to construct a dynamic thesaurus, a network of nodes and links. Selection of terms to build the dynamic thesaurus is done based on their frequencies and locations in a set of relevant documents provided by the user, and links between term-nodes are created based on co-occurrence of terms. Term information in the dynamic thesaurus is used together with a static thesaurus to generate associated keywords to expand the query at the time of retrieval. The structure of the dynamic thesaurus resembles more an associative network than a thesaurus and differs from the classical connectionist network architecture. The relations between terms (link types) are of a single type and there is no activation function, no learning procedure and the weights are on nodes rather than on links.

Fahlman [Fahlman et al. 1981] encoded a semantic network as a massively parallel network of simple processing (hardware) elements in his NETL system. It uses marker passing to perform simple inferences based on set intersection and transitive closure operations. The intersection operation locates items that share a set of properties whereas the transitive closure operation handles inheritance as well as closure of relations like "*part-of*". These operations are performed in parallel and allow the system to conduct a very fast search. The use of small numbers of discrete markers (Boolean conditions) for communication between network elements and the inability of the network elements to detect more than just the presence or absence of a marker in the input have been pointed as drawbacks of the model that limits its processing power.

Drawbacks of the SA Approach

SA is based on a networked representation of objects and relations between objects in the particular domain of the application. Success of the SA process critically depends on the "representativeness" of the representation. Building networks to represent underlying semantics of the data is known to be a difficult problem. If applying in a specific domain this requires in depth application domain knowledge that only experts in the application domain can provide. One alternative to manual construction is to use an existing thesaurus or a knowledge base on the same domain for automatic construction of the representation network. However, knowledge bases do not exist for many domains. Another alternative is to use machine learning for automatic creation of the representation. A second drawback is the high computing power needed for processing large network representations. These problems have limited the application of SA to smaller scale research prototypes rather than commercial systems.

4.5.3.2 Neural Networks (NNs) in IR

Artificial neural networks, which were inspired by the structures and functions of the human brain, are implementation realisations of connectionist modeling. They attempt to

achieve the primary objective of AI: to develop computational models that perform equally or better than humans in the tasks that humans are good at [Becker 1991]. Formal neurons of an information processing NN represent IR objects such as keywords, references, document excerpts or document classes. Knowledge is represented by a single layer of interconnected neurons (feature maps) or by a multi-layered network. The formal synapses (connections) are implemented through weighted links that represent the relevance levels of the associations that may be learned between the formal neurons.

Based on the learning paradigm used, NNs can be divided into two major categories, namely supervised NNs and unsupervised NNs. Supervised learning is used mainly in multi-layered NNs to implement spreading activation searches [Salton & Buckley 1988] in which the learning is guided by desired outputs of the training inputs. Spreading of the activation is via the excitatory and inhibitory links between nodes. Unsupervised learning procedures, on the other hand, use a "*winner-take-all*" approach. They rely only upon local information and internal control, and learn by capturing regularities in the input patterns. Self-Organizing Maps (SOM) and the Adaptive Resonance Theory (ART) networks are the most popular unsupervised NN architectures. In this section we summarise some of the innovative research in the application of NNs in IR. A good survey of the literature from which some of the present material is drawn can be found in [Crestani & Pasi 1999].

Supervised NN Models

One of the pioneer applications of NNs to IR is Mozer's Parallel Distributed Processing (PDP) Model [Mozer 1984]. In this model, each document and descriptor was represented by a node (Figure 4.4). The activation level of a document node indicates the system's belief in the relevance of the document. The weights between nodes were binary. Each document was connected with inhibitory links (with a constant weight) to all other documents allowing documents to compete each other. This helps to keep their activation levels under control during retrieval phase and to control the level of associativity among

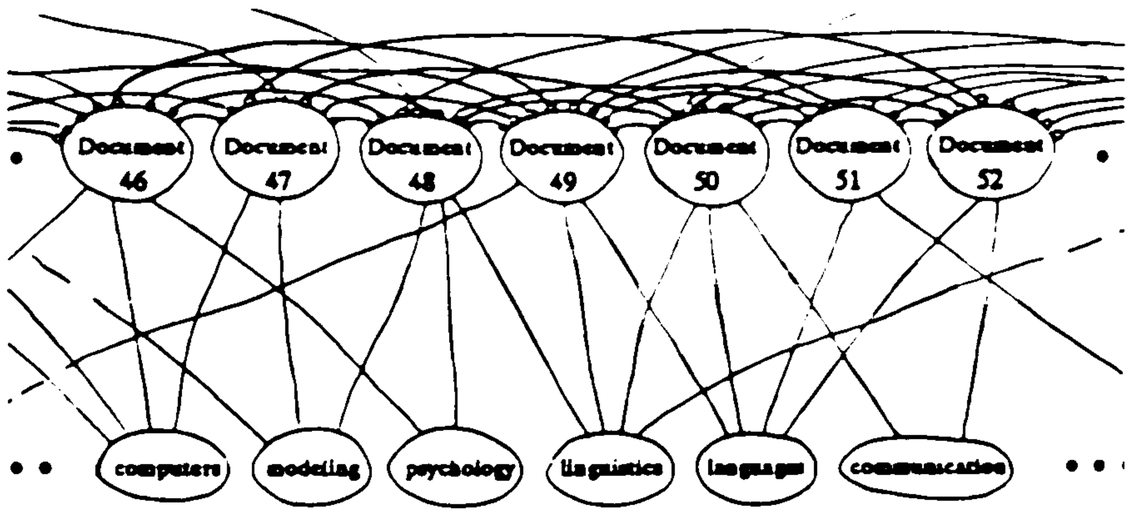


Figure 4.4 : Mozer's Inductive IR Model
(Reproduced from [Mozer 1994] with permission from M.C. Mozer)

them. Activation of a set of (initial) descriptors causes a set of new descriptors to be activated. These descriptors in turn activate a set of new documents, and also reinforce the activation of the already active ones. The result is that it suggests other potential descriptors that may help the search and other documents (other than the ones activated by the initial query descriptors) that may be relevant to the query. Indexing documents with a highly correlated set of descriptors help the model to gain semantic relationships between terms through their overlapping relationships with documents. However, the use of binary weights on links (all descriptors are of same importance), lack of a learning procedure (performance remains the same over time) and lack of links among descriptors (ignores direct inter-term semantics) have been identified as drawbacks of the model. Feasibility of applying this model to larger collections was later demonstrated by the experimental results of Bein and Smolensky [Bein & Smolensky 1988]. The model was further improved by incorporating relevance feedback into it by Hingston and Wilkinson [Wilkinson & Hingston 1991]. A similar approach (to Mozer's) for document retrieval was implemented by Stanfil and Kahle [Stanfil & Kahle 1986] on a Connection Machine (CM[1]).

Belew's AIR [Belew 2000] is considered the most influential work in employing NNs into IR. It is a three layer NN of authors, index terms (descriptors) and documents. A document is connected to each of its descriptors and also to each of its authors via two links

[1] A massively parallel supercomputer with 65,536 processors

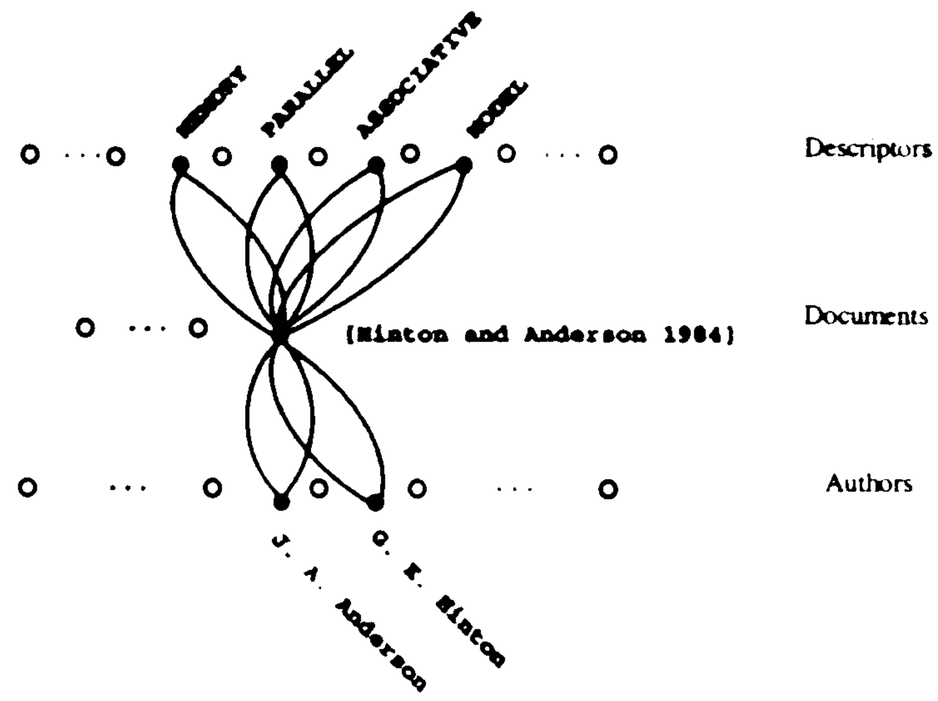


Figure 4.5 : Network Representation of a Single Book in Belew's AIR Model
(Reproduced from [Belew 2000] with permission © Cambridge University Press 2000)

(bidirectional). Weights of the links, initially computed based on an inverse frequency weighting scheme, are modified on the fly from the first user session by means of relevance feedback. Thus, as in our model (Chapter 7), AIR's representation results from the combination of two completely different sources of evidence: the word frequency statistics underlying its initial indexing and the opinions of its users. The sum of the weights on all links going out from a node is forced to be a constant (one). Figure 4.5 shows part of the network corresponding to the representation of the book "Parallel Models of Associative Memory" by G. E. Hinton and J. A. Anderson. The initial network is constructed from the superposition of many such document representations. The model works according to the basic SA principal by propagating activations of the nodes corresponding to the features of the query. Response of the system is the set of nodes that becomes active over a certain threshold during this propagation. A learning process, derived from the Hebbian rule and based upon relevance feedback, creates new connections between documents and descriptors resulting in a representation of the consensual meaning of descriptors and documents shared by some group of users [Crestani & Pasi 1999]. This work was carried forward by Rose and Belew (1991), and a hybrid connectionist and symbolic model called SCALIR [Rose & Belew 1991], a legal information system, was built.

Kwok's three-layer network [Kwok 1995] (Figure 4.6) uses a modified Hebbian Learning rule to reformulate probabilistic information retrieval. Nodes in the layers represent

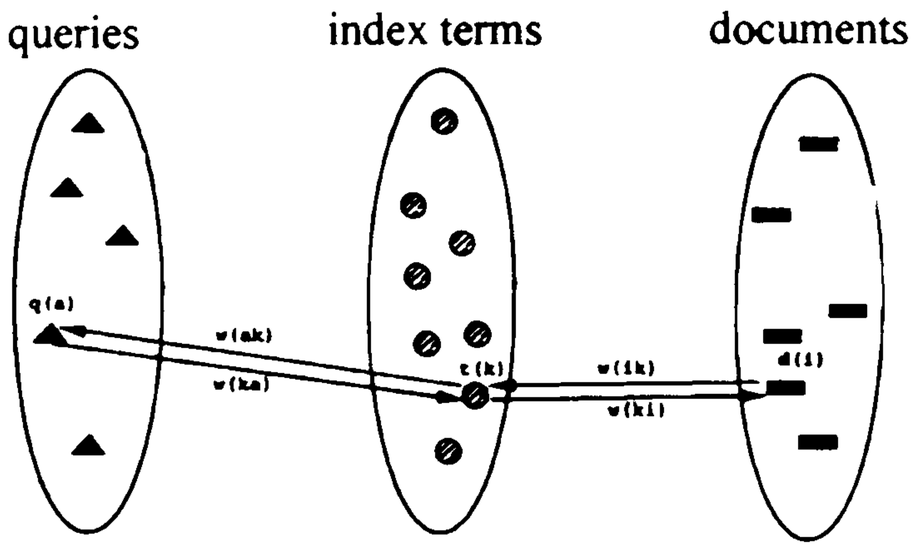


Figure 4.6 : Kwok's Model
(Reproduced from [Kwok 1995] with permission © 1995 by ACM Inc)

queries, index terms and documents respectively. Connections between nodes are bi-directional and asymmetric with no lateral connections. A weight on a connection is computed according to the classical probabilistic IR measure as the probability of presence of an index term given a particular query or document (i.e. w_{ka} and w_{ki}) or as the evidence that if the index term k is isused, it will be dealing with the contents of that query or documents (i.e. w_{ak} and w_{ik}). The weight computation has been criticised as too complex and thus far from reality.

Crestani [Crestani 1993, Crestani 1995] developed an adaptive IR system using a feed-forward NN. Figure 4.7 shows its three components: Query Processor, Matcher and Document Processor. The task of the query/document processor is to transform queries/documents into binary vector representations of diminutions equal to the number of input/output nodes in the Matcher. During training, binary vectors of queries and documents are trained in a 3-layer feedforward NN (the Matcher) using the

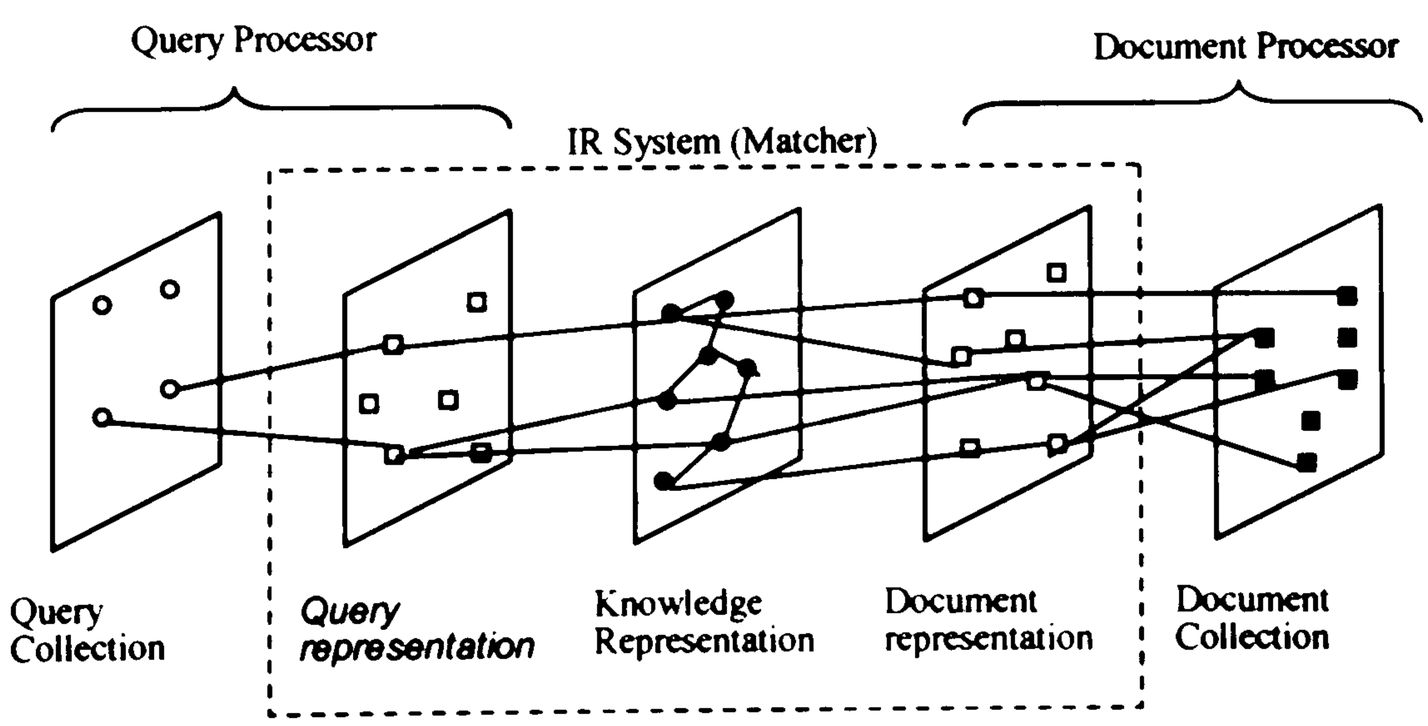


Figure 4.7 : Crestani's Adaptive IR Model
(Reproduced from [Crestani 1997] with permission © 1997 Kluwer)

backpropagation learning algorithm. The hidden layer learns associations between the terms in queries and terms in documents. At retrieval, the NN produces a new query representation on its output layer, better representing the given query according to the application domain knowledge it has learnt at the training phase. A ranked list of documents is then computed (by the Matcher) based on the activations of the output (document) layer nodes using Dice's coefficient. Crestani uses three different learning strategies (total learning, horizontal learning and vertical learning) in his experiments, out of which vertical learning (in which only a subset of the known relevant documents for each query was used for training) has shown best performance, comparatively similar to that provided by probabilistic relevance feedback. This system was later integrated into a more general network model for adaptive IR by Crestani and van Rigsbergen [Crestani & Rigsbergen 1997].

Jung and Raghava ([Jung & Raghavan 1990], as cited in [Crestani & Pasi 1999]), constructed a thesaurus like knowledge representation structure (a pseudo-thesaurus) using a single layer NN, and used it in conjunction with the vector space model to perform the ranking and document retrieval. Terms in the NN are represented as real numbers, and are determined by means of a learning procedure with relevance feedback from past users. The symmetry assumption of the relationships between terms has been identified as a major drawback of this model.

Chen et al [Chen et al 1999] reported a design of a NN for search engines within an index database. The engine uses the network, trained through user feedback, to classify the documents in its internal database, and ranks and returns those classified documents to the users. Any misclassifications reported by the user are fed back to the NN in order to learn the hidden nodes with proper refinements, and a new refined list of documents are returned to the user. Documents are represented as vectors of attributes of some dimension (d) in which each attribute may have a number (N) of discretised values. The network was

designed for learning any collection of documents represented by disjunctions of attributes. The goal of the training process is to find values for the attributes. For each real variable (X_i) at each dimension, N "virtual variables" are introduced (Figure 4.8) according to Maass and Warmuth [Maass & Warmuth 1995]'s virtual variables concept. The objective of the search is to find out those unknown weights for the virtual variables and the unknown threshold of the output layer node. For training the NN, the Elimination and Promotion strategy (of Littleatone's, find reference in Chen et al. 1999) based on relevance feedback is proposed. This strategy, which works well for smaller values of d and N, becomes computationally complex for large values. A three-layer NN version (Figure 4.8), in which the virtual variables are clustered into "blocks" by an additional layer, is proposed as a remedy for the dimensionality problem.

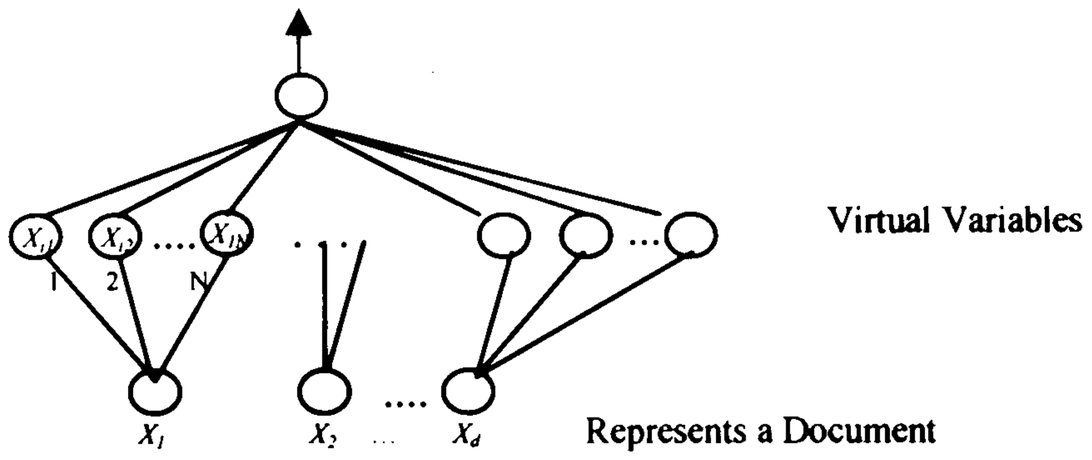


Figure 4.8 : Chen's Search Engine Model
(Reproduced from [Chen et al. 1999] with permission © 1999 IEEE)

A simple three layer backpropagation NN model called COSIMIR was proposed by Mandl [Mandl 2000] to directly calculate and output a similarity value for a given pair of a query and a document representation. Both query and document representations are given as input to the network. It does not use explicit mathematical similarity functions to compute similarity values (RSV) for a query-document pair, and neither does it assume term independency. The activation value of the output neuron (of the output layer) is interpreted as the similarity value (RSV). The high dimensionality of the input (document representation + query representation) is a major drawback of the model. Yang and colleagues [Yang et al. 1998] employed a fast constructive learning algorithm (known as DistAI) for classification in their information learning agents. DistAI adds hidden neurons

one at a time based on a greedy strategy to ensure that the hidden neurons correctly classifies a maximal subset of training patterns belonging to a single class.

Boughanem's "*Mercure*" [Boughanem et al. 2000] is a multilayer network consisting of an input layer, a term neuron layer, a document neuron layer and an output layer (Figure 4.9). A document neuron corresponds to a document and a term neuron to an index term. They are interconnected by bi-directional weighted links that represent indexing links. Activations of the output layer neurons result in a ranked list of documents for a given query. The model has a query optimisation strategy based on relevance feedback. It works

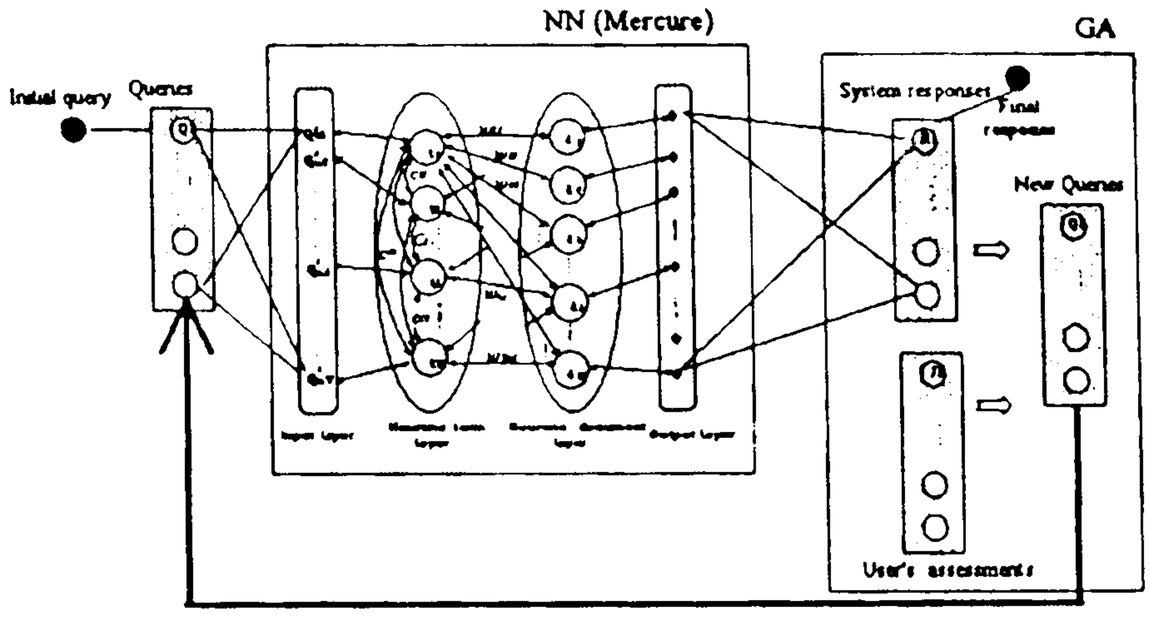


Figure 4.9: Architecture of Mercure
(Reproduced from [Boughanem et.al. 2000] with permission © 2000 Physica-Verlag)

by back-spreading the user judgments (relevance values) from the output layer to the input layer creating a new vector (new reformulated query) at the input layer. An alternative query reformulation strategy reported in the same paper uses the GA for generating a population of queries and then selecting the one that best matches the user's need. In this case, the NN model is used to evaluate each query separately in order to create a final response by merging the results of the best queries. Experiments conducted on rather small document collections have shown improvements over VSM model on single pass search.

Unsupervised Learning Techniques

Self-Organizing Maps

The Self-Organizing Map (SOM) forms a nonlinear projection from a high-dimensional data manifold onto a low-dimensional (usually 2D) grid. A representative "model" (say m_i) of some subset of data is associated with each grid point. This process, which leads to the

computation of an optimal collection of models, is carried out by: (1) approximating the data in the sense of some error criterion; and (2) taking into account the similarity relations of the models. This error criterion also involves the "spatial ordering" of the models in which the most similar models shall be found at adjacent grid points, and the more dissimilar ones shall be located farther away from each other on the grid [Kohonen 1998].

Some pioneer work on unsupervised learning for information processing tasks was reported by MacLeod and Robertson [MacLeod & Robertson 1991]. They designed an unsupervised NN model to perform document clustering by feature extraction. Their experimental results, obtained from clustering and subsequent querying on a classical test collection were comparable to that of hierarchical (sequential) clustering algorithms. Lin et al [Lin et al. 1991] used a Kohonen feature map for clustering 140 documents from the artificial intelligent literature. The documents which were represented by manually encoded 25-dimentional vectors were trained to an SOM of 10 x 14 (i.e.140) neurons.

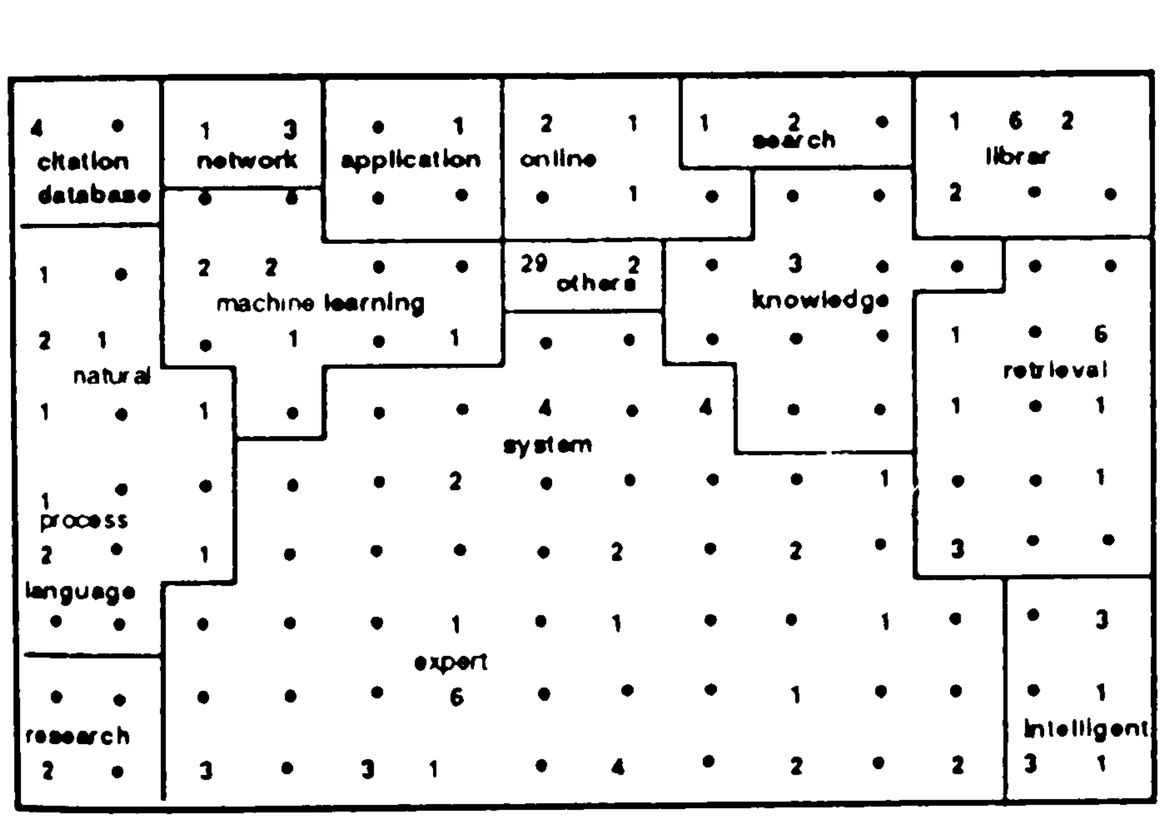


Figure 4.10: Lin's Document Map
(Reproduced from [Lin et al. 1991] with permission © 1991 ACM Inc.)

The numbers on Lin's map (Figure 4.10) shows the number of documents mapped onto each node and the regions show concept areas. The sizes of the regions correspond to the frequency of occurrence of the terms in that area. Despite the success shown by this model, it has not been evaluated either for the quality of clustering or for the retrieval effectiveness. Also the manual selection of index terms is a drawback in its implementation.

Merkl [Merkl 1995a] applied an SOM for clustering textual descriptions of C++ library components. Document vectors (of 498-dimention) were constructed by extracting terms from respective parts of the manual and trained onto a 10 x 10 (a 100-node) SOM (Figure 4.11). The evaluation results have shown that this approach is more effective than the clustering approach of the complete linkage method (often used clustering approach in IR), in the special category of the documents on which it was evaluated.

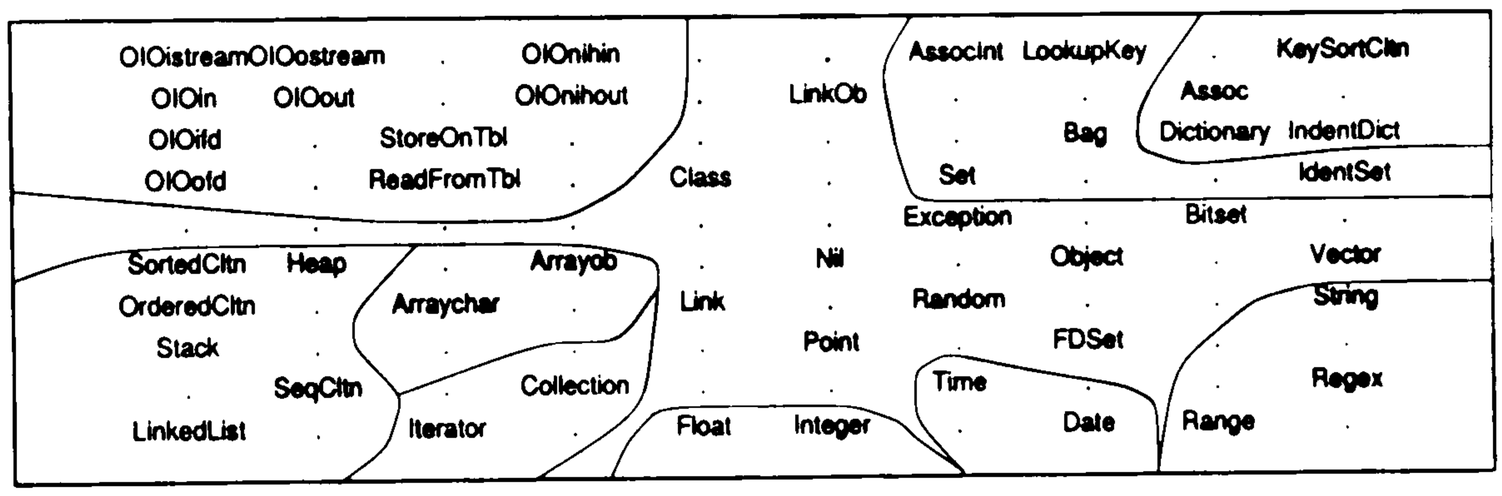


Figure 4.11: Merkl's Map of the NIH Class Library
(Reproduced from [Merkl 1995] with permission from D. Merkl)

Scholtes's "neural filter" and the "neural interest map" [Scholtes 1991] consist of Kohonen feature maps trained with natural language queries (interests) to derive an internal representation of the text. Documents are selected depending on the activity patterns they generate on the network, as they are passed through the trained network (feature map). Evaluation results (on Pravda CD-ROM) have shown better precision and recall figures than those of traditional statistical IR techniques. Troina and Walker [Troina & Walker 1996] constructed another Kohonen feature map for clustering index terms extracted from text documents. This map was used for query expansion and document classification. Terms picked from the same cluster as of a query term are considered similar and are used for query expansion. Documents are classified into subject-related groups based on the analysis of patterns of term occurrences in the document vector. Bordogna and Pasi [as cited in Crestani & Pasi 1999] have proposed a Neural Relevance Feedback model in which a neural network is dynamically constructed based on evidence of user's interests (judgments) on documents retrieved. In this network, neurons represent the most significant terms in the selected documents and the synapses

represent the relations between pairs of terms. At steady state, the terms corresponding to active nodes are considered meaningful and the degrees of the connections between these nodes and those corresponding to the original query terms indicate the strength of the associations between concepts of interest. A rule-based super-structure is then used to expand the original query evaluation with the meaningful terms by avoiding the explicit construction of a new query.

The WEBSOM [Honkela et al. 1998, Kohonen 1998, Lagus et al. 1996] is a considerably large project that uses an SOM to create a document map to support explorative full-text information retrieval and browsing. One of the interesting aspects of this project is its radically different document representation mechanism. Each word is encoded as an *n*-dimensional vector of random-number components to form a sparse representation of words. Then an average context vector ($X_i \in R^{3n}$) is created for each word based on the preceding and following words in the text.

$$X_i = \begin{bmatrix} E\{x_{i-1} \mid x_i\} \\ \varepsilon x_i \\ E\{x_{i+1} \mid x_i\} \end{bmatrix}$$

where $\boldsymbol{E}$ denotes the estimate of the expected value evaluated over the text corpus and ε is a small scalar number

These average context vectors are then used as inputs to an SOM to form a word category map based on the co-occurrences of words in documents. The SOM learning procedure tends to organise strongly related words that have similar contexts close to each other.

The SOM word category map is calibrated after the training process by inputting the X_i s once again to SOM and labelling the best-matching nodes with corresponding x_i parts (words) of X_i s. Each node may be labelled with several words, often synonymous or belonging to the same closed category, thus forming "word categories" in the nodes. Figure 4.12 illustrates how the WEBSOM model works.

The word category map thus created is used for encoding documents by mapping their text, word by word, onto the word category map forming a histogram of the "hits" on it. These

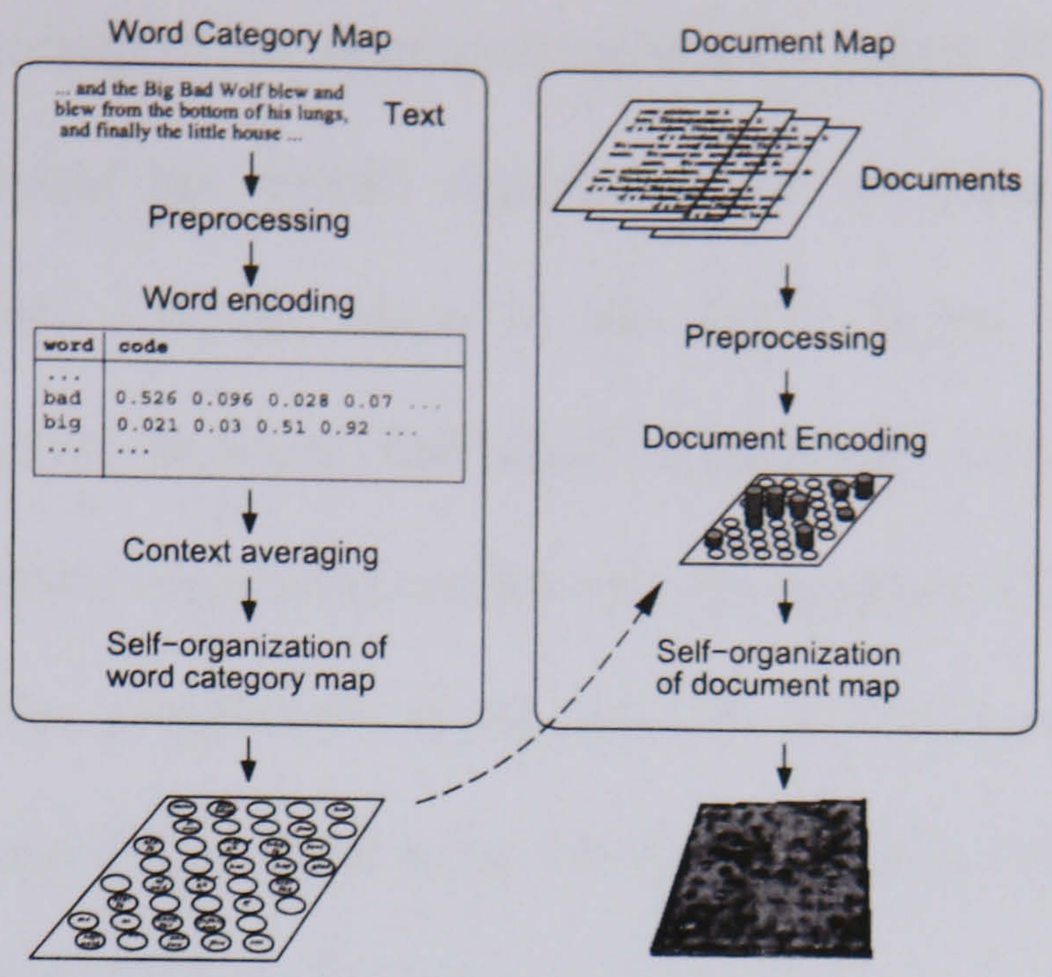


Figure 4.12: The Architecture of the WEBSOM Model
(Reproduced from [Honkela et al. 1997] with permission © 1998 Elsevier)

histogram representations of the documents are trained to a second SOM allowing them to self-organise resulting in a Document Map. A new document (query) can be mapped onto the document display and the position (node) it is mapped can be used as a starting point for exploring related documents in the nearby area [Honkela et al. 1998].

Hierarchical Feature Maps

Realizing the limitations of a 2-dimentional map metaphor for document space visualisation, Merkl and Rauber [Merkl & Rauber 2000] suggested the use of an "atlas" of feature maps each tuned to different portion of the full map at some level of resolution. This was realised using a hierarchical setup of multiple layers where each layer consists of a number of independent SOMs (Figure 4.13). At the root of this structure is only one

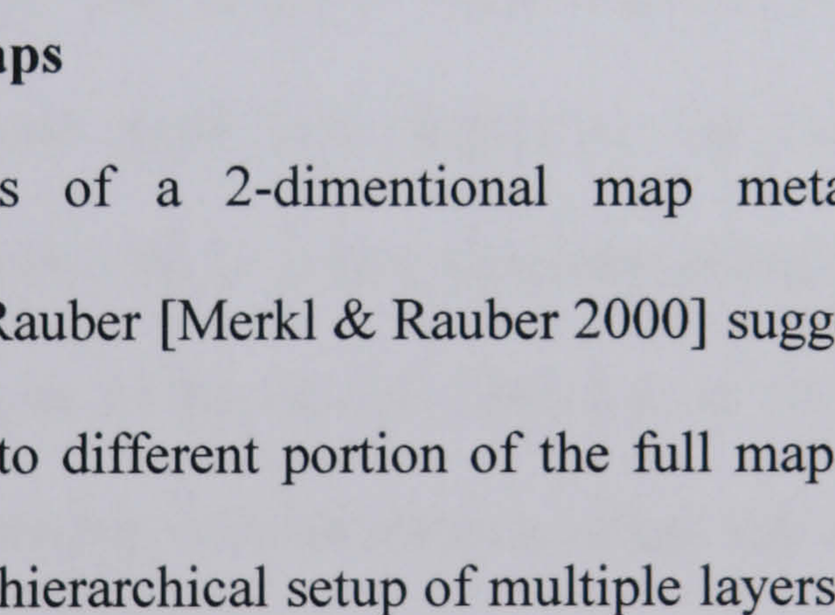


Figure 4.13: Architecture of a Three-layer Hierarchical Feature Map
(Reproduced from [Merkl & Rauber 2000] with permission © 2000 Physica-Verlag)

SOM representing a summary of the full map and every unit in this map is further expanded by an associated SOM located at the next layer of the hierarchy. This process can

be repeated to create third and further layers. The training process in this model results in a hierarchical arrangement of the document collection, where SOMs from higher layers of the hierarchy represent the overall organisation of the parts of the document archive represented by their siblings. Maps at the lower layers of the hierarchy provide fine-grained distinction between individual documents. This hierarchical feature map model has shown better evaluation results over the standard SOM on the CIA World Fact Book collection. The architecture of the network of this model is data dependent and therefore a prior knowledge of the input data is required to build an appropriate network before training begins.

Drawbacks of SOMs

One of the major shortcomings of self-organizing maps is the remarkable computational demand of its learning process. Possible solutions for this problem, as suggested in the literature, include the use of the biologically motivated concept of lateral inhibition at the level of the learning rule, and using a lower dimensional representation strategy to represent documents. Some work with regard to the first of these is described in [Merkl 1995b, Miikkulainen 1991]. Latent semantic indexing and principal component analysis (PCA) are examples of the second. [Bayer et al. 1996] reports the use of PCA in the area of document processing. Neural network realisations of the PCA can also be found in the literature [Oja 1982, Oja 1989].

Adaptive Resonance Theory (ART)

Most of the supervised learning algorithms suffer from catastrophic forgetting or interference as new patterns are accommodated. This demands that all the training examples be retrained once a new category of patterns is encountered. Adaptive Resonance Theory (ART) [Carpenter & Grossberg 1987] provides a way of learning new patterns without affecting the representation of previously learned patterns. Dunbar [Dunber 1999] reports an application of ART in the study of word meaning in which the model derives lexical relations between words from indirect subjective property rating judgments

provided by native speakers. An analogue version of the ART network, ART2, [Carpenter & Grossberg 1987] that is capable of learning analogue input is the most popular one. The ART2 network architecture has two interconnected layers with recurrent links between them. Resonance is achieved when the response of the recurrent links matches the input. Resonance determines the classification of a pattern. If the initial response does not create resonance, then an alternative category is tried and if there are no alternative categories left then a new one is created. [Vlajic & Card 1999] proposed a modified version of ART2 which is based on a recursive learning procedure with a dynamically changing vigilance parameter. This model has proven stability in hierarchical clustering, by means of which highly efficient multi-level document retrieval can be achieved. In their (Vlajic and Card's) work on the application of ART2 to Adaptive Hypertext Clustering, each web page was encoded in two separate representations; one based on the (words) content and the other based on the hyperlinks. The two representations were simultaneously processed in their own spaces, and the overall similarity of two web pages was computed in a compound function. The model is reported to have better identified both the main thematic categories and functional subgroups within them.

4.6 LATTICE-BASED RETRIEVAL

The use of lattice structures for information retrieval, though small in numbers, date back to Fairthorne (1956), followed by Mooers (1958), Salton (1968) and Soergel (1967) [as cited in Priss 2000b]. These applications were mainly for deriving a mathematical formalisation of a query language. They have not been well accepted by mainstream information retrieval community, perhaps due to difficulties in their practical utilisation. However, the invention of Formal Concept Analysis by Ganter and Wille (1999) has triggered concept lattices regain lost interest among IR researchers. A number of Concept Lattice based IR models has been developed by various researchers, including Godin [Godin et. al. 1993], Carpineto and Romano [Carpineto & Romano 1996,1998,2000], Priss

[Priss 1997, 2000b] and Cole, Eklund and Stumme [Cole & Eklund 1996, Cole et al. 2000, Becker et al. 2002]. Most of these models are based on navigating a lattice structure (AKA Galois Lattice [Ganter & Wille 1999]) for browsing. We do not detail these approaches here in this chapter, as a detailed description of the FCA formalism; the theory of concept lattices and a review of their application to document retrieval are given in the next chapter (Chapter 5).

4.7 SUMMARY

In this chapter, the traditional Boolean, VSM, Probabilistic and Logical models were summarised first, and then the more recent approaches such as Inference nets, Bayesian nets, GAs, Fuzzy theoretical approach, Spreading activation, Neural network and Knowledge-base approaches were reviewed. The main objective was to review innovative IR models, and their underlying theories and techniques that have enriched the field over the past half a century. Each of these approaches has been aimed at improving the effectiveness of IR by addressing one or more of the key problems discussed in Chapter 2. Each approach has its merits and drawbacks. Our intention was not to evaluate them against each other, but to present different approaches rendered to solve the IR problem and their underlying theories and techniques to give a good synthesis of the subject.

CHAPTER 5 - FORMAL CONCEPT ANALYSIS

In this chapter, we lay the foundation to our model by presenting the theory of Formal Concept Analysis (FCA) [Ganter & Wille 1999], the technique fundamental to our work. FCA in combination with lattice theory gives a hierarchically organised lattice structure of formal concepts known as a "Concept Lattice". It is these concept lattice structures that we use for representing documents and queries in our model. In the following, we first give a brief introduction to lattice theory, the underlining algebra on which FCA is based upon. We then define formal concept analysis and describe how formal concepts are organised into concept lattice structures according to a defined subsumption order relation. The analogy between the representation of formal concepts in concept lattices and the representation of ideas (concepts) in the human brain in the process of human understanding is then analysed. Finally, previous attempts made by various researchers to employ concept lattices into IR are reviewed.

5.1 LATTICE THEORY

5.1.1 Partially Ordered Set (poset)

A Partially Ordered set or poset $<P,\leq>$ is composed of a set of elements P and a subsumption relation $\leq$ defined on that set. The subsumption relation must posses the reflexivity, anti-symmetry and transitivity properties, i.e. for all $x_i, x_j, x_k \in P$:

i. $x_i \leq x_i$

ii. $x_i \leq x_j$ and $x_j \leq x_i \Rightarrow x_i = x_j$

iii. $x_i \leq x_j$ and $x_j \leq x_k \Rightarrow x_i \leq x_k$

5.1.2 The Hasse Diagram

Hasse diagrams (Figure 5.1) are used to visually represent posets. The Hasse diagram has the property that if one element is subsumed by another different element then the subsumed element will be positioned lower to that of the subsuming element.

The Hasse diagram can be drawn for any finite poset by calculating the covering relation (defined below in Section 5.1.4) and drawing lines between two elements if one covers the other. The elements are positioned so that if one element covers another then the covering element is placed above the covered element.

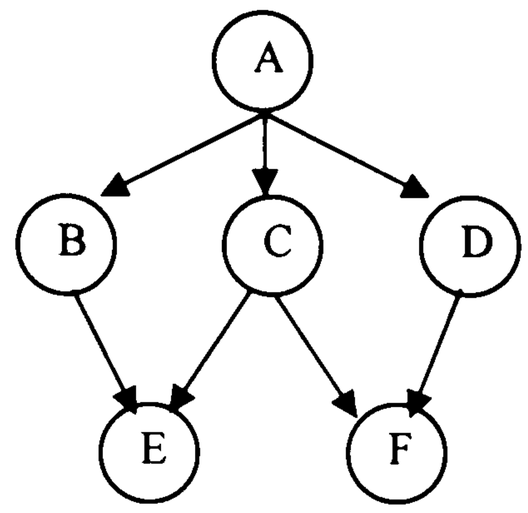


Figure 5.1 : A Hasse (Line) Diagram

The Hasse diagram given in Figure 5.1 illustrates an example of a poset with elements *A*, *B*, *C*, *D*, *E*, *F*. The subsumption relations are indicated by the paths between elements. Here *A* subsumes every element in the set while *C* subsumes both *E* and *F* and itself. An important feature of a poset, as can be seen in the above figure, is that an element may have multiple parents rather than one parent as is the case in Tree Diagrams. This makes posets a generalisation of a tree structure.

5.1.3 Down Set

The Down set of a subset of a partially ordered set is the set of elements which are less than all elements in the subset. Formally, let $<P,\leq>$ be a partially ordered set, then the down set of an element $e \in P$ is the set $\{a \in P \mid a \leq e\}$. The down set of $S \subseteq P$ is the set $\{a \in P \mid (\forall e \in S, a \leq e)\}$.

5.1.4 Covering Relation

The covering relation indicates the children of the elements. If $<P, \leq>$ be a poset and $x_i, x_j \in P$, then the covering relation defined on P is a relation $\sqsubseteq$ such that $x_i \sqsubseteq x_j$ iff $x_i \leq x_j$ and there is no $x_k (\neq x_j) \in P$ such that $x_i < x_k \leq x_j$.

5.1.5 Lattice

A Lattice is a partially ordered set with the special property that each pair of elements always has a least upper bound (join) and a greatest lower bound (meet). The following definitions will help understanding this.

- **Maximal and Minimal Elements**

If P be a partially ordered set and $S \subseteq P$, then $x \in S$ is maximal if for all $y \in S$, $y \leq x$. Similarly $x \in S$ is minimal if for all $y \in S$, $x \leq y$. The maximal and minimal elements of a subset S are known as the greatest and the least elements respectively.

- **Lower Bound and Upper Bound**

If P be a partially ordered set and $S \subseteq P$ then an element $x \in P$ is an upper bound of S if $s \leq x$ for all $s \in S$. Similarly a lower bound is defined dually. S^u and S^l denote sets of all upper bounds and all lower bounds of S respectively.

- **Least Upper Bound and Greatest Lower Bound**

If P be a partially ordered set and $S \subseteq P$ and if S^u has a least element then that element is called the least upper bound of S. If S^l has a greatest element then that element is called the greatest lower bound.

- **Join and Meet of Elements**

The join of two elements $x_1, x_2 \in P$ denoted by $x_1 \vee x_2$ (if it exists) is the least upper bound of $\{x_1, x_2\}$. Similarly the meet of the two elements, denoted by $x_1 \wedge x_2$ is the greatest lower bound of the two elements.

5.1.6 Complete Lattice

If the greatest lower bound and least upper bound exist for all $S \subseteq P$ then P is called a complete lattice.

5.2 FORMAL CONCEPT ANALYSIS (FCA)

Formal Concept Analysis (FCA) was proposed by Rudolf Wille in 1982 [Ganter & Wille 1999, Wille 1997] as a mathematical framework for performing data analysis. It provides a conceptual analytical tool for investigating and processing given information explicitly. FCA structures data into units that are formal abstractions of "concepts" of human thought allowing meaningful and comprehensible interpretation. FCA models the world as being composed of objects and attributes. An incident relation connects objects to attributes. The choice of what is an object and what is an attribute is dependent on the domain in which FCA is applied. Information about a domain is captured in a "formal context".

5.2.1 Formal Context

The theory of Formal Concept Analysis begins with the definition of a Formal Context based on objects and attributes as its elementary units. A formal context is merely a formalisation that encodes only a small portion of what is usually referred to as a "context". A formal context can be considered as (a mathematical model of) a table, which relates objects and attributes of a "real situation" [Burmeister 1998]. The following is a formal definition of a formal context.

A formal context is a triplet $<G,M,I>$ consisting of two sets G and M and a relation I between G and M (i.e. defined on $G \times M$), where G is a set of objects and M is a set of attributes, gIm denotes that object g has attribute m, i.e. $(g,m) \in I$.

Note : 1. this is not a partially ordered set as a partially ordered set is defined on a single set with an order relation.

2. it is not mandatory for a relationship to exist between every pair of elements in $G \times M$.

5.2.2 Formal Concept

FCA defines a Formal Concept on a Formal Context. A formal concept consists of a pair of sets (A,B), where A is a set of objects and B is a set of attributes. In this concept (i.e. the pair (A,B)), attributes in B are maximally possessed by the set of objects in A, and consequently the objects in A are the maximal set of objects possessing the set of attributes in B. A formal definition is given below.

A pair (A,B) of sets in which $A \subseteq G$ and $B \subseteq M$ is a formal concept if $A^{\downarrow} = B$ and $B^{\downarrow} = A$ (completeness constraint), where $A^{\downarrow} = \{ m \in M \mid gIm \text{ for all } g \in A \}$ (i.e. the set of attributes common to all the objects in A) and $B^{\downarrow} = \{ g \in G \mid gIm \text{ for all } m \in B \}$ (i.e. the set of objects which have all attributes in B). $A^{\downarrow}$ is known as the intent of the set of objects in A, and $B^{\downarrow}$ the extent of the set of attributes in B.

The set of all formal concepts of a context (G,M,I) is denoted by $B(G,M,I)$. This set consists of all pairs (A,B), where $A \subseteq G$ and $B \subseteq M$, such that $A = B^{\downarrow}$ and $B = A^{\downarrow}$.

An Important result:

For every set of objects $A \subseteq G$, $A^{\downarrow}$ is an intent of some concept in $B(G,M,I)$, and $(A^{\downarrow\downarrow},A^{\downarrow})$ is always a concept. $A^{\downarrow\downarrow}$ is the smallest extent containing A. Consequently, a set $A \subseteq G$ is an extent of a concept if and only if $A = A^{\downarrow\downarrow}$.

5.2.3 The Subsumption Relation and Sub/Super Concepts

FCA models the specificity and generality relationships between two related concepts by means of a sub-super order relationship. This sub-super concept relationship is formally defined as:

If (A_1,B_1) and (A_2,B_2) are concepts of a context, then (A_1,B_1) is called a sub concept of (A_2,B_2), if $A_1 \subseteq A_2$ (or $(B_1 \supseteq B_2)$. In this case (A_2,B_2) is a super concept of (A_1,B_1), and this sub-super order relation is written as $(A_1,B_1) \leq (A_2,B_2)$.

This means a sub concept actually is a concept with fewer objects than any of its super concepts; equivalently, a subconcept is a concept with more attributes than any of its super concepts.

5.2.4 Concept Lattice

A set of all concepts of the context *(G,M,I)* (i.e. *B(G,M,I))* when ordered with the order relation $\leq$ (the subsumption relation) defined above forms a concept lattice of the context, and is denoted by $\underline{B}(G,M,I)$. Recall that a lattice is an ordered set V with an order relation in which for any given two elements x and y, the supremum and the infimum elements always exist in V (Section 5.1.5). Furthermore, such a lattice is called a "complete lattice" if supremum and infimum elements exist for any subset X of V (Section 5.1.6).

5.2.4.1 Fundamental Theorem on Concept Lattices

The fundamental theorem of FCA states that the set of all the formal concepts created from a formal context forms a complete lattice (see Ganter & Wille 1999 pp20-22 for a proof). This complete lattice, as it is composed of formal concepts, is called a concept lattice.

Properties of a concept lattice include:

1. A concept lattice arranges its elements in a structured manner, showing the sub-super concept (order) relationships.
2. A concept lattice can be illustrated in a Hasse (Line) diagram (recall, every finite ordered set can be represented by a Hasse (Line) diagram).
3. A node in the line diagram of a concept lattice structure represents a formal concept (i.e. it consists of a set of objects A and a set of attributes B such that $A = B^{\downarrow}$ and $B = A^{\downarrow}$)
4. A node connected to a second node is:
   - a sub concept of the second node if it appears below the second node in the Hasse diagram of a lattice structure.
   - a superconcept of the second node if it appears above the second node.

     i.e. the concepts that are in the lower part of the lattice are more specific concepts and the concepts that are in the upper part are more generic concepts (note that the concepts in the upper part cover concepts in the lower part).

An example of a formal context (Table 5.1) and its concept lattice structure (Figure 5.2) (extracted with permission from Ganter & Wille 1999) are given below. The context of the example is based on an educational film "*Living Beings and Water*".

Objects \ Attributes		a. Needs water to live	b .Lives in water	c. Lives on land	d. Needs chlorophyll to produce food	e. Two seed leaves	f. One seed leaf	g. Can move around	h. Has limbs	i. Suckles its offspring
1	Leech	×	×					×		
2	Bream	×	×					×	×	
3	Frog	×	×	×				×	×	
4	Dog	×		×				×	×	×
5	Spike-weed	×	×		×		×			
6	Reed	×	×	×	×		×			
7	Bean	×		×	×	×				
8	Maize	×		×	×		×			

Table 5.1 : Formal Context of the Film "Living Beings and Water"

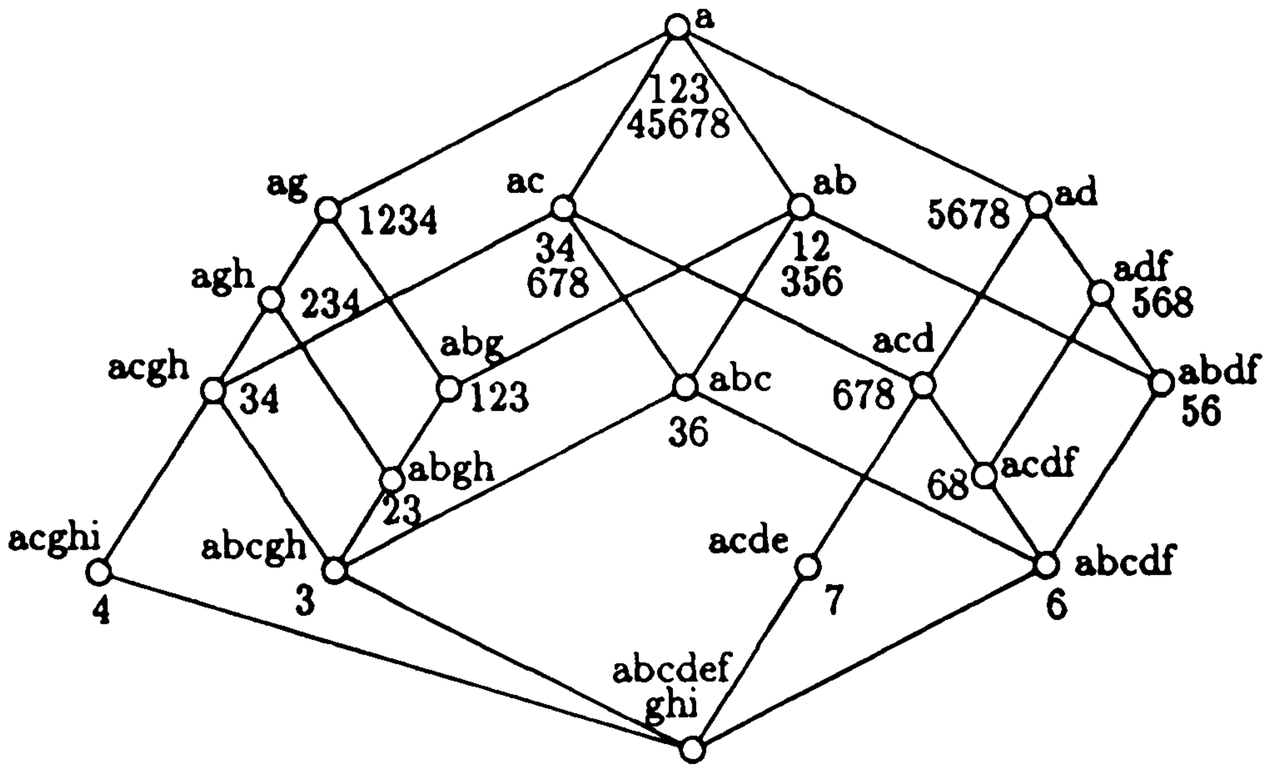


Figure 5.2 : Concept Lattice of the Context in Table 5.1
(Reproduced from Ganter & Wille 1999] with permission © 1999 Springer-Verlag)

The line diagram given above (in Figure 5.2) indicates the intents and extents of all the concepts. However, the labelling can be simplified by writing each element (object/attribute) only once in the diagram as shown below in Figure 5.3.

The extent of a given node (a node/circle represents a concept) consists of the objects located at this circle or the circles which can be reached by descending line paths from this circle. Correspondingly, the intent can be found by following all line paths going upward from the circle and noting down the attributes assigned to these circles.

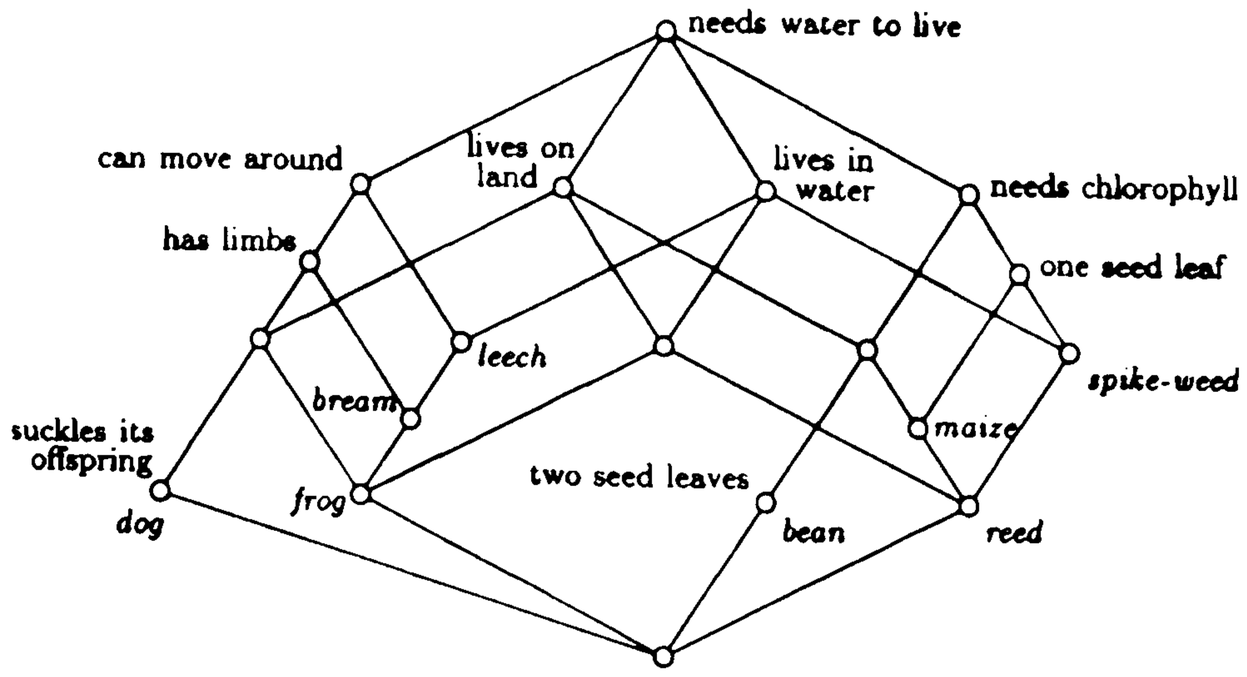


Figure 5.3 : Concept Lattice with Simplified Labelling
(Reproduced from [Ganter & Wille 1999] with permission © 1999 Springer-Verlag)

5.2.5 Join and Meet Concepts

The concepts of the join and meet of concepts inherit from the join and meet elements defined in Lattice Theory (Section 5.1.5). Concepts at the nodes from which two or more lines run up are called *meet* concepts (i.e. nodes with more than one parent) and concepts at the nodes from which two or more lines run down are called *join* concepts (i.e. nodes with more than one child). An interesting feature of concept lattices is that we do not need to know all meet and join concepts explicitly in order to build the complete lattice. They can be inferred from the set of concepts. A join (parent) node is inferred given all of its child nodes and a meet (child) node is inferred given all of its parent nodes (Figure 5.4).

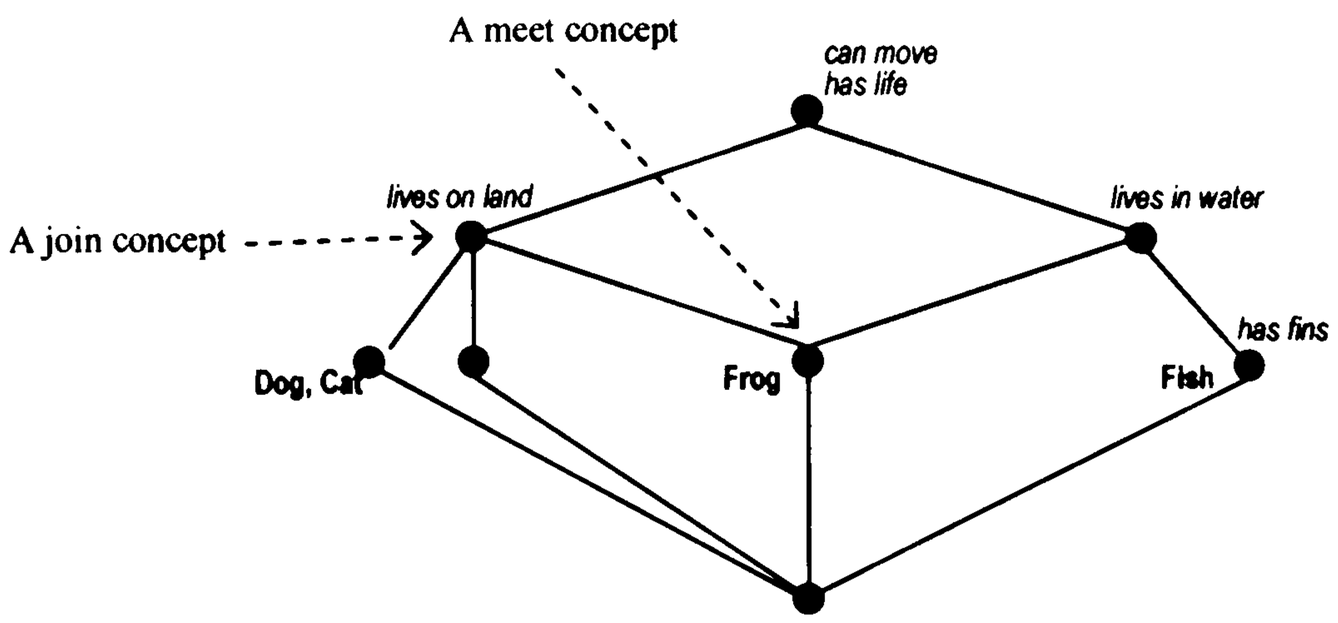


Figure 5.4 : Example Concept Lattice

5.2.6 Object Concepts and Attribute Concepts

An object concept is the most specific concept present in a concept lattice containing a given object. Conversely, an attribute concept is the most generic concept containing a given attribute. For instance, the object concept of the object "*frog*" in the above concept

lattice (Figure 5.4) is {*frog*} → {*lives in water, lives on land, can move, has life*}, and the attribute concept of the attribute *lives in water* is {*frog, fish*} → {*lives in water, can move, has life*}. We use these two properties in our work to extract most specific concepts from lattice representations of queries and documents to match between.

5.3 CONCEPTUAL SCALING

Conceptual Scaling is a process of turning many-valued contexts into a number of one-valued formal contexts or scales [Prediger & Wille 1999, Ganter & Wille 1999, Stumme 1996]. Many valued contexts are the ones in which objects may have different values for the same attribute (e.g. the attribute "colour" can contain any colour value). These many-valued contexts are modelled with an additional set W that, for instance, contains all the permissible coloursn and written as (G,M,W,I), where the relation I is now a triplet $(g,m,w) \in I$ and $I \subseteq G \times M \times W$. The attribute m can be viewed as a partial function from G to W, written as $m(g) = w$. A concept lattice cannot be defined on a multi-valued context; instead a multi-valued context is translated into a one-valued context(s) and a concept lattice(s) created. The translation process may result one or more one-valued contexts which are called "conceptual scales". A concept lattice of combined scales can then be embedded in the direct product of the concept lattices of the individual scales, and be organised into a nested line diagram. A formal definition of a conceptual scale is as follows:

A conceptual scale for a set $Y \subseteq M$ is a one-valued context $S = (G_s,M_s,I_s)$ with

$$\times_{m \in Y} m(G) \subseteq G_s$$

The derived relation $J_s \subseteq G \times M_s$ is defined by

$$(g,n) \in J_s \Leftrightarrow ((w_m)_{m \in Y},n) \in I_s \text{ with } (g,m,w_m) \in I \text{ for all } m \in Y$$ [Stumme 1996].

Conceptual scaling has also been used in single-valued contexts to reduce the number of attributes involved in a single investigation [Prediger & Wille 1999, Cole et al. 2003]. In Toscana and ToscanaJ [Becker et al. 2002], for instance, one can choose a list of scales

($(S_t)_{t\in T}$, where T is a set of index terms), and the nested line diagram of the concept lattice of the context will be displayed on the screen.

This idea in particular has strength in visualising parts of the full concept lattice for browsing purposes. Although we do not use conceptual scaling in our work, it deserves mentioning, and is useful in understanding related work. More detail of conceptual scaling can be found in the book by Ganter and Wille (1999), pp 36-56.

5.4 Towards Concepts (Ideas) Expressed in Natural Language

The formal concept analysis and the concept lattices, defined and described above, present an interesting formal framework for representing and analysing formal concepts defined on formal contexts. In this research, we investigated the suitability of this framework in the representation and analysis of ideas (concepts) extracted from free text documents (in the context of IR). A number of questions have to be addressed before employing formal concepts in IR tasks. These include: what is an object; what is an attribute; what is a concept (an idea); how can we extract objects and attributes from textual material and form formal concepts; what is the analogy between the order relationship defined for formal concepts and the ideas (concepts) extracted from textual documents; what are the roles of *join* and *meet* concepts in the human understanding of natural language text etc. This section attempts to answer these questions. The objective is to justify the suitability of FCA formalisations for representing concepts (ideas) written in natural language, and thus for representation of textual material in terms of formal concepts in an IR setup.

5.4.1 Abstracting Ideas (Concepts) in Human Understanding

The theory of concept lattices has been founded based on the traditional understanding of concepts, by which a concept is determined by its extent and intent. The extent of a concept (e.g. *dog*) is the collection of all objects covered by the concept (the collection of all *dogs*), while the intent is the collection of all attributes (e.g. *has tail, can bark, is a*

mammal etc.) covered by the concept. This interpretation of a concept can be employed directly in representing concepts expressed in natural language. The formation of an idea or a concept in the human mind during the understanding of natural language text may initially be triggered by the objects (real or abstract objects) and attributes (properties) of objects present in the text. The overall context of the subject being read and the reader's background knowledge of the subject are secondary means that help clarifying ambiguities.

5.4.2 Why Two Entities: Objects and Attributes?

In general, an object corresponds to the subject or topic (or the main participants of an idea) of the context, and the attributes modify the meaning of the object to express the intention or the context in which the object is being used. For instance, a particular set of attributes associated with the object "*dog*" may deal with the context of say *eating habits of dogs* while another set of attributes may deal with say the *sleeping habits of dogs*. The human thought (understanding) process should necessarily be able to understand these two entities or features (i.e. objects and attributes or more precisely extensions (participants) and intentions) in order to make sense of natural language expressions. Therefore, it is important to capture them separately in order to formulate an abstraction of human thought. FCA captures these two important aspects, the subject and the context by its two entities, objects and attributes, respectively, in the definition of a formal concept. This makes FCA a suitable candidate for abstracting human thoughts (concepts) for computer manipulation.

5.4.3 Super-Sub Order Relationship in Formal Concepts and Natural Ideas

In FCA, a sub concept is defined as a concept with fewer objects and more attributes than its super concepts. This means we need more attributes (or properties) to define something specific, compared to few attributes needed to define something generic. We can argue that this property holds in human understanding as well, since the more generic an idea is, the

more examples we can find to support it (i.e. a broader category is defined when a few generic properties are specified). On the other hand, a specific idea within this broader category needs a specific example with at least one additional more specific property to distinguish it from the rest.

Concepts stored in human mind may be structured in a similar manner whereby we need more specific detail to learn, understand or express a more specific idea. For instance, to define a *bird*, we need to say it *can fly* and it *has a beak*, in addition to the information necessary to say that it's an *animal*. However, the frequently used generic attributes (in this case the attributes to specify that a bird is a living animal) are usually not explicitly mentioned by humans to express an idea during normal human communication. This is because they are implicit, and an average human brain has gained all the necessary background knowledge to understand frequently used common ideas when expressed just by the referent (textual label) of the main object. For instance, we never define what an "animal" is during conversations; instead we simply use the term "*animal*". Everyone knows what an animal is. At some point during our learning process (implicit or explicit) we have absorbed all the necessary attributes to understand what an animal is. It is obvious, however, that encoding concepts (ideas) in a computer requires all the information necessary to distinctly identify a particular idea (concept) to be explicitly specified. These background general ideas (concepts) are analogous to the super concepts and the sub ideas/categories of them or more specific cases of them are analogous to the sub concepts defined in the FCA formalisation. In other words, sub concepts correspond to specific ideas and super concepts correspond to general ideas.

5.4.4 What does More/Less Objects in a Concept Mean?

In FCA, objects having the same set of attributes are categorised together and a single formal concept with all such objects sharing the same set of attributes is formed. Each object in such a concept, together with its associated set of attributes, represents the

concept. Additional objects (in the extent of the same formal concept sharing the same set of attributes) only say that those objects also comply for the same concept, and thus help its correct interpretation, i.e. having more objects in the extent of concept help clarifying any ambiguities of the interpretation (of meaning) of the concept attributed by that particular set of attributes. Therefore, having more objects in a concept can be regarded as having more examples or evidence to understanding clearly the concept (idea) that it represents.

5.4.5 The Roles of Join and Meet Concepts in Human Understanding

Categorising common objects (or ideas) together is a natural phenomenon in the human understanding. For instance, if you are asked to name some animals you can give a vast number of different animals as examples of animals. If you are then asked to name some carnivorous animals, you certainly have no problem of naming a set of animals that eat meat. You may have given names of some of these carnivorous animals as examples of animals for the first question as well. This means your brain knows how to categorise the same set of objects depending on the context, i.e. depending on the attributes that each object possesses. This categorisation is not rigid in which a given object may be categorised into more than one category. For instance, a *frog* may be categorised into an animal *living in land* as well as *living in water*. It is this very same phenomenon that the *meet* and *join* concepts formulate in FCA. The concept of "*animal*" is a join of all categories of living beings (such as *birds*, *reptiles*, *mammals*, *fish* etc.). However, as of *frogs* (see Figure 5.4), certain animals belong to more than one sub category of animals.

The similarities between the properties of FCA and ideas or concepts of human understanding discussed above suggest that the way that humans formulate concepts, structure them and use them in the process of understanding and expressing ideas is analogous to the way concepts are formulated in FCA and are structured in a Concept

Lattice. This motivated us to base our IR model (Chapter 7) on an FCA-based representation scheme.

5.5 Related Work - Applications of FCA in IR

Concept lattices, which have originally been created as a data visualisation and analysis tool, have later been employed in information retrieval tasks such as browsing, categorisation and document retrieval/ranking. Use of a conceptual map, which is structured according to generalisation and specialisation relationships between formal concepts, makes FCA naturally suitable for user interface design to help the user navigate through the concept lattice to locate desired documents. As a result, most of the past research have been on developing browsing mechanisms for domain specific IR [Cole et al. 2003, Kim & Compton 2001, Becker et al. 2002]. However, a drawback of this approach is that as the documents in the search space grow, the size of the lattice structure grows and it becomes impossible to display the entire structure on a computer screen. A number of researchers have used conceptual scaling (Section 5.3) as a remedy for this problem. Previous applications of FCA can therefore be described under two categories: (i) those that generate one large concept lattice; and (ii) those that employ conceptual scaling. TOSCANA [Becker et al. 2002], CEM [Cole et al. 2000], Kim and Compton's Web-based browsing mechanism [Kim & Compton 2001] and the Cole and Eklund's work [reported in Cole & Eklund 1993,1996] are few examples of browsing systems that use conceptual scaling while Carpineto's retrieval model [Carpineto & Romano 2000] is an example for the first category. It is also an example of a conventional type of an IR model, which is not based on browsing but produces ranked list of documents to a given query. A detailed review of this work is given below. Association Rule Mining [Jitender et al. 1998, Pasquier et al. 1999] is another application area related to information processing for which Concept Lattices have been applied successfully, but discussion of it is outside the scope of our work.

Godin et al. [Godin et al. 1993] implemented a lattice-based retrieval method known as the Galois Lattice Retrieval Method. In their work, the advantage of the lattice method against hierarchical classification was analysed, and the retrieval performance was evaluated compared to a conventional boolean retrieval. No significance performance difference has been reported, but the lattice structure was suggested as being an attractive alternative because of the potential advantages of lattice browsing. The objects of the concepts were documents (Doc IDs) and attributes were keywords (controlled terms) extracted from the documents. The user interacts with the system by navigating through the vertices of the lattice. The navigation can be done by either directly selecting a neighbouring vertex in the graph or by specifying (adding) a new term refining the current query. Addition of a new term results in a direct jump to the smallest vertex containing all the terms in the current vertex plus the new term. The interface of this model showed only the direct neighbours (of the current vertex) in the lattice.

Carpineto and Romano [Carpineto & Romano 1996, 1998] used a thesaurus as background knowledge to formulate browsing and presented experimental evidence that adding a thesaurus to a concept lattice improves its retrieval performance. The interface ULYSSES developed by Carpineto and Romano (1995 as cited in Priss 1997) showed the lattice graph similar to a fisheye view [Furnas 1986] of individual nodes. As in Godin's work [Godin et al. 1993], controlled terms of a database were employed to construct the lattice.

In a separate attempt, Carpineto [Carpineto & Romano 2000] implemented another IR model in which the whole document collection is represented in a single concept lattice structure using only the keywords/attributes extracted from the contents of the documents (no thesauri is employed in this work). In this work, each document is characterised by the set of (controlled) terms it contains. The extent of a concept consists of documents (IDs) and the intent consists of terms/keywords. The concept lattice is automatically built from a document-term matrix (inverted index) using Godin's incremental lattice-building

algorithm called GALOIS. A query, also characterised by the set of terms it contains, is then mapped onto the document lattice. The query will either exactly map on to an existing node (in the case if there is a node in the lattice containing exactly the same attributes as of the query) or it will cause a new node to be created for the query (otherwise) at the appropriate position determined according to the terms present in the query (Figure 5.5). The distance (in terms of links in the shortest path) of a document (document node) to the query node determines a score (similarity measure) of the document to the query. Documents most similar to the query are the ones that appear in the extent of the same node as of the query (if the query were mapped onto an existing node), or the documents that appear in the extents of the nodes which are closest (by distance of one), i.e. directly linked to the newly created query node (otherwise). The following line diagram (Figure 5.5) illustrates the mapping of a query with terms "NNS" and "FINANCE" into the concept lattice of the context shown at the left.

	D1	D2	D3	D4	D5	D6	D7
Neural-Network-Systems	x	x	x				x
Knowledge-Based-Systems				x	x	x	x
Credit				x	x		x
Finance	x			x			x
Account	x		x				
Bank	x	x	x			x	
River		x					
Waters						x	

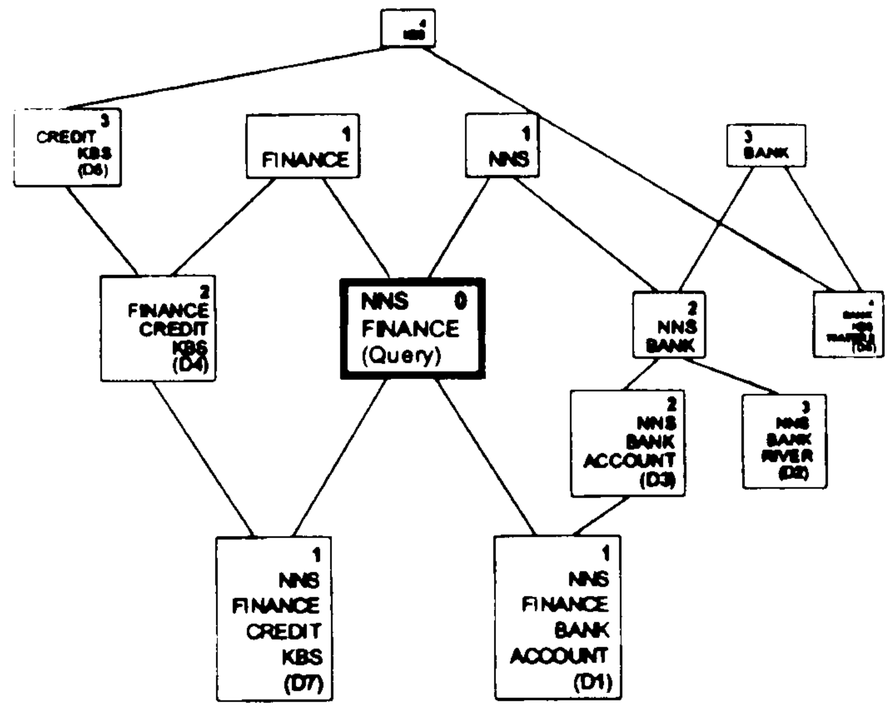


Figure 5.5 : Carpineto's Model (Reproduced from [Carpineto & Romano 2000] with permission © 2000 John Wiley Sons Inc.)

An important distinction of this similarity measure, to that of other best match similarity measures, is that this is a conceptual distance within the concept space rather than one based on common term counts. The number of terms common to the concepts of two linked nodes can vary depending on the data used for building the concept structure and therefore do not say much about the closeness of them in the concept space. For instance, there may well be two near nodes (concepts) in the lattice that differ by a larger number of terms than more distant concepts do.

Experimental results from this model have shown better performance compared to cluster-based and best match approaches when the top retrieved documents (i.e. top 5, 10 & 20 retrieved documents) were considered. Also, the ability of the model to rank documents that did not match the query is shown to be superior over the other two approaches [Carpineto & Romano 2000].

Christian Lindig [Lindig 1995] applied FCA for building a simple IR prototype to retrieve software components. Objects are names of software components and attributes are keywords extracted (manually) from the documentation of the software. The software concepts are then organised in a concept lattice and retrieved by the user formulating a query with a set of keywords selected from a list of permissible keywords. The interface of the implementation shows interactively the software components that contain the selected set of keywords as the user selects keywords.

The ANACONDA and TOSCANA [Becker et al. 2002] are two of the most widely used FCA-based programs developed by the Darmstadt research group. ANACONDA is an editor for creating and managing concepts, and TOSCANA is a tool for visually displaying the underlying conceptual structure(s) (created with ANACONDA) and interactively browsing on them. TOSCANA uses conceptual scaling for displaying the user with only a part of the full concept lattice structure, and allows the user to explore it by expanding the concepts with a list of scales. TOSCANA has recently been re-implemented in Java (TOSCANAJ) using a new file format based on the www consortium (W3C) specifications of XML.

Cole and Eklund [Cole & Eklund 1996] incorporated a medical language thesaurus known as SNOMED (Systematized Nomenclature of Medicine) [http://www.snomed.org/] for the retrieval of medical discharge summaries. Medical concepts in SNOMED are described by a set of phrases and are structured according to a subsumption relation corresponding to a specialisation/generalisation hierarchy. The medical discharge summaries, indexed with

SNOMED concepts, are used to construct a concept lattice to represent the document collection. In this work, documents (IDs) are considered objects and medical concepts of SNOMED are considered attributes. A particular SNOMED concept is assigned to a document if that concept (string) is found in that document. The generalisation/specialisation hierarchy of concepts of SNOMED are imposed on the document lattice by assigning the documents with all the concepts (as indices) in the SNOMED that are more general to each of the found (SNOMED) concepts (Figure 5.6). Therefore, the structure of the concept lattice well reflects the specialisation/generalisation information present in SNOMED.

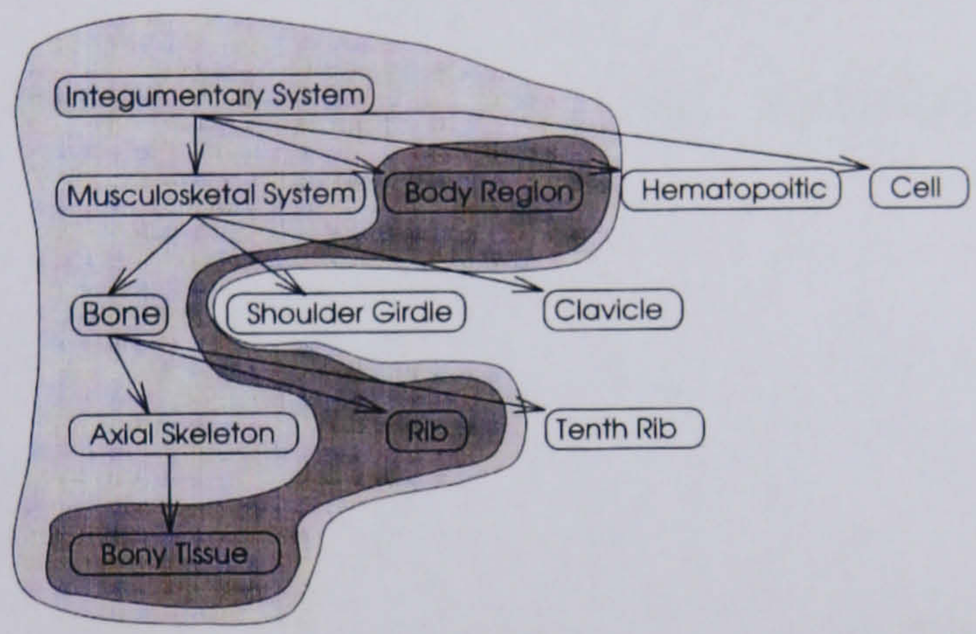


Figure 5.6 : SNOMED Concept Hierarchy and Assignment of Concepts
(Reproduced from [Cole & Eklund 1996] with permission © 1996 Springer-Verlag)

Note that, the dark shaded area in Figure 5.6 shows the concepts that exist in the document, and the lightly shaded area shows the additional concepts that are included by the generalisation. The user formulates a query by navigating through the SNOMED hierarchy specifying the level of specialisation for each concept he is interested in. The documents that are at the "*meet*" of the maximal elements of each SNOMED concept on the boundary of the region of selected concepts are the ones that are retrieved (Figure 5.6). Hence the retrieved documents contain the specialised concepts specified by the user and all the generalised concepts (in the SNOMED hierarchy) of them.

In this work, incorporation of expert knowledge (through SNOMED) for document representation and also for query formulation is considered advantageous for the retrieval of the documents in the particular domain it is designed for. A disadvantage of this

approach is that it cannot be used in domains in which such domain specific thesauri are not available for indexing the documents.

This work has later been extended [Cole et al. 1997] to deal with large contexts by using conceptual scaling. In this work, instead of the SNOMED, they used the thesaurus MeSH (Medical Subject Headings of the National Library of Medicine) for indexing. During query formulation, the user partitions the MeSH concepts by assigning one of a group of colours to each concept (Figure 5.7 left). This partitioning defines a number of different conceptual scales. Each conceptual scale defines a context of interest containing all medical documents with only those indicated attributes. The various contexts of interests, formulated as conceptual scales, are then combined in a nested line diagram

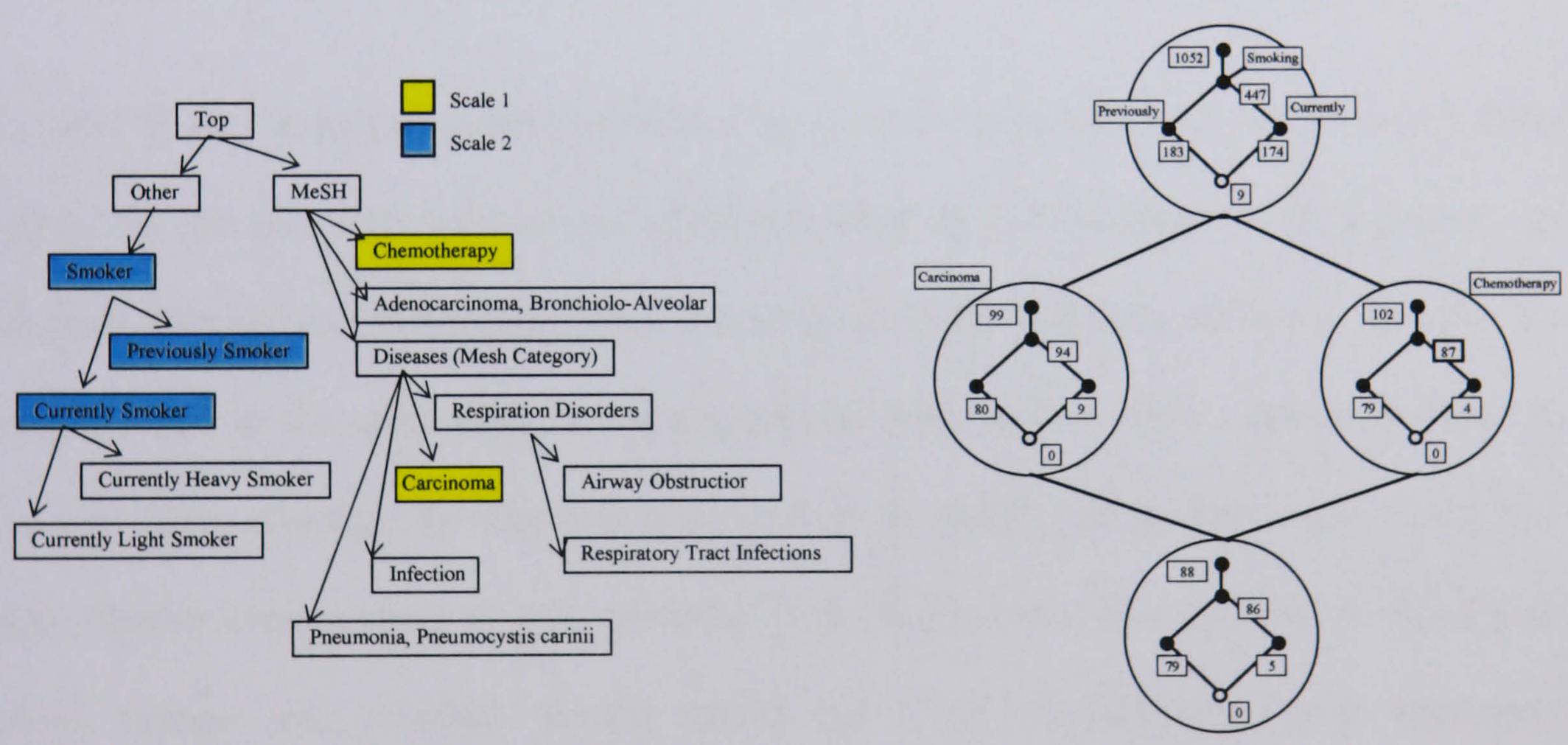


Figure 5.7 : Scale Assignment (left) Nested Line Diagram with an Expanded Scale (Right)
Reproduced from [Cole et al. 1997] with permission © 1997 Springer-Verlag)

(Figure 5.7 right) reducing the visual and computational complexity. This allows a larger number of attributes to be explored. The underlying theory used (Conceptual Scales) in this model is the same as in TOSCANA and ANACONDA, but the way conceptual scales are created is different. In this model, the implications that exist in the scale are restricted to those that exist in MeSH's medical taxonomy, whereas in ANACONDA conceptual scales are generated by user-inserted arbitrary implications. This is a dynamic approach in which the user creates scales, examines derived concept lattices, and changes the composition of

the scale iteratively, whereas in TOSCANA the scales are created separately in advance (using ANACONDA), prior to the combination of the scales in nested line diagrams.

A further extension of this work is notable for its application to analysing a collection of email [Cole & Eklund 1999]. In this work however, instead of a domain specific thesauri a hierarchy of classifiers were used for concept extraction. These classifiers extract useful terms referred to as "catchwords" (e.g. "*conference*" or "*organisation*") from the email, and encode known implications. The relations are generated and assigned in a semi-automatic process, in which the user assigns the relations either by accepting the suggestions made by the system (extracted catchwords) or modifying them, or even attaching his own attributes. The result of classification (term extraction) is stored in an inverted index and a hierarchy is defined by a set of subsumption rules defined by the user.

In order to see the ways in which attributes (the user is interested in) combine in the email collection, the user first searches for attributes either by their location in the hierarchy or by their description, by entering into a text search phase and then selecting the ones he needs to add to the scale from the search results. This scale is then used to construct a concept lattice showing the concepts generated by the email and attributes selected by the user. Further development of this work (by Cole, Eklund and Stumme) has resulted two email storage and retrieval models called the CEM (Conceptual Email Manager) [Cole et al. 2000] (Figure 5.8) and HIERMAIL [Cole & Eklund 2001]. The main objective

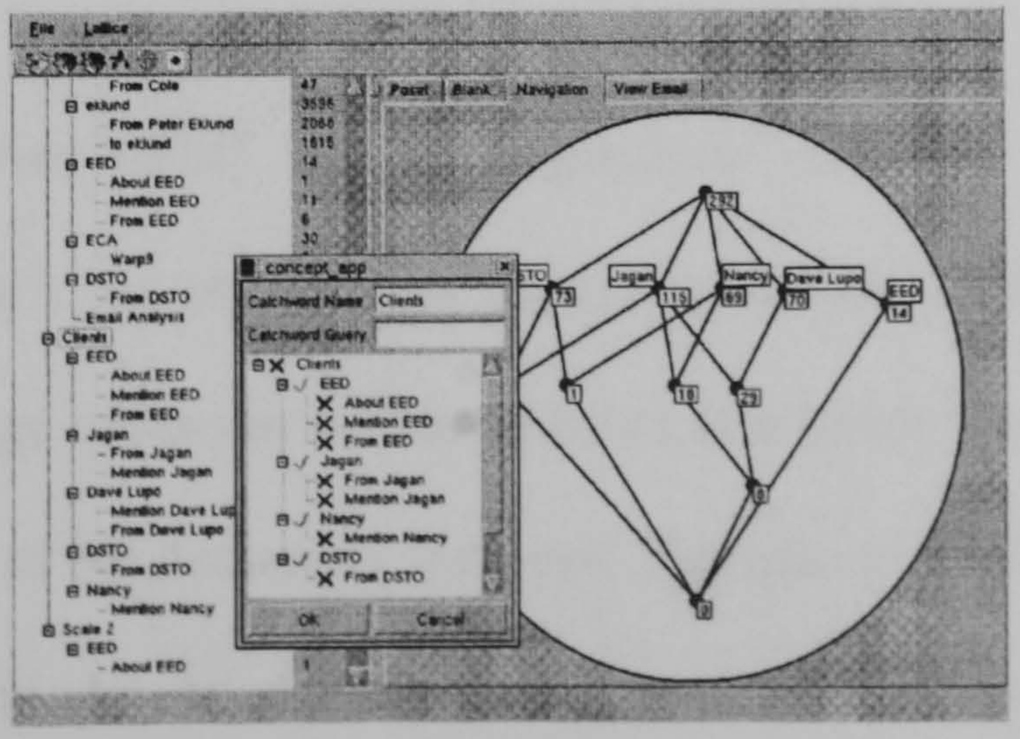


Figure 5.8 : Scale, Catchwords and Concept Lattice of CEM
(Reproduced from [Cole et al. 2000] with permission from R. Cole)

of these two tools has been to use a concept lattice based data structure to store email in "virtual folders". Navigation space in this case is a concept lattice instead of a tree of disjoint folders and the formal concepts replace the folders. This set-up allows users to follow different paths by using different combinations of the "catchwords" or descriptors to reach the same desired email as well as to store the same email under more than one virtual folder.

Kim and Compton [Kim & Compton 2001] reported a browsing mechanism for incrementally developed domain-specific document retrieval using FCA. In this model, a hierarchical conceptual clustering of the documents is built dynamically with the results corresponding to the user's query. Similar to Godin's approach [Godin et al. 1993], display of the lattice graph is restricted to only the direct neighbours and the graph is displayed using hyperlinks. A concept lattice is built with documents as objects and their keywords as attributes. The concept lattice is then scaled up (using conceptual scaling) with other attributes such as *author*, *proceeding title* and *publication year* and a nested structure is formed. The system has two modes of operation; a general search mode with Boolean queries and a lattice-based retrieval mode. The general search mode can be used in conjunction with the lattice-based retrieval mode to initially move on to a portion of the concept lattice associated with the user's query. An implementation of a working prototype on a test domain (papers presented at the Banff Knowledge Acquisition Workshops in recent years) can be found on the web (*http://pockey.cse.unsw.edu.au/servlets/Search*).

Priss [Priss 1997, Priss 2000a] reports a graphical interface similar to Godin's and Carpineto & Romano's for information browsing. In this model, a knowledge base or thesaurus of knowledge structures also formulated as a lattice is used in conjunction with a lattice representation of the document database. The role of the thesaurus is to incorporate common sense semantic knowledge into the system. Additionally, many-valued attributes (such as publication year of books) are also included in the document representation. This

model was later improved by using a faceted-thesaurus and the "FaIR" IR system [Priss 2000b] was developed. A facet here is simply a scale, not created for a selected set of attributes of the data, but a hand crafted structure of common knowledge. A faceted thesaurus is a collection of complete (i.e. all necessary combinations are enumerated in the facet) concept lattices (facets) constructed by partitioning a set of terms. Documents are indexed using the concepts of the thesaurus. A prototype of FaIR that has been implemented as an interface for a small knowledge base can be found at *http://kb.indiana.edu/*.

Van der Merwe and Kourie [Merwe & Kourie 2001] have proposed a lattice-based data structure, referred to by them as "compressed lattice", for scaling down a concept lattice to a two-layered graph with fewer nodes [see Figure 5.9 left]. It is a bipartite graph with an embedded-lattice achieved by removing concepts in a concept lattice in such a way that the resulting structure retains the desirable properties of the lattice. In their paper, the data structure is described in reference to query operations on a database represented as a

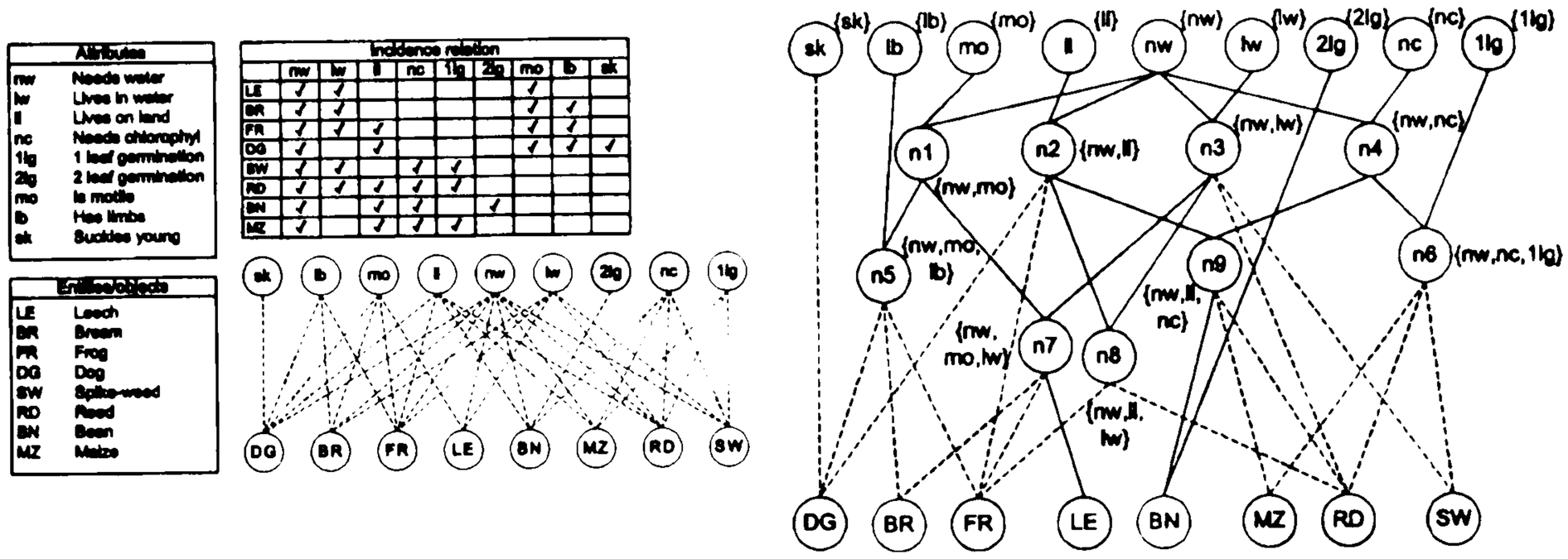


Figure 5.9 : Merwe's Bipartite Graph of the Context "Living Beings" (left) Lattice with less than 4 Attributes in its Nodes(right)
(Reproduced from [Merwe & Kourie 2001] with permission © 2001 Taylor & Francis Group)

bipartite graph as well as a concept lattice. It shows how a concept lattice can be compressed down to a bipartite graph and also how a compressed bipartite graph can be expanded to obtain the original concept lattice, using the defined operations. The answer for a query is computed based on the set of meets (in the compressed lattice) of all the subsets of the query terms. In this set up, removing concepts containing more than ***k***

attributes from the graph results in the return of the minimal concepts that have at most k of the n ($k < n$) query terms in common with the documents [see Figure 5.9 right].

A fully compressed concept lattice, compressed down to a bipartite graph with no intermediate nodes, is equivalent to the BAM data structure we use in our implementation. The difference is that, in our work, we embed the concept lattice in a bipartite graph by calculating appropriate weights between the two layers of nodes in the BAM rather than explicitly defining operators to do it. By doing this, we avoid the need to use expensive lattice building algorithms, but at the expense of an additional set of weights to be computed and stored. We achieve the same final compression (as of van der Merwe's work) in a two-layer graph structure with less computational effort.

Another interesting work that is worth mentioning is Wille's attempt to combine the FCA and Conceptual Graphs (Semantic Networks) [Wille 1997] to make use of the benefits of both disciplines. Both conceptual graphs (semantic networks) and FCA have been used for knowledge representation and processing, and so it is desirable to combine them together. He showed how conceptual graphs and FCA may be combined to obtain a formalisation of Elementary Logic which is useful for knowledge representation and processing. In the paper [Wille 1997], he describes a process of translating conceptual graphs to concept lattices via formal contexts.

5.6 SUMMARY

In this chapter, the theory of FCA was presented with sufficient detail to understand the underlying principles of our approach. An analogy with natural language understanding by humans was discussed, and the suitability of FCA to represent concepts written in natural language text for the purpose of representing them in a computer for IR tasks was justified. Finally, a detailed literature review of the FCA-based IR models was given.

In all the approaches reviewed above, except van der Merwe's work, a concept is formulated in such a way that objects in its extent are documents (document identities) and attributes in its intent are keywords (controlled terms) present in the documents. In contrast to this approach, we attempt to capture concepts analogous to how the human brain might work, by defining objects of a concept as natural language textual symbols (i.e. terms or phrases) of real or abstract objects that are present in the content of the document, and attributes as the properties of those objects as mentioned in the text. This is similar to how van der Merwe has formulated concepts. The two-layer structure that van der Merwe obtained by compressing a concept lattice can be regarded as the one most related to the way we encode them in two layered data structures (the differences were highlighted above).

In addition, most of the reported models work by creating a single large concept lattice, based on one-valued contexts with or without conceptual scaling. Using a large single concept lattice demands high memory requirements and computational power for creating, traversing and maintaining the lattice (see [Kuznetsove & Ob"edkov 2001] and [Hemkemeier & Vallentin 2000] for reviews of complexity of Lattice building algorithms).

The need for high computational power and the difficulty of automatic or manual extraction of formal concepts have restricted the application of concept lattices to small-scale document collections. Even though the use of conceptual scaling has helped to overcome the sizing problem to some extent, it is only a solution for partial display of the full concept lattice. A major drawback of conceptual scaling as it is used in the reported models is that the scales have to be pre-computed. In contrast to the existing approaches, we represent each individual document by a separate independent concept lattice. Therefore, at any time during processing we interact with only one of those smaller concept lattices. In addition, each concept lattice is encoded in a two-layer neural network called a Bidirectional Associative Memory (BAM) to overcome lattice building and

navigation overheads. The next chapter is devoted to describe what a BAM is and how a concept lattice could be embedded in a BAM network.

CHAPTER 6 - BIDIRECTIONAL ASSOCIATIVE MEMORIES (BAMS)

One of the major problems that limit application of concept lattices to real world problem solving is the complexity of lattice building algorithms. A number of algorithms, both incremental and non-incremental, can be found in literature for generating concept lattices [Godin et al. 1995, Kuznetsov & Ob"edkov 2001, Hemkemier & Vallentin 1998]. The complexity of these algorithms increases with the sizes of M, G and L (the size of the concept lattice). The algorithm which has the smallest time complexity is the Nourine algorithm with $O((|G|+|M|)|G||L|)$ [Kuznetsov & Ob"edkov 2001]. In addition, as the concept lattice becomes bigger, it takes longer to traverse the lattice to extract specific concepts of interest. Therefore, it is our best interest to find a solution that can access the desired concepts in the lattice efficiently, while avoiding the use of a complex and expensive lattice building algorithm. We achieve this goal by embedding concept lattices of document/query representations in BAM structures. In this section, we describe what a BAM is, how a concept lattice can be embedded in a BAM structure, and how specific or generic concepts of a given set of objects or attributes can be extracted from a concept lattice encoded in a BAM without the burden of traversing the lattice.

6.1 WHAT IS A BAM?

Associative memories represent a class of neural networks that aim at modelling the association phenomenon. Based on the early models of Amari [Amari 1972] and Hopfield [Hopfield 1984], Kosko [Kosko 1987, Kosko 1988] proposed a bi-directional associative neural network called a Bidirectional Associative Memory (BAM). A BAM consists of two layers of neurons (Figure 6.1). The states (activities) of the first and the second layers (say containing k and l neurons respectively) are denoted by x_i (i=1,..., k) and y_j (j=1,...,l). The states x_i and y_j can be encoded in either binary (0 or 1) or bipolar (+1 or -1) encoding. Each (i^{th}) neuron of the first layer is connected to each (j^{th}) neuron of the second layer by a

connection weight. A real threshold $\theta_i^x(\theta_j^y)$ is assigned to the i^{th} neuron of the first layer (j^{th} neuron of the second layer). A number of different weight computation methods have been suggested for setting up the connection weights. Amongst these are the ones proposed by Kosko the originator of BAMs [Kosko 1987] and Bělohlávek [Bělohlávek 2000]. In this work, the latter one is used due to the reasons mentioned later in this chapter.

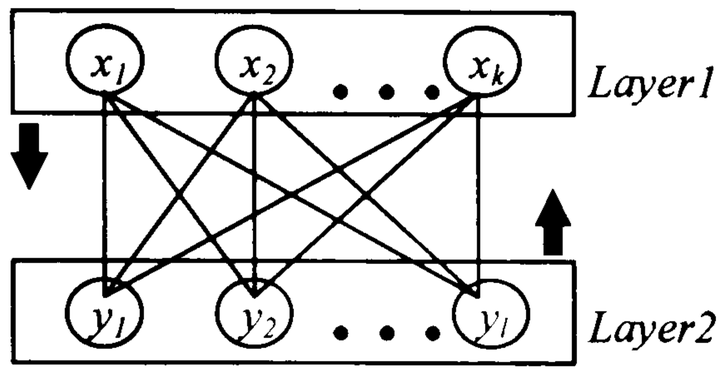


Figure 6.1 : Structure of a BAM

6.2 DYNAMICS OF A BAM

Given a pair $<X,Y> = <<x_1,\ldots,x_k>,<y_1,\ldots y_l>> \in \{0,1\}^k \times \{0,1\}^l$ of patterns of signals, the signal X is fed to the first layer to obtain a new pair $<X,Y^{\downarrow}>$, then $Y^{\downarrow}$ to the second layer to obtain $<X^{\downarrow},Y^{\downarrow}>$, and so on. The dynamics is given by the formulas:

$$y'_j = \begin{cases} 1 & for \quad \sum_{i=1}^{k} w_{ij}x_i > \theta_j^y \\ y_j & for \quad \sum_{i=1}^{k} w_{ij}x_i = \theta_j^y \\ 0 & for \quad \sum_{i=1}^{k} w_{ij}x_i < \theta_j^y \end{cases} \qquad x'_i = \begin{cases} 1 & for \quad \sum_{j=1}^{l} w_{ij}y_j > \theta_i^x \\ x_i & for \quad \sum_{j=1}^{l} w_{ij}y_j = \theta_i^x \\ 0 & for \quad \sum_{j=1}^{l} w_{ij}y_j < \theta_i^x \end{cases}$$

The pair of patterns $<X,Y>$ is called a stable point if the states of neurons, when set to $<X,Y>$, do not change under the defined dynamics. Using an appropriate energy function, [Kosko 1988] proved that such a network is stable for any weights w_{ij} and any thresholds θ_i^x, θ_j^y. Stability means that given any initial pattern $<X,Y>$ of signals, the net eventually stops after a finite number of steps (feeding signal from layer to layer back and forth).

The aim of learning in the context of associative memories is to set the parameters of the network so that a prescribed training set of patterns is related in some way to the set of all stable points. However, not all training patterns (pairs) become stable points. Kosko

proposed a form of Hebbian learning, by which the weights w_{ij} are determined from the training set $T = \{<X^p,Y^p> \mid p \in P\}$ by $w_{ij} = \sum_{p \in P} bip(x_i^p) bip(y_j^p)$, where *bip()* maps 1 to 1 and 0 to -1, i.e. changing the binary encoding to a bipolar one, and $P = \{1,2,3\ldots,$number of patterns in $T\}$. Thresholds of all nodes are set to zero.

Using this encoding, Kosko showed that the minimal two-layer nonlinear feedback network that achieves the task of storing and recalling paired-data associations (A_i,B_i) is the BAM [Kosko 1988].

6.3 A Review Of Research on BAMs

Kosko's original BAM suffers from low storage capacity, low recall reliability, and highly spurious memories [Shi et al. 1998]. Many efforts have been made to improve the performance of the original BAM, since its publication by Kosko [Kosko 1987, Kosko 1988]. Theese improvements have been mainly in the directions of developing new learning algorithms [Wang & Don 1995] and enhancing the original architecture by adding dummy neurons, more layers, or interconnections among neurons inside each layer [Wang 1996] (find more references in Shi et al. 1998 and in Wang 1996). They attempt to achieve high storage capacities, high quality recall from noisy patterns and low spurious memories [Shi et al. 1998, Wang 1996]. Hardware implementations of the BAMs have also been proposed [find references in Leung et al. 1997].

Most of these models assume logical symmetry of interconnections, that the weights from one layer to the other are the same as the weights from the latter layer to the former layer. This symmetry of interconnections hampers the efficiency of the BAMs capability in pattern storage and recall and limits their use for knowledge representation and inference [Shi et al. 1998]. Xu and all [as cited in Shi et al. 1998] have proposed an asymmetrical BAM model (ABAM) to overcome this drawback, but its requirement of linear independence among stored patterns limits its storage capacity [Shi et al. 1998]. Shi, Zhao

and Zhuang [Shi et al. 1998] propose another general model for a BAM that does not assume this linear independence requirement.

Wang [Wang 1996] designs two BAMs, a linear BAM and a nonlinear BAM, by using an optimal associative memory matrix in place of the commonly used Hebbian or quasi-correlation matrix. He shows that the introduction of a nonlinear characteristic enhances the ability of the BAM to suppress the noise occurring in the output patterns, and reduces largely the problem of spurious memories.

In addition to the drawbacks discussed above, the restriction to binary or bipolar pattern pairs and the limitation to two input/output patterns have led BAMs to be less popular and less utilised, as most applications require to operate on more (more than two) real-valued input/output patterns. Various researchers have investigated these issues to overcome the limitations. Abedin and Ahsen [Abedin & Ahson 1993] propose a new coding strategy that enables the BAM to operate on real-valued inputs and Humpert [find reference in Kulkarni & Yazdanpanahi 1993] has suggested a generalisation of the BAM to a bi-directional associative memory with several input/output patterns (called BAMg). Kulkarni and Yazdanpanahi [Kulkarni & Yazdanpanahi 1993] investigated the interconnections and updating of neuron fields of a number of different BAMg architectures in the image processing domain to store and retrieve sets of images, and achieved good performance.

Despite the limitations discussed above, BAMs have been used in several application domains successfully. Bavarian [Bavarian 1988] reports the use of BAMs in intelligent systems, Mathai and Upadhyaya [Mathai & Upadhyaya 1989] report the use of BAMs in spectral signature recognition, and Wu et al. [Wu et al. 1990] report the use of BAMs in image classification to name a few.

6.4 BAMs for Encoding Concept Lattices

It is conceived that, in general, the concept interpretation of the patterns of states is not possible as there can be stable points that cannot be interpreted as formal concepts. The following example by Bělohlávek 2000 [Bělohlávek 2000] illustrates this fact.

E.g. Consider k=1, l=2, w_{11}=1 and w_{12}=-2 and all thresholds set to zero. Then {<0,<0,0>>, <0,<0,1>>, <0,<1,1>>, <1,<1,0>>} is the set of all the stable points. Now consider the pairs $\langle A_1,B_1\rangle = \langle 0,\langle 0,0\rangle\rangle$ and $\langle A_2,B_2\rangle = \langle 1,\langle 1,0\rangle\rangle$. We have $A_1 \subset A_2$ but $B_1 \not\supset B_2$, which contradicts the rule for subsumption relation (valid for formal concepts) that the more common objects there are, the fewer the common properties.

On the other hand take for example, k=l=2, w_{11}=w_{22}=1, w_{12}=w_{21}=-3 and all thresholds set to -1/2. Then the set {<<0,0>, <1,1>>, <<0,0>, <0,1>>, <<1,0>, <1,0>>, <<1,1>, <0,0>>} is the set of all the stable points (denoted by Stab(W, θ)). It can be verified easily that this set of stable points correspond exactly to the concept lattice defined by the given concepts/stable concepts.

This had raised the question whether there is a BAM corresponding to each concept lattice $\underline{B}(G,M,I)$ such that the set of all concepts of $\underline{B}(G,M,I)$ is precisely the set of all the stable points of the BAM. Using the weighting scheme given below, Bělohlávek [Bělohlávek 2000] has proved that there is a BAM given by the weights W and thresholds θ such that $Stab(W, \theta)=\{\langle A,B\rangle \mid \langle A,B\rangle \in \underline{B}(G,M,I)\}$ corresponding to the concept lattice given by the context $\langle G,M,I\rangle$ with G and M finite. This is an interesting and useful property of the BAM that drew our attention to use BAMs for encoding concept lattices.

Following is the weight training/computation formula used by Bělohlávek:

$$w_{ij} = \begin{cases} 1 & if \langle g_i, m_j \rangle \in I \\ -q & if \langle g_i, m_j \rangle \notin I \end{cases} \quad \text{for } i=1,\ldots,k,\ j=1,\ldots,l$$

where $q = max\{k,l\}+1$. All the thresholds are set to -1/2.

6.5 A TRAINING SET FOR A BAM TO LEARN A CONCEPT LATTICE

A training set T consists of a set of concepts in the form (A,B), where elements of A come from the set of objects G and elements of B from the set of attributes M of the context (G,M,I). Here the set G contains all the objects necessary to define the extents A of the concepts (A,B) in the training set and M contains all the attributes necessary to define the intents B of the concepts (A,B) in the training set. This means the elements in T are defined on (G,M,I). However, a set of training pairs $T = \{(A^p,B^p) \mid p \in P, P \subset \mathbf{N}\}$ needs its training patterns to obey the fundamental rule (completeness constraint) of the formal concept $A^{\downarrow}=B$ and $B^{\downarrow}=A$, in order for it to be a conceptually consistent training set to form a concept lattice. This condition, as described before, is what defines the subsumption relation ($\leq$) to form the sub-super concept hierarchy between formal concepts. It ensures that the set of formal concepts in the training set are storable in the BAM. Bělohlávek has stated (Theorem 2 of Bělohlávek 2000) the necessary and sufficient conditions for a training set T to be conceptually consistent as:

$$A^p = \{g \in G \mid \forall m \in B^p \exists p' \in P : g \in A^{p'}, m \in B^{p'}\}$$
$$B^p = \{m \in M \mid \forall g \in A^p \exists p' \in P : g \in A^{p'}, m \in B^{p'}\}$$

Any given set of arbitrary concepts (a training set T) which satisfies the above condition defines a concept lattice on the context (G,M,I) formed by all the objects and attributes of the concepts contained in T. This training set is a subset of $B(G,M,I)$ (set of all the concepts of the context (G,M,I)). It also is a subset of the concept lattice $\underline{B}(G,M,I)$ of the context (G,M,I).

Described above is theoretically what a training set should comply to. However, Bělohlávek's weighting scheme does not need to pay explicit attention to the above description. It only needs to know all the individual objects in the context and all of the attributes possessed by each of them. Given this information, we can simply set the weights between object nodes and attribute nodes to +1 if the object possesses the attribute

or $-q$ (where $q=max(k,l)+1$) if it does not posses the attribute. An example of weight settings necessary for learning a given concept lattice is shown in Figure 6.3.

6.6 What Actually a BAM Learns and Returns ?

6.6.1 Learning

Given a set of training pairs of object sets and attribute sets, that satisfy the completeness constraint (i.e. they are formal concepts in a given context), a BAM learns the underlying concept lattice. As mentioned before, join and meet concepts that are inferred by the patterns in the training set are automatically detected and learnt by the BAM. For example, given four training patterns, one for each object: *dog*, *cat*, *frog* and *fish* (with all of their corresponding attributes as indicated in Figure 5.4) the concept lattice given in Figure 5.4 is derived.

6.6.2 Stable Points

Stable points are just the nodes in the underlying concept lattice that the training set infers. This indeed is what Bělohlávek reports in his paper [Bělohlávek 2000]. In terms of training patterns, it creates stable points for the patterns or concepts that are conceptually consistent (as stated in Section 6.5) and concepts that are inferred from them (i.e. join and meet concepts).

An arbitrary pair consisting of a set of objects and a set of attributes extracted from text does not necessarily become a stable point in a BAM, as it might not comply with the completeness constraint and so will not form a node in the concept lattice. It may however be a part of a concept represented by a node. The example given below illustrates this further.

E.g. consider the sample context of planets given in Table 6.1. A list of objects (names of planets) and their properties that can be extracted from this context is given below.

<{Me},{ss,sn,mn}>, <{V},{ss,sn,mn}>, <{E},{ss,dn,my}>, <{Ma},{ss,dn,my}>, <{J},{sl,df,my}>, <{S},{sl,df,my}>, <{U},{sm,df,my}>, <{N},{sm,df,my}>, <{P},{ss,df,my}>

This list can be used as a training set for a BAM to learn the underlying concept lattice (Figure 6.2) using the Bělohlávek's algorithm. As can be seen in Figure 6.2, except for

Planet	**Size**			**Distance from Sun**		**Moon**	
	small	**medium**	**large**	**near**	**far**	**yes**	**no**
Mercury	×			×			×
Venus	×			×			×
Earth	×			×		×	
Mars	×			×		×	
Jupiter			×		×	×	
Saturn			×		×	×	
Uranus		×			×	×	
Pluto	×				×	×	
Neptune		×			×	×	

Table 6.1 : Context of the Planets

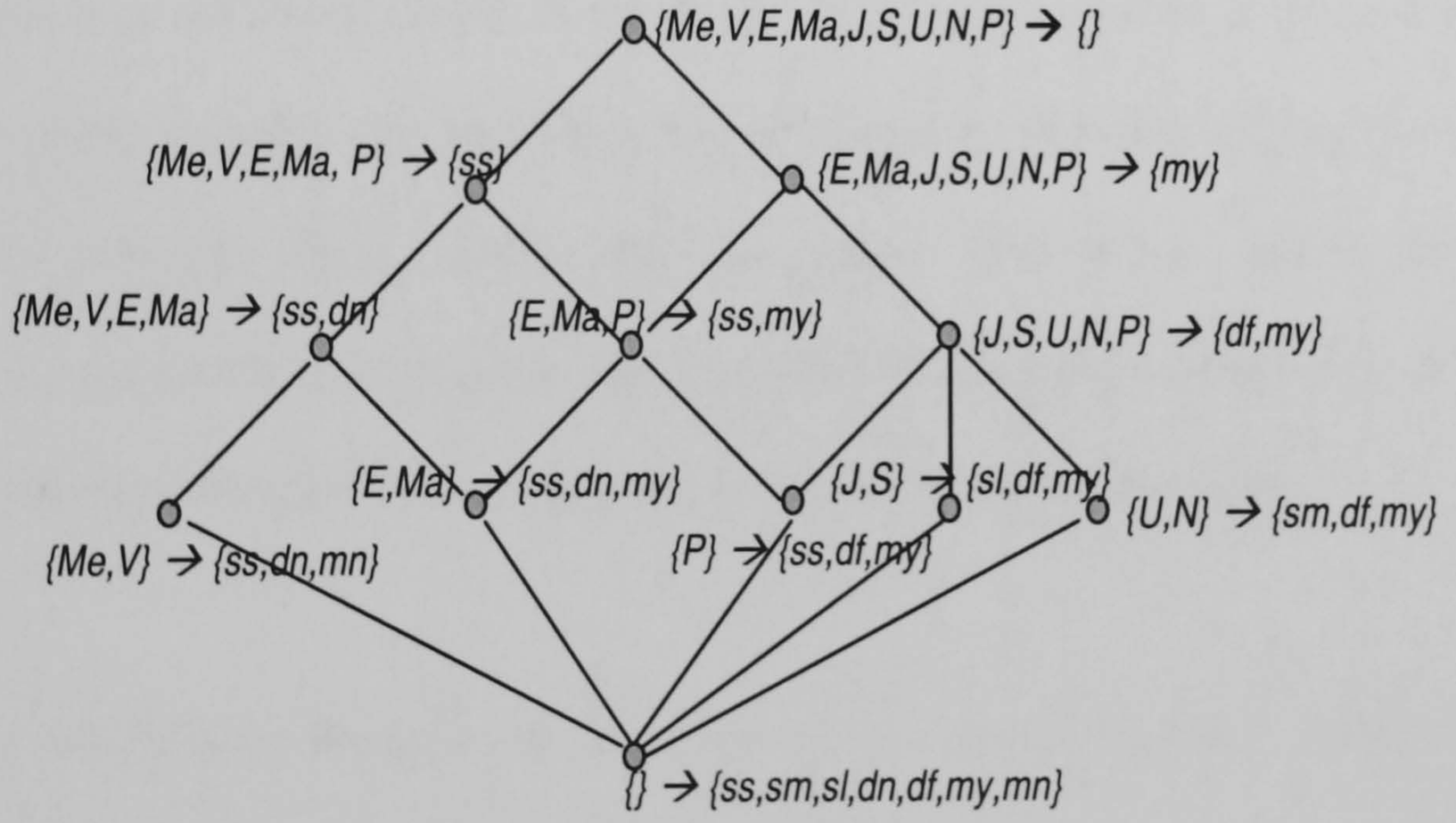


Figure 6.2 : Concept Lattice of the Context of Planets

$<\{P\},\{ss,df,my\}>$ no node is explicitly created (i.e. no stable point is created) for representing other training patterns. This is because those patterns are not conceptually consistent. Therefore, one cannot use them (as they are) to train a BAM if Kosko's algorithm (Section 6.2) is used. They need to be pre-processed to create a consistent set of patterns in this case. On the other hand, the following set of patterns are conceptually consistent, and therefore they can be used to train a BAM with the given concept lattice using Kosko's algorithm.

<{Me,V},{ss,dn,mn}>, <{E,Ma},{ss,dn,my}>, <{P},{ss,df,my}>, <{U,N},{sm,df,my}>, <{J,S},{sl,df,my}>

As can be seen in Figure 6.2, these correspond to nodes in the concept lattice and are stable points in the BAM. Other nodes in Figure 6.2 (e.g. $\{Me,V,E,Ma\} \rightarrow \{ss,dn\}$) can be derived from this set of training patterns. They also become stable points even though they are not present explicitly as training patterns in the original training set.

6.6.3 An Important Property of a BAM

Unlike purely feed-forward neural network architectures, BAMs can accept input patterns from either layer. We can present a pattern with objects to the first layer (referred to as the object layer) or a pattern with attributes to the second layer (referred to as the attribute layer). The BAM returns the most specific concept containing all the objects of the input pattern in the first case and the most generic concept containing all the attributes of the input pattern in the second case. A formal proof of this property is given below. We use this interesting property for retrieving specific/generic concepts during the extraction of candidate concepts from query and document BAMs to match between them (Section 7.1.3). Notice that, a given input to a BAM may lead to retrieving an inferred *join* or *meet* concept instead of a concept/pattern used for training the BAM.

Proof:

Lets take $B(G,M,I)$ be the set of all concepts of the context $<G,M,I>$, where G is the set of all objects in the context, M is the set of all attributes in the context and I is the incident relation. Let us assume that a BAM is trained with the concept lattice of this context. Then the stable points in the BAM represent (with one-to-one correspondence) all formal concepts (A,B) of the context $<G,M,I>$, and thus they hold the completeness constraint $A^{\downarrow}=B$ and $B^{\downarrow}=A$,

where $A^{\downarrow}=\{m \in M \mid \forall g \in A \text{ s.t. } <g,m> \in I\}$ and $B^{\downarrow}=\{g \in G \mid \forall m \in B \text{ s.t. } <g,m> \in I\}$

Now, take an arbitrary object g_o. Assume, without loss of generality, that the BAM gives $\{g_o,g_i\} \rightarrow \{m_x,m_y\}$ when the object g_o is presented to the object layer of the BAM. This means $\{g_o,g_i\} \rightarrow \{m_x,m_y\}$ is a formal concept. Then by the completeness constraint ($A^{\downarrow}=B$ and $B^{\downarrow}=A$), we get:

$$\{g_o,g_i\}^{\downarrow} = \{m_x,m_y\} \text{ ----------(1) and } \{m_x,m_y\}^{\downarrow} = \{g_o,g_i\} \text{ ----------(2)}$$

Now we will prove that no other formal concept more specific to that of $\{g_o,g_i\} \rightarrow \{m_x,m_y\}$ exists in the concept lattice containing the object g_o.

By definition, a formal concept which is more specific to a second formal concept should contain either less objects or/and more attributes to that of the second formal concept. We will prove that no such subconcept containing the object g_o can exist in the lattice, given the assumption that the BAM returns $\{g_o,g_i\} \rightarrow \{m_x,m_y\}$ for the input $\{g_o\}$.

There are three possible forms that a subconcept of a given concept can take.

1. a concept with less objects and same attributes
2. a concept with same objects and more attributes
3. a concept with fewer objects and more attributes.

Proving non-existence of a subconcept under the first two cases is straightforward. Given the assumption that $\{g_o,g_i\} \rightarrow \{m_x,m_y\}$ is a formal concept, we can easily prove that the above two ((1) and (2)) are not formal concepts in the given context and therefore they do not exist in the lattice.

Case 1 :

Lets take $\{g_o\} \rightarrow \{m_x,m_y\}>$ to be a formal concept in the concept lattice. This is more specific to the concept $\{g_o,g_i\} \rightarrow \{m_x,m_y\}$ as it satisfies the subsumption relation $A_1 \subseteq A_2$ (and $B_1 \supseteq B_2$) (Section 5.2.2).

Since this was taken to be a formal concept, from the completeness constraint we get:

$$\{m_x,m_y\}^{\downarrow} = \{g_o\} \text{----------(3)}$$

(2) and (3) leads to a contradiction, meaning that $\{g_o\} \rightarrow \{m_x, m_y\}$ cannot be a formal concept of the context given the assumption that $\{g_o, g_i\} \rightarrow \{m_x, m_y\}$ is a formal concept.

Case2:

Now take $\{g_o, g_i\} \rightarrow \{m_x, m_y, m_z\}$ to be a formal concept of the context. This is more specific to the concept $\{g_o, g_i\} \rightarrow \{m_x, m_y\}$ as it satisfies the subsumption relation $B_1 \supseteq B_2$ (and $A_1 \subseteq A_2$).

Since this was taken to be a formal concept, from the completeness constraint, we get

$$\{g_o, g_i\}^{\downarrow} = \{m_x, m_y, m_z\} \quad \text{----------(4)}$$

(1) and (4) leads to a contradiction, meaning that $\{g_o, g_i\} \rightarrow \{m_x, m_y, m_z\}$ cannot be a formal concept of the content given the assumption that $\{g_o, g_i\} \rightarrow \{m_x, m_y\}$ is a formal concept.

Case 3:

Finally, take without loss of generality, that $\{g_o\} \rightarrow \{m_x, m_y, m_z\}$ to be a formal concept in the concept lattice. This does not lead to a contradiction with the conditions (1) and (2) as in the first two cases and hence is a potential candidate to be a formal concept that can possibly exist in the lattice. Also this is a more specific concept to $\{g_o, g_i\} \rightarrow \{m_x, m_y\}$ as it satisfies the subsumption relation $B_1 \supseteq B_2$ (and $A_1 \subseteq A_2$).

Now, we will prove that, this formal concept cannot exist in the lattice given the assumption that the concept $\{g_o, g_i\} \rightarrow \{m_i, m_j\}$ exists and is returned when the object g_o (only) is presented to the object layer.

Since $\{g_o, g_i\} \rightarrow \{m_x, m_y\}$ was assumed to be a formal concept, both g_o and g_i must possess at least the attributes m_x and m_y. The concept $\{g_o\} \rightarrow \{m_x, m_y, m_z\}$ says that the object g_o possesses m_z as well.

Let us see what the BAM returns according to the dynamics of the BAM. Note that we use Bělohlávek 's weight computation rule given in Section 6.4. Recall, a weight between an

object and an attribute nodes are set to one (+1) if the object possesses the attribute. Otherwise the weight is set to the negative value of the maximum number of nodes in either layer +1 (see Figure 6.3). A node fires if the weighted sum of incomes ($\sum x_i w_i$) exceeds -1/2.

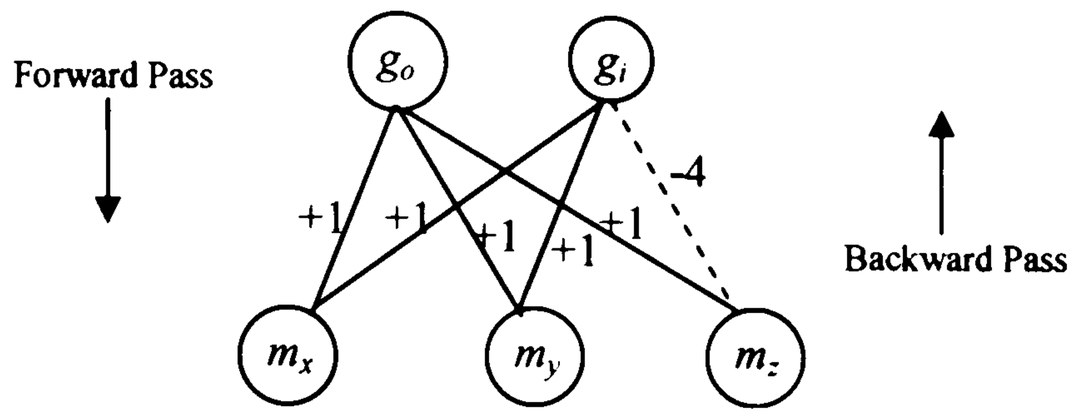


Figure 6.3 : BAM with Weight Settings for Case 3 of the Proof

1st Forward pass:

g_o is presented to the object layer (i.e. $g_o = 1$ and $g_i = 0$)

=> only the nodes correspond to m_x, m_y and m_z fires.

1st Backward pass:

(m_x, m_y and m_z are presented to the attribute layer)

=> only the node correspond to g_o is fired.

Dynamics of the BAM does not change thereafter, and so {1,0} → {1,1,1} is a stable state, i.e. the BAM returns $\{g_o\} \rightarrow \{m_x, m_y, m_z\}$ when g_o is presented. This contradicts with our very first assumption that the BAM returns $\{g_o, g_i\} \rightarrow \{m_x, m_y\}$ for the input object g_o. Hence, if the BAM returns the concept $\{g_o, g_i\} \rightarrow \{m_x, m_y\}$ for the input object $\{g_o\}$ as we first assumed, then the concept $\{g_o\} \rightarrow \{m_x, m_y, m_z\}$ cannot exist in the concept lattice.

The cases 1, 2 and 3 prove that given an input object (g_o) a BAM returns the most specific concept in the lattice containing the given input object (g_o). This property holds even for a given any subset of objects and the proof is the same.

With a similar argument we can prove that given a set of attributes as the input to the attribute layer of a BAM returns the most generic concept containing the given set of attributes (proof is not given here)

6.7 A BAM IN OPERATION - AN EXAMPLE

The following two figures (Figure 6.4 and Figure 6.5) illustrate the BAM's action to learn the associated concept lattice given in Figure 6.2 based on the context given in the Table 6.1. Firstly, Figure 6.4 illustrates what the nodes represent and how the weights are set between object nodes and attribute nodes. Note that, the nodes in grey are the active (firing) nodes, dashed-lines represent links with negative (*-q*) weights, and solid lines represent links with weight +1. Secondly, Figure 6.4 and Figure 6.5 together show how the BAM works in its forward pass and backward pass. Initially, the object *Ma* is given as the input to the BAM (i.e. activity of *Ma* is set to +1). This makes the attributes *ss*, *dn* and *my* fire in its forward pass and those attributes in turn make both *Ma* and *E* active in its backward pass. The recurrent process stops here (i.e. further recurrences will not cause any change to its current state). Lastly, Figure 6.4 and Figure 6.5 together demonstrate the

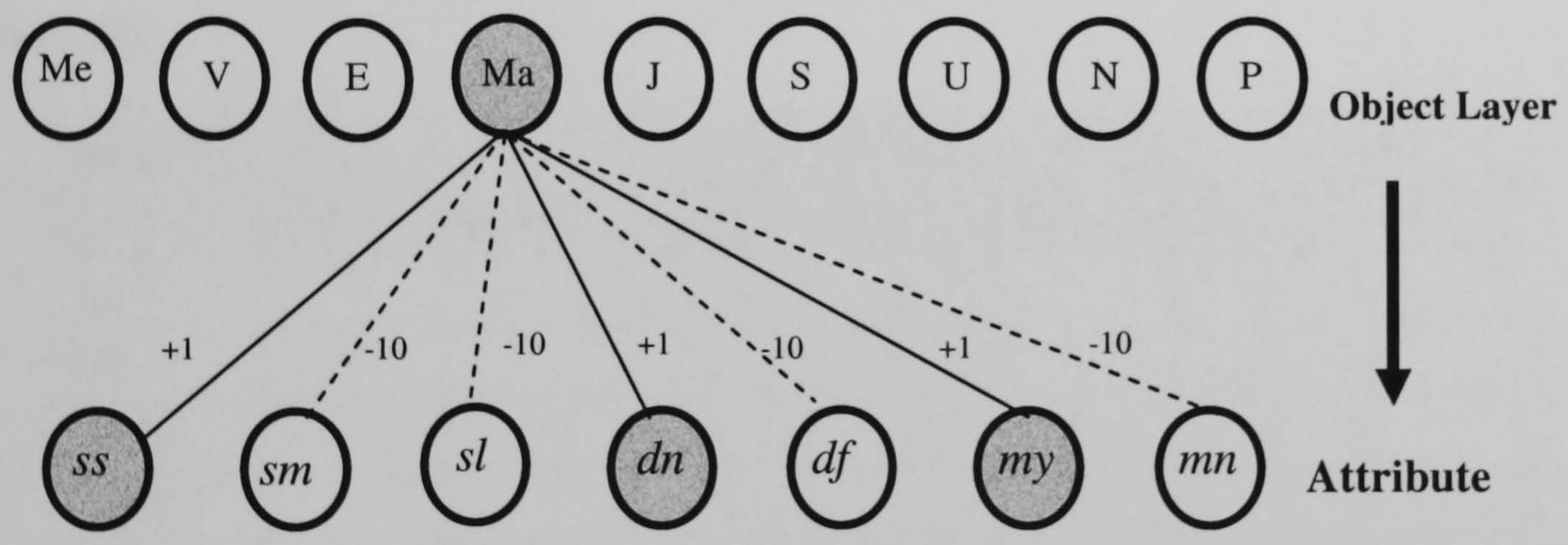


Figure 6.5 : BAM in Operation - Forward Pass

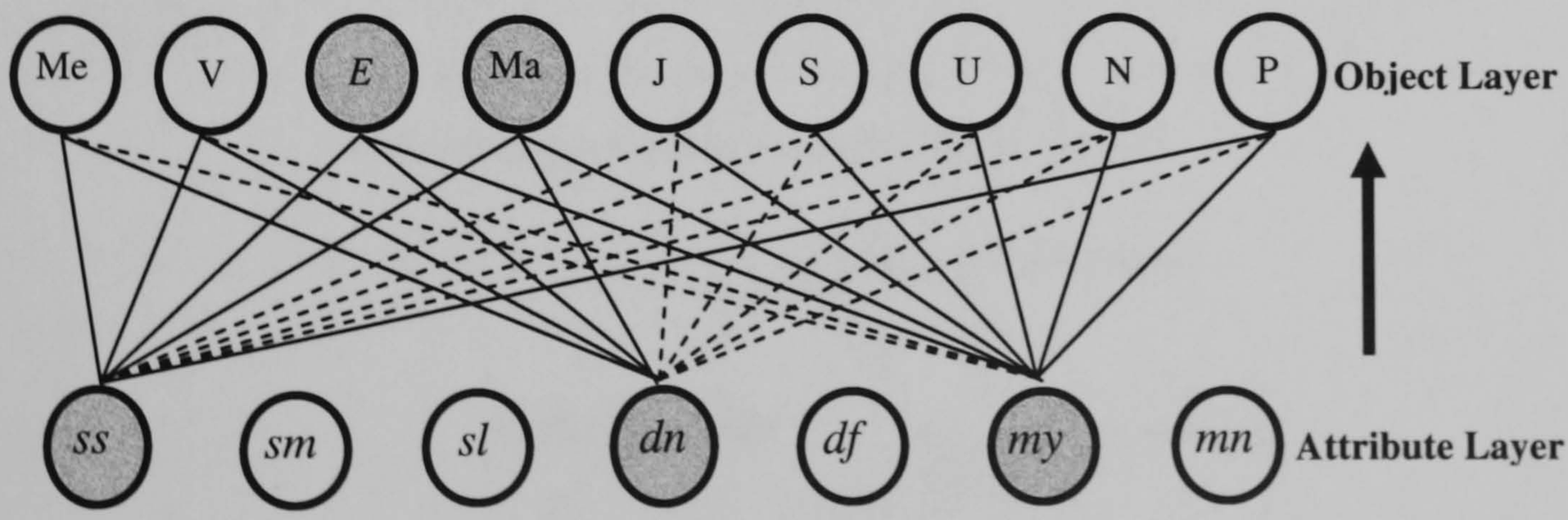


Figure 6.5 : BAM in Operation - Backward Pass

property that for a given set of input objects (in this case a pattern with only *Ma* active), the BAM returns the most specific concept available in the lattice containing the given input objects (*Ma*) in its extent (in this case $\{E,Ma\} \rightarrow \{ss,dn,my\}$). The reverse of this,

i.e. given a set of attributes the BAM returns the most generic concept containing the given input attributes, can be demonstrated similarly (not shown here). Note that only the links from active node(s) in question are shown.

6.8 Adding New Concepts to an Existing (Trained) BAM

Adding new concepts into a concept lattice, in general, involves updating many nodes and therefore is a computationally expensive process. However, the task is much easier and faster with a BAM, in particular with Bělohlávek's weight-setting mechanism. It only needs a node to be added to the corresponding layer for each new object or attribute and links recalculated. The following example illustrates how this takes place.

Consider a reduced context (Table 6.2) obtained by removing the objects *Me* and *V* from the context given in Table 6.1. The concept lattice of this context (Table 6.2) is given in Figure 6.6.

Planet	Size			Distance from Sun		Moon	
	small	medium	large	near	far	yes	no
Earth	×			×		×	
Mars	×			×		×	
Jupiter			×		×	×	
Saturn			×		×	×	
Uranus		×			×	×	
Pluto	×				×	×	
Neptune		×			×	×	

Table 6.2 : Context of the Planets Excluding Mercury and Venus

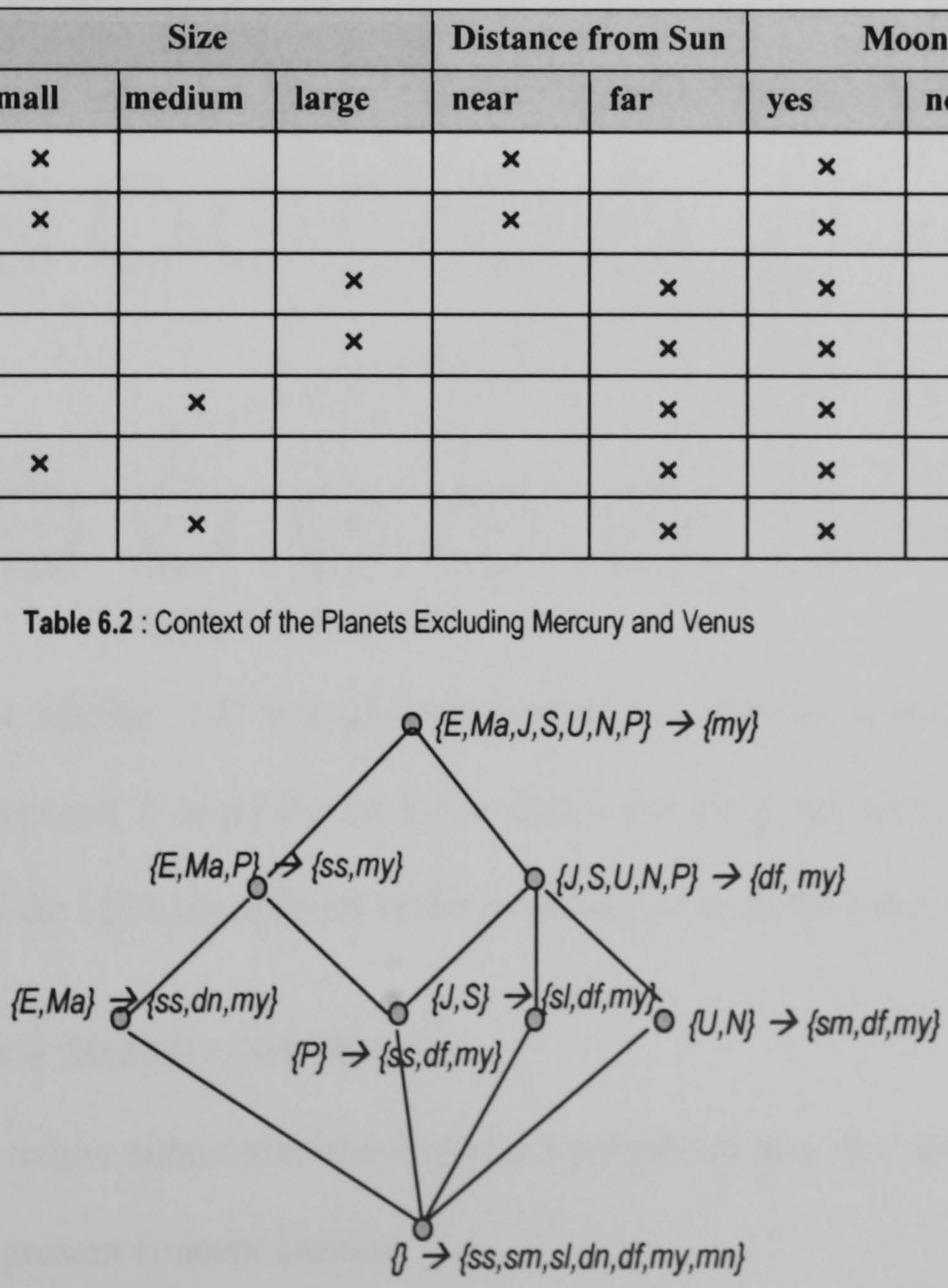


Figure 6.6 : Concept Lattice of the Context in Table 6.2

A BAM that learns this (reduced) concept lattice do not have nodes in the object layer for *Me* and *V*, and in the attribute layer for *mn*. Negative weight values of the BAM in this case are -8 (i.e. $-q = -(\max(k,l)+1) = -8$).

Let us consider adding the planet *Me* (mercury) first; i.e. add $\{Me\} \rightarrow \{ss,dn,mn\}$ into the BAM which has already learnt the lattice shown in Figure 6.6. This requires two nodes to be addeded into the BAM, one to the object layer for representing the object *Me* and the other to the attribute layer for representing the attribute *mn* (shown in dotted circles in Figure 6.7 below). Then, the weights of the links between the node *Me* and its attribute nodes *ss*, *dn* and *mn* are set to one (shown in solid lines), and the weights of the links from the node *Me* to the other attributes are set to $-(\max(k,l)+1)$ (shown in scattered lines). Also, the weights of the links from other object nodes (other than the node being added) to the attribute nodes that they do not possess should be updated with $-(\max(k,l)+1)$ as they do depend on the number of nodes in the BAM. But, weights of the links from other object nodes to the attribute nodes that they possess remain the same and so do not need updating.

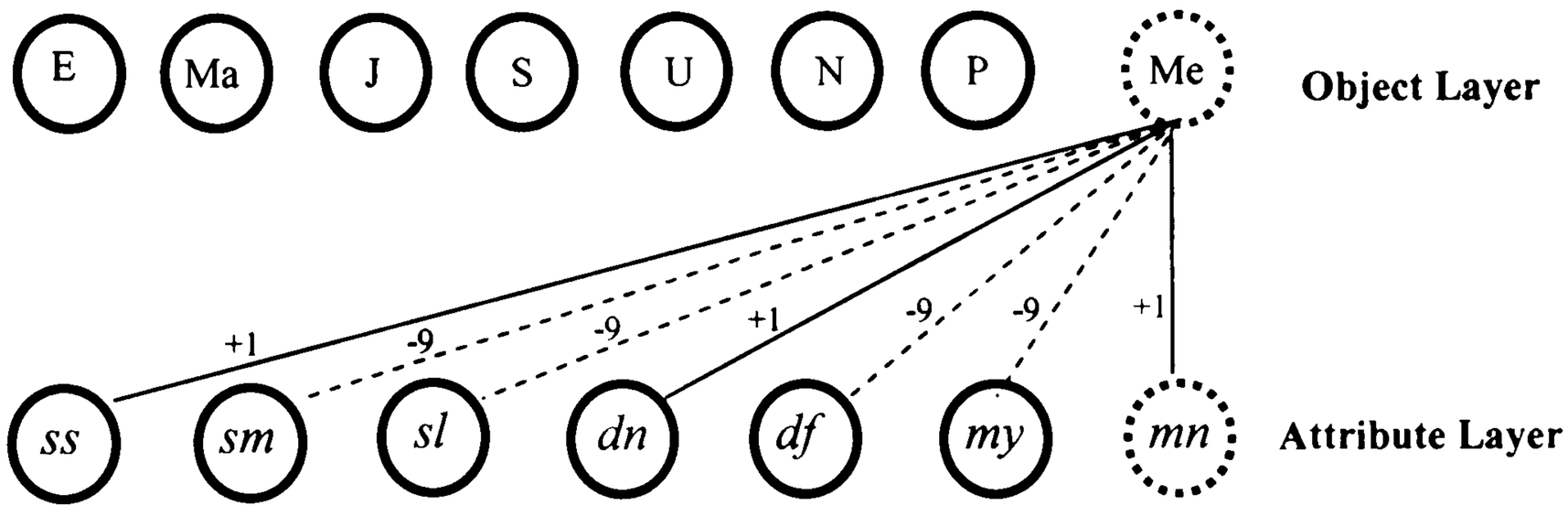


Figure 6.7 : Updating a BAM

Adding Venus (i.e. adding $<\{V\} \rightarrow \{ss,dn,mn\}>$) needs the addition of only one node to the object layer to represent *V* as all the attributes that it possesses are already present in the BAM. Weights of the links are updated in the same way as described above.

6.9 Summary of Important Points

The points listed below summarise the important properties that one should know when using BAMs to represent concept lattices.

1. An arbitrary training pattern does not necessarily become a stable point in the BAM. Only the patterns generated by enforcing completeness constraint ($A^{\downarrow}=B$ and $B^{\downarrow}=A$) become stable points in the BAM.

2. A storable set of training patterns should consist only of patterns that satisfy the condition $A^{\downarrow}=B$ and $B^{\downarrow}=A$.

3. A BAM learns a complete concept lattice, not just a set of training patterns, i.e. in addition to the stable points created for each of the (conceptually consistent) training patterns, a BAM also creates stable points for *meet* and *join* concepts that are inferred by those training patterns.

4. When an input pattern consisting of some objects is presented to the object layer of a BAM, it returns the most specific concept as its output containing all the input objects in its extent. Conversely, when an input pattern consisting of some attributes is presented to the attribute layer of a BAM, it returns the most generic concept containing all the input attributes in its intent.

5. It is possible for a BAM to return a concept with no objects and all attributes, or a concept with all objects and no attributes. (These represent the lowermost/minimal and uppermost/maximal elements of the concept lattice).

6.10 SUMMARY

In this chapter, we described Bidirectional Associated Memories and how a concept lattice could be embedded in a BAM structure by training it with a set of training patterns (formal concepts). Bělohlávek's weight setting algorithm (Section 6.4) enables a BAM to learn exactly all the formal concepts in a given concept lattice as its stable points. This is an interesting and useful feature of the BAM that helps us to avoid using complex lattice building algorithms. In addition, the ability of a BAM to return the most specific or generic concept that exists in the learnt concept lattice, given a set of objects or attributes, is an extremely useful property of the BAM, in particular, it helps avoiding the use of conventional lattice traversing algorithms for searching desired concepts. How these ideas are used to design our IR model is described in the next chapter.

CHAPTER 7 - THE CONCEPTUAL MODEL

This chapter is intended to present the design of our conceptual model. Describing the design of an IR system involves explaining how the documents in the target source are characterised; how the queries are formulated and presented to the system (Query Language); how the core process of comparing queries with documents is carried out to find out which documents to retrieve (matching strategy); how and in which order the documents found to be useful (by the system) are presented to the user (ranking); and how the decisions of the user about the relevance of the retrieved documents are used for learning or query reformulation. These issues should be described in the context of what the model is intended or designed to achieve. Recall that our primary objective is to use more informative constructs (concepts), which correspond to how human thinking and understanding might work, to represent the concepts, ideas or thoughts present in documents, and to employ the representations created with these more informative constructs to achieve more elaborate "*concept matching*". The theories and techniques underlying our model were presented in Chapters 5 and 6, and in this chapter we present how they are used to design a prototype system to achieve our goals. A complete description of the design requires detailed treatment of all its components, together with some implementation detail. However, before flooding the reader with implementation detail, we first describe our design in this chapter at a conceptual level, in order to give the reader a high level view of what the model is about rather than how it is implemented. Within this chapter and the next, details of all the features and components of the model are covered, including the crucial decisions made to realise our goal of concept matching within the framework of FCA, and also the simplifications necessary for making the implementation feasible. Justifications for such decisions are given at the relevant places in the text.

7.1 Concept Matching

A node in a Concept Lattice represents a formal concept which can be thought of as analogous to a concept that might arise in the human brain during the reading of a document. The process of reasoning about the usefulness of a given document to an information need (query) can therefore be achieved based on how similar the nodes in a query concept lattice are to the nodes in a document concept lattice (see Figure 7.1).

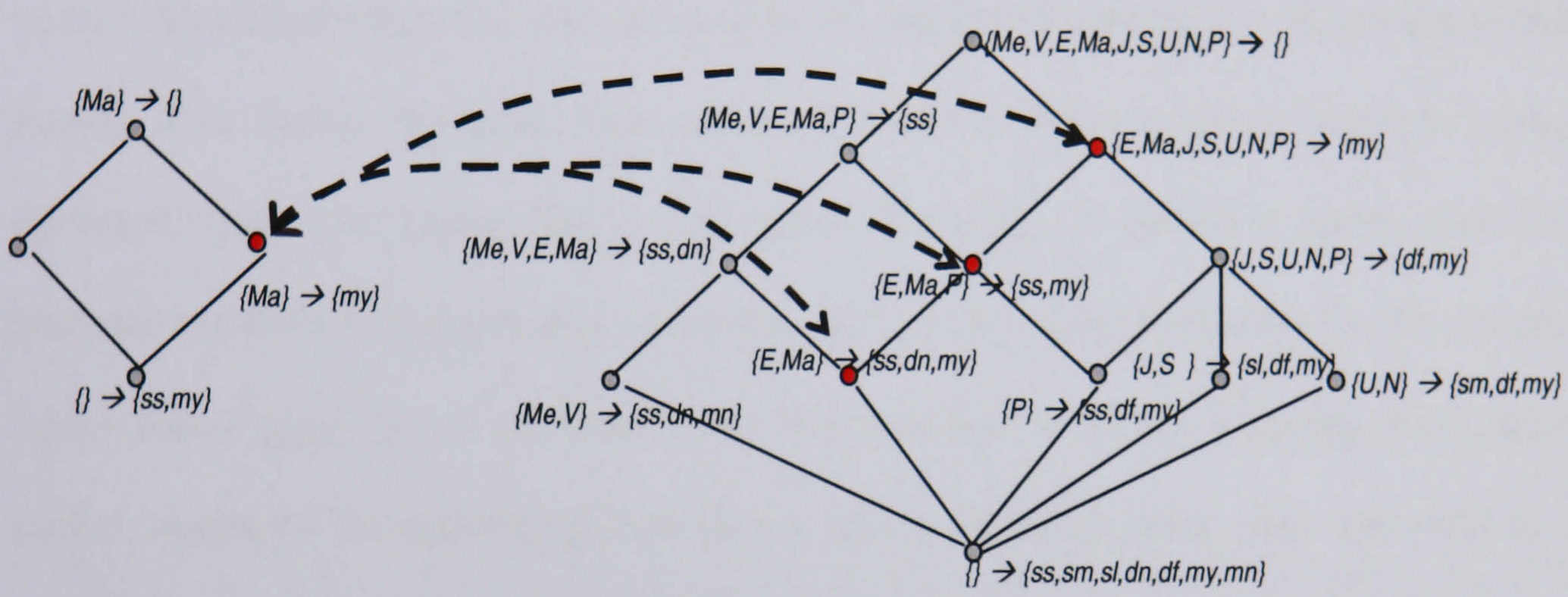


Figure 7.1 : Node Matching between Two Concept Lattices

Comparing query nodes with document nodes in this scenario is not as simple and straightforward as comparing just simple terms or keywords. We need to maintain the consistency of our treatment of certain terms as objects and certain terms as attributes. In addition, we may need to take into account the superiority/generality of concepts within the concept hierarchies in order to match more specific concepts and also to avoid duplications. Problems of natural language such as synonymy, polesemy and other problems related to the variability of vocabulary cause mismatches between components (objects and attributes) of concepts, which lead to concept mismatches. In addition, size variability between query and document concepts in terms of numbers of objects and attributes means that a complete match between a query and a document node is impossible.

7.1.1 Partial Matching

Queries in general are short expressions of information needs. Documents on the other hand are much longer and more descriptive than queries. Therefore a document lattice is likely to have more nodes and a more detailed order hierarchy between them than a query lattice. Note that it is the containment of common attributes or objects that relates a given node to another in a hierarchy according to the order relation. This causes two problems. Firstly, it leads certain nodes that are supposed to represent the same idea to have different terms (objects and elements) and also a different number of elements in their extents and/or intents. This means that most often a partial match is likely to occur between a query-document node pair rather than a full match. Secondly, it causes a query node to be partially matched with more than one node, often in the same hierarchy, in the document lattice (see Figure 7.1). A solution to the first problem is first to measure the degree of partial similarity (commonality) between a query-document node pair and then to use similarity degrees of all (partial) node matches to compute the final retrieval status value (RSV) for the query-document pair. A solution to the second problem is to use relationships in the hierarchy to match the most specific concepts whenever a query node (partially) matches with more than one related document node (Section 7.1.3).

Definition

We define a partial match between two concepts as a concept *m*, in which the objects in its extent are those common to the query and the document extents, and the attributes in its intent are those common to the query and the document intents. This can be formally expressed as:

Let $q = \langle A,B \rangle$ and $d = \langle C,D \rangle$ be two formal concepts, where A,B,C,D are sets of terms in which terms in A and C are interpreted as objects, and terms in B and D are interpreted as attributes according to the FCA formalism, then the partial match between the two concepts is given by the "concept" $m = \langle A \cap C, B \cap D \rangle$. Here the term "concept" is loosely

used, as this "concept" may or may not exist as a formal concept in either context of the two lattices that the two concepts being matched are from.

For example, consider the two formal concepts {*Dog*, *Cat*, *Frog*} → {*lives on land*, *has life*, *can move*}, and {*Fish*, *Frog*} → {*lives in water*, *can move*, *has life*} found in the example concept lattice given in Figure 5.4. The partial match between these two concepts is given by {*Frog*} → {*has life*, *can move*}. This is a more specific concept to the two concepts being matched. There is a node in the concept lattice (Figure 5.4) that contains only the two attributes *has life* and *can move*. The fact that the object *Frog* can have more attributes (in the context of the lattice that the two concepts being matched are from) than what the intent of this concept contains is irrelevant. In particular, when matching concepts between two concept lattices (e.g. between a query lattice and a document lattice), it is unlikely for either lattice to have a node representing solely the resultant concept given by the partial matching defined above. But, it is guaranteed that more exhaustive formal concepts (to the resultant concept) are present in each lattice (i.e. the concepts being matched).

The individual object-attribute pairs (unit-concepts) in m determine how similar the two concepts q and d are. If the query concept is identical to the document concept (ideal case), a complete concept match occurs, i.e. $m=q=d=<A,B>=<C,D>$. A statistical measure for this degree of similarity can be computed by using the statistics of matching object and attribute counts in the two nodes, preferably with a term (concept) weighting mechanism. This is the most frequently used strategy for quantifying similarity between two sets of (query and document) keywords/terms in conventional keyword-based IR systems. The resultant similarity degrees of individual node matching can be combined in some aggregation function to compute a final measure for the similarity (RSV) of the document to the query. We have extended this simple mechanism to "unit-concepts". We use the

weights of matching unit-concepts (Section 7.2.1) instead of the weights of individual terms (objects and attributes).

7.1.2 Possible Levels or Degrees of Partial Concept Matching

Given below are the possible levels or degrees of partial matches that can take place between two concepts due to different levels of specificities/exhaustivities between them. Note that the examples given are for illustrative purposes only. They show the different possibilities of partial matching by taking into account the different possible compositions of queries (left) that can be compared to the given document concept $\{E,Ma\} \rightarrow \{ss,dn,my\}$.

Case 1: Perfect Match:

E.g.1 : $\{E,Ma\} \rightarrow \{ss,dn,my\}$ with

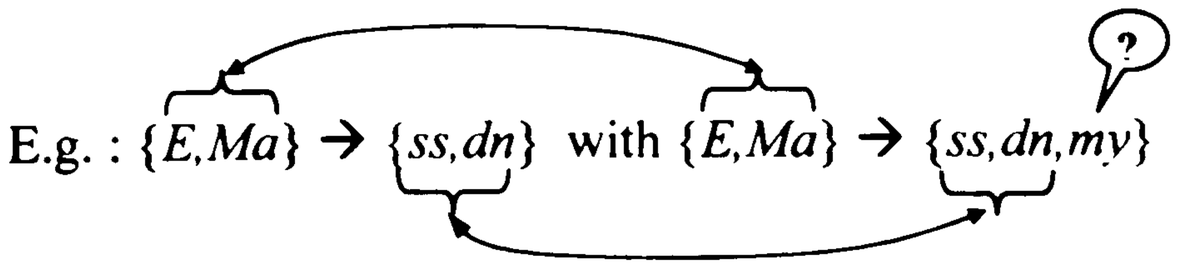
$\{E,Ma\} \rightarrow \{ss,dn,my\}$

This is the ideal case, in which the query and document concepts are identical and therefore a perfect match occurs. This indeed is the strongest possible match between two concepts. However, the overall contribution of this match for retrieving the document depends on the significances (weights) of constituent object-attribute pairs in that document (Figure 7.2).

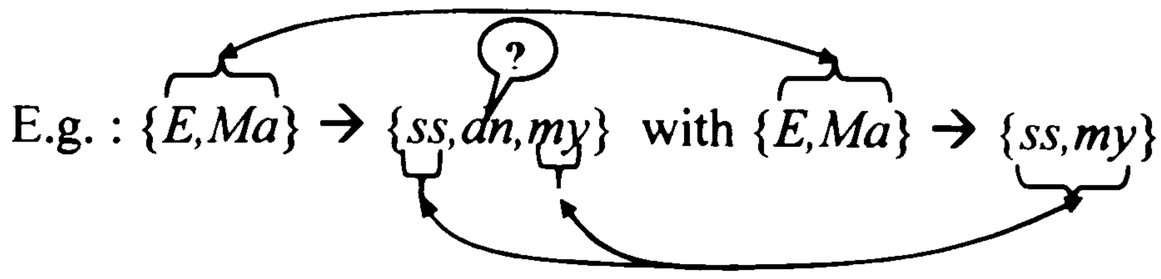
Case 2: Full Extent and Partial Intent:

There are three possibilities for this case:

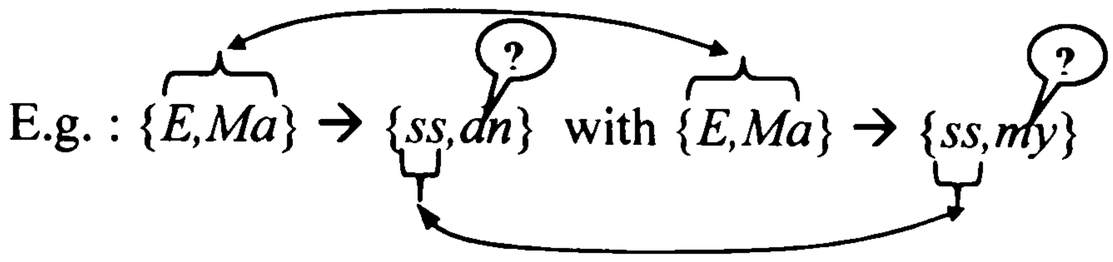
i. The intent of the document concept (strictly) contains the intent of the query concept, i.e. the intent of the document concept is more exhaustive to the intent of the query concept.



ii. The intent of the query concept (strictly) contains the intent of the document concept, i.e. the intent of the document is specific to the intent of the query.



iii. No intent contains the other, but there are elements common to both intents



One can treat these three possibilities differently and quantify their degree of similarity differently according to a particular view of the matching process. For instance, if we view the matching process as one looking for the presence of query concepts in the document lattice, we can say that the first possibility given above (Case 2.i) is a perfect match because the document concept fully contains the query concept. In other words, what we are looking for is present in full in the document concept. In this case, we would not mind having additional objects or attributes in the document concept. Conversely, one can view it as a process of looking for the presence of document concepts in the query. However, a moderate and more general view that makes partial matching easier is to disregard the presence of additional objects and attributes both in the query and document concepts. In other words, consider matching elements only regardless of the additional objects and attributes. This generalises the first two views and simplifies the implementation of the matching process. Partial matching can then be achieved by taking into account the object-attribute pairs common to both the query and the document concepts. This fits well into the partial matching process defined above in Section 7.1.1.

For instance, consider the following concept pair, in which the document concept (in right) is more exhaustive to the query concept (in left). Our matching process treats it according

to the first view, i.e. as if we were looking for the presence of the query concept in the document concept:

E.g.: {***Ma***} → {***ss***} and {*E*,***Ma***} → {***ss***,*dn.my*}

Now, if we reverse the document and query concepts, for instance, we get the following pair of concepts in which the document concept is more specific to the query concept. Our matching process can now be viewed as looking for the presence of the document concept in the query concept, ignoring any additional (not matching) elements. So they both will result in the same outcome ({***Ma***} → {***ss***}).

E.g.: {*E*,***Ma***} → {***ss***,*dn*,*my*} and {***Ma***} → {***ss***}

However, given the fact that documents are more detailed than queries, this type of situation (i.e. query concepts being more exhaustive to document concepts) is less likely to arise, compared to the first view.

Case 3: Full Intent and Partial Extent:

E.g. : {***E***} → {***ss***,***dn***,***my***} with {***E***,*Ma*} → {***ss***,***dn***,***my***}

Intents of formal concepts may be considered more crucial in defining concepts according to the FCA formalism. Objects in extents can be regarded as just examples or instances that belong to the same concept or category defined by intents. However, in the context of IR, objects play a major role in identifying relevant documents. Research on the use of noun phrases (only) for indexing has demonstrated the importance of nouns (in our case, objects). Therefore, we treat them both with equal importance in our work.

This case can be subdivided into three sub-cases and further analysed as with Case 2, but as everything stated above for Case 2 applies to this case as well, we will avoid any further analysis at this point.

Case 4: Partial Extent/Intent

E.g.: {***E***} → {***ss,dn***} with {***E,Ma***} → {***ss,dn,my***}

This is the most general form of all the cases mentioned above. It is simply a combination of Case 2 and Case 3 described above and so is easily understood. In the example given above, the document concept is more exhaustive to the query concept. But all other combinations (e.g. {***E***} → {***ss,dn,my***} with {***E,Ma***} → {***ss,dn***}) are possible with this case. What is important here is to understand that the idea of unit-concept matching suits this case as well.

7.1.3 Many-to-Many Node Matching

As mentioned before, and as can be seen in Figure 7.1, a given object (or attribute) may appear in more than one location (node) in a concept lattice. As a result, a given query concept may match fully or partially with more than one node in the document lattice. Conversely, more than one query node may match partially or fully with the same node in the document lattice. This many-to-many relationship between node matching leads to a duplication of concept matches and thus favours the retrieval of documents with large lattice representations (as they have more nodes in their representations and thus a greater chance for more nodes to match). A large lattice tends to contain more nodes, between which more sub-super relationships are likely to be present than in smaller lattices. This raises the need to restrict node matching to a carefully selected set of appropriate node pairs.

A sensible approach to solving this problem is to use the relationship information in the concept hierarchy of the document lattice in some meaningful way to impose restrictions on node matching, as it is the properties of the relationship hierarchy that cause the duplication. We achieve this by matching the query concept with the most specific document concept in the concept hierarchy of the document lattice in the case that a query concept matches with more than one (related) document concept (node). The intuition

behind this idea is based on the fact that it is the most specific concepts that best describe what the document is specifically about. Ideally, the most specific query concepts should be matched with the most specific document concepts, and in this work we attempt to achieve this goal.

Such a matching process, however, requires well representative query and document concept lattices. The short length of queries, in particular, causes the concept hierarchies of query lattices to be less informative and less useful. This does not mean that the concept hierarchies of documents are complete and self-contained. The level of completeness of a concept lattice representation depends on the information content of the source document; the ability of the feature extraction process to extract objects and attributes and to represent the content of the document in the form of formal concepts; and the subsequent amount of learning the representation has undergone through user interactions in the past. However, in general, documents are more descriptive and lengthier than queries in an IR set-up. In our case in particular, document lattices tend to grow in time as they are subject to learning, through which they are expected to learn a more exhaustive set of concepts from user needs. Despite the problems caused by incomplete representations of document and especially query representations, matching the most specific concepts between query and document lattices is the most sensible approach to take. This is achieved using the properties "object concepts" and "attribute concepts" defined in Section 5.2.6.

To illustrate, consider the case of matching the concept containing the object Mass (*Ma*) in the query lattice with the document lattice given above in Figure 7.1. The object concept of the object *Ma* in the query lattice is $\{Ma\} \rightarrow \{my\}$. This query concept is present in all three nodes indicated in black in the document lattice in Figure 7.1. However, the (black) node at the bottom indicates that *Ma* is smaller in size (*ss*), posses a moon (*my*) and is near to the sun (*dn*); whereas the top black node represents a more general concept in that it shows all the planets that have moons. *Ma* in that concept is just an example for a planet

containing a moon. A document containing this concept could be about any other planet (contained in its extent), or simply a document about planets containing moons in general. The bottom black node, on the other hand, precisely represents *Ma* and *E* with all their properties given in the context. This is as far as we can go in discriminating planets based on this particular query concept and the given document lattice. There is no information in the context of the document to distinguish between the *Ma* and *E* based on just this one given query concept. Given the fact that *Ma* is mentioned in the query, it makes sense to match it with the most specific concept containing *Ma* rather than matching it with a more general document concept. In a real situation, however, it is not just one query concept that decides the relevancy of the document. There is often a set of query concepts to match with the document, and it is a combination of the matching degrees of all the query concepts that decides a similarity score (RSV) for the query-document pair.

7.2 LEARNING

A characteristic of machine learning is the changing of weights of certain features to make them more or less significant. In the context of IR this relates to learning the weights of features that are used to match between queries and documents or learning the strengths of relationships between features. Another aspect of learning in IR is to learn new features for query enhancement. In this section we describe how our learning strategy deals with these different aspects of learning.

7.2.1 Concept Weighting

7.2.1.1 Concept Weighting in our Model

A concept in our case is not just a single term, but two related collections (sets) of terms: objects and attributes. Therefore, we need a mechanism to weigh the significances of such concepts in such a way that partial matching between concepts can be accounted for. Simple term weighting strategies (such as *tf-idf)* are not suitable in our case. On the other hand, assigning a single weight for individual formal concepts represented by nodes in the

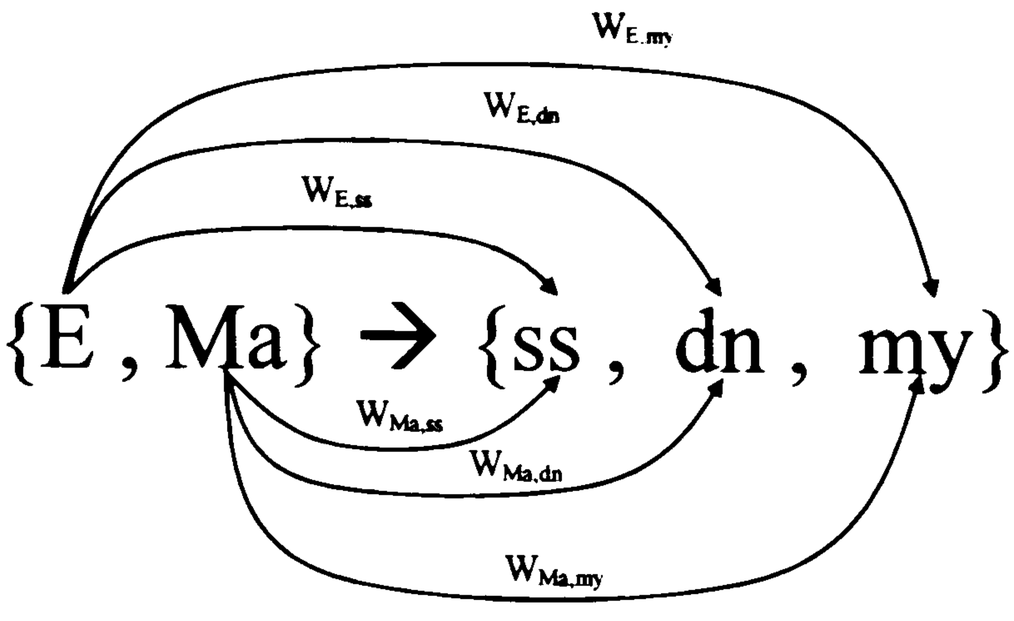


Figure 7.2 : Weight Assignments to Object-attribute Pairs

concept lattice does not permit computation of the degrees of partial matching. Instead, our solution is to assign a weight to each object-attribute pair (Figure 7.2). Note that we use the term "*unit-concept*" to refer to a pair consisting of an object and related attribute.

The overall weight of a formal concept can then be defined as the sum of the weights of each distinct *unit-concept* that it is composed of. The overall weight of the concept given in Figure 7.2 will therefore be:

$$\Sigma W = W_{E,ss} + W_{E,dn} + W_{E,my} + W_{Ma,ss} + W_{Ma,dn} + W_{Ma,my}$$

This composite weight, however, may not contribute towards the computation of a similarity measure between a query and a document concept unless a perfect match occurs. Instead, the treatment of a formal concept as a collection of unit-concepts makes it possible to quantify the significance of a partial match based on the weights of matching object-attribute pairs, and thus fits well with the partial concept matching process described above in Section 7.1.1. For instance, the degree of similarity of the query concept $\{Ma\} \rightarrow \{ss,dn\}$ to the document concept $\{E,Ma\} \rightarrow \{ss,dn,my\}$ is $W_{Ma,ss} + W_{Ma,dn}$. This is not the final similarity measure (RSV) between the query and the document, as there may be other matching query-document concept (node) pairs. The final RSV value will therefore be an aggregation of all such degrees of similarities arising from partial matching between individual concept pairs between the same query-document pair (Section 8.7.2). The RSV values are then subject to thresholding in order to make the decision to retrieve

the document. We use a dynamic thresholding mechanism (Section 8.7.4) as a way of normalising to deal with the size (length) variability problem of documents and queries.

It should also be noted that in our final RSV calculations, we use not only the weights of the unit-concepts but also the weights of the keywords (i.e. the weights of individual objects and attributes) that concepts are made up of. Section 8.5 gives details of why we use them and how individual terms are weighted and used in computing RSV values.

7.2.1.2 Factors That Affect Concept Weighting

Assigning the correct weights for concepts in documents is a difficult problem in IR. There are a number of factors that we should consider in deciding a weighting strategy. They include:

1. The balance between the expressive power and the retrieval power of features/concepts.
2. The balance between the locality (to a document) and globality (to the collection) of features.
3. The ability to capture the variability of the significance of the same feature in different documents.
4. Adaptivity (static versus dynamic weights).
5. Usage factor (the chance of an average user using a feature/concept in formulating a query).

This is a highly correlated set of properties that defines a complex set of conflicting demands. Finding the right balance between these is extremely difficult.

- **Expressive Power Versus Discrimination Power**

Quantifying the significance of concepts in respect of their ability to convey an understanding of the important message a document carries is a major objective of text understanding research. However, in IR, the primary interest has been in giving a higher significance to the features that support the correct retrieval of documents (against queries) than to the features that best represent the main theme(s) of the documents. How far these

two coincide is a research question that is outside the scope of our work. One might expect that the features that best describe the content of a document are the best features that can uniquely identify the document. But "best" here is a relative concept that depends heavily on the users. It is unlikely that all users always use the same concepts or terms (to describe their information needs) that the authors think are "best" in describing the contents of their documents. In addition, the differences of vocabularies between individual authors may result in different authors reporting the same information using different terms (vocabularies). The differences between users and authors make the situation more complex and highly dependent on such factors which are outside the IRS.

- **Locality Versus Globality**

The fact that a given feature (concept) is not equally significant in all the documents that it appears demands some degree of locality in weight values, i.e. significance weights must be decided with respect to the individual documents. How much a weight computation is local and how much it is global is a well-known problem in IR research. Statistical methods attempt to balance this by taking into account both global and local information (frequency counts) to estimate the weight values, i.e. they estimate the significance of concepts in retrieving a document based on the level of their "uniqueness" in representing individual documents. However, estimations based on the frequency statistics are often incomplete and easily misleading. Such estimations are not tolerant to the deliberate repetition of features, the length differences between documents and other complexities of natural language writing, such as the use of pronouns, analogies and the use of examples etc. As a result, false hits are made by retrieval systems. In addition, certain documents such as bibliographies, summaries and surveys, might have higher frequency counts of certain terms, which influence their retrieval even when the user might not be interested in them.

- **Usage Factor of Concepts**

Prior assignment of weights to features by statistical methods tends to allocate higher weights to "rare" concepts. These "rare" concepts might never be used by users to find a document. Therefore, assigning a high weight for a concept is not of much value if that concept is not used by the users. A more useful approach therefore would be to assign higher weights for features/concepts that are used on average by more users for finding a document. One approach in this direction is to allow users to assign their own weights to search terms. The intuition behind this approach is based on the assumption that the importance of a term is a relative concept and that it is the user who knows exactly how crucial a search term is to his information need. The INQUERY system [Callan et al. 92, Croft 1995] can be given as an example for a practical implementation of this approach, in which the user can modify the *tf* factors of term weights. This approach has the desired feature that two distinct users can weight the terms of the same query expression differently according to their perspectives of the query [Fagin 2000]. Although this sounds more practical, its success depends on the user's ability to formulate his query correctly, identifying the significant concepts that the authors might have used. This is a difficult and unrealistic goal to achieve. As a result, more often than not, the user will have to repeat his query a number of times by changing the significance weights of his search terms or concepts before he gets them right. In addition, this approach is typically implemented via ad hoc heuristics and demands excessive user interaction.

- **Document Ageing**

Another important factor that has not drawn much attention from the IR community is document ageing. As documents age, information contained in those documents becomes obsolete and therefore less useful. Nevertheless, certain classic documents may remain useful despite their age. IR systems that are based on static weighting mechanisms cannot deal with this problem as the RSV value of a given document to a given query is fixed

regardless of when the query was made. A dynamic weighting scheme that learns weights based on user interests is required to deal with this problem.

7.2.2 A Reinforcement Learning Strategy

7.2.2.1 Learning Concept Weights

A more practical solution, that helps to overcome most of the above problems, is to decide which concepts (or keywords) are significant and to what extent based on the common views of the users. We achieve this goal by employing a reinforcement learning strategy based on relevance feedback, in which the concepts that support retrieval of a document that the user finds useful are rewarded and concepts that cause a false retrieval are penalised. Rewarding is achieved by increasing the weights of the matching unit-concepts that helped retrieval of a useful document, and penalising is achieved by decreasing the weights of the matching unit-concepts that led to a false retrieval of a document. In this scenario, the weight of a given unit-concept in a given document at a given time depends solely on the retrieval of that document in the past as a result of that unit-concept being matched, and the decisions of the users about the usefulness or relevancy of the document to their information need(s). This keeps the weights dynamic and lets them evolve over a period of user interactions. The balance between positive and negative hits of a given object-attribute pair in a given document in the past decides its present weight, i.e. the significance of a concept is based on the average view of all the users. Therefore, on average, a fair response can be expected by an average user for an information need.

Figure 7.3 given below illustrates how our concept learning strategy works. The four boxex at the top show the concepts in a query, the documents (IDs) retrieved, and what concepts caused the documents Doc#35 and Doc#20 to be retrieved. Only Doc#35 is found to be relevant to the query by the user. The weight adjustments are shown at the bottom. Since Doc#35 is a relevant and retrieved document, the weights of the unit-concepts common to both the query and Doc#35 (only $p \rightarrow q$ in this case) are rewarded. On the other

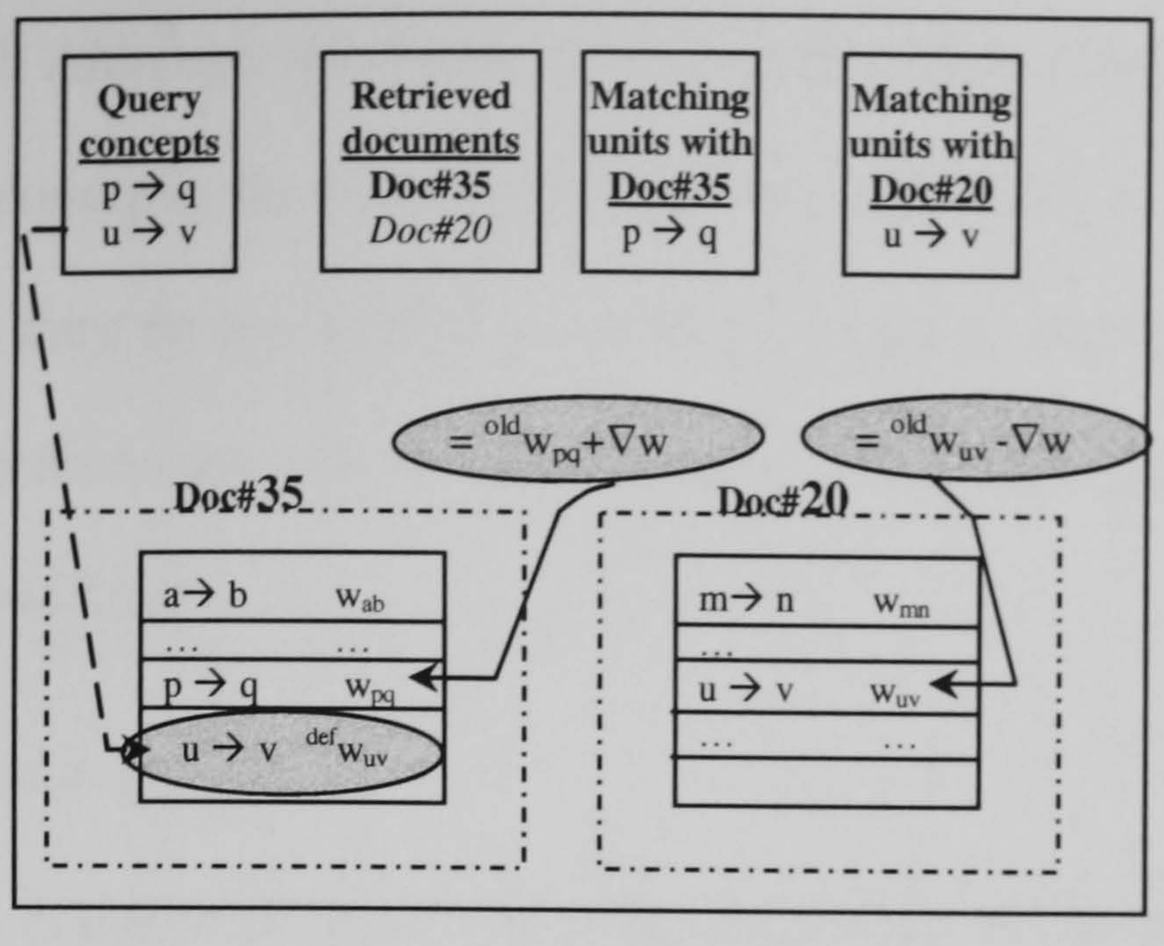


Figure 7.3 : Concept Learning Strategy

hand, Doc#20 has also been retrieved by the system as relevant to the query, but the user has not found it useful (i.e. it is a false hit). Therefore, the weights of the unit-concepts common to both the query and Doc#20, i.e. the weights of the unit-concepts that caused it to be retrieved (in this case only $u \rightarrow v$) are penalised. This makes the similarity of Doc#35 stronger and Doc#20 weaker to the query.

This weight learning policy allows the same unit-concept (object-attribute pair) in different documents to have different weight values. In addition, the dynamic nature of the weights results in the RSV value of a document declining in time as the document ages and gets less attention from the users. Also, by learning weights, we escape from the need to pay special attention to balancing the trade-off between locality-globality and the trade-off between the expressivity-retrieval powers in weight setting. In fact, we have passed them over to the users to decide, without giving any burden to them.

Implementation details of our weight learning mechanism are given in Section 8.6.

Rare Concepts

A property much desired in characterising documents is to assign a higher weight for rare concepts or features. As mentioned before, statistical methods achieve this by taking into account the occurrence counts of features in the whole collection. In our case, however, since rare concepts are rarely used by users in formulating their information needs, they

rarely get reinforced. Although this looks to be an undesirable property, in fact it is not. Since they are rare concepts, the chance of negatively reinforcing a rare concept is even rarer and as a result they do not tend to get as many negative reinforcements as positive reinforcements. Therefore, weights of rare concepts will often eventually emerge with higher values in the long run.

7.2.2.2 Learning New Concepts

The weight learning described above enables the documents in the collection to learn the significances of their concepts according to user decisions. However, learning weights of unit-concepts alone is not sufficient for concept learning in IR. A weight of a given unit-concept in a given document is reinforced only if the following three conditions are satisfied:

1. The document is retrieved by the system for one or more queries.
2. The unit-concept in question matches with one or more of those queries which retrieved the document.
3. The document is judged as useful to those queries by the user, in which case the weight is increased; otherwise the weight is decreased.

It is unlikely that all the unit-concepts of a query are present in a document, due to various problems already mentioned that cause mismatches. This slows down learning, as reinforcing just a few concepts (features) in a document is not sufficient for picking that same document up again by a similar query. A new query looking for the same document may not contain any of those unit-concepts in the document that have been reinforced at some point in the past by previous queries. As a result, a query will fail to pick up a document useful to the user just because of the mismatch problems. Therefore, we need a way of making the similarity of the documents that were judged by the users as relevant to their queries much stronger, so that a similar query that has some unit-concepts in common with some of the previous queries (to which the document was previously retrieved and found useful) could retrieve the document. We achieve this by expanding the document

representations with all the unit-concepts of corresponding queries that are judged as relevant to them. For instance, in the example shown in Figure 7.3, Doc#35 is enhanced by adding the unit-concept $u \rightarrow v$ into its representation. This helps to eliminate the gap between the users and authors, and makes documents learn from the users about the different possible ways of describing their contents according to user perspectives.

7.2.2.3 The Problem of Wrong Judgements

The success of relevance feedback relies on the way relevance judgements are made. In most automatic relevance feedback models the system's heuristics make the judgements. But in our case, we use users' judgements as to which documents are relevant out of the retrieved documents. In both cases, there is a danger caused by misjudgements. Wrong judgements adversely affect learning and degrade the performance of the system. A way round this is to make the system learn from experts rather than from novice users. This is possible with the systems that are trained in advance before being deployed, in which case the knowledge learnt is fixed thereafter. In an adaptive set-up like ours, in which the system learns continuously while it is in use, we cannot expect all of its users to be experts. However, we believe that this problem will not cause a severe danger to learning in our model, given that the chance of all users making the same error is unlikely. Although there is a possibility of fooling the system by making wrong judgements deliberately, the system is capable of recovering from such acts as it learns from correct judgements made by other users, provided that the majority of them do not make the same incorrect judgements.

7.3 SUMMARY

This chapter presented in detail the design of our model. Special features of our model are its explicit concept matching and the use of a reinforcement learning strategy for learning the significances of concepts as well as learning different possible ways of describing the contents of documents according to the perspectives of the users.

This learning strategy is based on relevance feedback information. Traditionally, relevance feedback has been used in IR for query reformulation purposes. Although it has shown improvements of as much as 20% on recall and precision, the system's learning through relevance feedback is temporal and does not last long. It does not retain what it learns from important user decisions, as they (user decisions) are used only within one query session during searching for a single information need. The learning gained by relevance feedback in one query session is usually not available for subsequent query sessions and hence a separate learning mechanism is required to make the system adaptive. In contrast to the system's learning, the knowledge gained by the user from previous retrieval sessions stays longer in his mind and will be useful to him in formulating future queries. This feature is, however, local to individual users.

In our model, document representations are updated continuously in accordance with user decisions and thus the knowledge gained through relevance feedback is retained for later use. We expect the document representations to converge to a fully representative set of concepts for each document over a period of time. Such a set of concepts, in fact, will become more customised to the vocabulary and the writing style(s) of the end user(s), as it is the concepts of the user formulated queries that are appended to the (relevant) document representations. The following can be listed as the favourable properties of our reinforcement learning strategy:

1. The ability to assign different weights to the same concept in different documents.
2. Implicit balancing of locality and globality of document characterisation, and the expressive and retrieval power of concepts.
3. The use of dynamic weights that help document aging and customising (document representations) for changing users and changing user interests.
4. Document representation is locally held for each individual document. This makes a distributed representation of the document collection possible.
5. It assists in narrowing the gap between users and authors by letting documents learn the way users describe the information that they find useful in documents.

CHAPTER 8 - IMPLEMENTATION DETAIL

A detailed discussion of the design of our model was given in the previous chapter. In this chapter we present the implementation detail of the model. Firstly, we present a schematic diagram of our implemented prototype (Figure 8.1), and the detail of each of its components follows.

8.1 SCHEMATIC DIAGRAM OF THE IMPLEMENTED PROTOTYPE

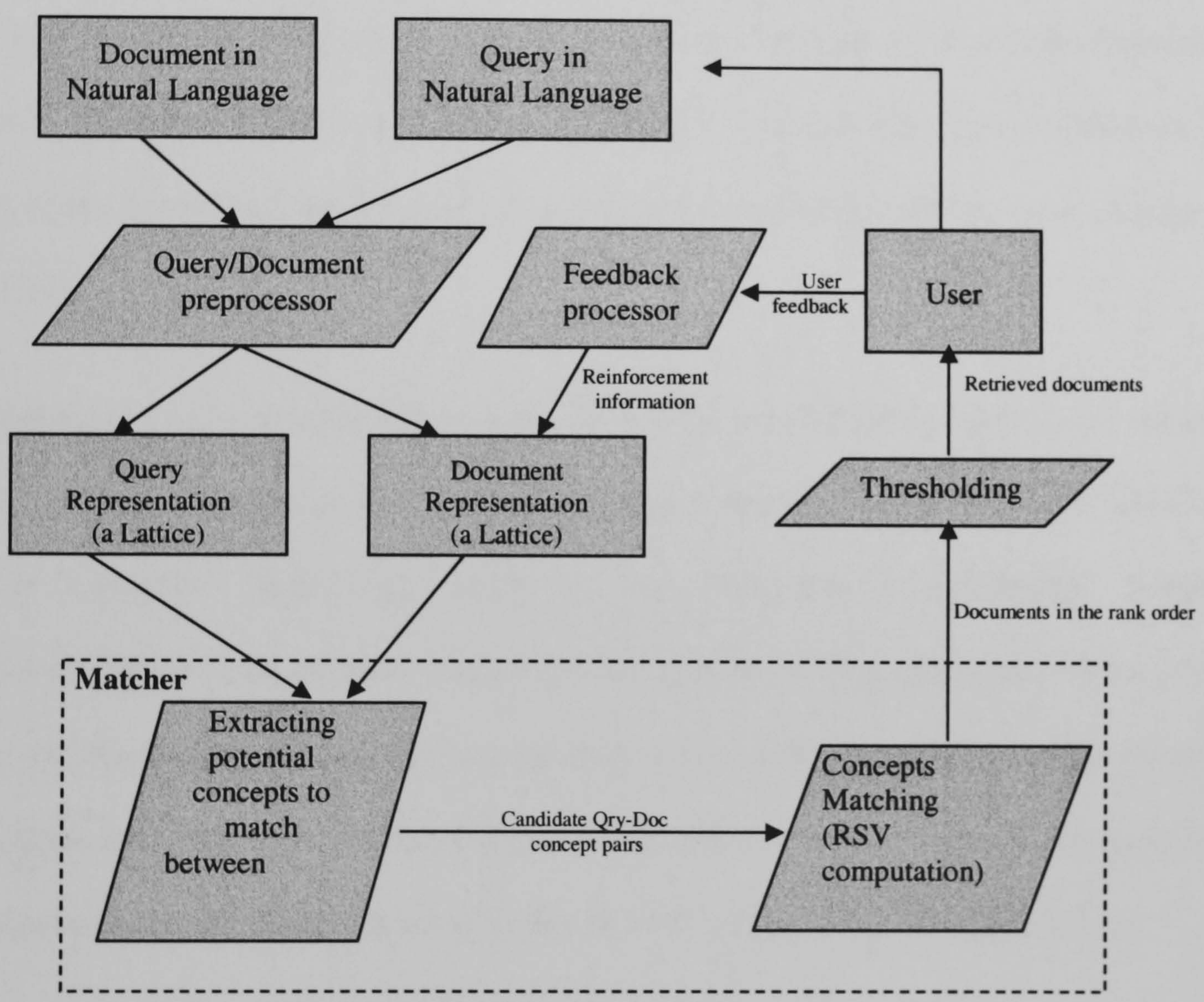


Figure 8.1 : Schematic Diagram of the Model

8.2 FEATURE EXTRACTION FOR CONCEPT GENERATION

Extraction of features for generating concepts of the kind we are interested in (i.e. object-attribute pairs (unit-concepts)) is a difficult problem. This task, which is related to the more complex natural language understanding problem, is an unsolved research question beyond the scope of our work. However, the fact that a deep and complete understanding of the text may not be mandatory for IR means that it is possible to work with a shallow and

partial representation of document contents. We make use of techniques such as part-of-speech (POS) tagging and noun and verb phrase analysis, together with a set of selected prepositional terms (such as *in, on, at, of*) that frequently appear between noun and/or verb groups as connectors, to extract a set of features (objects and attributes) and their possessive relationships (which objects possess which attributes). Even though the use of individual noun-phrases has been proven to be superior to full text indexing [Evans & Zhai 1996], this alone would not be sufficient in our case, in particular for extracting intentions (attributes). It ignores the important roles that verbs play in natural language. In fact, it is verbs that express the intentions or objectives of subjects (participant objects). Since we are interested in the relationships between objects and attributes, it is useful to take into account verbs (verb phrases) and connections between noun phrases and verb phrases as well.

Our feature extraction process starts by pre-processing text with POS tagging and chunking terms into noun and verb groups. We used the tagger and chunker LTCHUNK (developed by the Language Technology Group of the University of Edinburgh, Scotland; http://www.ltg.ed.ac.uk/software/chunk/) for this purpose. It tags each term with a symbol (using the Penn Treebank tag-set [Marcus et al. 1993, 1994]) to identify its part-of-speech and groups them into noun phrases (NG) and verb phrases (VG). We gather the tagged and chunked information of each sentence in the following data structure (Figure 8.2):

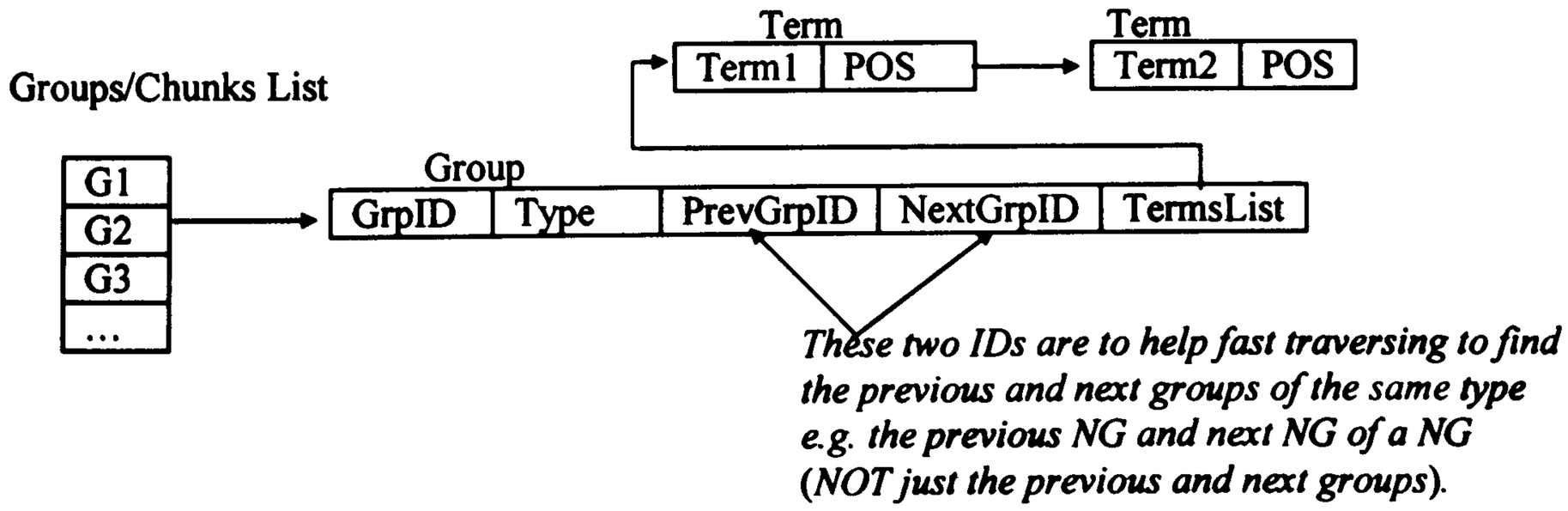


Figure 8.2 : Sentence Data Structure

A document consists of a list of such structures, one for each sentence in the text. Terms that do not belong in any (NG or VG) chunk are grouped under a third category, "Other

Group" (OG). This group is very important for our concept extraction process, as this is where the connecting terms (prepositional words that connect different groups) are grouped in general.

For instance, consider the sentence, "*The tractor hit the gate of the farm*". A tagged and chunked version of this would look like *[The_DT tractor_NN] (hit_VB) [the_DT gate_NN] of_IN [the_DT farm_NN],* where noun groups are enclosed with square brackets and verb groups in round brackets. The symbols *DT*, *NN*, *VB* and *IN* are the POS tags for a determinant, a noun, a verb and a preposition respectively (from the Penn Treebank tag-set [Marcus et al. 1993, 1994]). The representation of this sentence in the data structure is illustrated in Figure 8.3.

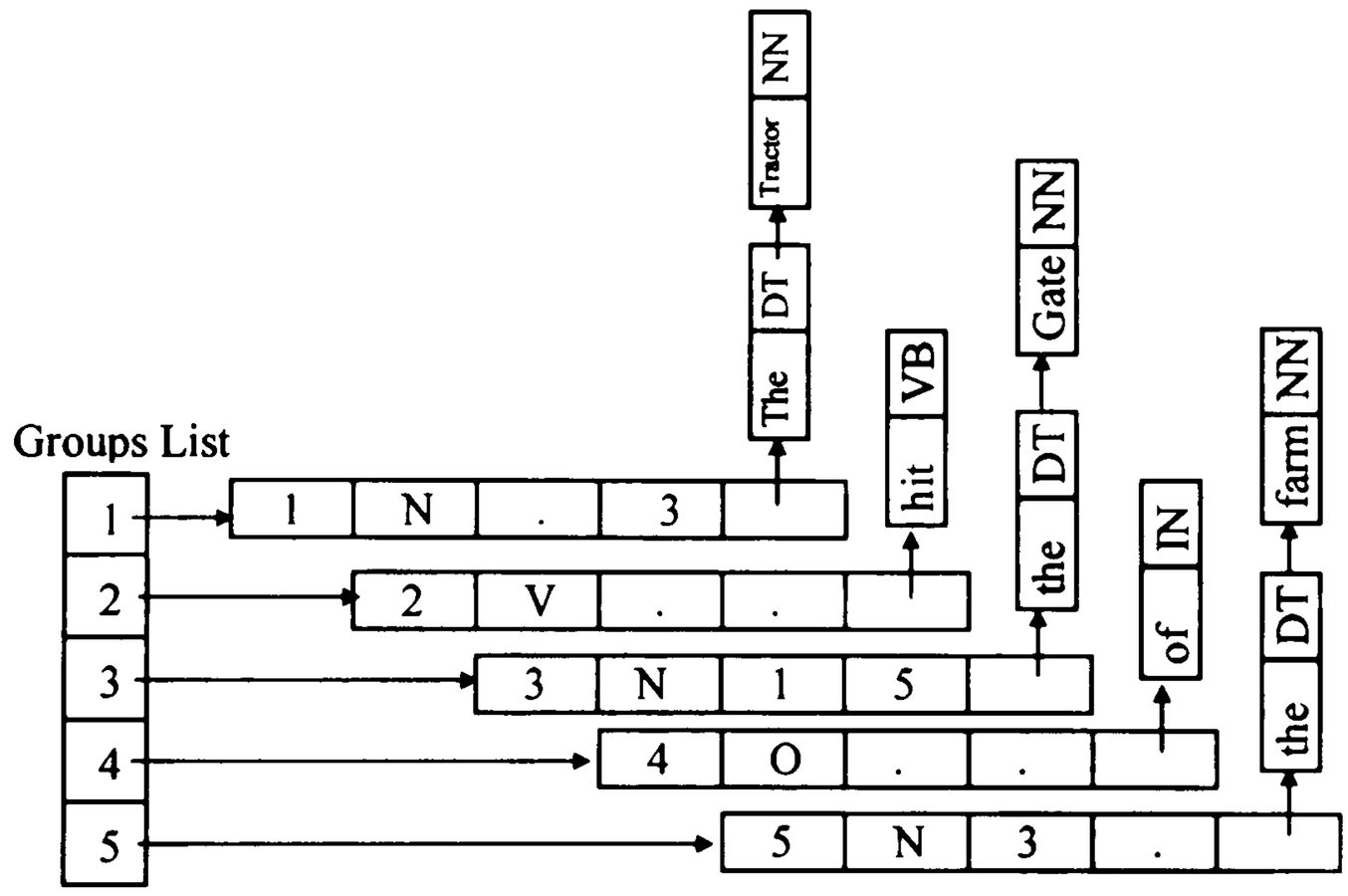


Figure 8.3 : Data Structure of a Sentence – an Example

As can be seen in the above example (Figure 8.3) the extracted sentence structure contains not only the terms of the sentence, but also its syntactic structure in terms of POS tags and chunks. A sample of sentences (from the Cranfield collection) were further analysed and a number of rules were developed to identify objects and their attributes in order to automate the concept extraction process. Syntactic information about the sentence structures, syntactic structures within components (i.e. within NGs, VGs and OGs), and semantic relationships between terms within groups and between groups indicated by connecting words were taken into consideration in developing these rules. Most of these rules can be described under three general categories based on the grounds on which they were

developed. However, there are certain specific rules that do not belong to either of these general categories, for example, the constructs formed by "*such as*" followed by some examples or a few examples followed by "*etc*". Constructs such as these that do not belong to the general rules but contain useful information for the extraction of concepts are captured by using special rules. In this section the general categories only are described. The complete list of rules used for concept extraction is given in Appendix A with examples illustrating each rule.

8.2.1 Syntactic Structure of Noun Groups

As noted above, most noun groups contain more than one noun word and adjective (modifier) within them. They are further analysed using POS tags attached to each word in order to detect useful object-attribute relationships between the words within them. A frequently found relationship in noun phrases (NGs) is the adjective-noun construct. In this case the adjective usually acts as a modifier that specifies a special property of the noun term. Therefore, an adjective in such constructs is considered a property (attribute) of the noun term (object) that it modifies.

For example, if the syntactic structure of a noun group is DT|JJ|NN (e.g. "*The*_DT *tall*_JJ *man*_NN"), then a concept is formed as NN → JJ (i.e. {*man*} → {*tall*}). Here "*tall*" is an attribute of the object "*man*". This is read as the object "*man*" having the property "*tall*". Here, the tag DT stands for a determinant, JJ for an adjective and NN for a noun, and the structure of the chunks is written using the part-of-speech tags of individual terms separated by vertical bars ("|"). Note that we use the convention {<extent>} → {<intent>} to write a formal concept. An extent (or intent) can contain more than one object (attribute) in which case they are written comma separated.

In case of noun groups comprising more than one noun word, concepts are formed according to the following rule:

$[<noun_1> < noun_2> < noun_3>] \Rightarrow$ $\{<noun_3>\} \rightarrow \{<noun_1>, <noun_2>\}$;
$\{<noun_3>\} \rightarrow \{<noun_1>+<noun_2>\}$; and
$\{<noun_2>+<noun_3>\} \rightarrow \{<noun_1>\}$

For example consider the noun phrase [*central*_NN *capital*_NN *city*_NN]. The concepts extracted from it are as follows:

1. {*city*} → {*central, capital*}
2. {*city*} → {*central capital*}
3. {*capital city*} → {*central*}

In addition to the above, a concept is formed by taking the entire noun group (i.e. all the terms in the group) as the object as well as the attribute (self-concept) when a noun group has only noun terms (according to part-of-speech tagging results). This is to ensure that such descriptive keyphrases, which carry useful information, are retained for possible keyword matching. For example, consider the noun group [*London*_NN *Bridge*_NN]. Only the concept {*Bridge*} → {*London*} is extracted from the first rule described above. However, it would be more meaningful, in this case, to keep the two terms of this phrase together (i.e. *London Bridge* as a single entity). Note that the phrase "*London Bridge*" will not be used as a single element to create a concept if this noun group is not connected with another noun group by a connector (preposition) that we use (Section 8.2.2) for extracting information. Forming a self-concept such as {*London Bridge*} → {*London Bridge*} in particular enables a meaningful keyphrase match in the absence of a unit-concept.

8.2.1.1 Possessive Relationships

The possessive relationships between two noun words indicated by a trailing *'s* or *s'* are also detected and processed separately to form concepts containing all the words in the noun group up to the apostrophe (') as the object and the rest of the words in the group after the possessive tag "POS" as the attribute. These always appear within a single noun group (NG). In addition to this rule, the rules described above are also applied if any part (object or attribute) of the concept thus formed is multi-termed.

E.g.

1. [*The*_DT *man*_NN*'s*_POS *hair*_NN]

   ⇨ {*man*} → {*hair*}

2. [*Seattle*_NNP *'s*_POS *central*_JJ *business*_NN *district*_NN]

   ⇨ {*Seattle*} → {*central business district*}

   ⇨ {*Seattle*} → {*district*}

   ⇨ {*Seattle*} → {*business , district*}

   ⇨ {*Seattle*} → {*business district*}

An interesting outcome of these concept extraction rules is that the same concept happens to be formed from several different ways of writing the same idea (verbal groups). For instance, both noun groups "*The man's hair*" [*The_DT man_NN's_POS hair_NN*] and "*The hair of the man*" ([*The_DT hair_NN*] *of_IN* [*the_DT man_NN*]) produce the same concept {*man*} → {*hair*}.

8.2.2 Prepositional Connectors between Chunks

Use of individual noun phrases (only) for document indexing does not take into account the relationships between noun groups. We analysed the relationships between those noun groups that are connected by prepositional connectors such as *in, on, of, with, to, into, from* etc, for extracting useful concepts using terms in two connected chunks. Relationships inferred by such connectors between two noun groups can be interpreted as *object-attribute* relationships. Both the set of prepositions used as connectors and the rules that decide the roles of terms or phrases as objects or attributes were selected by analysing the syntactic structures of sentences and the semantic relationships between chunks/groups of a resultant pre-processed (tagged and chunked) sample of text. The list of connecting words selected [given in Appendix A] is not complete by any means, but consists of the most frequent and useful connectors found in the text.

Usage of prepositions in natural language is such that the roles of the terms (as objects and attributes) in most connectors (such as "*in*" and "*at*") usually appear in a particular written

order that can be interpreted either as *object-connector-attribute* or *attribute-connector-object*. However, there are a few connectors (e.g. "*of*") for which the usage is not precise. With these connectors an object (attribute) can appear on either side. For example, in the phrase "[*millions*] *of* [*visitors*]", *visitors* is more suitable as the object and *millions* as the attribute. But in the noun group "[*University*] *of* [*Plymouth*]", either the *University* or *Plymouth* can be taken as the object and the other term as a property/attribute of the object. In such cases, both possibilities can be used for concept extraction (and both were used in the initial experiments). However, we only employed the most frequently used practice in the final experiments as no significant improvement in the retrieval results was found when both possibilities were used. For instance, for the operator "*of*", i.e. in the form NG_1 *of* NG_2, NG_2 is considered the object and NG_1 is considered the attribute.

As already noted in Section 8.2.1, in most cases noun groups are comprised of several adjectives and one or more noun words etc. (E.g. *The_DT red_JJ old_JJ car_NN*). Using all the terms in a noun group as a single object or attribute is not very useful when creating a concept based on two connected noun groups. Instead we use only the noun words of the noun groups. For instance, consider the following two pieces of text:

E.g. 1. [*the*_DT *red*_JJ *car*_NN] *of*_IN [*the*_DT *tall*_JJ *man*_NN]

2. [*a*_DT *lobby*_NN] *of*_IN [*dark*_JJ *marble*_NN *walls*_NNS]

In the first example, creating a concept as {*tall man*} → {*red car*} causes matching problems when the same adjectives (*red* and *tall*) are not used by the user in his query statement. A better formulation therefore would be: {*man*} → {*car*}. Note that determinants (like "*a*" and "*the*") are always ignored. The processing of the syntactic structures within the noun groups described above extracts {*man*} → {*tall*} and {*car*} → {*red*} as concepts. The first two concepts will then be joined during concept formation, resulting in the single concept {*man*} → {*car, tall*}. This allows the query

concepts {*man*} → {*car*} or {*man*} → {*tall*} to partially match with the concept {*man*} → {*car, tall*}.

However, using this method alone also causes a problem if a particular document talks about more than one distinct car, say, described by two different colours, that should be identified separately. The above-mentioned method constructs a concept with the object *car* together with all colours as its attributes. Therefore, it is useful to use both methods, i.e. in this example {*tall man*} → {*red car*}, {*man*} → {*car, tall*} and {*car*}→ {*red*}. In this case, if the document talks about a *green car*, a concept will be formed taking *green car* as a single attribute of an object to which it relates. During concept matching, we must be careful to take into account only the most expressive matching concept(s) in cases where concepts share the same terms (in their extents or intents), i.e. if both {*tall man*} → {*red car*} and {*man*} → {*car*} matches with a query, we should take only {*tall man*} → {*red car*} into account, as otherwise we may be duplicating the same concept.

8.2.3 Verbs in Verb Groups (VGs) and Other Groups (OGs)

Verb terms in the English language take many different forms depending on the tense and other language rules. Examination of tagged and chunked text reveals that verb terms appear frequently in VGs and OGs, but rarely in NGs. In NGs verb terms usually appear as adjectives (e.g. *running man* or *cracked bottle*). These are correctly identified by the tagger as adjectives (and tagged with _JJ) and so do not need special attention. Verbs that appear in VGs tend to follow a few generic patterns/rules and therefore can be captured by a set of generic rules, while for others specific rules are needed.

The most generic construction of noun and verb group combination is of the form $NG_1|VG|NG_2$, with the VG containing a single verb (_VBZ type) that creates an "*is_a*" or "*has_a*" type of relationship between the surrounding NGs. In this case, a concept is created using the noun word(s) of NG_1 as the object(s) and the noun words of NG_2 as the

attribute(s). For instance, consider the simple sentences "[*John*_NN] (*is*_VBZ) [*an*_DT *engineer*_NN]" or "[*John*_NN] (*has*_VBZ) [*a*_DT *car*_NN]". We extract {*John*} → {*engineer*} from the first sentence and {*John*} → {*car*} from the second.

Appendix A gives full details of the different constructs of noun/verb groups and the different constructs of connecting groups (Other Groups (OGs)) used, together with the rules developed for extracting concepts from those constructs. Note that the connecting words used were grouped into five main (overlapping) groups [Appendix A] depending on the syntax patterns of the text they appear in, in order to create rules for common patterns.

8.3 DATA STRUCTURE(S)

There are three data structures central to our implementation: (1) a Sentence Structure - a structure for gathering information about the details of sentences; (2) an Elements Table - a structure for keeping constituent elements (objects and attributes) of concepts and their weights (i.e. keyword weights); and (3) a Concepts Table - a structure to store unit-concepts and their weights. The Sentence Structure, which is central to our concept extraction, was described in Section 8.2 (Figure 8.2 and Figure 8.3), and the other two structures are described below.

Note that we used the same structures for Queries. The reasons for this are twofold. Firstly, it is easier and is a frequently used practice to use the same structure for both queries and documents. Secondly, it allows for the provision of future enhancements to use (user specified) weights for query concepts/keywords.

8.3.1.1 Elements Table

This is the structure that keeps information of each element. Each object and attribute is assigned an identification number with respect to its role (as object or attribute) and has a weight (keyword weight). This information, together with the role of the element and its

status (i.e. whether an original one or initially extracted one) or a reinforced (i.e. one added later-on during learning) are stored in this structure as illustrated below in Table 8.1.

TERM	*ROLE*	*OBJECT ID*	*ATTRIBUTE ID*	*OBJECT STATUS*	*ATTRIBUTE STATUS*	*KEYWORD WEIGHT*
man	1	1	-999	1	-999	0.25
hair	2	-999	1	1	1	0.38
…						
<trem$_n$>	3	12	15	1	2	0.18
…						

Table 8.1 : Elements Table Data Structure

where,

ROLE : 1-Object, 2-attribute, 3-both

STATUS : 1-Original, 2-updated/later-added one through reinforcement process

-999 indicates "Not Relevant/Not Used"

Note that a given term can play different roles in different unit-concepts depending on the situation and therefore can take both roles (Role = 3, i.e. it is an object in one unit-concept and an attribute in another). Since it is useful to know the role of an element, we assign identification numbers for the elements with respect to their roles. An element that plays both roles is therefore assigned with two identification numbers; one as an object and the other as an attribute. However, only one (keyword) weight is associated with it as we do not distinguish between the roles of elements outside the context of unit-concepts when weights of constituent elements are considered as keywords. Numbering objects and attributes separately is not essential; they all can be numbered with a single series and the role information can be kept separately. However, separate numbering simplifies the implementation.

8.3.1.2 Concepts Table

As can be seen in Table 8.1, the Elements Table does not store object-attribute relationships. They are kept separately in the Concepts Table as shown below in Figure 8.4. It is possible to combine these two structures and use a single (more complex) structure. We avoided using such a complex structure for reasons of simplicity of implementation and tractability of data.

This structure holds the attributes related to each object and weights between each object-attribute (i.e. weights of unit-concepts) pair as shown below.

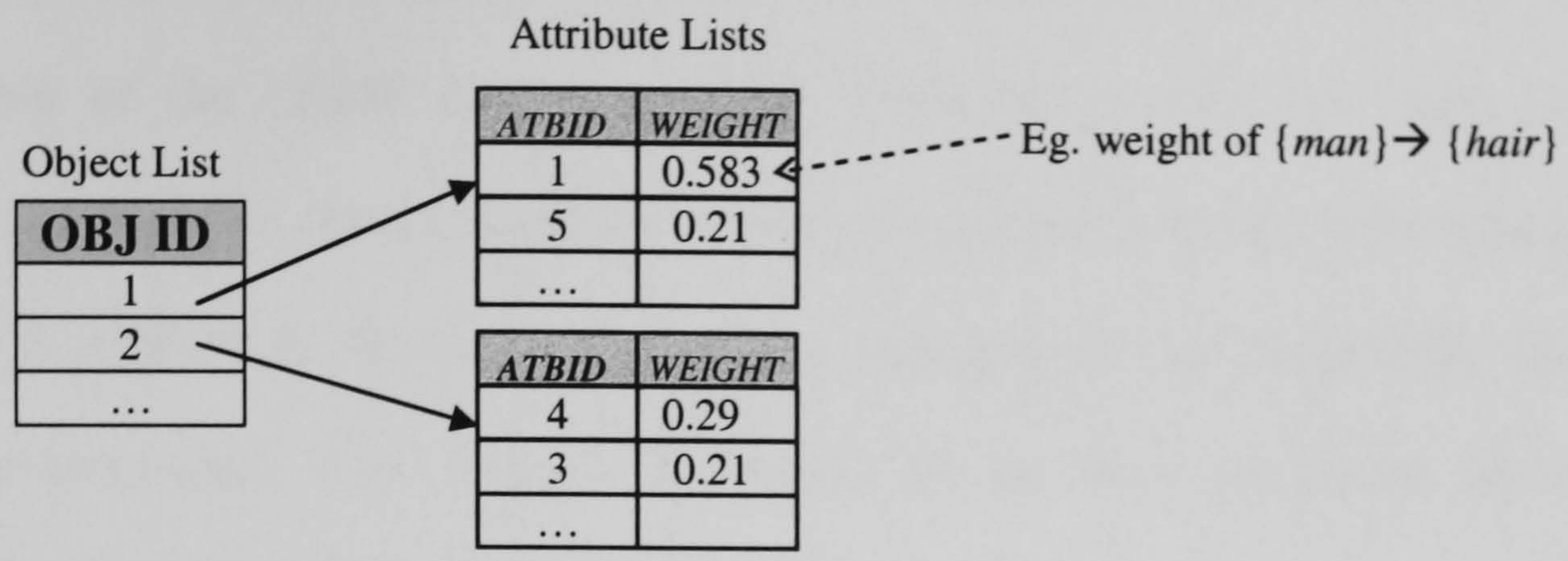


Figure 8.4 : Concepts Table Data Structure

8.4 Interactions between the Data Structures and Reinforcements

Adding a new concept involves adding its constituent elements to the Elements Table and adding the relationships between constituent objects and attributes (unit-concepts) to the Concepts Table with initial weight values. In the case where one element of the unit-concept is already present then we only need to add the other element into the Elements Table, and amend the Concepts Table accordingly. If the element of the concept being added is the "object" and is present as an object in the Elements Table, then there is already a Concepts Table for that object. We only need to add the attribute of the unit-concept to the Elements Table and Concepts Table, and to set an initial weight for the unit-concept in the Concepts Table and an initial (keyword) weight for the attribute in the Elements Table. If it is the "attribute" that is present in the Elements Table, then the object needs to be added into the Elements Table and to the Objects List of the Concepts Table, and a new table needs to be created to store the attribute and the weight of the unit-concept. If one or both elements of the unit-concept being added are present in the Elements Table with opposite roles, then their roles need to be amended. Amendments needed for the Concepts Table follow the same process described above, with those elements treated as though they were not present in the Elements Table.

If both elements are present in the Elements Table, but the Concepts Table has no entry relating the object to the attribute, then an entry is created by adding the attribute to the corresponding table. If both elements are present in the Elements Table and the related attributes table of the Object in the Concepts Table already has an entry for the unit-concept being added, then no concept addition takes place; instead, the weight of the unit-concept is reinforced. Reinforcing matching unit-concepts and keywords involves increasing or decreasing their weights depending on the user judgement about the usefulness of the document to the information need.

8.5 The Need For Keyword Matching

Common unit-concepts are not always present between the nodes of two concept lattices (i.e. a partial match does not always take place between a query and a document concept). We often find cases where two concepts share either a common object(s) or attribute(s) but not both (Figure 8.9). This may happen for various reasons, including word mismatch problems due to synonymy or the chance representation of distinct concepts by the same term (due to polysemy). Regardless of the cause, we do not want to ignore the possibility of the positive contribution that such common features between nodes might make towards the retrieval of a useful document. Therefore such a single object or attribute match is considered a keyword match and its significance is modelled with keyword weights. For this reason, we maintain weights for both unit-concepts and their constituent elements (objects and attributes). The weights of the unit-concepts model the significances of the object-attribute pairs (unit-concepts) with respect to the individual documents within which they are present, while the weights of the keywords (individual elements of unit-concepts) model the significance of the elements the unit-concepts are composed of.

8.6 All About Concept/Keyword Weighting and Weight Learning

8.6.1 Weight Ranges and Initial Values

Both keyword and unit-concept weights are initialised at the beginning with an initial value of 2.5 and they are subject to learning over user interactions. The same initial value (2.5) is used for new unit-concepts and keywords that are added during the learning process. As mentioned before, the values of weights are subject to increase or decrease by small steps (Δw) depending on the user feedback. The range of a weight was arbitrarily selected to fall in the range between 0.1 and 5.0. The minimum value of a weight was set at 0.1 instead of zero (0) to avoid complete unawareness of the existence of a unit-concept or keyword. One can use any other positive range instead and accordingly set the thresholds or normalise the RSV values to lie in a particular range (such as [0,1]).

8.6.2 Step Size of Weight Changes

The step size is determined proportional to the current value of the weight. The idea is to make weight changes based on how far it is from the top boundary (if increasing) or the bottom boundary (if decreasing). This makes learning faster if the difference between the current value of the weight and the boundary towards which the modification is made is larger, and makes learning slower otherwise. The weight modification formula is given below and is illustrated in Figure 8.5.

$$W^{new} = W^{old} + \eta \Delta W$$

$$\text{where} \begin{cases} \Delta W = W^{max} - W^{old} & \text{if a positive reinforcement or} \\ \Delta W = W^{old} - W^{min} & \text{if a negative reinforcement, and} \end{cases}$$

η is the learning rate.

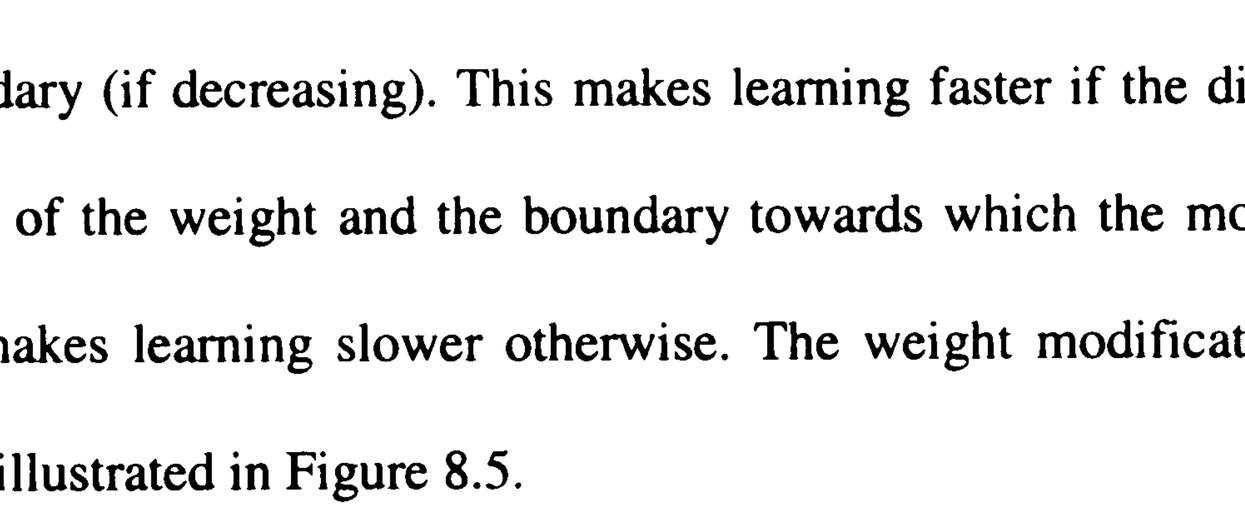


Figure: 8.5 : Weight Modification Policy

8.6.3 Learning Rates for Rewarding and Penalising

The learning rate is a constant that identifies the proportion of the weight difference to take into account for the actual step size of the modification. However, the nature of IR is such that usually not all documents retrieved by the system will be opened by the user and therefore not many documents (of a retrieved set of documents) will be judged as useful by an average user. As a result, only those (few) documents that are judged by the user as useful are rewarded (i.e. the weights of matching concepts and keywords are increased and the query concepts that are not present in them are added). All other documents in the retrieved set are regarded as false hits (i.e. implicit negative feedback) and therefore are penalised (i.e. weights of matching concepts and keywords are negatively reinforced) by our learning algorithm. As a result of this, on average, the weights of unit-concepts and keywords are likely to be negatively reinforced more often than they are positively reinforced. If not controlled, this imbalance of rewarding and penalising would lead all the weights to end up with the minimum weight value allowed (0.1) in the long term. This problem has also been reported by Scott Weiss [Weiss et al. 1997] in his attempt to use Littlestone's "Winnow" algorithm for a newsgroup classification task. A way to resolve this problem is to use different learning rates for positive and negative reinforcements. However, deciding precise values for positive (η) and negative (β) learning rates is difficult. It depends on a number of factors including the number of queries tried, the composition of the queries, user judgments, etc. Based on the results of a few preliminary experimentations on the Cranfield collection, they were set to $\eta=0.04$ and $\beta=\eta/3=0.0133$.

8.6.4 Informative Factors of Comparison Units

The weight reinforcement strategy described so far treats each matching entity (unit-concepts and keywords) as equals regardless of their informativeness, i.e. the weight of each unit-concept/keyword is reinforced by a computed amount based only on the learning rate and the current value of the weight. However, not every concept (or keyword)

is equally informative. We have identified some differences (four levels) in the degree of informative-ness of the information items we use (i.e. unit-concepts and keywords). A way to deal with this problem is to use further weighting factors to re-weight the weight modification steps based on their informative-ness. Depending on the number of individual words they were composed of, they were categorised into four groups of informative-ness and four significance weighting factors were used to re-weight the weights initially computed based on the learning rate. We call these significance factors (weights of weights) "***Informative Factors***" in order not to confuse them with others. Given below are the four different levels of informative-ness considered. They are listed in increasing order of their informative-ness.

a. One-term keywords
b. Keyphrases (Keywords with more than one term)
c. Unit-concepts with one-term components (one-term object and one-term attribute)
d. Unit-concepts with multi-term components (at least one component constitutes more than one term)

The parameter values used in the final experiments were:

Positive Reinforcement Rate: $\eta = 0.04$

Negative Reinforcement Rate: $\beta = \eta/3 = 0.0133$

Informative Factor of unit-concepts with multi-terms: $\mathbf{d} = 3.0$

Informative Factor of unit-concepts with single-terms: $\mathbf{c} = 2.0$

Informative Factor of KeyPhrases: $\mathbf{b} = 1.6$

Informative Factor of single-term keywords: $\mathbf{a} = 1.0$

These values are not optimised for their performances but selected based on empirical results of preliminary experimentation. As can be seen, keyphrases (multi-term keywords) are weighted higher than one-term keywords, as they are likely to be more meaningful than one-term keywords. Unit-concepts are weighted even higher than keyphrases as they are more flexible and therefore considered rich in expressive power. This is simply because they are not restricted to consecutive words as in the case of a keyphrase. Unit-concepts

with multi-term objects or attributes are obviously the most informative among the four categories, and so have the highest significance of all.

The values of the weights of the Informative Factors were decided so that a desired set of (pre-decided) incremental and decremental steps were obtained for a unit-concept or a keyword with an average weight value of 2.5. The following table (Table 8.2) shows the complete set of equations used for the weight modifications for each case of informative-ness. The values of the step sizes are also shown for each case for a concept with an average weight of 2.5.

	Weight Increase Formula		**Steps when W^{old}=2.5**	
	Positive Reinforcement	Negative Reinforcement	Weight Increase	Weight Decrease
Unit-Concepts with multi-term objects/attributes	$W^{old} + \eta.d.(W^{max}-W^{old})$	$W^{old} - \beta.d.(W^{old}-W^{min})$	0.3	0.096
Unit-Concepts with single-term objects and attributes	$W^{old} + \eta.c.(W^{max}-W^{old})$	$W^{old} - \beta.c.(W^{old}-W^{min})$	0.2	0.064
Keyphrases	$W^{old} + \eta.b.(W^{max}-W^{old})$	$W^{old} - \beta.b.(W^{old}-W^{min})$	0.16	0.0512
Single-term Keywords	$W^{old} + \eta.a.(W^{max}-W^{old})$	$W^{old} - \beta.a.(W^{old}-W^{min})$	0.1	0.032

Table 8.2 : Weight Reinforcement Formulae

8.7 Retrieval Process and Similarity (RSV) Computation

8.7.1 Setting up Concept Lattices of Queries and Documents

The retrieval process begins when a query (a natural language expression) is issued by the user. This query expression is pre-processed as described in Section 8.2 and concepts are extracted. A concept lattice of the extracted concepts is then created as described in Section 6.4 by setting up a BAM with the object-attribute link information (Figure 6.4). Then, in order to compare the query concept lattice with the document concept lattices, the concept lattice of each document (one at a time) is also set up with a BAM. The nodes of the query concept lattice are then compared with the nodes of the document concept lattices for partial matching (Section 7.1).

8.7.2 Candidate Node/Concept Pairs for Comparison

As already mentioned in Section 7.1.1, not all query concepts match with all document concepts, and therefore attempting to match all query nodes with all document nodes is a waste of time. Instead, we extract "candidate" concept pairs to match between the query and the document based on the presence of common features (unit-concepts and keywords) between them. The candidate concept extraction process works mainly by looking for the most specific concept in the document lattice for each query object (i.e. using object concepts). Attribute concepts (i.e. the most generic concept containing a given attribute) are also used in the cases where a related object concept is not available in the document. There are a number of cases to consider here when developing an algorithm to extract "candidate concept pairs".

Firstly we look for the presence of query objects and attributes in the document representation. For each object and attribute common to the document and the query, object concepts and attribute concepts (respectively) are extracted from both the query and the document BAMs. Such object and attribute concept pairs are the candidate concept pairs to match between the query and the document. During this process, we make sure tthat extract the most specific concepts are extracted wherever possible and also that the same concept pair is not extracted more than once. Also we avoid extracting document (query) concepts that are general (in the general-specific hierarchy in the concept lattice) to any of the document (query) concepts already extrcted to match with the same query (document) concept. If an object or attribute in the query appears as both object and an attribute in the document representation, we check whether there is any order relation (in the concept hierarchy) between them in order to avoid matching two related document concepts with the same query concept. Only the most specific concept is considered for matching in such cases. However, there are some cases where we find the same term (word or phrase) appearing both as an object and attribute in document representations, but are

not related in the concept hierarchy. Such concepts are regarded as two distinct concepts (ideas). In this case, the attribute concept given by the document BAM is also taken into account as a candidate concept to match with the object concept given by the query BAM, in addition to the object concept given by the document BAM.

The following are the different possible cases (eight cases) identified and taken into account in the algorithm that extracts candidate concepts to match between queries and documents in our implementation (see also Figure 8.6 and 8.7).

1. The query object is present ONLY as an object in the document representation
2. The query object is present ONLY as an attribute in the document representation
3. The query object is present as both an object & attribute in the document and they are related (i.e. the query object plays both object and attribute roles in the document and they are present in the same concept or in different but related concepts in the super/sub concept hierarchy)
4. The query object is present as both an object & attribute in the document and they are not related
5. The query attribute is present ONLY as an object in the document representation
6. The query attribute is present ONLY as an attribute in the document representation
7. The query attribute is present as both an object & attribute) and they are related
8. The query attribute is present as both an object & attribute and they are not related

The following two charts illustrate these cases and how candidate concept pairs can be extracted to match between the query and the document. Some cases have more than one alternative to process them and some processing options cover some others. The algorithm we have developed (given in Figure 8.8) covers all the cases with no alternative options (i.e. cases 1.1, 2.1, 2.3, 3.1, 4.1, 4.3, 5.1, 5.2 and 6.1) and the following alternative options of the cases with alternative options.

1. Case 2.2 : Option 2
2. Case 3.2: Both options 1 and 2
3. Case 4.2: Option 2
4. Case 5.3: Option 2
5. Case 6.2: Both options 1 and 2

A working example illustrating which candidate concept pairs are extracted by this algorithm to match between a query and a document (from Cranfield collection) is given in Appendix B.

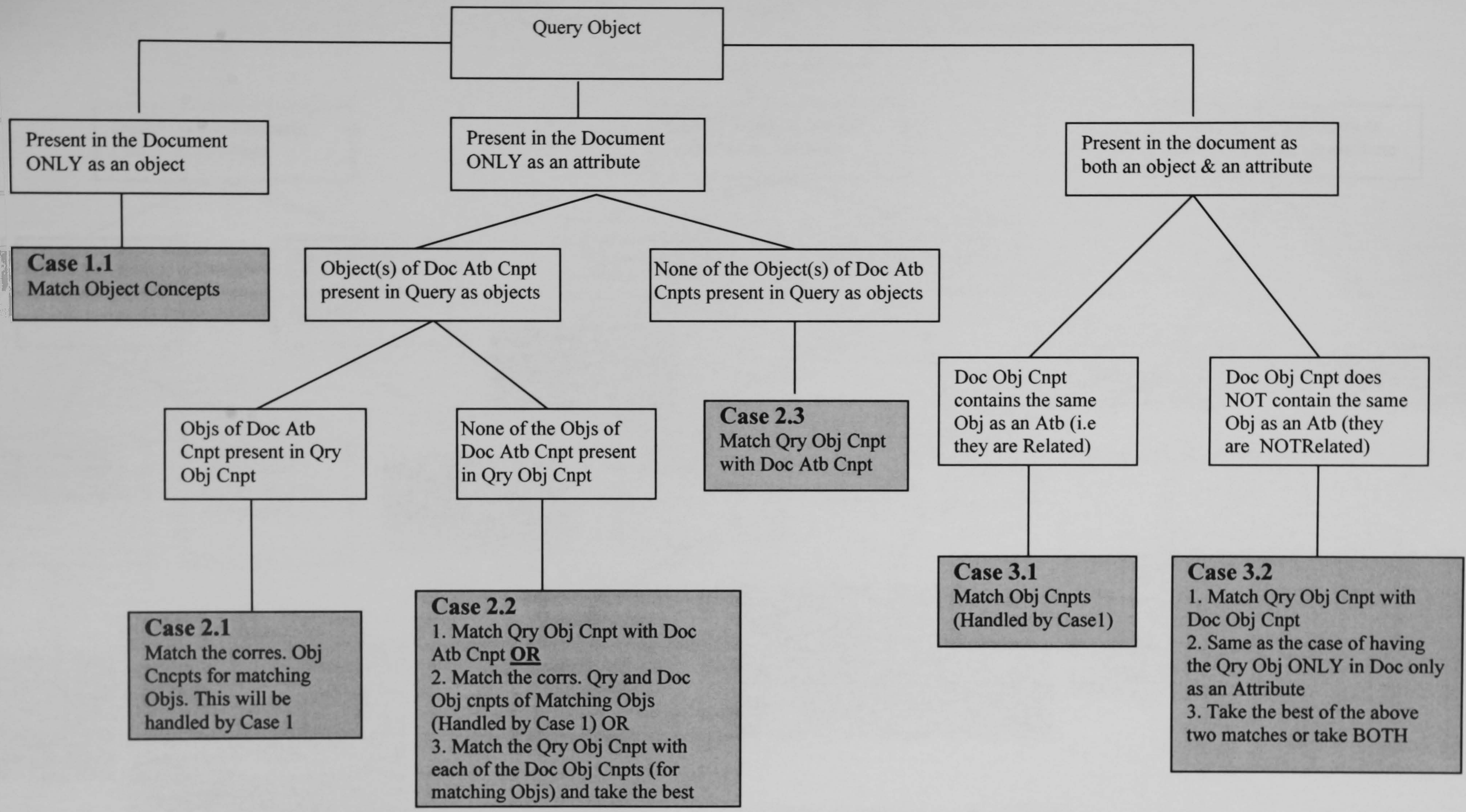


Figure 8.6 : Candidate Concept Pairs Extraction Based on Query Objects

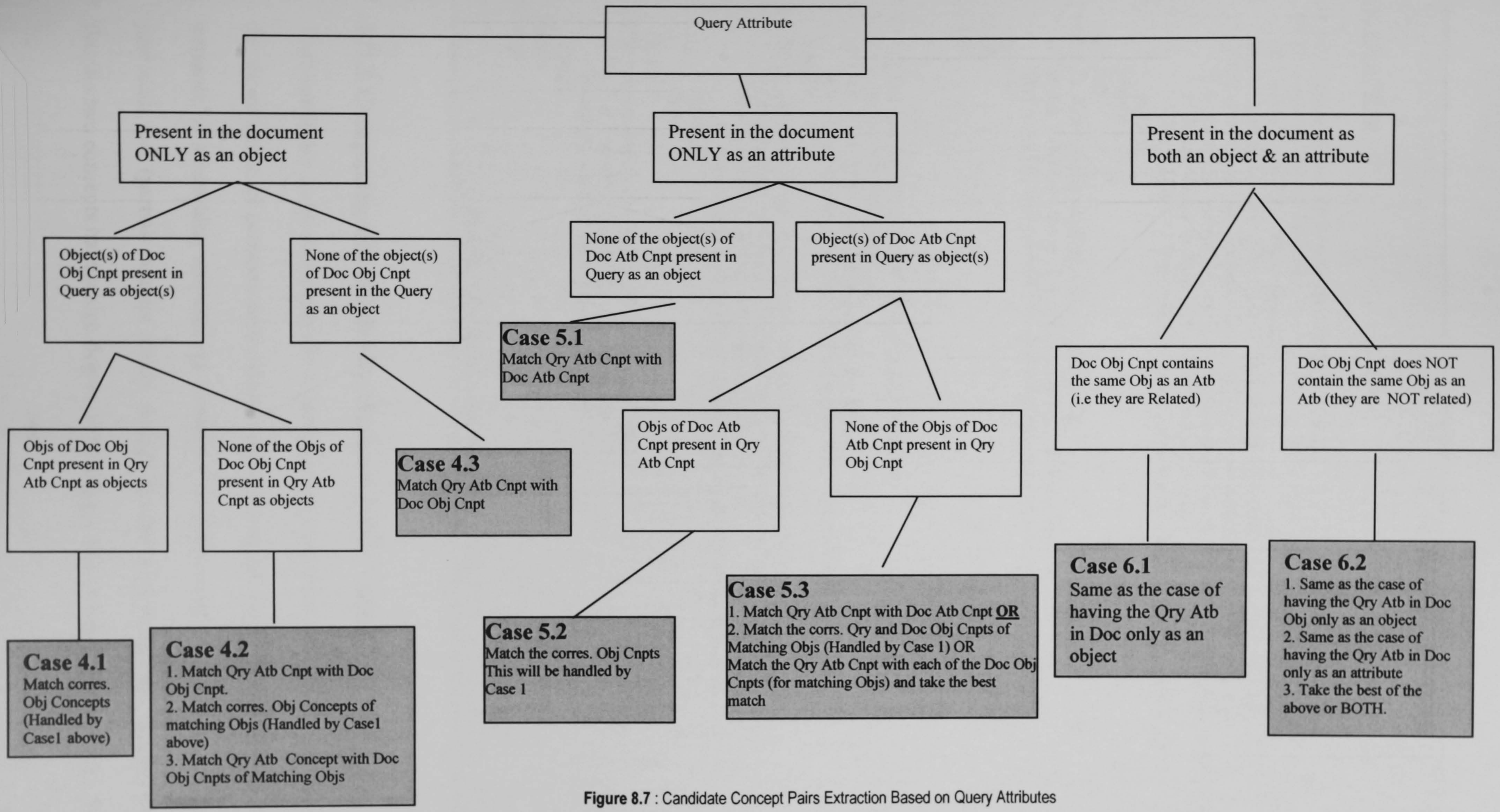


Figure 8.7 : Candidate Concept Pairs Extraction Based on Query Attributes

The Algorithm

```
FOR each Query Object (QO) in the Query Object List
    IF QO is present in Doc Object List
        Match the Qry and Doc Object Concepts
        (this covers case1.1, 2.1, 2.2,3.1 and 1st case of 3.2 (3.2.1) AND case 4.1, 4.2, 5.2, 5.3)
        IF QO is present in Doc Attribute List (i.e. QO is present in both Doc Obj and Atb Lists)
            IF they are NOT related (i.e. Doc Obj Cnpt for the Object QO do not contains the QO as an Attribute)
                IF Objs of Doc Atb Cnpt NOT present in Qry Obj List
                    Match Qry Obj Cnpt with Doc Atb cnpt (covers the other part of case 3.2 (3.2.2))
                ENDIF
            ENDIF
        ENDIF
    ELSE (i.e. QO is present ONLY in Doc Attribute List)
        IF Objects of Doc Attribute Cnpt NOT present in Qry Object List
            Match Qry Object Cnpt with Doc Attribute Cnpt (covers case 2.3)
        ENDIF
    ENDIF
ENDFOR


FOR each Query Attribute (QA) in the Query Attribute List
    IF QA is present in the Doc Obj List
        IF Objs of Doc Obj Cnpt NOT present in Qry Obj List
            Match Qry Atb Cnpt with Doc Obj Cnpt (case 4.3, 6.1 and part of case 6.2 (6.2.1))
        ENDIF
        IF QA is present in the Doc Attribute List (i.e. present in both Lists)
            IF Doc Obj Cnpt NOT contain the same Obj as an Atb (i.e. they are not related)
                IF Objs of Doc Atb Cnpt NOT present in Qry Obj List
                    Match Qry Atb Cnpt with Doc Atb Cnpt (other part of case 6.2 (6.2.2))
                ENDIF
            ENDIF
        ENDIF
    ELSE (i.e. QA is present only in the Doc Attribute List)
        IF Objs of Doc Atb Cnpt NOT present in Qry Obj List
            Match Qry Atb Cnpt with Doc Atb Cnpt (case 5.1)
        ENDIF
    ENDIF
ENDFOR
```

Figure 8.8 : Candidate Concept Extraction Algorithm

8.7.3 Computing the Similarity Measure (RSV Value)

Each candidate query and document concept pair extracted for matching is then examined for the presence of common unit-concepts (partial concept matching) and keywords. The presence of a matching unit-concept between a candidate query and a document concept pair leads to a (partial) concept match. A high number of unit-concept matches indicates that the two concepts have a high degree of similarity. The presence of a common object or

attribute that does not participate in a matching unit-concept leads to a keyword match (see the illustration in Figure 8.9). The matching unit-concepts and keywords between candidate concept pairs are stored in two lists (*MatchingConceptsList* and *MatchingKeysList*) and are pruned for any duplication. Pruning does the following:

1. Removes keywords that are contained in other more expressive keywords or in unit-concepts
   e.g. The keyword "*bridge*" is removed from the *MatchingKeysList* if the keyphrase "London Bridge" is also present in the *MatchingKeysList* or one of the unit-concepts {*London*} → {*bridge*} or {*bridge*} → {*London*} or {*London bridge*} → {*fall*} is present in the *MatchingConceptsList*.
2. Removes unit-concepts that are contained in other more expressive unit-concepts
   e.g. {*bridge*} → {*fall*} is removed from the *MatchingConceptsList* if {*London bridge*} → {*fall*} is present in the list.

It is the unit-concepts and keywords in the pruned lists that contribute to the RSV computation. See Appendix B for an actual example.

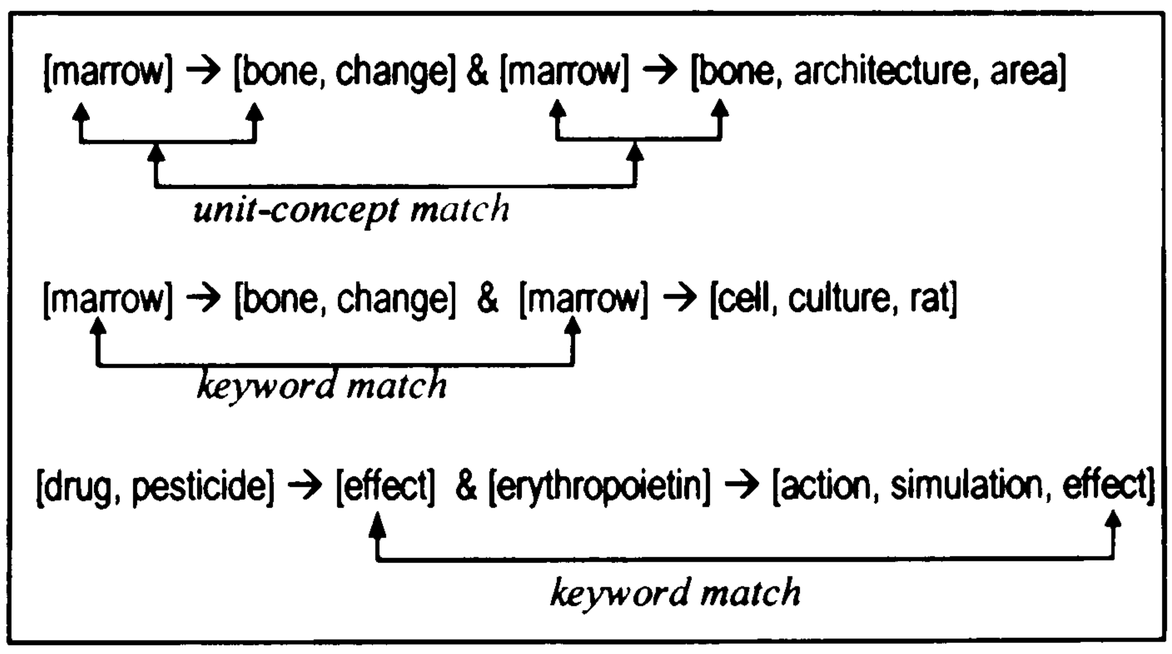


Figure. 8.9 : Unit-concept Matches and Keyword Matches

A similarity measure for each candidate query-document concept pair is considered to be the sum of the weights of the matching unit-concepts and keywords. The final RSV value for a query-document pair is then computed as the sum of the similarity measures of all the candidate query-document concept pairs considered between the query and the document. The RSV values thus computed for each query-document pair are then subject to thresholding in order to decide which documents to retrieve. The documents whose RSV values exceed their thresholds are presented to the user ranked simply in the decreasing

order of their RSV values. Notice that we do not constraint the RSV value to a particular range (e.g. [0-1] as in many other systems) and therefore they can take any positive value.

8.7.4 Concept Size Variability and Thresholding

The similarity computation described above depends on the sizes of the concepts (i.e. the number of objects and the number of attributes present in the concepts) and the weights of the unit-concepts and keywords. Different concepts are of different sizes in terms of the number of objects and the number of attributes they contain. Query concepts in particular tend to be shorter due to the fact that most of the time, queries are short expressions (compared to documents) and thus contain less detail. The number of unit-concepts that the concept extraction process can extract from such short expressions is smaller compared to much lengthier query expressions. This feature tends to favour lengthier queries as they are likely to be represented with more unit-concepts thus giving them a higher chance to match with documents. A common solution for this type of sizing problem is to use some kind of size normalisation. In the implemented prototype of our model, however, concepts were not normalised for lengths; instead, a concept length dependent thresholding mechanism was used to compensate for the varied sizes of queries.

This is a dynamic thresholding strategy computed by taking into account the total number of unit-concepts available in all the candidate query concepts considered for comparing with document concepts (i.e. the total number of unit-concepts we are looking for in the candidate document concepts). Note that the size of a query is determined by the number of distinct unit-concepts that its representation comprises. This value is multiplied by a predefined base threshold value (see the equation given below). The use of a base threshold value allows us to experiment with the best thresholding value to use by varying the base threshold. Based on the results of a few preliminary experimentations, the base value was fixed to 1.3. No further experiments were conducted to evaluate the impact of thresholding. Such work is reserved for future experimentation.

$$\text{Dynamic Threshold} = (\text{Base Threshold}) \times \left(\begin{array}{l} \text{Number of unit-concepts in all the candidate} \\ \text{query concepts considered for matching} \\ \text{between a given query-document pair} \end{array} \right)$$

See Appendix B for a threshold calculation in an actual example.

8.8 FEEDBACK PROCESSOR

The task of the feedback processor is to implement the reinforcement learning strategy described in Section 7.2.2. It accepts the user feedback in the form of *yes* or *no* (i.e. accepts a document as useful or rejects it as not useful) and modifies the document representation accordingly. The modifications made include:

1. adding unit-concepts of the query into the document if the document is judged as useful;
2. increasing the weights (rewarding) of the unit-concepts and the keywords that helped the retrieval of a document when the document is judged as useful; and
3. decreasing the weights (penalising) of the unit-concepts and keywords that helped the retrieval of a document when the document is not judged as useful.

These were already described in Section 7.2.2 for unit-concepts. In addition to the weights of unit-concepts, the contributions of matching keywords are also counted for the computation of RSV values and therefore they are also reinforced in the same way as unit-concepts. Implementation details of rewarding (weight increment) and penalising (weight decrement) weights are described in Section 8.6 and therefore are not repeated here. The complete reinforcement learning process with keyword weight learning is illustrated in Figure 8.10.

As already indicated, an important feature of the way we use relevance feedback information is that the modifications made based on them are retained for later use. We expect the document representations to converge to a well representative set of concepts (for each document) over time through these modifications. Such a set of concepts, in fact, will become more customised to the vocabulary and the writing style of the end users, as it

is the concepts of the user formulated queries that are appended into relevant document representations.

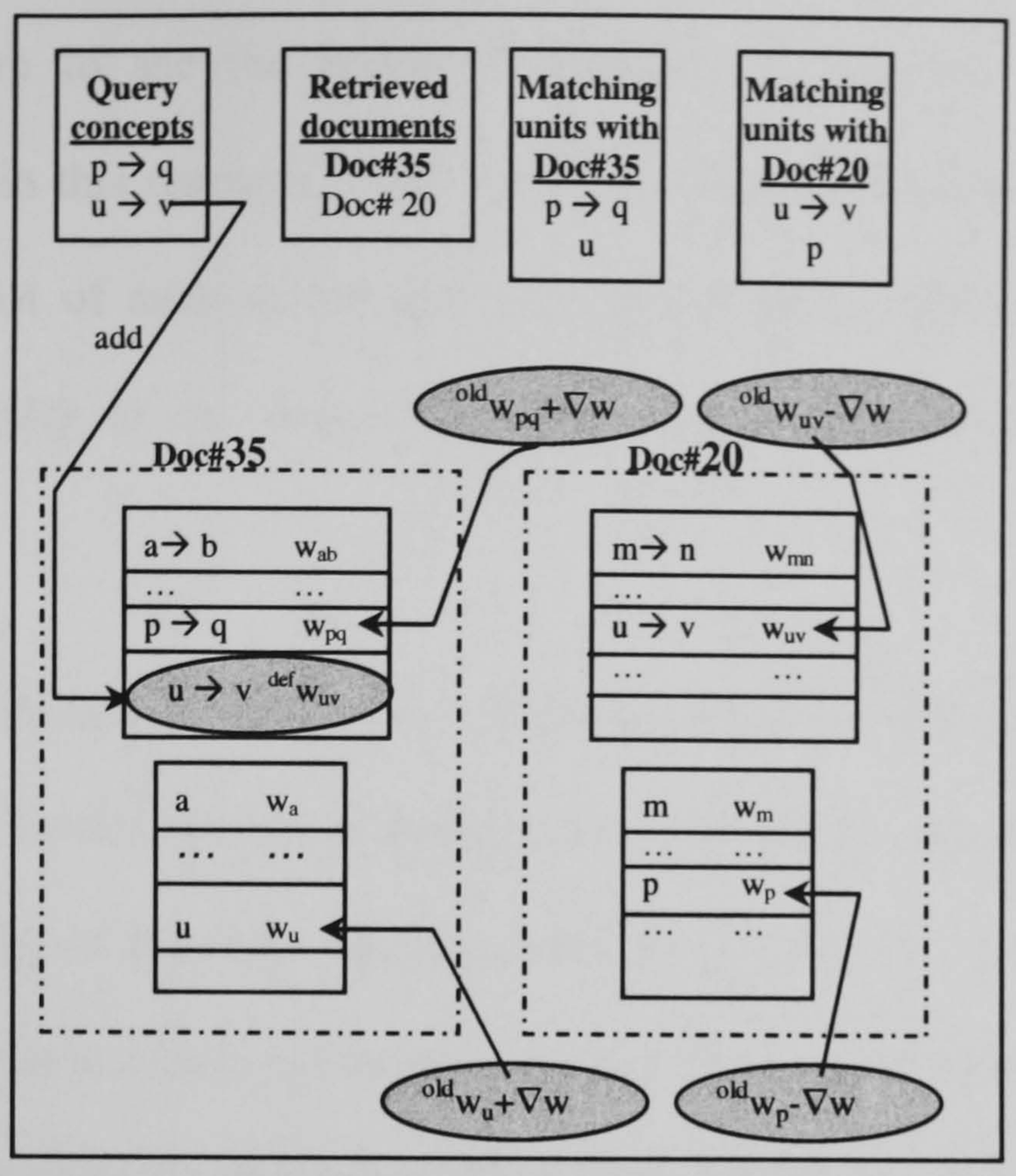


Figure 8.10 : Full Reinforcement Strategy
(Concept Addition/Concept Learning and Keyword Learning)

An adverse effect of adding all the unit-concepts of queries into the documents judged as relevant to them is that it may result in unnecessary unit-concepts creeping into the document's representation. Such unimportant unit-concepts are expected to end up with low weights in the long run as a result of the weight learning process described above, and thus can easily be pruned off.

8.9 SUMMARY

Firstly, this chapter presented in detail how pat-of-speech tags, noun (NG), verb (VG) and other (OG) chunks, and connecting (prepositional) words were used for extracting objects, attributes and their relationships to form unit-concepts. These concepts may not represent the content of a document as completely and accurately as a human would perceive it. Also, they are far from what can be achieved by manual concept extraction within the same FCA formalism. The representation of concepts in the human brain will be even more complex than such a formulation. Therefore the results of the implemented prototype

might not represent the true power of concept matching and concept learning. Given the deficiencies of automatic concept extraction, we have no choice but to extract concepts manually if we are to see the actual performance of concept matching within the framework set out in this research. However, this was not attempted during this research due to the high cost of such an attempt. Our goal in this particular task is to extract concepts automatically to an extent that is sufficient to demonstrate the potential of concept matching.

Secondly, the implementation detail of concept matching was described. Node or concept matching between concept lattices as described in this chapter was found to be a difficult task. Direct matching of the most specific query nodes with the most specific document nodes is impossible as it is difficult to build a concept lattice which sufficiently represents a query due to the inadequate amount of information present in queries and the problem of duplicate matches. This made us develop an algorithm to extract candidate concept pairs to match between the query and the document. This, as we have found, is sufficient to help concept matching despite its considerable expense in terms of computational resources.

In addition, the details of the formulae used for weight modifications, RSV calculation, and dynamic threshold calculation were given in this chapter. As seen in these formulae, there are a number of parameters that can be optimised, but we have used only a selected set of values based on the empirical results of preliminary experiments conducted on a small document collection. They will need customising to the environment (document collection) as appropriate and optimised for better performances.

The following chapter evaluates various properties of our model using the results of the experimentations of our implementation. Our major interest in those experimentations is to evaluate the impact of concept matching and concept learning on retrieval performance over time.

CHAPTER 9 - RESULTS AND EVALUATION

This chapter presents our experiments, results and observations on the experimental results. We conducted a carefully designed set of experiments based on the traditional evaluation model to examine the effect of individual components of the system, as well as the performance of the system as a whole in its full capacity. Two main classes of experiments were designed: one based on an "Incremental Learning-Testing" strategy and the other based on a "Probe Testing" strategy. Tests based on the first strategy were used to evaluate the performance improvements of the system as it gains experience through user interactions. The probe testing was used mainly for producing comparable results to evaluate the system against published results. The primary test collection used for evaluating performance improvements was the Cranfield test collection. The CACM and CISI collections were used to produce results for comparison with published results.

The first part of this chapter presents the test collections used, their detailed properties and statistics. The properties desired of a test collection for the purpose of evaluating learning systems, and details of the two testing strategies mentioned above are presented next. Individual experiments conducted, their results and observations are given in the latter part.

In this chapter, the performance of "keyword matching", as used in our system, is compared with the performance of other aspects of the system. The "keyword matching" discussed here should be distinguished from the traditional keyword matching used in conventional keyword-based IR models. In our case a keyword is a constituent component of a unit-concept. The way it is extracted and processed is radically different to conventional keyword-based systems. We do not explicitly extract each term as a keyword from document contents, but keyword extraction is a result of unit-concept extraction. Also, we do not use a *stoplist* for removing non-significant keywords or unit-concepts, and terms are not stemmed to obtain their roots. A keyword match takes place only when one

of the two constituent components of a unit concept in a query matches with one in a document concept (but not both - in this case a unit-concept match takes place). Recall that, a term or a phrase that makes a keyword match between a query concept and a document concept is pruned out if it participates in a concept match between the same query and the document (Section 8.7.3). This is to make sure no feature is counted more than once. In addition, the weights of individual components (keywords) of such a matching unit-concept are not rewarded (the weight is not increased); only the weight of the unit-concept is rewarded. The weight of a keyword is rewarded only if it makes a keyword match. Our intention here is only to compare and contrast the effectiveness of the two comparison units (entities) used in our model on performance. Therefore, results reported here for keyword matching should not be confused and compared with results of of conventional keyword-based systems. Also note the alternative use of the terms "document collection", "test collection" and "test database" to refer to the same.

9.1 Test Collections

There are a number of document collections around for evaluating IR systems. These include the Cranfield, Reuters, CACM, CISI, MEDILINE and TREC collections (Table 9.1). Compared to the TREC collections, the others are very small in size (in the order of less than 5000 documents). Despite the availability of bigger collections such as TREC, high computational and storage demands needed for using bigger collections for systems evaluation have restricted most researchers to smaller collections. Computational demands are severe in the evaluation of learning models (such as ours) compared to static IR models, as they (learning systems) need to undergo training before their performance can be evaluated.

Any test collection that is meant for evaluating IR systems should possess: (1) a collection of documents, (2) a collection of queries and (3) relevance assessments (i.e. user decisions on which documents are relevant to which queries). The larger the collection in terms of

the number of documents and queries the better (closer to the reality). The number of documents and queries is critical in evaluating learning systems, as they naturally need more queries and/or documents so that the collection may be divided into training and testing sets.

Creating a test collection with query-document relevance assessments is a difficult and time-consuming task that requires a lot of human effort. Most of the test collections listed below in Table 9.1 were created decades ago mainly for the purposes of document categorisation or evaluation of static IR systems. Newer collections, such as Reuters and some of the TREC collections that have also been designed for document categorisation do not contain queries and relevance assessments. No document collection (or corresponding

Collection	Documents	Queries	Size in MB	Bytes per Doc.	Details
LISA	5,872	35	3.4	610	Library and Information Science Abstracts
CACM	3,204	64	2.2	717	Titles and abstracts from Communications of the ACM journal CACM. Important feature: a number of articles reference each other
CISI	1,460	112	2.2	1,526	Document abstracts in library science and related areas published between 1969 and 1977, extracted from Social Science Citation Index by the Institute for Scientific Information
Cranfield	200/1,400	225	1.6	1,203	Document abstracts in aeronautics and related areas, originally used for tests at the Cranfield Institute of Technology in Bedford, England.
Time	423	83	1.5	3,663	Articles from the Time magazine's world news section in 1963
Medline	1,033	30	1.1	1,079	Document abstracts in biomedicine from National Library of Medicine
ADI	82	35	0.04	466	Document abstracts from library science and related areas
OHSUMED	348,566 references				References from 270 medical journals over 5 years (1987-1991)
Reuters	22,173	No Queries	-		Documents from the Reuters newswire
	21,578	No Queries	-		An improved version
TREC	Huge		~ 4GB		This has 5 volumes with material from various sources. Relevant assessments of documents for queries is not complete

Table 9.1 : Test Collections

evaluation methodology) seems to have been created specifically for evaluating interactive learning systems. Therefore, the present test collections lack certain desirable features (given in Section 9.1.1) for evaluating learning systems that adapt to their environment. This, together with lack of a suitable evaluation technique, hampers the evaluation of the full potential of adaptive IR systems.

9.1.1 Desired Properties of a Test Collection

Depending on the target of the IR system, certain specific properties become more desirable over certain others. There may be certain properties that are desirable to produce, for instance, more robust or more representative evaluation figures, while certain other properties may be necessary for specific learning algorithms used by an IR system, and yet others may be required to test for specific features of an IR model. Among them, domain independency, nature or type of queries and documents (e.g. keyword type queries or natural language queries) and overlaps in documents and queries (i.e. presence of more query-document cross relations in relevance assessments) are the major properties that we are concerned with.

9.1.1.1 Domain Independency

It is best if the documents in the collection are not from a specific domain. In specific domains, the meaning of certain domain-specific words may be more precise than they are in the general use. Therefore, performance measures obtained on such a domain-specific document collection might not represent the true performance of the system as the system can be tuned for the specific domain. A system that is tuned for a specific domain is unlikely to show a similar performance in another domain. Therefore, one cannot conclude that a particular system is better than another based on the performance results obtained only on one collection. Unfortunately, most of the publicly available collections (including Cranfield, CISI, and CACM) are domain-specific (see Table 9.1). The Reuters collection

seems to cover a wide range of topics, but lack of queries and relevance assessments have prevented it from being used for evaluating IR systems.

9.1.1.2 Expressiveness

Test collections that were originally created for evaluating keyword-based retrieval models lack adequately expressed natural language queries. In addition, short documents might not contain sufficient background information to identify correctly what the document is mainly about. The length of a query has drawn much attention of the IR community. In particular, present test collections have been crtitisised to have too long queries. This is in accordance with the finding that the length of an average (keyword-based internet query) query is between 1.5 to 2.5 words long. However, these findings were mainly based on the investigations made on general Internet populations [Silverstein et al. 1998, Spink & Xu 2000, Jansen et al. 1998]. The circumstances and context between searches on more regular IR systems such as DIALOG and searches on the Web done by the general Internet population are known to be very different [Jansen et al. 1998]. A number of other studies have shown that the mean number of search terms in the searching of regular IR systems is 7 to 15 (find details in [Jansen et al. 1998]). The nature of query processing and the facilities provided for query formulation in the IR systems used in these studies may have affected the way queries were formulated by the users. However, changes in the way queries are formulated and searching is conducted can be expected as more natural language interfaces that encourage longer queries are made available [Wolfram 2000]. In line with this expectation, our model is designed to work on reasonably well-expressed natural language query expressions. Such queries are richer in terms of the semantic relationships between words and are naturally longer than the Web queries issued to keyword-based search engines. According to the past studies which have indicated that longer queries are used by familiar users, such long queries would be more suitable for (though not limited to) expert users. Our target, however, is to use such more realistic natural language queries with richer semantics.

9.1.1.3 Overlaps

A primary objective of any learning system and of our system in particular, is that learning should trigger the retrieval of previously missed-out (un-noticed) but relevant documents. We attempt to achieve this goal by enhancing document representations based on query-document interactions, where knowledge acquired by a document through previous query interactions is expected to help its retrieval by a subsequent (relevant) query.

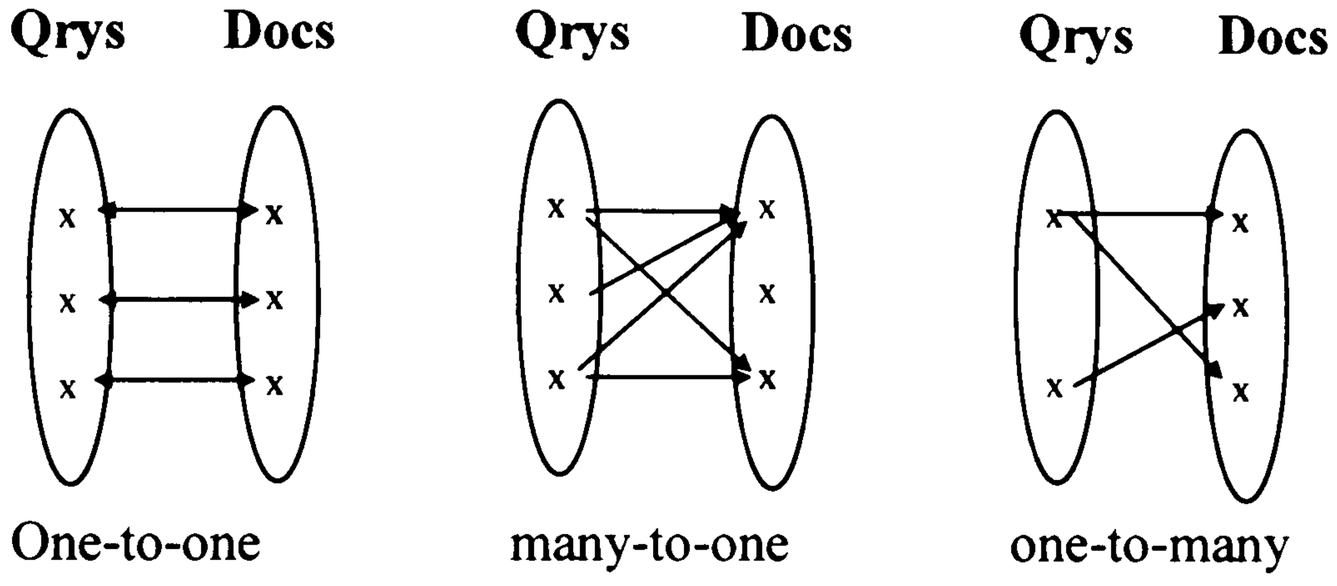


Figure 9.1 : Different Query-Document Cross Relations

This important feature cannot be tested if the document collection lacks sufficient cross-relations. Medline is an example of a collection in which no document is relevant to more than one query (1-to-1). The diagram given above (Figure 9.1) shows different query-document relationships that a test collection might possesses. A collection with more many-to-one relations is desired for our learning strategy.

9.1.1.4 Similar Queries

The overlaps mentioned above occur mainly as a result of similar information needs. Similar information needs may or may not be similarly formulated using the same vocabulary. Our learning mechanism is expected to take care of the differently formulated queries that are targeted at the same documents. A result of this learning scheme is that the system tends to favour queries it has seen before. This is not at all an undesired feature, as similar queries do occur in reality. Even humans are better and quicker at answering familiar queries in a known context than answering unfamiliar queries.

The likelihood of similar queries to occurring has been found to be greater on more regular search services compared to Web queries. Based on the results of the analysis of approximately one billion queries collected over 43 days from the AltaVista Query Log, Silverstein [Silverstein et al. 1998] conjectured that a small set of queries is repeated many times over the course of the day. An average query frequency (over 43 days) is 3.97. Twenty five (25) most common queries formed 1.5% of the total number of queries asked in the 43-day period, despite being 0.00000016% of the unique queries. In this study, two queries were considered the same if they contained the same words with the same capitalisation, ignoring the word order and operators. Also, Jansen et al. [Jansen et al. 1998] give some statistics about modified queries, i.e. a subsequent query in succession, created by the same user by adding terms to or removing terms from the previous query, and also identical queries issued by the same user. The identical queries reported in this study include queries generated by the search engine (Excite) when the second and further pages are viewed by the user. Such queries cannot be considered as identical queries formulated by the user. The identical queries issued by different users were not analysed in this study. In a study on query clustering, Wen and colleagues [Wen et al. 2002] analysed 2.7 million user queries (FAQs) directed to the Encarta encyclopaedia. A particular result of this analysis placed 66% of queries in 1756 clusters. The average number of queries per cluster is 8.24, confirming their hypothesis that many users tend to use similar or identical queries.

The use of relevance feedback (or more like this) and successive searches also account for similar queries, in addition to the identical queries created by the users themselves. Wolfram [Wolfram et al. 2001] found an increasing trend of the use of relevance feedback (5% to 9.7% from 1997 to 1999). An interactive survey conducted by Spink and Xu [Spink & Xu 2000] found that many users had conducted two, three or more searches over time when seeking information on a particular topic. These results support positively for a system's favourable response to queries similar to the ones it has seen before.

9.1.2 Test Collections Used

After an initial investigation of the document collections for the basic properties, the four collections: MED, CISI, CACM and Cranfield (CRAN) were initially chosen for evaluating our system. The collection size (TREC - too big, OHSUMED - too big, ADI - too small, TIME -too small) and lack of queries (Reuters) were the major factors that caused us to drop the other collections. Further examination of the four chosen collections revealed that Cranfield is better in terms of cross-references, and the number and nature of queries (natural language query expressions). The MED collection does not possess the desired overlaps or a sufficient number of queries. The CISI is poor in terms of expressive natural language queries and number of queries. The CACM lacks sufficient overlaps (cross-relations) and has too few queries. Compared to the other three, the Cranfield collection has more queries, a greater degree of overlaps and reasonably well-expressed natural language queries (see Table 9.2 and Table 9.3 below for comparison statistics). Based on these facts, the Cranfield collection was chosen as our main test collection, primarily for the experiments that were aimed at evaluating the performance of interactive learning.

No of relevant queries	No of Documents			
	CRAN	CISI	CACM	MED
>10	0	0	0	0
10	0	0	0	0
9	1	1	0	0
8	2	2	0	0
7	7	2	0	0
6	11	10	1	0
5	16	23	8	0
4	62	49	16	0
3	137	114	32	0
2	271	266	92	0
1	417	458	406	696

Number of documents which are relevant to more than three queries

Table 9.2 : Document-Query Cross Relations in Test Collections

Note that Table 9.2 above should be read starting from the number of documents. For instance, in the Cranfield collection there are 2 documents judged as relevant to 8 queries each.

Test Collection	No. of Words per Query
CACM	26.25
MED	19.63
CRAN	17.97
CISI	14.28

Table 9.3 : Word Density in Queries in Test Collections

9.2 PUBLISHED RESULTS

When choosing a set of published results to evaluate the comparative effectiveness of an IR system, it is best if the results of similar model(s) can be chosen. In particular, we were interested in a model that uses the same or similar theoretical concepts (as ours) and learns the document representations interactively from experience (adaptive). Nevertheless, we were unable to find a set of published results for a model that satisfies all of these requirements. But a set of results published by Carpineto and Romano [Carpineto & Romano 2000] was appealing to us for two reasons. Firstly, this model is based on formal concept analysis and concept lattices, and secondly, the authors have given comparative results of three models: (1) a FCA-based model (referred to as CLR - Concept Lattice-Based Ranking), (2) a Cluster-based model (referred to as HCR - Hierarchical Cluster Based Ranking), and (3) a Best Match model (BMR - Best Match Ranking). The implementation of the BMR model is a simple vector-based model in which the similarity between queries and documents is computed by taking the inner product with cosine normalisation between their vector representations. In the implementation of the HCR model, similarity between a query and a cluster is computed by taking the inner product with cosine normalisation between the query and each cluster in the cluster hierarchy. Also, the degree of similarity between documents has been determined by taking the inner product with cosine normalisation, and using the single link method (i.e. linking the most similar items together) for cluster formation [Carpineto & Romano 2000].

Carpineto and Romano have evaluated their model on the CISI and CACMcollections, and results reported. A problem with these two collections, however, is that their relevance

assessments are not complete. They both have queries to which no documents are assessed to be relevant. Such queries (with no relevant documents) have been discarded from their evaluation. Only the first 35 of the 112 queries of CISI and 52 of the 64 queries of CACM (12 queries do not have associated relevance assessments) have been used. In producing comparable results, we used the same performance metrics on the same collections using the same queries in our evaluation (Section 9.5.9). However, it should be noted that the underlying indexing languages and retrieval criteria are different between Carpineto and Romano's implementation and ours. Also note that, the documents as well as some queries in the CISI, CACM and Cranfield collections contain information about authors and journals (see the example given in Figure 9.2). These were removed from all documents in our evaluations, as they do not help creating concepts in our case.

```
.I 1
.T
Preliminary Report-International Algebraic Language
.B
CACM December, 1958
.A
Perlis, A. J.
Samelson,K.
.N
CA581203 JB March 22, 1978  8:28 PM
```

Figure 9.2 : Fraction of a Document in CACM Collection

9.3 Performance Metrics and Evaluation Techniques

Since recall and precision formed the basis of the Cranfield tests, a number of performance metrics have been defined based on the notion of "relevance" to support evaluation of IR systems. A number of evaluation methodologies/techniques have been suggested using those metrics to compare performance of IR systems. The technique, "Precision at 11 standard recall levels (interpolated)", that facilitate the averaging and plotting of P-R curves is the most frequently used. Other techniques include Average precision over all relevant documents (non-interpolated), Precision at specific retrieval points, R-Precision, Fallout Rate, F-Measure, E-Measure, Novelty Ratio, Coverage Ratio, Average Precision Histogram etc. The computation of these metrics is primarily based upon binary relevance judgements, i.e. given a document collection and a query, some documents are relevant to

the query and others are not (binary relevance judgments). This (binary) assumption leads to derivation of the following four statistical quantities for a given query session. They provide the necessary inputs for the computation of performance figures for the precision and recall based evaluation metrics mentioned above.

- The number of relevant and retrieved documents (true positives)
- The number of non-relevant and retrieved documents (false positives)
- The number of relevant and non-retrieved documents (true negatives)
- The number of non-relevant and non-retrieved documents (false negatives)

We used the Average Precision (non-interpolated) over all relevant documents and Precisions at Retrieval Points 5, 10 and 20 as our primary evaluation measures. Precisions at standard 11-point recall levels were also used to create a few P-R curves. Details of these techniques are given in the following sections.

9.3.1 Precision and Recall Definitions

$$\text{Precision} = \frac{\text{No. of Relevant Items Retrieved}}{\text{Total no. of Items Retrieved}}$$

$$\text{Recall} = \frac{\text{No. of Relevant Items Retrieved}}{\text{Total no. of Relevant Items in the Collection}}$$

9.3.2 Precision at 11 Standard Recall Levels (Interpolated)

This is based on calculating precisions for each query at 11 standard cut-off recall points. These cut-off points are determined based on a set of pre-defined percentages taken from the total number of relevant documents present in the collection for each query. The 11 standard percentages are {0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100%}. For instance, if there are 3 documents relevant to a given query in the collection, then the recall points (cut-off values) x are 0, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7 and 3.0. Precision at a given recall level x is then calculated by going down the ranked retrieved document list until the x^{th} relevant document is found and then dividing x by the position of that document in the rank order. For instance, the recall point is 3 for the recall level 100% in the above example and if the third relevant document appears at the 10^{th}

position in the ranked list, then the precision at the 100% recall level is 3/10. The precisions of each individual query at each recall level thus computed are then averaged to obtain a mean precision figure to represent the performance of the system at that recall level.

One of the problems with this method is that the cut-off points take fractional values. For instance, what does the 0.3^{rd} relevant document mean in the above example? This problem is tackled by "interpolation" in which those fractional values are mapped (rounded-off) to the nearest upper integer. As a result, the precisions at recall levels 0.0, 0.3, 0.6 and 0.9 are all taken to be the precision when the first relevant document is retrieved. This causes unrealistic precision figures to be assigned for queries at recall levels with fractional values. The problem is severe if there not many documents are assessed relevant to the queries. For example, if there are many queries in the collection, say, with only one document each judged as relevant, the P-R curves of those individual queries will be flat horizontal lines. (Recall, cut-off points are 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 and the interpolated cut-off point is one (1.0) for each recall level. As a result the precision at all standard recall levels will be the same for those queries).

Despite this deficiency, the P-R curve on standard 11-point recall levels is the most widely used and accepted technique for comparing the performance between IR systems. Although a P-R curve does not show the actual performance of an IR system, it does not affect too much when the results were computed on the same test collection for the purpose of comparing performance between systems.

9.3.3 Precision-Recall Graph (P-R Graph)

The Precision-Recall graph is created using the precisions at the 11 cut-off values described above. Typically these graphs slope downwards from left to right, enforcing the notion that as more relevant documents are retrieved, more nonrelevant documents are also retrieved. The plots of different test runs can be superimposed on the same graph to

determine which test run is superior. Curves closest to the upper right-hand corner of the graph indicate the best performance. Comparisons are best made in three different recall ranges: 0 to 0.2 (high precision), 0.2 to 0.8 (middle recall), and 0.8 to 1 (high recall).

9.3.4 Average Precision over all Relevant Documents (non-interpolated)

This is a single-valued measure that reflects the performance over all relevant documents. It rewards systems that retrieve relevant documents quickly (highly ranked). The measure is not an average of the precision at 11 standard recall levels. Rather, it is the average of all the precision values obtained after each relevant document is retrieved. As an example, consider a query to which four documents have been assessed relevant, and that they are retrieved at ranks 1, 2, 4 and 7. The actual precisions obtained when each of the relevant documents is retrieved are 1, 1, 0.75 and 0.57 respectively, and the mean of these is 0.83. Thus, the average precision over all relevant documents is 0.83. Usually an average over a sample of queries is reported.

9.3.5 Precision and Recall at Specific Retrieval Points

Precision computed after a given number of documents have been retrieved is considered to reflect the actual measured system performance as a user might see it. Precision at a given cut-off retrieval point is computed by taking the average precisions over all the topics/queries at that cut-off point. TREC uses 9 retrieval points (5, 10, 15, 20, 30, 100, 200, 500, 1000), but we used only 4 retrieval points (1, 5, 10 and 20) as they are the most important for the user, and retrieval points 5, 10 and 20 are the ones Carpineto and Romano have used.

However, a problem with this technique is that it does not reflect the actual performance if the collection does not have as many relevant documents (judged as relevant for a given query) as the value of the retrieval point. For instance, assume that there is only one document in the collection judged as relevant to a given query and that this document is ranked at the top in the retrieved list. The precisions of this query at the retrieval points 1,

5, 10 and 20 are 1, 1/5, 1/10 and 1/20 respectively. The precision figures obtained for this query at retrieval points 5, 10 and 20 do not give a correct impression of the effectiveness of the retrieval despite the fact that the only document relevant to the query has been retrieved. In fact, not many queries in the three collections we used have 20 relevant documents. Usually, precisions at retrieval points are given together with recalls at the same retrieval points. Recall at retrieval point x (for a given query) is computed as the number of relevant documents within the top x retrieved documents divided by the total number of relevant documents present in the collection (for the given query). Typically, averages over a sample of topics (queries) are reported.

9.4 Training-Testing Strategies

Evaluating learning systems requires that systems be trained prior to testing for producing performance figures. Common practice in machine learning has been to divide the sample data set into two halves: one set to be used for training and the other for testing. Since the data points in the test set are not shown to the system during training, this method allows the system to be tested on unseen data. However, in order to produce representative performance figures, the set of data points should be distributed between the training and testing sets in such a way that they both equally represent the domain space.

In our case, it is the set of queries that act as inputs to our learning system. Therefore, it is the set of queries that is to be split between training and testing sets, if we are to follow the above mentioned Training-Testing evaluation strategy. However, this strategy, in its original form, is not suitable for evaluating an interactively learning system or for producing comparative results, because: (1) it is designed for training systems once (only) before the system is put in operation; and (2) it does not allow the production of results based on testing all data points. Therefore, we used three different strategies here: (1) *Incremental Learning-Testing strategy* for evaluating the performance of the system as it gains experience; (2) a *Probe Testing strategy* for producing performance figures based

on testing all queries for the purpose of comparing them with published results; and (3) *Full-Training Full-Testing strategy* for examining the performance of the system on queries that the system has seen (learnt) before. These strategies are described in detail in the following sections.

9.4.1 Incremental Learning-Testing Strategy

9.4.1.1 Methodology

The main objective of this test strategy is to examine how the experience gained through user interactions affects the performance of the system. This was achieved by first splitting the set of queries into two sets (training set and testing set) and then training the system on a (cumulative) subset of the queries in the training set at each training session. The system is tested on all test queries at the end of each training session. This means that the system undergoes a number of training-testing phases during its evaluation process. At each subsequent training session, a set of (new) queries (which were not used in the previous phases), picked randomly from the full set of training queries, are added into the actual training subset. The number of training-testing phases required for producing one set of performance figures depends on the number of queries in the full training set and how many new queries are added to the actual training set at each training session. The more training-testing phases there are, the better. But this requires a greater number of queries in the full training set.

9.4.1.2 Splitting Queries between Training and Testing Sets

Splitting the query set into training and testing tests so that they both equally represent the full set of queries (query space) in terms of desired properties is a difficult task. This includes deciding which features or properties of queries should be considered for equal distribution, how many queries should go in the training set and how many in the testing set etc. There are 225 queries in the collection we used (Cranfield) in all tests that were conducted according to this strategy, and out of them 65 queries were used for testing and

160 queries for training. More queries were placed in the training set simply because we needed more queries to create more training-testing sub sessions.

No. of relevant documents (x) (Category)	No. of Queries with x no. of relevant documents	% of queries with x no. of relevant documents in the collection	No. of Queries for test set (out of 65)	Actual no. of test queries taken
<=2	6	2.67	1.73	2
3	29	12.9	8.38	8
4	19	8.44	5.49	6
5	26	11.6	7.51	8
6	28	12.4	8.09	8
7	21	9.33	6.07	6
8	15	6.67	4.33	4
9	14	6.22	4.04	4
10	15	6.67	4.33	4
11	8	3.56	2.31	2
12	7	3.11	2.02	2
13	5	2.22	1.44	1
14	5	2.22	1.44	1
15	9	4	2.6	3
Between 15 & 20	12	5.33	3.47	4
>=20	6	2.67	1.73	2
Total	225			65

Table 9.4 : Query-Document Overlaps and Selection of Test Queries

Queries were divided between the training and testing sets based on their degree of overlap in relevant assessments, as this is the most important factor that helps interactive learning in our model. The degree of overlap was measured in terms of the number of documents assessed as relevant to each query. A proportionate number of queries were randomly selected from each set of queries that has the same degree of overlap (see Table 9.4).

The number of queries picked from each category for testing is shown in the above table (Table 9.4), and the actual queries thus picked up for testing are given below. The rest of the 160 queries were used for training.

Test Queries:

[2, 6, 10, 12, 14, 18, 21, 24, 26, 29, 31, 34, 36, 41, 46, 48, 53, 57, 61, 64, 67, 72, 76, 79, 82, 85, 88, 91, 94, 97, 101, 103, 106, 110, 115, 118, 122, 126, 128, 133, 139, 142, 145, 150, 154, 157, 159, 163, 166, 170, 174, 178, 181, 184, 186, 191, 194, 200, 203, 205, 207, 211, 214, 218, 222]

Although, this gives a fairly representative set of queries for testing in terms of cross-relations, it does not guarantee that the queries are equally distributed (between the training and testing sets) in terms of their similarity (i.e. the use of the same vocabulary). The resultant test and training sets were also expected equally to represent the query space in terms of their expressiveness in natural language, though no effort was made to enforce this.

9.4.1.3 Training (Sub) Sets and Testing Process

A training (sub) set for each training phase was created by adding 40 randomly selected queries from the full training set (of 160 queries in Cranfield) into the training set used at the previous training session. No query was picked more than once. So, the number of queries trained at the first training phase was 40, at the second phase 80 (previous 40 + new 40), at the third phase 120 (previous 80 + new 40) and at the final phase 160. Each query was iterated 20 times at each training session (in all the experiments except for those cases where the number of iterations were irrelevant). The order of presentation of the queries to the system was random. At the end of each training session, the system was tested with the 65 test queries, and the similarity measures and the numbers of unit-concept matches and keyword matches were recorded for each query-document pair.

9.4.1.4 Incremental Learning-Testing Experiments

A number of tests were conducted according to this test strategy for the purpose of testing the impact of different aspects (components) and their combinations on the performance of the system. The two comparison or matching entities (i.e. unit-concepts and keywords) and the three components of the learning strategy (i.e. concept addition, unit-concept weight learning and keyword weight learning) were the main components of the system. Consideration of these components left us with a number of possible combinations to be tested. Table 9.5 shows tests and the components used in each test run. The same test queries (65) and the training queries (160) stated above were used in all these tests. The

results of the tests were combined in a number of charts (in Section 9.5) to compare the effectiveness of the various components of the system. Also, the results were statistically analysed for the significance differences between each case/test (refered to as "technique").

	Matching Entity		**Learning Components**			Comment
	Concept Matching	Keyword Matching	Concept Addition	Concept Learning	Keyword Learning	
Test1	×					
Test2	×		×			
Test3	×			×		
	×				×	No Use
Test4	×		×	×		
	×		×		×	Same as Test2
	×			×	×	Same as Test3
	×		×	×	×	Same as Test4
Test5		×				
Test6		×	×			
		×		×		No use
Test7		×			×	
		×	×	×		Same as Test6
Test8		×	×		×	
		×		×	×	Same as Test7
Test9		×	×	×	×	Same as Test8
Test10	×	×				
Test11	×	×	×			
Test12	×	×		×		
Test13	×	×			×	
Test14	×	×	×	×		
Test15	×	×	×		×	
Test16	×	×		×	×	
Test17	×	×	×	×	×	

Table 9.5 : Experiments on Incremental Learning-testing

9.4.2 Probe Testing

9.4.2.1 Methodology

Probe testing is a technique used for testing learning systems especially when not enough data points are available to split between training and testing sets. It consists of a number of training-testing sessions. At each session, a pre-decided (usually small) number of data points are left out from the full data set and the rest are trained to the system. The data points that were left out from training are used as testing inputs to test the system at the end of each training session. No data point is left out from more than one training session (i.e. no data point is tested more than once), and training-testing sessions are repeated until all

the data points are tested. The smaller the number of data points left out (at each training session) the better, as this allows the system to learn from more samples. A special case of probe testing, known as Leave-one-out testing, leaves only one data point out from training at each session. An interesting feature of probe testing is that it allows all data points to be tested, rather than a subset of them. This was an appealing feature for us, in particular, for producing comparative results. The set of published results, with which we compare ours, has been produced based on the (average) performance of all the queries, not on a subset of them.

However, a major disadvantage of probe testing is that it requires the system to undergo a number of training-testing sessions. This is a computationally very expensive task that can take weeks or even months, depending on the size of the data set and the amount of processing required in each training iteration. For example, testing our system on the Cranfield collection took about 45 days on a 1.5 GHz PC, when 9 queries were left out from training (to be used for testing) at each session and when each query was iterated 50 times at each training session. The complexity of this training is $50 \times (225 - 9) \times 25 = 270{,}000$ in terms of the number of query iterations (where 25 is the number of training sessions needed to cover all 225 queries when 9 queries are left out at a time). At each training iteration, each training query is compared with 1400 documents, which requires 1400 document lattices to be set up, candidate query-document concept pairs to match between to be extracted, a similarity measure for each query-document pair to be computed, and retrieved documents to be reinforced. Thus, the complexity is $270{,}000 \times 1400 = 378{,}000{,}000$ in terms of query-document interactions. For this reason, probe testing was not conducted to examine the performance of different aspects of the system, as was the case with interactive learning-testing, but only to produce the performance measurements of the system with all its components in place.

9.4.2.2 Probe Testing Experiments

Three tests were carried out (one on each of CISI, CACM and Cranfield) according to the probe testing strategy using the full capacity of the system (i.e. using both concept and keyword matching and all three learning components). The numbers of queries left out from learning at each training session (to be used for testing) were 9 for Cranfield, 2 for CACM and 1 for CISI. This made the test on Cranfield to have 25 (=225/9) training-testing sessions, CACM to have 26 and CISI to have as many training-testing sessions as the numbers of queries considered (i.e. 35). Each training query was presented (in a given training session) 20 times (i.e. 20 iterations). The selection of queries for leaving out from training at each training session was done randomly, and the order in which training queries were presented to the system was also made random in order to make sure that the presentation order does not affect performance figures.

Note that, the set of published results chosen for comparison with ours contains results obtained on the CISI and CACM collections only. However, since these two collections do not possess the desired features/properties (Section 9.1.1) for testing learning systems of our kind, our system was tested on the Cranfield collection as well (Table 9.23, Table 9.24 and Table 9.25). Table 9.6 lists the individual tests conducted according to the probe testing strategy.

Test	Features / Components Used	Test Collection	Queries Left Out from Training	Training Iterations
Test18	All	CISI	1	20
Test19	All	CACM	2	20
Test20	All	Cranfield	9	20

Table 9.6 : Experiments on Probe Testing

9.4.3 Full-training and Full-testing

9.4.3.1 Methodology

The aim of this test strategy was to measure the performance of the system on queries that it has already seen (i.e. learnt) before. This is achieved by simply training the system with all queries, and then testing the system on the same (trained) set of queries. One might

expect a near perfect, if not 100%, retrieval accuracy when the same queries that were shown to the system during training are tried, because the relevant documents have been reinforced with all the corresponding query concepts (i.e. all query concepts are added to the relevant documents, and the weights of the unit-concepts common to the queries and relevant documents are rewarded during training). This, however, is not always the case due to three major reasons: (1) no system could recall all relevant documents, (2) conflicts in relevance assessments, and (3) the presence of common concepts. These problems make 100% retrieval accuracy an impossible task. Therefore, it will be interesting to see how far our approach can cope with those problems, and improve performance on seen queries.

9.4.3.2 Full-training Full-testing Experiments

Three tests were conducted according to the full-training full-testing strategy, one on each of the three collections: Cranfield, CISI and CACM. At each test, all the queries of the respective collection were shown to the system for 100 iterations (in total), and the same set of queries (all queries in the collection) were tested after a pre-decided number of training iterations (5, 10, 20, 30, 50, 75 and 100). The following table (Table 9.7) lists the details of the three individual tests.

	Features / Components Used	**Test Collection**	**Training Iterations**
Test21	All	CISI	100
Test22	All	CACM	100
Test23	All	Cranfield	100

Table 9.7 : Experiments on Seen Queries

9.5 RESULTS

9.5.1 Effect of Learning on Performance

In the following, the precision averages of Test17 were plotted against a number of training queries (Figure 9.3) to observe the performance of the system in its full capacity. Figure 9.4 plots average precisions at retrieval points 1, 5, 10 and 20 to see the consistency of the performance at different retrieval points. P-R curves were produced on the results of

testing at the four levels of training, (Figure 9.5) to see the improvements of precisions during training.

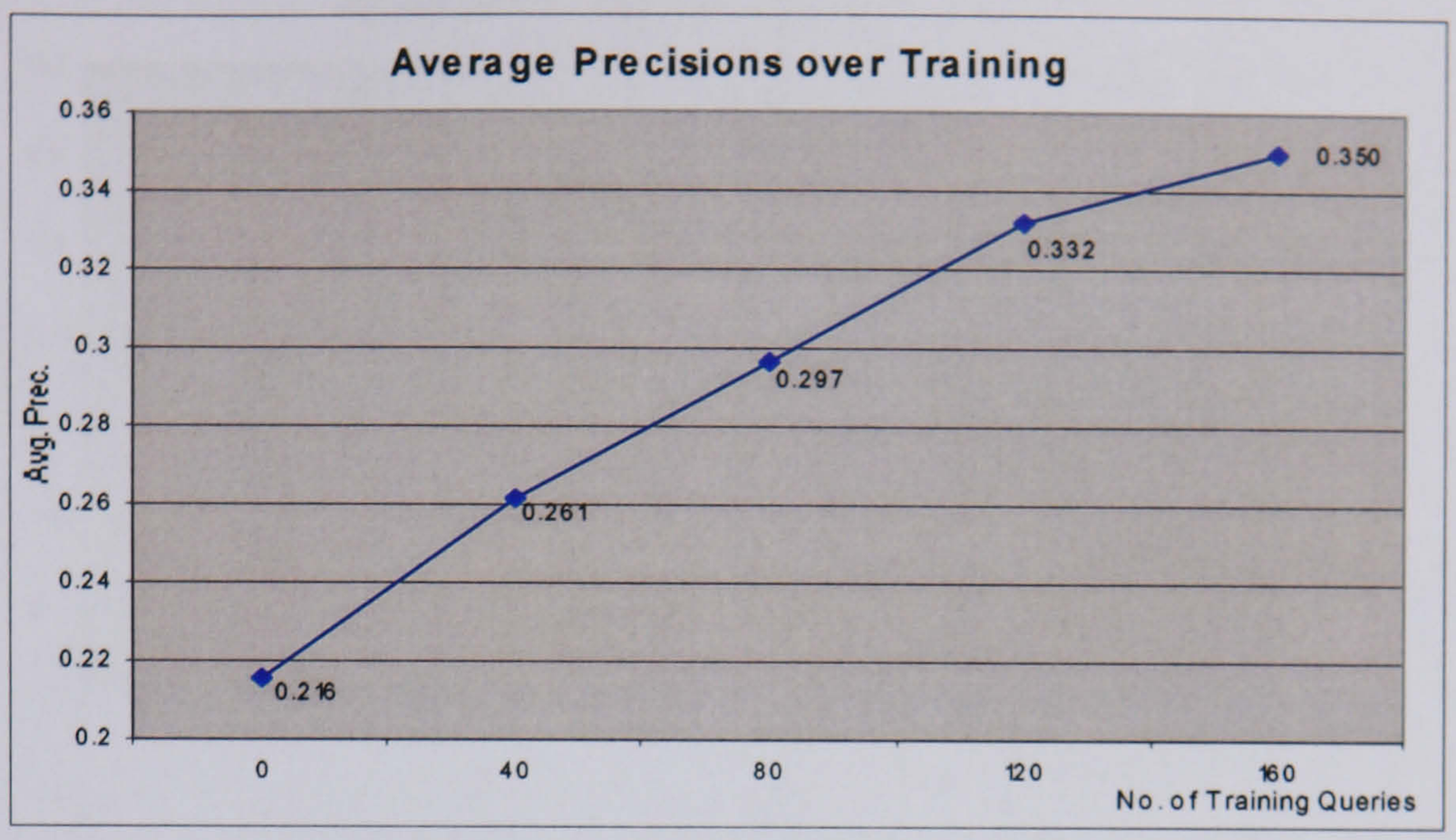


Figure 9.3 : Average Precisions over Learning

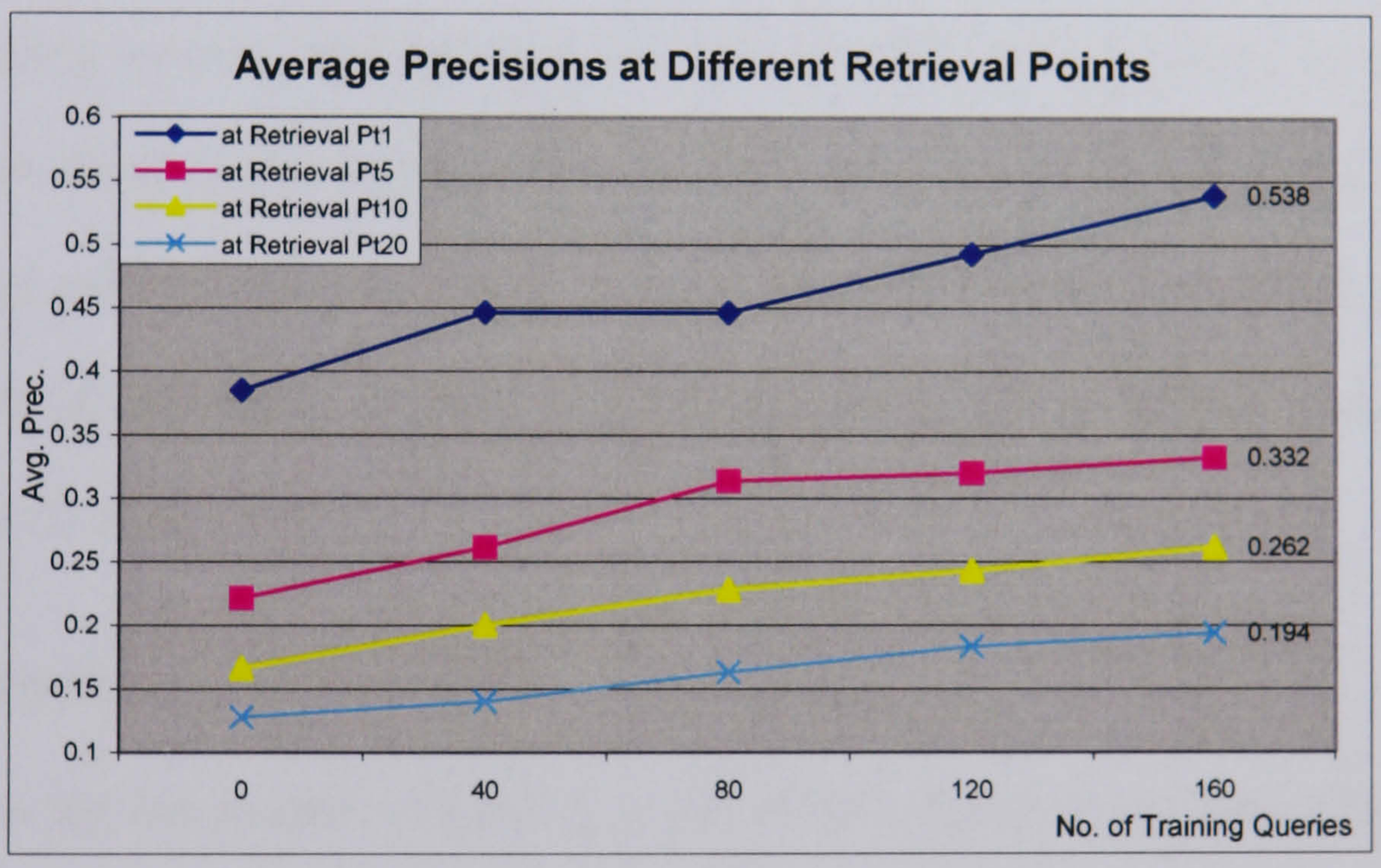


Figure 9.4 : Average Precisions over Learning at Different Retrieval Points

Observations

Figure 9.3 shows that the performance of the system (i.e. average precisions (non-interpolated)) increases considerably during training. Note that, the performance gains at different parts of the curve vary depending on the amount of learning taking place at each training session and how much that learning helps in retrieving relevant documents for test queries (Figure 9.16 in Section 9.5.5 shows that the curves of the same test at different test runs are different). Figure 9.4 shows that not only the overall performance given by average precisions, but also performances at retrieval points 1, 5, 10 and 20, are increased

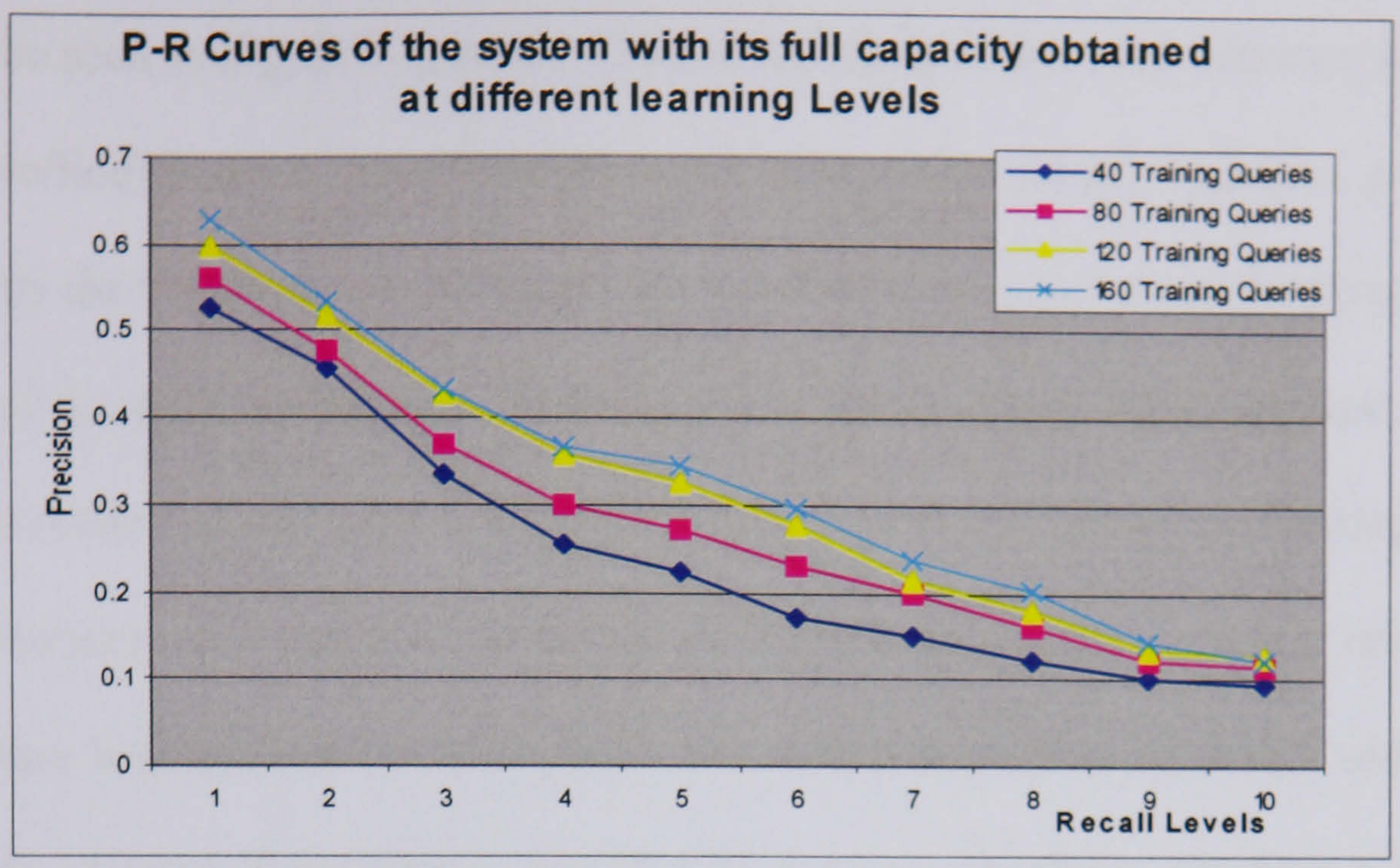


Figure 9.5 : P-R Curves at Different Levels of Learning

consistently. Also, Figure 9.5, which plots the P-R curves of test results obtained at the end of each training session, highlights the same performance increases, and confirms that the performance of the system increases over training. This, however, is a combined result of both concept and keyword matching with all three learning components of the system. The contribution of individual components towards this result is further analysed in the following section.

9.5.2 Contribution of Learning Components

Given below are the results of tests 2, 3, 4, 10 and 17 plotted in one chart (Figure 9.6) to see the performance gains given by different learning components. A random baseline, computed by arranging the documents in a random order for each test query, is also plotted in the same graph. The aim of these plots was to show that the performance of the system improves as each training component is added. Eexcept for Test17, all tests (i.e. tests 2, 3, 4 and 10) used concept matching only (no keyword matching). In addition to overall average precisions, the average precisions at retrieval points 1, 5, 10 and 20 of the same tests were also plotted in the 4 charts given in Figure 9.7 to show the performances at those retrieval points.

Observations

As can be seen in Figure 9.6, concept matching alone without any learning component (the flat curve/line) is not of much use. The problems with concept extraction, and mismatches caused by the vocabulary differences and word ambiguity in natural language are the main causes of the poor performance of concept matching (only). These problems are severe in our case (concept matching) compared to simple keyword matching. Concept extraction is more complex and difficult, as it needs identification of a semantic relation or some connection between two terms or phrases to interpret them as an object and attribute pair. The term mismatch problem is doubled in concept matching, because a concept match needs both the object and attribute constituents of a query concept (unit-concept) to match with an equivalent in a document.

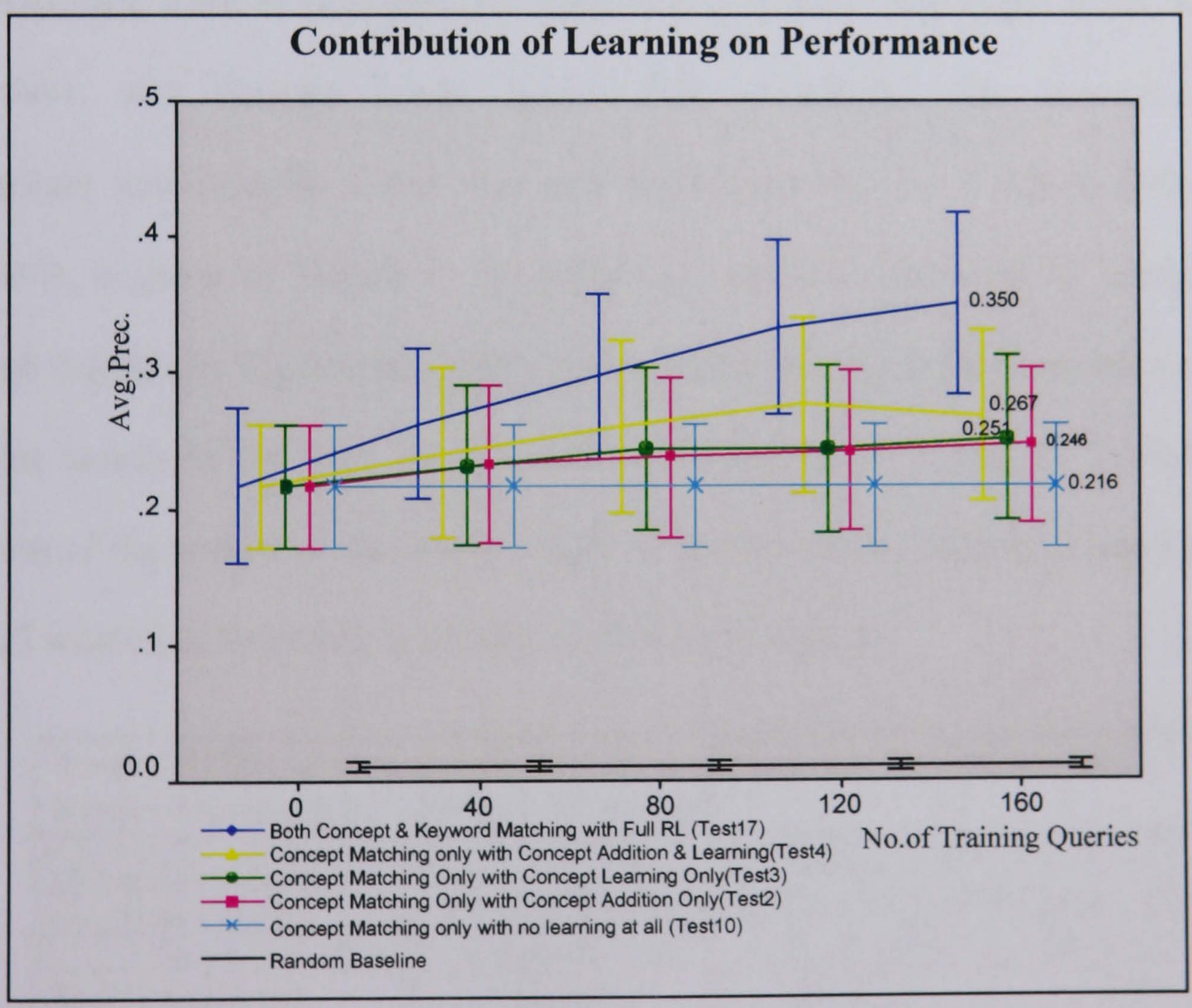


Figure 9.6 : Contribution of Learning on Performance

A 5 x 5 within subjects analysis of variance (ANOVA) was performed on the performance figures of 5 tests each with 5 training levels. Note that the baseline was not included in the computation of ANOVA as it is clearly much lower from the rest. Instead of the Random Baseline, performance of the system with no prior learning could be considered as the

baseline, as our objective here is to compare the performances between different learning components.

Effect	Repeated Measures Analysis of Variance Sigma-Restricted parameterization Efective hypothesis decomposition				
	SS	Degree of Freedom	MS	F	P
Intercept	98.15935	1	98.15935	102.0559	<0.00001
Error	61.55643	64	0.96182		
Technique(Test)	1.04141	4	0.26035	6.6482	<0.00001
Error	10.02522	256	0.03916		
Training Level	0.53715	4	0.13429	9.3826	<0.00001
Error	3.66396	256	0.01431		
Technique*Training Level	0.44668	16	0.02792	3.8166	<0.00001
Error	7.49032	1024	0.00731		

Table 9.8 : ANOVA on the Results of Tests 2,3,4,10 and 17

As expected, there were significant main effects of both: Technique (test), $F_{4,256}$=6.648, p<0.0001; and Training Level, $F_{4,256}$=9.3826, p<0.0001). The interaction between Technique and Training Level was also significant, $F_{16,1024}$ = 3.816, p<0.0001). The ANOVA is given in Table 9.8. The follow-up analysis conducted by using the Tukey Honest Significant Difference (HSD) test revealed that the differences between tests are present mainly at the later training levels (training levels 4 and 5). Table 9.9 gives a fraction of the results of the Tukey HSD test for comparing individual pairs of points of Test17 with other four tests, at training level three and above.

Tukey (HSD) test. Approximate Probabilities for Post Hoc Tests Error: Within MS = 0.00731, df =1024.0			
Comparison Pair of tests/techniques	**Training Levels** Training Level3 (80 Queries)	Training Level4 (120 Queries)	Training Level5 (160 Queries)
Test17 and Test4	0.663145	0.031214	0.000028
Test17 and Test3	0.063944	0.000019	0.000019
Test17 and Test2	0.015393	0.000019	0.000019
Test17 and Test10	0.000037	0.000019	0.000019

Table 9.9 : Probabilities of Follow-up Tukey HSD Analysis on the Results of Tests 2,3,4,10 and 17

According to the probability figures of Table 9.9, the performance of "both matching" (i.e.Test17) is significantly different to the other four tests at training levels 4 and 5. Except with Test4 and Test3, it is significantly different to other tests (i.e. tests 2 and 10) at training level3 too.

The results of the system, using the concept weight learning and concept addition components each alone, have shown only marginal improvements (Figure 9.6). Concept weight learning does not (and it is not expected to) solve the two main causes of the poor performance stated above. Although concept addition helps documents to learn (through user interactions) the different ways that users might refer to them (i.e. learn from experience), and thereby helps both the word mismatch problem and poor concept extraction, it has not shown a significant improvement. This is mainly because these results were produced by testing unseen queries. The lack of sufficient overlaps in the collection (i.e. the use of the same unit-concepts to represent similar documents) does not help the retrieval of a document by a query as a result of the document being reinforced (updated) by another query (this is the main objective of our learning strategy). However, it is interesting to observe that they both show, though small, increasing trends in performance. As a result of improvements of each component, their combination shows even greater improvement.

Finally, the curve of Test17 in Figure 9.6 shows that taking keyword matching into account helps to improve performance. The reasons for this are: (1) keywords help the initial picking up of documents for reinforcing; and (2) keyword matches, that take place in the absence of unit-concept matches, help increase the similarity scores (RSVs) of documents, thus helping the system to rank the documents, with more features common to the query, above the documents with fewer features. The second point is valid only if more keyword matches occur with relevant documents than with non-relevant documents - a well-known observation in IR [Krovetz & Croft 1992, Savoy 1997]. Although no experiments were

targeted at examining the validity of this feature, the improvements shown by the system with keyword matching demonstrates its validity.

Except in the case of precisions at retrieval point one, we see better results for concept matching, when both concept addition and concept learning were taken into account

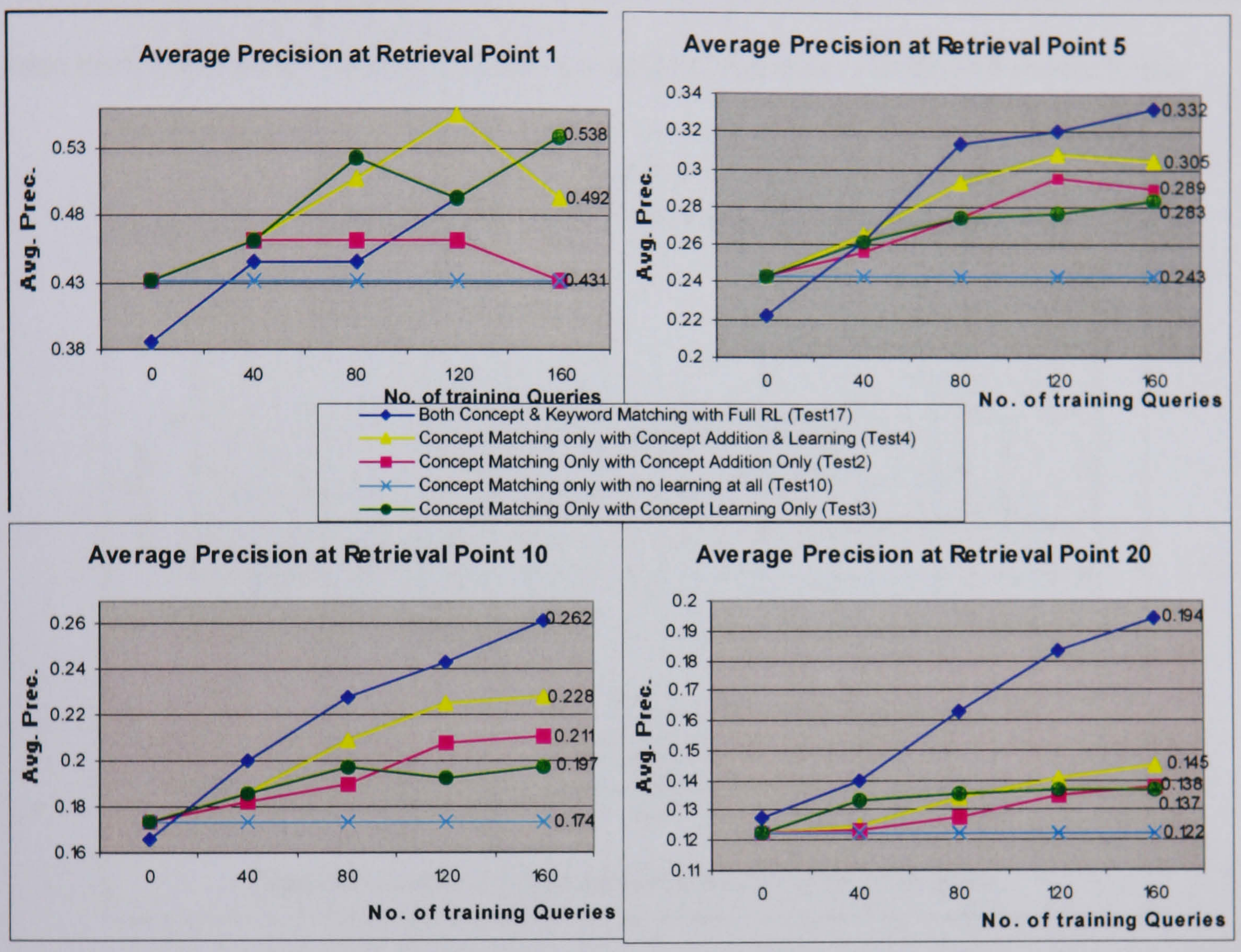


Figure 9.7 : Average Precisions at Different Retrieval Points

(charts in Figure 9.7). This is in addition to the case of the system's performance in its full capacity (i.e. using all the components). Although higher precisions were shown by the cases "concept learning only" and "concept addition and concept learning" at the early stages of learning (as shown in the top two charts of Figure 9.7), the results of the system with its full capacity outperforms them as the system gains more experience. These charts evidence that the better performance shown by concept matching with both learning components (seen in Figure 9.6) is not just an arbitrary result, but is consistent with the results of the system at different retrieval points. They also show that concept matches are superior in finding a few relevant documents (better precision).

9.5.3 Effect of Learning Components on the two Matching Entities

The aim of the following two charts was to see how the different learning components and their combinations affect each matching entity. The results of Test1, Test2, Test3 and Test4 were used in the first chart (Figure 9.8), and the results of Test5, Test6, Test7 and Test8 were used in the second chart (Figure 9.9). Two 4 x 5 ANOVAs (one for each chart) were performed on the data (test results) to analyse the main effects and interactions.

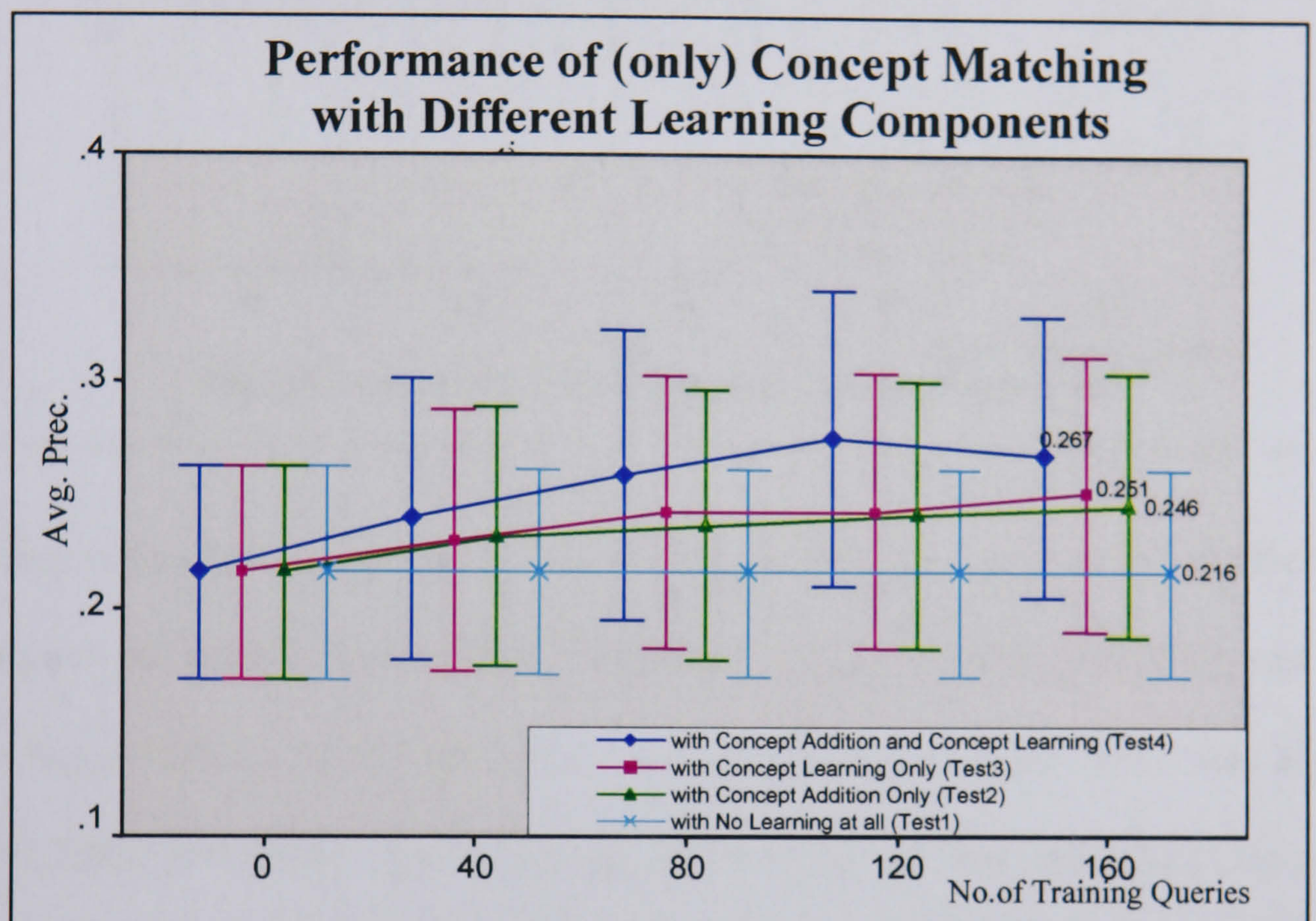


Figure 9.8 : Contribution of Learning Components on Concept Matching Only

According to the ANOVA performed on the results shown in the charts in Figure 9.8, no significant main effect is detected in either variable, but an interaction exists ($F_{12,768}$=1.8924, p=0.032) between them. The follow up Tukey HSD tests show significant differences only between Test4 and Test1 at training levels 3, 4 and 5. A fraction of the resultant probabilities of the follow-up Tukey HSD test is given in Table 9.10, for the purpose of comparing significances between tests at training levels 3 and above.

Comparison Pair	Trn. Lev3 (80 Queries)	Trn. Lev4 (120 Queries)	Trn. Lev5 (160 Queries)
Test4 and Test3	0.994434	0.177731	0.990571
Test4 and Test2	0.855681	0.113576	0.862363
Test4 and Test1	**0.005234**	**0.000044**	**0.000141**
Test3 and Test2	1.000000	1.000000	1.000000
Test3 and Test1	0.475850	0.477375	0.076535
Test2 and Test1	0.881770	0.608546	0.287287

Table 9.10 : Probabilities of Follow-up Tukey HSD Analysis on the Results of Tests 1,2,3 and 4

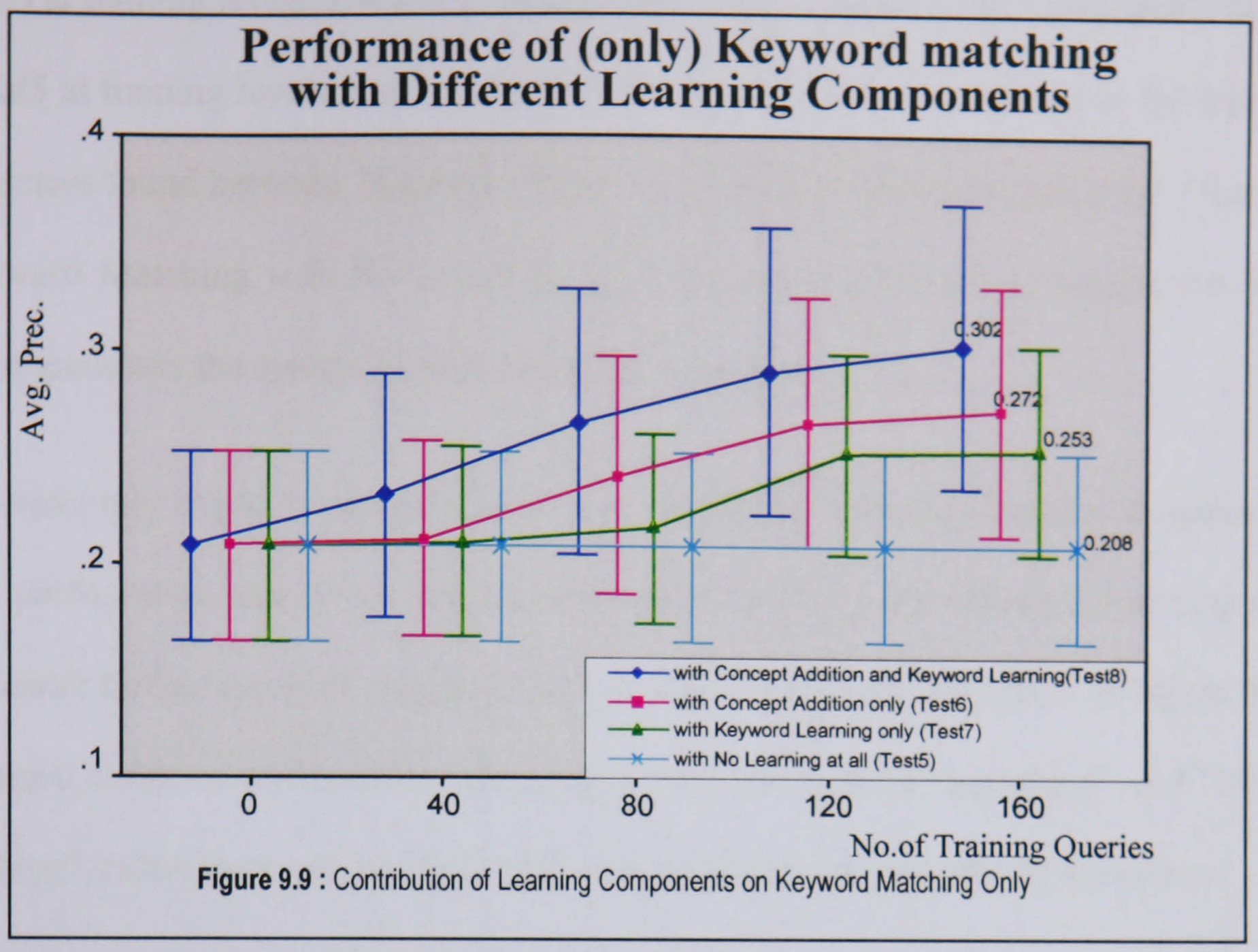


Figure 9.9 : Contribution of Learning Components on Keyword Matching Only

According to the ANOVA performed on the results shown in the charts in Figure 9.9, there were significant main effects of both: Technique, $F_{3,192} = 4.4208$, $p=0.0049$; and Training Level, $F_{4,256}=9.9835$, $p<0.00001$. Also, the interaction between them was significant, $F_{12,768}=3.2692$, $p=0.00012$. The follow up Tukey HSD test was performed (for point wise comparison), and the probabilities necessary to compare each test (technique) with each other at training levels 3 onwards are given in Table 9.11.

Comparison Pair	Trn. Lev3 (80 Qreries)	Trn. Lev4 (120 Queries)	Trn. Lev5 (160 Queries)
Test8 and Test6	0.953242	0.970069	0.807846
Test8 and Test7	**0.034604**	0.344702	**0.044802**
Test8 and Test5	**0.002894**	**0.000043**	**0.000042**
Test6 and Test7	0.960830	0.999965	0.998158
Test6 and Test5	0.622352	**0.003277**	**0.000502**
Test7 and Test5	1.000000	0.135298	0.105133

Table 9.11 : Probabilities of Follow-up Tukey HSD Analysis on the Results of Tests 5,6,7 and 8

According to the probabilities given in Table 9.11 for the respective pairs of points, "Keyword Matching with Concept Addition and Concept Learning" (Test8) is not significantly different from "Keyword matching with Concept Addition Only" (Test6) at any of the training levels. It differs from "Keyword Matching with Keyword Learning Only" (Test7) at training levels 3 and 5; and from "Keyword Matching with No Learning"

(Test5) at training levels 3, 4 and 5. In addition, Test6 is found to be significantly different to Test5 at training levels 4 and 5. An interesting observation, according to the significant differences found between "Keyword Matching with Keyword Learning only" (Test7) and "Keyword Matching with No Learning" (Test5), is that as keyword weights are learned from experience, the system performs significantly better.

The above two charts, in general, show that combining learning components gives better mean performance results for both concept matching (only) and keyword matching (only). This result further confirms that the better performance shown by Test17 in Figure 9.3 is a combined result of all learning components on both "concept matching" and "keyword matching", rather than a result of one or a few particular components of the system.

The performance gain shown by the system with only the keyword learning component (i.e. Test7) is also interesting. The keyword matches at each test session were the same in this case, as it is the same set of queries that were tested on the same collection, and no concepts (and hence no keywords) were added to the documents by learning. Despite the fact that no additional keyword matches have taken place during training, the results have improved. This is solely due to keyword weight learning. This mean, learning seems to have assigned higher weights to keywords that are significant, at least in terms of their discriminating power.

However, not all of the keyword matches that take place during testing for the case "keyword matching only", are considered as keyword matches when "both keyword and concept matches" are taken into account. The keywords that participate in concept matches are not treated as keyword matches in the later case, and are pruned out. Therefore, the result of the case with "keyword matching and concept matching" is not the same as the sum of the individual cases of "concept matching only" and "keyword matching only".

9.5.4 Impact of Matching Entities on each Learning Component

The aim of the charts given under this section is to see the impact of individual learning components on each matching entity. The results of the same tests, discussed in the above section (Section 9.5.3), were used here, but they were organised differently in order to compare the performance of "concept matching", "keyword matching" and both of them together, on a given individual learning component (or on a given combination of them) at a time. As before, ANOVAs were performed on the corresponding results of the tests to analyse statistically the differences between each case (technique).

9.5.4.1 Impact of Concept Addition

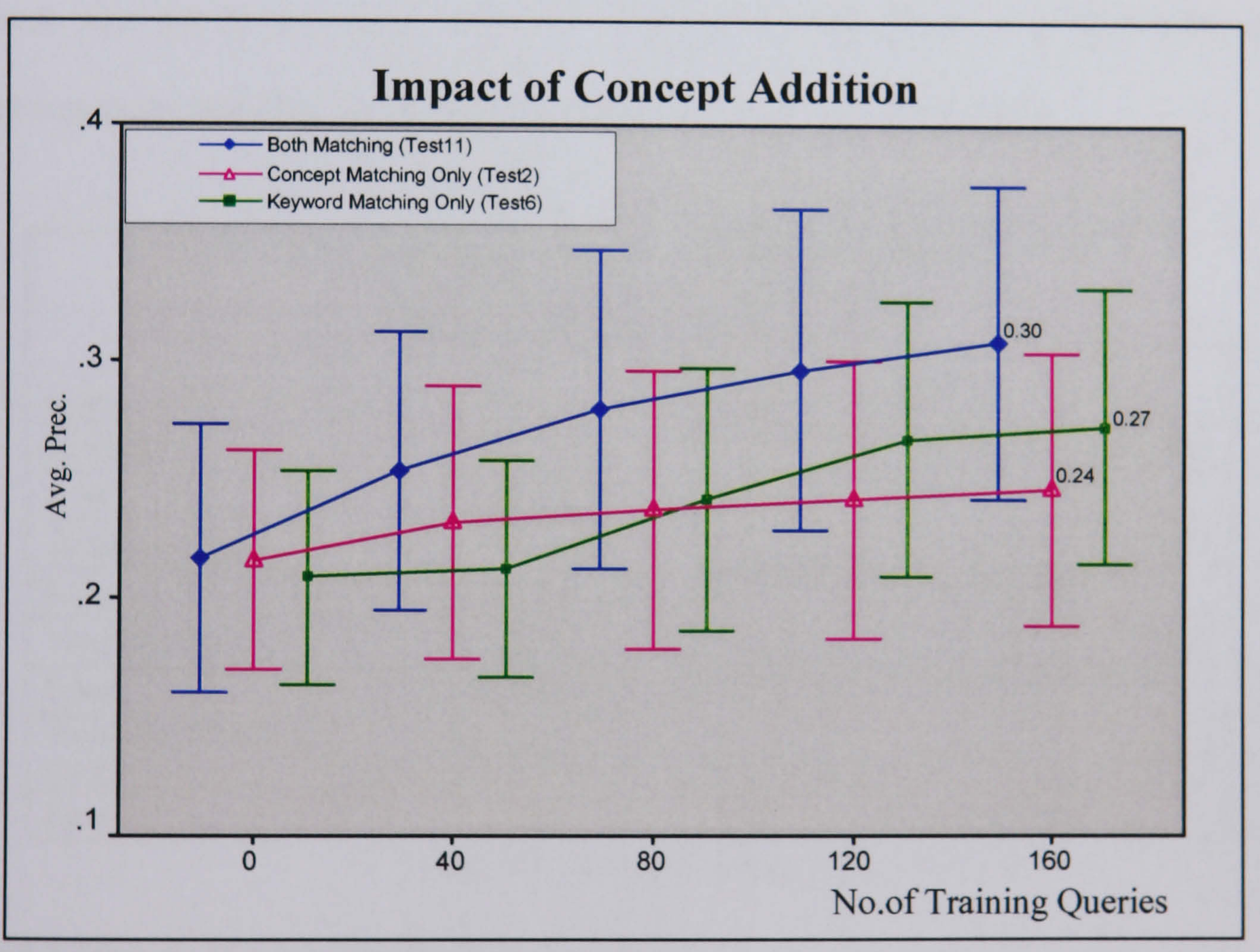


Figure 9.10 : Impact of Concept Addition

According to Figure 9.10, keyword matching seems to outperform concept matching in the case when only the concept addition component is taken into consideration. The main reason for this could be insufficient learning caused by the inadequate number of queries and inadequate expressiveness in queries for learning. Since these results were based on testing unseen queries, the chance of more concept matches taking place is less if queries similar to testing (unseen) queries have not been learnt by the system. Although there are

cross relations (overlaps) in the test collection, those queries that are assessed as relevant to a given document may have not been formulated in a similar way to each other (i.e. they may not share the same concepts). Therefore, concept addition leads more keyword matches to take place, than concept matches (with relevant documents). On the other hand, both "concept matching only" and "keyword matching only" show significant performance increases over training and hence best performance is shown when both of them are used in combination (curve on Test11).

The results of significance testing performed with a 3 x 5 within subjects analysis of variance (ANOVA), given below in Table 9.12, indicates no main effect of the Technique (tests). But there is a main effect of Training Levels. Since, no significant interation between these two were indicated, no follow up tests were performed.

	Repeated Measures Analysis of Variance Sigma-Restricted parameterization Efective hypothesis decomposition				
Effect	**SS**	**Degree of Freedom**	**MS**	**F**	**P**
Intercept	60.09127	1	60.09127	105.5956	<0.00001
Error	36.42048	64	0.56907		
Technique(Test)	0.24656	2	0.12328	2.2677	0.107697
Error	6.95845	128	0.05436		
Training Level	0.50607	4	0.12652	9.6161	<0.00001
Error	3.36813	256	0.01316		
Technique*Training Level	0.10678	8	0.01335	1.3204	0.230591
Error	5.17571	512	0.01011		

Table 9.12 : ANOVA on the Results of Tests 2,6 and 11

9.5.4.2 Impact of Concept Weight Learning

Concept weight learning affects concept matching only (not keyword matching). Therefore, the contribution of concept matching is the same in the two cases plotted (in Figure 9.11). Nevertheless, "Both Matching" shows a slightly better result, on average, due to the additional keyword matches that it takes into account. The statistical analysis performed with a 2 x 5 within subjects ANOVA (Table 9.13) shows a main effect of Training Level. No interaction between the two variables was found.

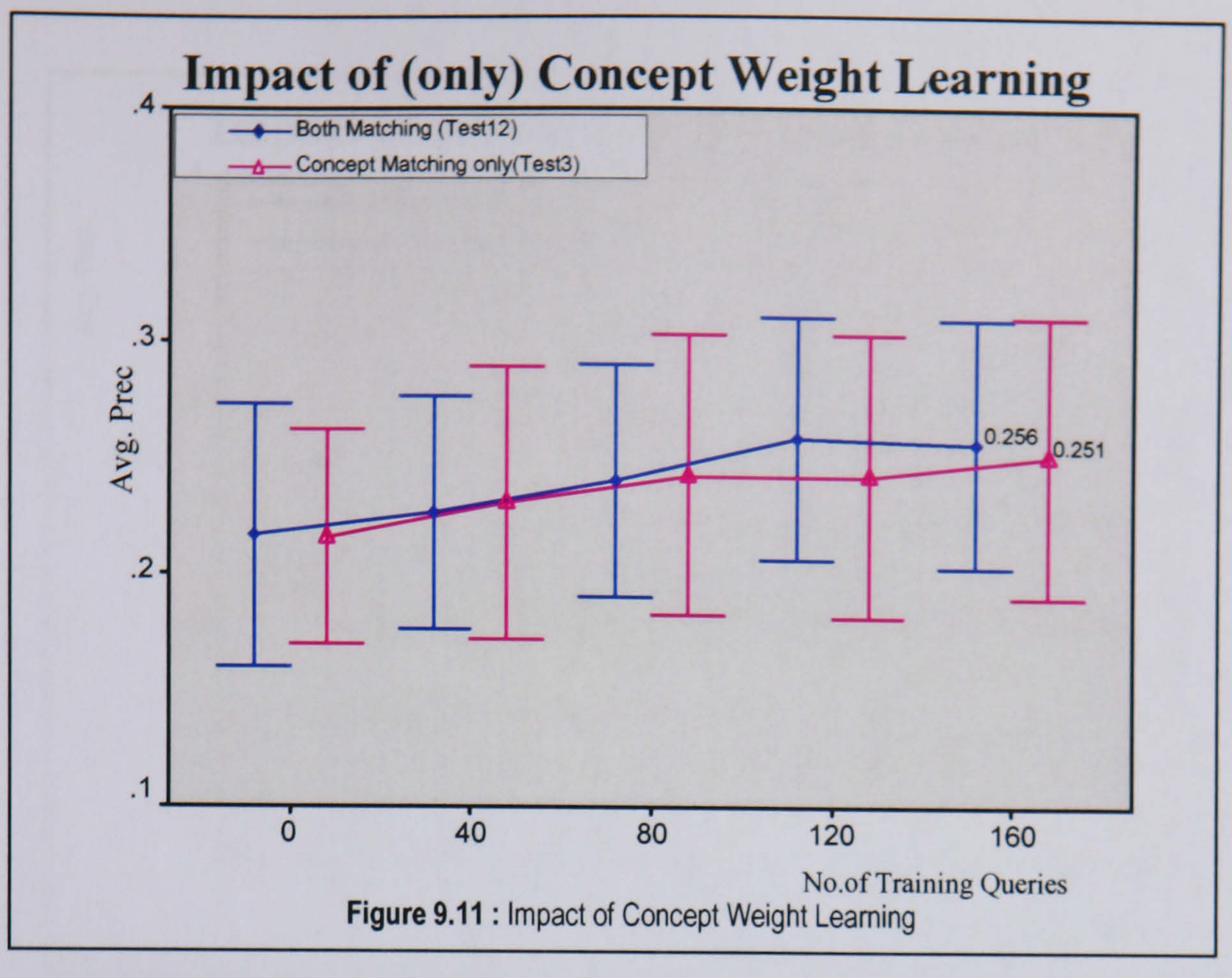


Figure 9.11 : Impact of Concept Weight Learning

An interesting observation, however, is that the concept weight learning has helped to improve the system's performance. As the concept matches between the test queries and the documents are the same in this case, as a result of no new concepts being added to the document representations, the performance increment shown is solely a result of concept weight learning.

9.5.4.3 Impact of Keyword Weight Learning

Effect	Repeated Measures Analysis of Variance Sigma-Restricted parameterization Efective hypothesis decomposition				
	SS	Degree of Freedom	MS	F	P
Intercept	36.84508	1	36.84508	89.48792	<0.00001
Error	26.35088	64	0.41173		
Technique(Test)	0.00148	1	0.00148	0.06325	0.802234
Error	1.49983	64	0.02343		
Training Level	0.12953	4	0.03238	8.72784	<0.00001
Error	0.94980	256	0.00371		
Technique*Training Level	0.00881	4	0.00220	0.020603	0.934929
Error	2.73569	256	0.01069		

Table 9.13 : ANOVA on the Results of Tests 3 and 12

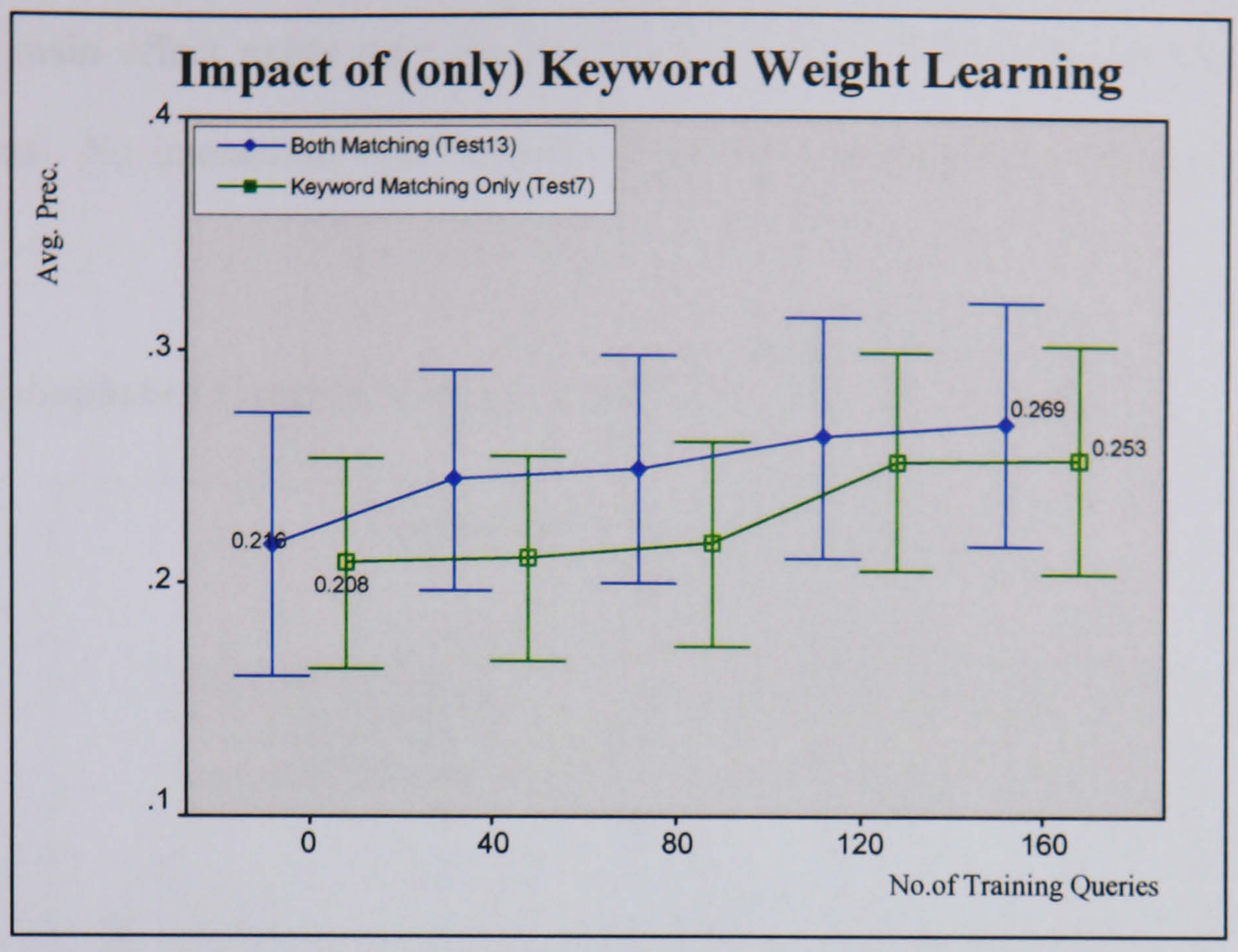


Figure 9.12 : Impact of Keyword Learning

The contribution of keywords is the same in both cases plotted in Figure 9.12 as no new keyword matches take place on testing over training. However, as can be seen in Figure 9.12, the system seems to perform better in the case with "both matching" due to additional (original) concept matches that take place in this case (compared to the case with keyword matching". Also, both curves show increasing trends over training queries (experience), meaning that keyword weight learning helps improving performance.

	Repeated Measures Analysis of Variance Sigma-Restricted parameterization Efective hypothesis decomposition				
Effect	**SS**	**Degree of Freedom**	**MS**	**F**	**P**
Intercept	36.88214	1	36.88214	119.7826	<0.00001
Error	19.70617	64	0.30791		
Technique(Test)	0.06535	1	0.06535	2.1546	0.147043
Error	1.94128	64	0.03033		
Training Level	0.22164	4	0.05541	8.4101	<0.00001
Error	1.68663	256	0.00659		
Technique*Training Level	0.01940	4	0.00485	0.9410	0.440736
Error	1.31964	256	0.00515		

Table 9.14 : ANOVA on the Results of Tests 7 and 13

The statistical analysis performed with a 2 x 5 within subjects ANOVA (Table 9.14) shows that a main effect exists only on Training Level (i.e. two curves are not significantly different). No interaction was found between the two variables, Technique and Training level.

9.5.4.4 Impact of Concept Addition and Concept Weight Learning

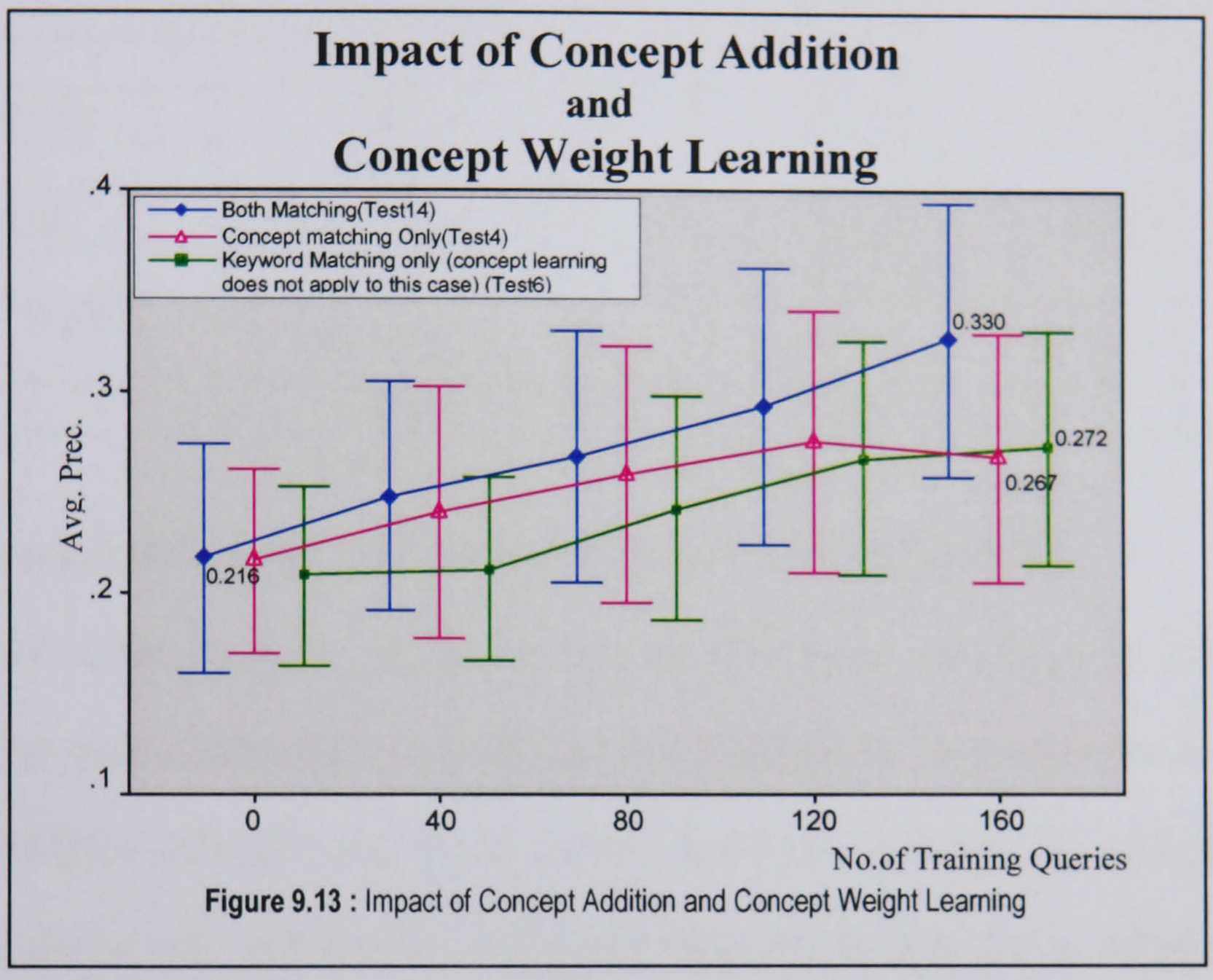


Figure 9.13 : Impact of Concept Addition and Concept Weight Learning

A better improvement is shown for the case of "both matching", in particular at training level5 when both concept addition and learning were considered. This is a result of improvements made by concept addition on both “keyword matching and concept matching” (Figure 9.10) and “concept weight learning” on concept matching (Figure 9.11). It seems that the curve of “both matching” tends to depart from the curve on “keyword matching only”, beyond the training level3. Also, the performance increases shown by “keyword matching only” (Test6) as the system learns is interesting despite the fact that concept weight learning does not help keyword matching at all. This is a result of more keyword matching taking place as more concepts are added (as noticed before). In addition, “concept matching with concept addition and concept learning” (Test4) has performed as well as keyword matching (Test6).

The statistical analysis performed with a 3 x 5 within subjects ANOVA (Table 9.15) show a main effect only on the Training Level (but not on the Technique). No interaction was found between the two variables.

Effect	Repeated Measures Analysis of Variance Sigma-Restricted parameterization Efective hypothesis decomposition				
	SS	Degree of Freedom	MS	F	P
Intercept	62.77500	1	62.77500	111.3542	<0.00001
Error	36.07947	64	0.56374		
Technique(Test)	0.14877	2	0.07438	1.3842	0.254237
Error	6.87827	128	0.05374		
Training Level	0.74328	4	0.18582	10.4127	<0.00001
Error	4.56840	256	0.01785		
Technique*Training Level	0.08182	8	0.011023	0.8233	0.582245
Error	6.36040	512	0.01242		

Table 9.15 : ANOVA on the Results of Tests 4, 6 and 14

9.5.4.5 Impact of Concept Addition and Keyword Weight Learning

Figure 9.14 shows improved performance of all three cases over training. Since concept addition has only a little effect on (only) concept matching (as noticed before in the case of concept addition only in Figure 9.10), due to insufficient learning, the curve for concept matching shows only a little effect. Note that, keyword learning has no effect on concept

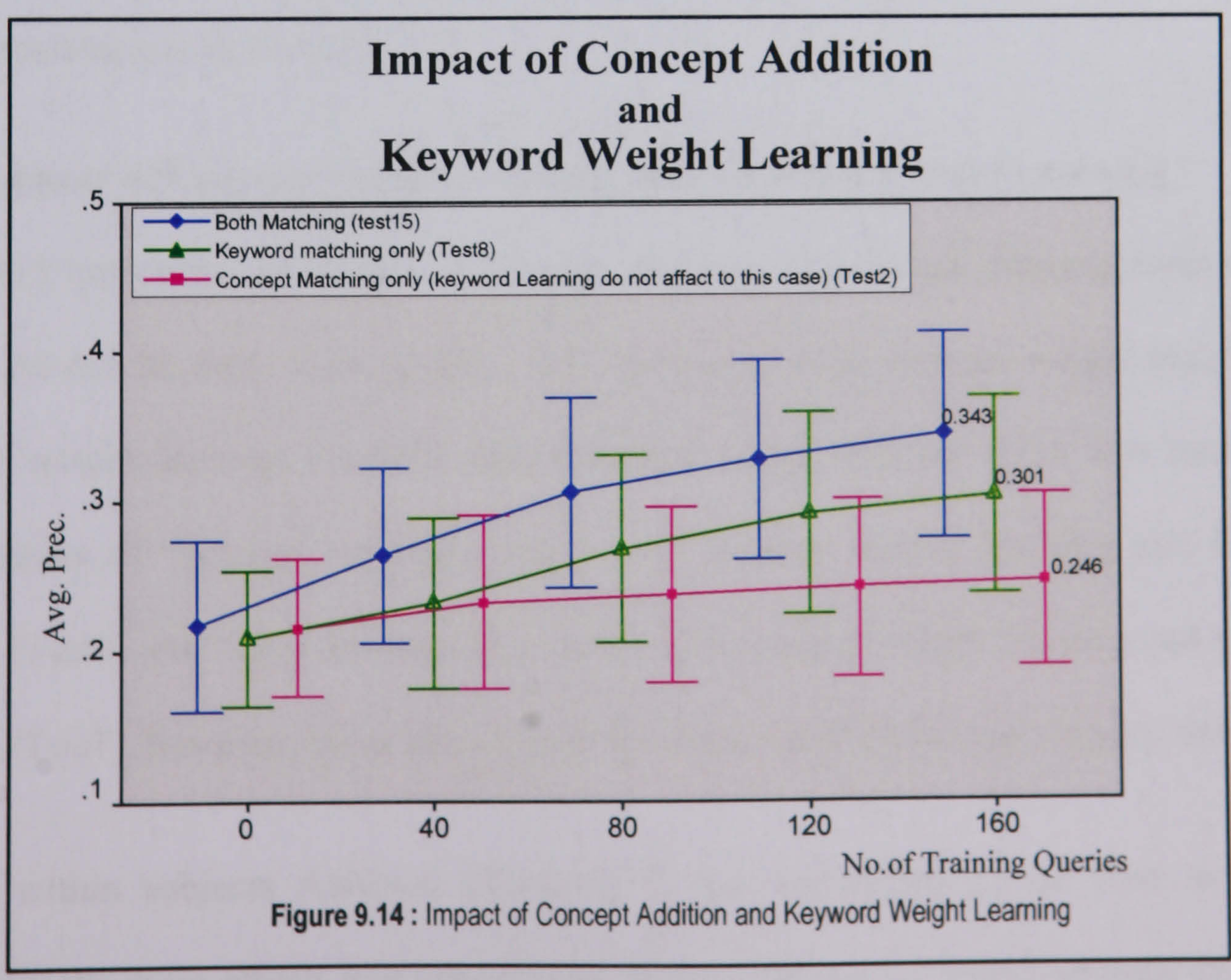


Figure 9.14 : Impact of Concept Addition and Keyword Weight Learning

matching. On the other hand, keyword matching has shown a better improvement over training due to concept addition and keyword weight learning.

A 3 x 5 within subjects ANOVA was performed to analyse the data statistically. There were significant main effects of both: Technique (test), $F_{2,128}=5.0427$, $p=0.0077$; and Training Level, $F_{4,256}=12.8268$, $p<0.00001$. The interaction between Technique and Training Level was also significant in this case, $F_{8,512}=2.6347$, $p=0.0077$. Table 9.16 shows a selected set of probabilities extracted from the results of the follow-up Tukey HSD tests, in order to compare the significances between tests at training levels 3 and above.

Comparison Pair	Trn. Lev3 (80 Queries)	Trn. Lev4 (120 Queries)	Trn. Lev5 (160 Queries)
Test15 and Test8	0.711139	0.805446	0.635490
Test15 and Test2	**0.012639**	**0.000261**	**0.000034**
Test8 and Test2	0.946143	0.291288	0.112741

Table 9.16 : Probabilities of Follow-up Tukey HSD Analysis on the Results of Tests 2, 8 and 15

According to the probabilities given in Table 9.16, "both matching with concept addition and keyword weight learning" (Test15) is significantly different from "concept matching only with concept addition and keyword weight learning" (Test2) at training levels 3, 4 and 5. This indicates significantly different performance by both matching (Test15) over concept matching only (Test2).

9.5.4.6 Impact of Concept Weight Learning and Keyword Weight Learning

Figure 9.15 compares the impact of concept and keyword weight learning over training queries. As can be seen in the graph, "both matching" with concept weight learning and keyword weight learning (Test16) has performed better than the other two throughout. Performances of "concept matching only" with concept weight learning and keyword learning (Test3) and "keyword matching only" with concept weight learning and keyword learning (Test7), however, show mixed results with no clear difference between them.

A 3 x 5 within subjects ANOVA (Table 9.17) was performed on the data to analyse statistically the main effects and interactions. Main effects were found on both Technique

and Training Level, meaning that the curves (techniques) are significantly different from each other. However, there was no significant interaction between the two variables.

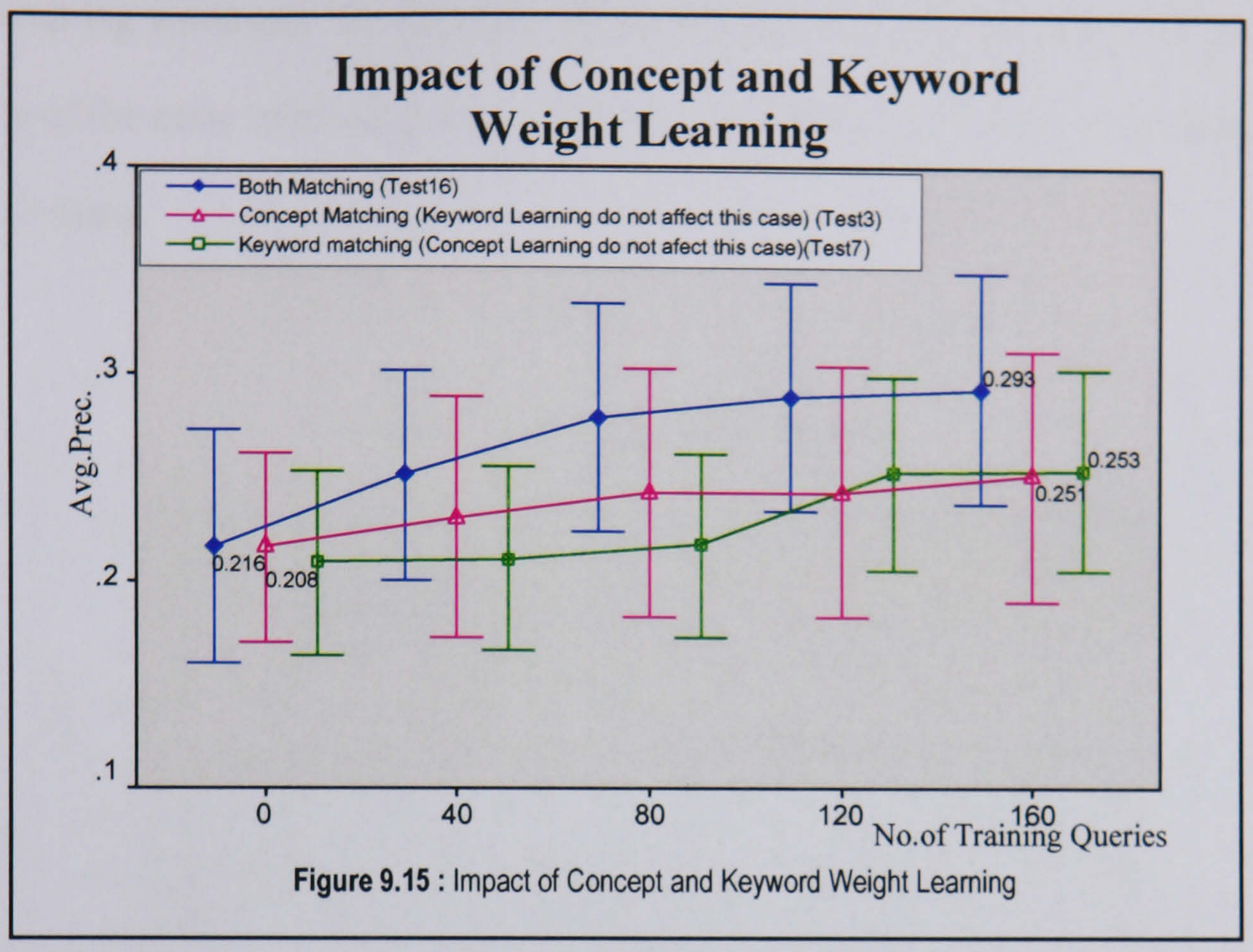


Figure 9.15 : Impact of Concept and Keyword Weight Learning

Effect	Repeated Measures Analysis of Variance Sigma-Restricted parameterization Efective hypothesis decomposition				
	SS	Degree of Freedom	MS	F	P
Intercept	57.76009	1	57.76009	111.8288	<0.00001
Error	33.05629	64	0.51650		
Technique(Test)	0.24819	2	0.12410	3.1494	0.046221
Error	5.04364	128	0.03940		
Training Level	0.36537	4	0.09134	11.3675	<0.00001
Error	2.05705	256	0.00804		
Technique*Training Level	0.08042	8	0.01005	1.4938	0.156544
Error	3.44561	512	0.00673		

Table 9.17 : ANOVA on the Results of Tests 3, 7 and 16

9.5.5 Effect of Presentation Order on Performance

This experiment was aimed at examining whether the presentation order affects the performance of the system. Test17 was repeated three times and the results of the three runs were plotted on the same chart (Figure 9.16). Although the set of training queries was the same at each run, the individual training sets at corresponding training sessions (note that each test run involves four training sessions with 40, 80, 120 and 160 training queries at the four respective training sessions) were different in the three runs (except the training

sets of the final training sessions with the same 160 queries), as each time we picked up queries randomly from the full training set when creating the training sets for individual (sub) training sessions. In addition, although the last training session of each test run consists of the same training queries, they are presented to the system in a random order.

Observations

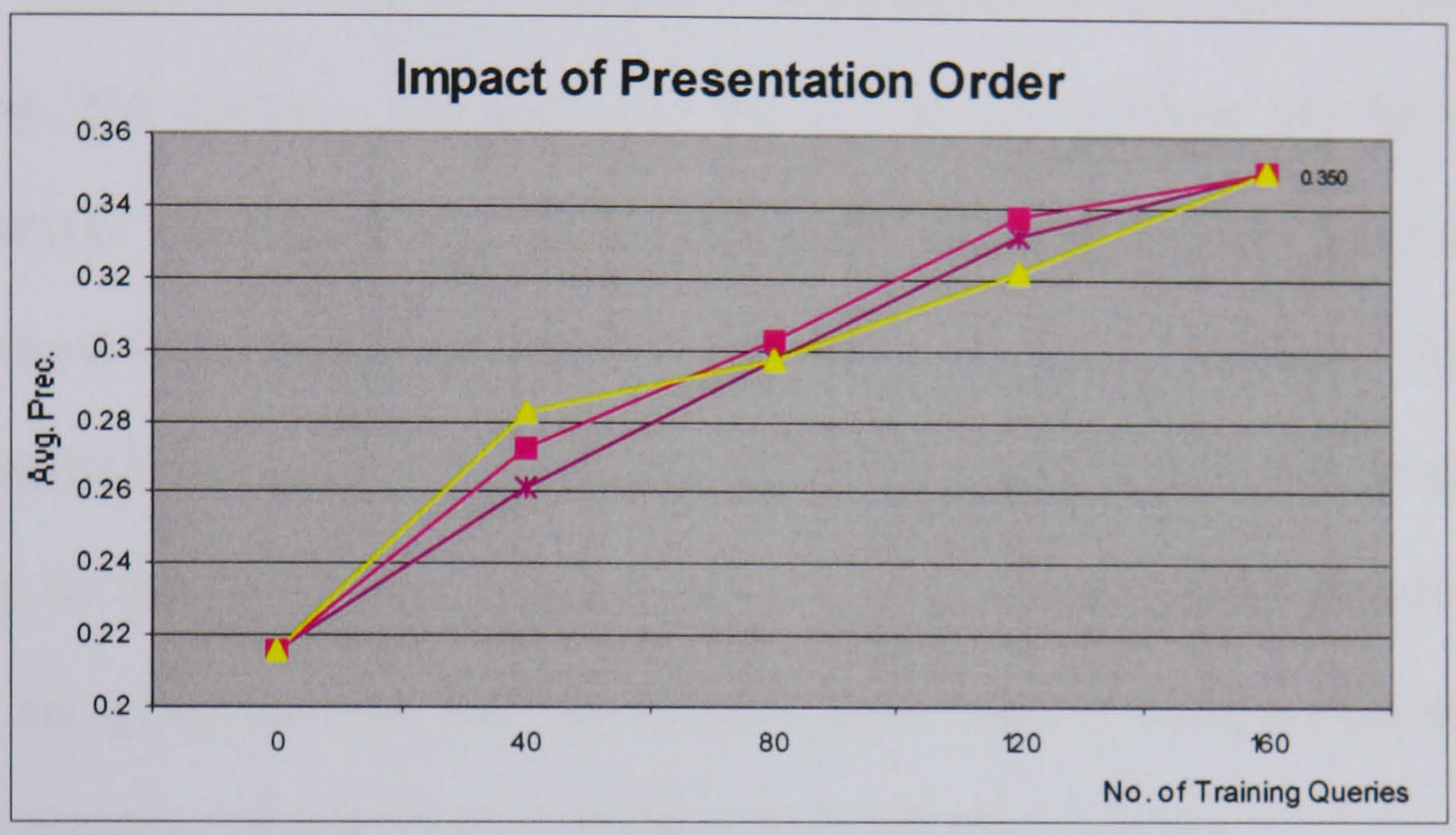


Figure 9.16 : Impact of Presentation Order

As expected, the results of the three runs are slightly different to each other. This is a result of the fact that the training (sub) sets of the training sessions are different (except at the last training level) and the contribution of different training queries on training is different. Since early reinforcement of a document helps other (relevant) queries to pick it up earlier in the training process, it allows the weights of those matching concepts to undergo more tuning. This causes the order of presentation of the queries to make a difference. However, despite the random presentation order, the results at their last training-testing sessions (i.e. when the training set is the same at all three test runs) reaches the same value when adequate training iterations are used (i.e. when training queries are shown repeatedly until the weights are converged).

Note that since our experimental set-up consists of a fixed set of queries with fixed (predecided) relevance assessments, and that we iterate this set of queries over and over again during training, using the same set of relevance judgements (for user feedback), the

weights of concepts tend to converge in our case. However, in a practical situation in which the user feedback is not fixed and the same set of queries does not appear iteratively, weights will never converge to fixed values. Instead, they will keep changing dynamically, depending on user interactions.

9.5.6 Effect of Learning on (numbers of) Concept and Keyword Matches

In this section, we carry out a detailed comparison of the contributions of keyword matching and concept matching by looking at the average numbers of matching unit-concepts and keywords per query-document pair with relevant documents as well as with non-relevant documents. We use the results of Test18 through Test23 and report corresponding average counts on the CISI, CACM and Cranfield collections, which were obtained by testing the system at the end of a pre-decided number of learning iterations. Counts are given on seen and unseen queries in order to compare the performance of concept matching (and keyword matching) between the two cases.

Note that for average count calculations with relevant documents, all relevant documents in the database (for each query) were taken into account regardless of whether they were recalled (RSV value > zero) or not. This is to include the documents which have nothing in common with a query but are still assessed as relevant. But with non-relevant documents, only the non-relevant documents with non-zero RSV values were considered, as there is no benefit of considering non-relevant documents that have nothing in common with the query.

The results are listed in the following four tables (Table 9.18 - Table 9.21) and a summary of results is given in Table 9.22. Note that the figures shown in the tables were obtained after 20 training iterations for the case of unseen query testing, and after 100 training iterations for the the case of seen query testing.

Iterations	CRANFIELD		CACM		CISI	
	Concepts	Keys	Concepts	Keys	Concepts	Keys
0	0.710348	1.912856	0.189849	1.821537	-	-
1	1.140801	2.2125603	0.425843	2.269914	-	-
2	1.142984	2.130332	0.426393	2.271176	0.244544	2.182932
5	1.142984	2.130332	0.426393	2.271176	0.244544	2.182932
20	1.142984	2.130332	0.426393	2.271176	0.244544	2.182932

Table 9.18 : Average Number of Unit-Concept and Keyword Matches per Query-Document Pair with Relevant Documents (on Unseen Queries)

Iterations	Cranfield		CACM		CISI	
	Concepts	Keys	Concepts	Keys	Concepts	Keys
0	0.710348	1.912856	-	-	-	-
1	5.871865	0.506639	-	-	-	-
2	5.929075	0.495198	7.416438	1.302963	3.81725	0.986864
5	5.929075	0.495198	7.416438	1.302963	3.836858	0.986864
100	5.929075	0.495198	7.416438	1.302963	3.836858	0.986864

Table 9.19 : Average Number of Unit-Concept and Keyword Matches per Query-Document Pair with Relevant Documents (on Seen Queries)

Iterations	Cranfield		CACM		CISI	
	Concepts	Keys	Concepts	Keys	Concepts	Keys
0	0.081282	1.413301	0.010099	1.272748	-	-
1	0.092168	1.559275	0.051313	1.315085	-	-
2	0.092268	1.560897	0.051598	1.316208	0.111251	1.668411
5	-	-	0.051598	1.316208	0.111251	1.668437
20	0.092267	1.560907	0.0516	1.316192	0.111251	1.668437

Table 9.20 : Average Number of Unit-Concept and Keyword Matches per Query-Document Pair with Non-Relevant Documents (on Unseen Queries)

Iterations	Cranfield		CACM		CISI	
	Concepts	Keys	Concepts	Keys	Concepts	Keys
0	0.081282	1.413301	-	-	-	-
1	0.092679	1.564219	-	-	-	-
2	0.092688	1.565913	0.52672	1.316891	0.112511	1.567456
5	-	-	0.52672	1.316891	0.112511	1.567508
20	0.092688	1.565913	0.52672	1.316891	0.112511	1.567508

Table 9.21 : Average Number of Unit-Concept and Keyword Matches per Query-Document Pair with Non-Relevant Documents (on Seen Queries)

	With Relevant Documents			With Non-Relevant Documents		
	Concepts	Keywords	Concepts Keywords	Concepts	Keywords	Concepts Keywords
CRAN-Unseen Query Testing	1.142984	2.130332	0.5365	0.092267	1.560907	0.05911
CRAN-Seen Query Testing	5.929075	0.495198		0.092688	1.565913	0.05919
CACM-Unseen Query Testing	0.426393	2.271176	0.1877	0.0516	1.316192	0.03920
CACM-Seen Query Testing	7.416438	1.302963		0.052672	1.316891	0.03999
CISI-Unseen Query Testing	0.244544	2.182932	0.11202	0.111251	1.668437	0.066
CISI-Seen Query Testing	3.836858	0.986864		0.112511	0.1567508	0.7177

Table 9.22 : Summary of Unit-Concept and Keyword Counts

Observations

With relevant documents, more keyword matches have taken place than concept matches when unseen queries were tested (see Table 9.18). This means that concept addition (learning) has failed to make unit-concepts the driving entity for retrieval. As noticed before, the main reason for this may be the lack of sufficient overlaps between queries (in the set of queries). Recall that the criteria used for selecting test queries is based on the number of documents assessed as relevant to each query, rather than the actual similarity of queries. Therefore, given a set of queries with sufficient overlaps (in terms of similarity in the use of the same concepts), and queries picked up for testing based on their similarity so that the similar queries are equally distributed among the training and testing sets, the result would have been different. For instance, numbers of concept matches are much higher than keyword matches in the case testing on the seen queries given in Table 9.19. In fact, no keyword matches take place between a query and a relevant document once the document is updated with the query concepts. However, there are always cases where certain documents are not picked up at all by certain queries to which those documents are judged as relevant.

Few concept matches have taken place with non-relevant documents compared to keyword matches both on testing seen and unseen queries (see Table 9.22). These results show that a smaller number of false hits are made by concept matching compared to keyword

matching. This means that concept matching makes more accurate retrieval (increase precision) compared to keyword matching.

Another interesting result is that only the first couple of training iterations have shown the biggest changes in the numbers of concept and keyword matches made with relevant documents (see Table 9.18 and Table 9.19). This is because most concept additions take place at the early iterations. Once concepts of a query have been added to its relevant documents, the numbers of concept and keyword matches between the query and its relevant documents are the same at the subsequent iterations. Learning at subsequent iterations only helps tuning weights. This is the reason for the big jump in average precisions observed at the first iteration in Figure 9.18 and Figure 9.19.

Furthermore, relatively more concept matches and fewer keyword matches have taken place with relevant documents in the Cranfield collection, in the case of seen query testing (Table 9.19). Although more concept matches have occurred with relevant documents in the CACM collection in the case of testing on seen queries (Table 9.19), it has reported fewer concept matches (and more keyword matches) with relevant documents in the case of testing on unseen queries. As a result, the ratio of the number of concept matches to the number of keyword matches is higher on the Cranfield collection (with relevant documents when unseen queries were tested). These figures (ratios) indicate how good the system is on concept matching against keyword matching. In addition, the P-R curve on the CACM in the case of seen query testing (Figure 9.17) is no better than the corresponding P-R curve on the Cranfield. The greater number of concept matches that it has made with relevant documents have not improved the performance of CACM on seen query testing. This could be due to the presence of more common concepts between queries and documents (as also evident by the counts with non-relevant documents).

The system has performed relatively badly on the CISI collection compared to CACM and Cranfield. It has made fewer concept matches with relevant documents in seen query

testing and more concept matches with non-relevant documents in the unseen case compared to the other two collections.

9.5.7 Performance Comparison between Seen and Unseen Test Queries

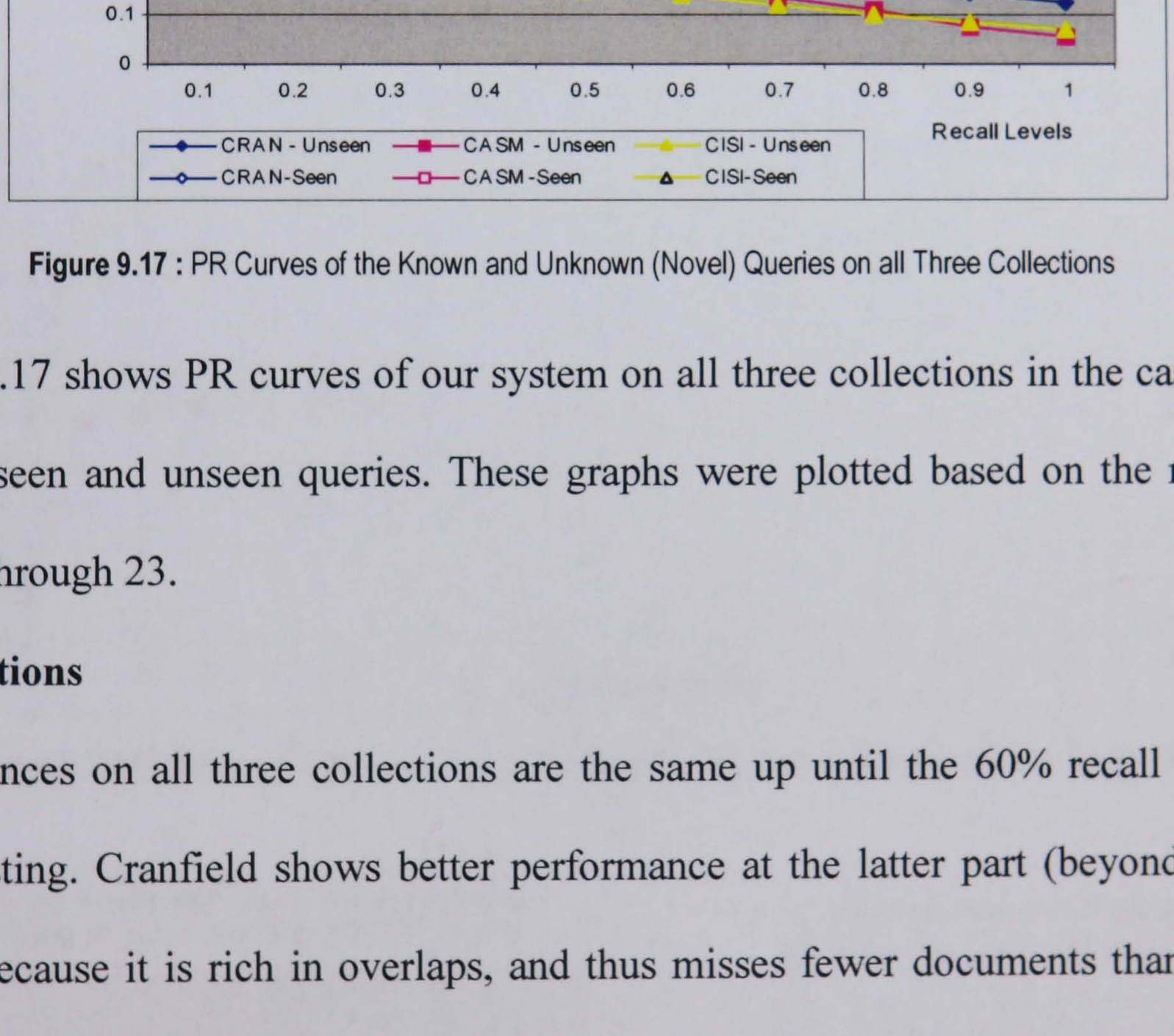


Figure 9.17 : PR Curves of the Known and Unknown (Novel) Queries on all Three Collections

(Figure 9.17 shows PR curves of our system on all three collections in the cases of testing on both seen and unseen queries. These graphs were plotted based on the results of the tests 18 through 23.

Observations

Performances on all three collections are the same up until the 60% recall level in seen query testing. Cranfield shows better performance at the latter part (beyond 60% recall) mainly because it is rich in overlaps, and thus misses fewer documents than in the other two collections. This result suggests that if there were more overlaps between queries, better results could be obtained. Also, CISI performs better than CACM until the 90% recall level. This may be because it is richer in overlaps than CACM. The performance of the system on unseen query testing is comparatively poor on the the CISI and CACM than on Cranfield. The CACM has performed better than CISI, perhaps, due to the more

expressive queries (in terms of the length) that it has compared to the queries in CISI (see Table 9.3).

9.5.8 Performance Dynamics over Training Iterations

In this section, we examine and analyse the dynamics of the system's performance over training iterations. Only the results on Cranfield (of Test20) are used here. Results on the other two collections (CISI and CACM) also show similar patterns (not given here). Figures 9.18 and 9.19 show how the performance curves (average precisions) at different retrieval points behave over training iterations for the two cases of unseen and seen query testing respectively.

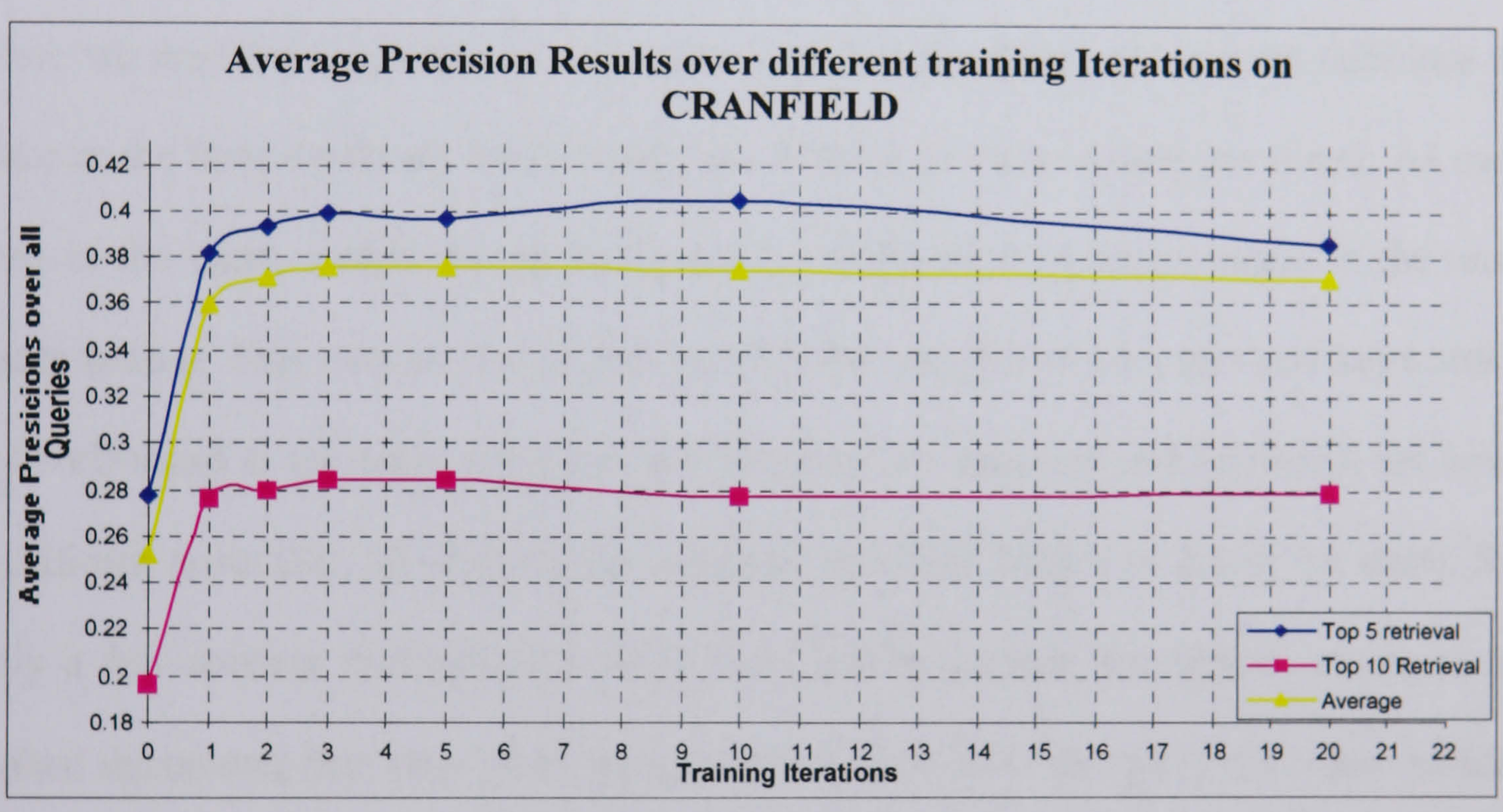
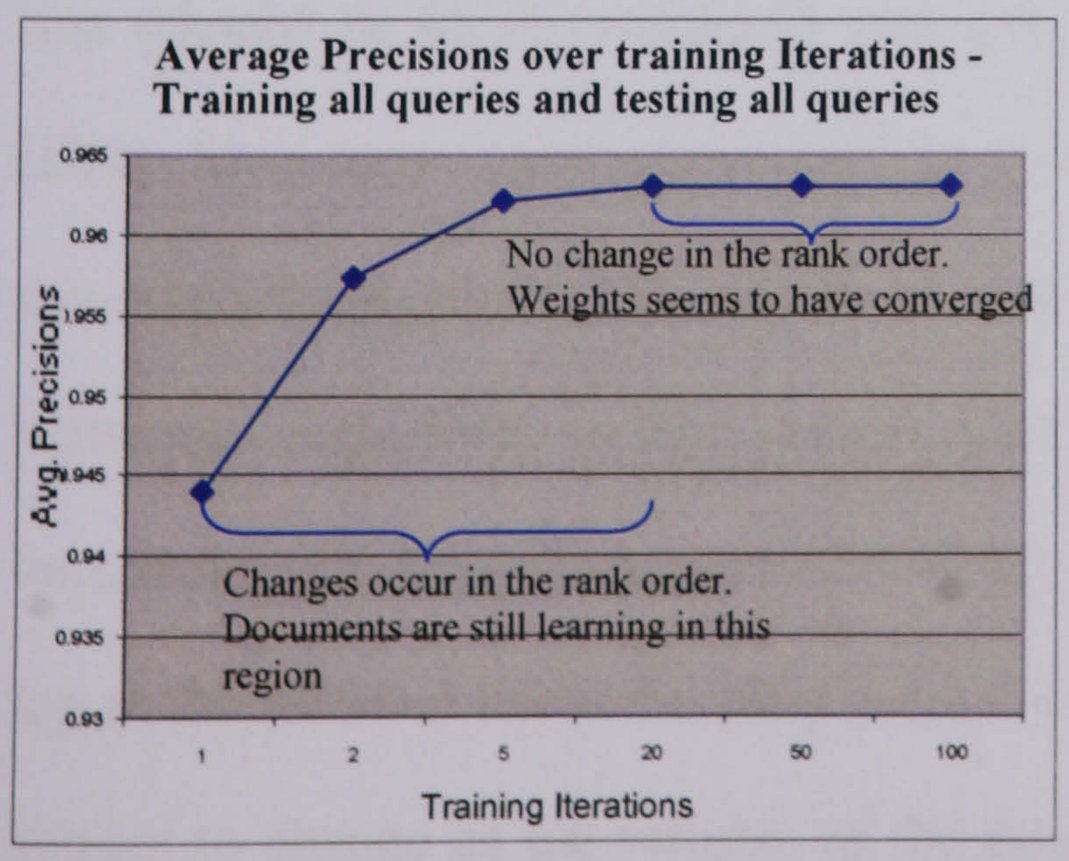


Figure 9.18 : Performance Dynamics over Training Iterations on Cranfield Collection (Unseen Query Testing)



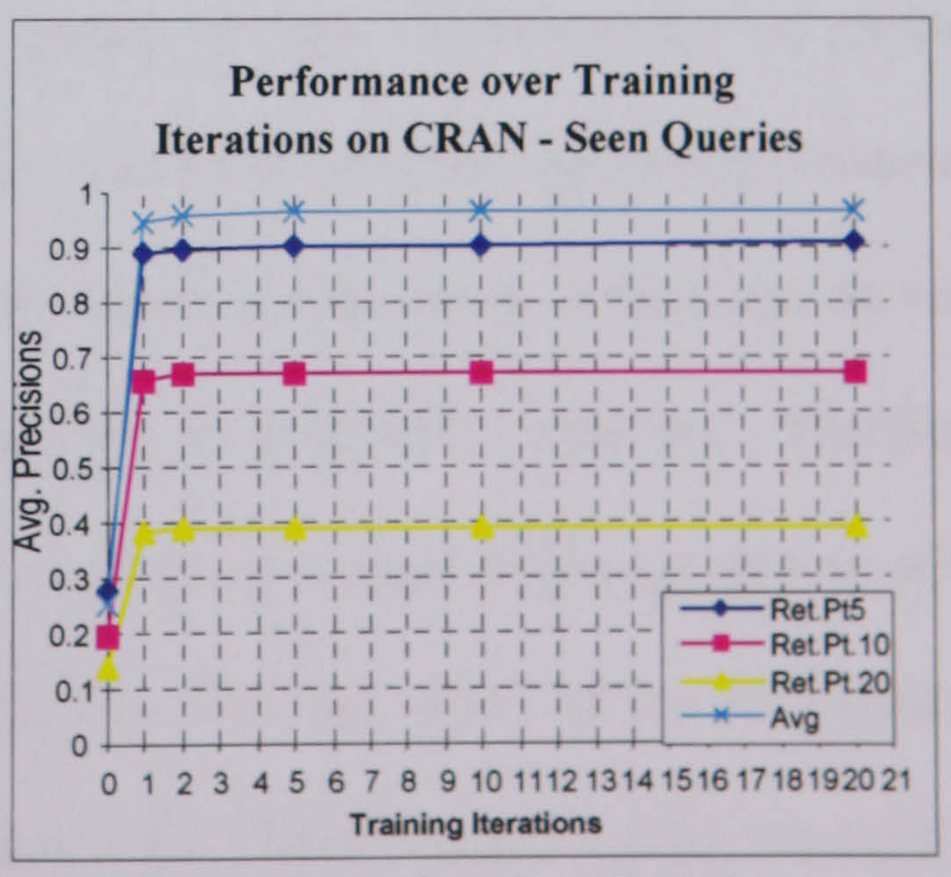


Figure 9.19 : Performance Dynamics over Training Iterations on Cranfield Collection (Seen Query Testing)
(Figure on the right shows the same result of the first 20 iterations at different retrieval points)

Observations

There is a sharp increase of performance at the first iteration. followed by a little increase in the next couple of iterations. This is because the maximum learning (concept additions) takes place the very first time a query is shown, i.e. concepts of queries are added to the relevant documents the very first time they (relevant documents) were retrieved for the query. This result is also observed in the unit-concept counts given in Table 9.18 and Table 9.19. The addition of query concepts at early iterations (retrieval sessions) trigger other similar queries (queries sharing the same unit-concepts) to retrieve those (reinforced) documents at subsequent iterations.

Since we repeated the same set of queries over the iterations, no concept additions take place at the later iterations, instead only the weights of the concepts are tuned. As can be seen in the chart, weight tuning has caused a slight drop in performance in the unseen query testing. This may be due to the fact that the weights of concepts and keywords are not well-tuned at the early iterations, and therefore all concepts and keywords are equally significant (note that, all of them are assigned an initial weight of 2.5 at the start). Since only a few concept matches take place with unseen queries, the system seems to have ranked documents that have more features in common with the query (i.e. more matching unit-concepts and keywords) the top. Some of these matching unit-concepts may not be the best (should be highly-weighted) concepts in terms of their representation or retrieval ability. Since more concept and keyword matches occur with relevant documents than with non-relevant documents, having more matches causes performance to increase at early iterations, regardless of the actual significances of matching elements. Therefore, documents with better information to satisfy the information need might not appear at the top of the retrieval list at the early stages of learning. The rank order of the documents is subject to change as the weights of the documents are (better) tuned during the training iterations (see Figure 9.20).

This phenomenon in which the performance drops as the queries are shown a greater number of times (iterated more) can also be described in machine learning terms as a result of "overfitting". Overfitting causes the system to be less robust for average novel situations.

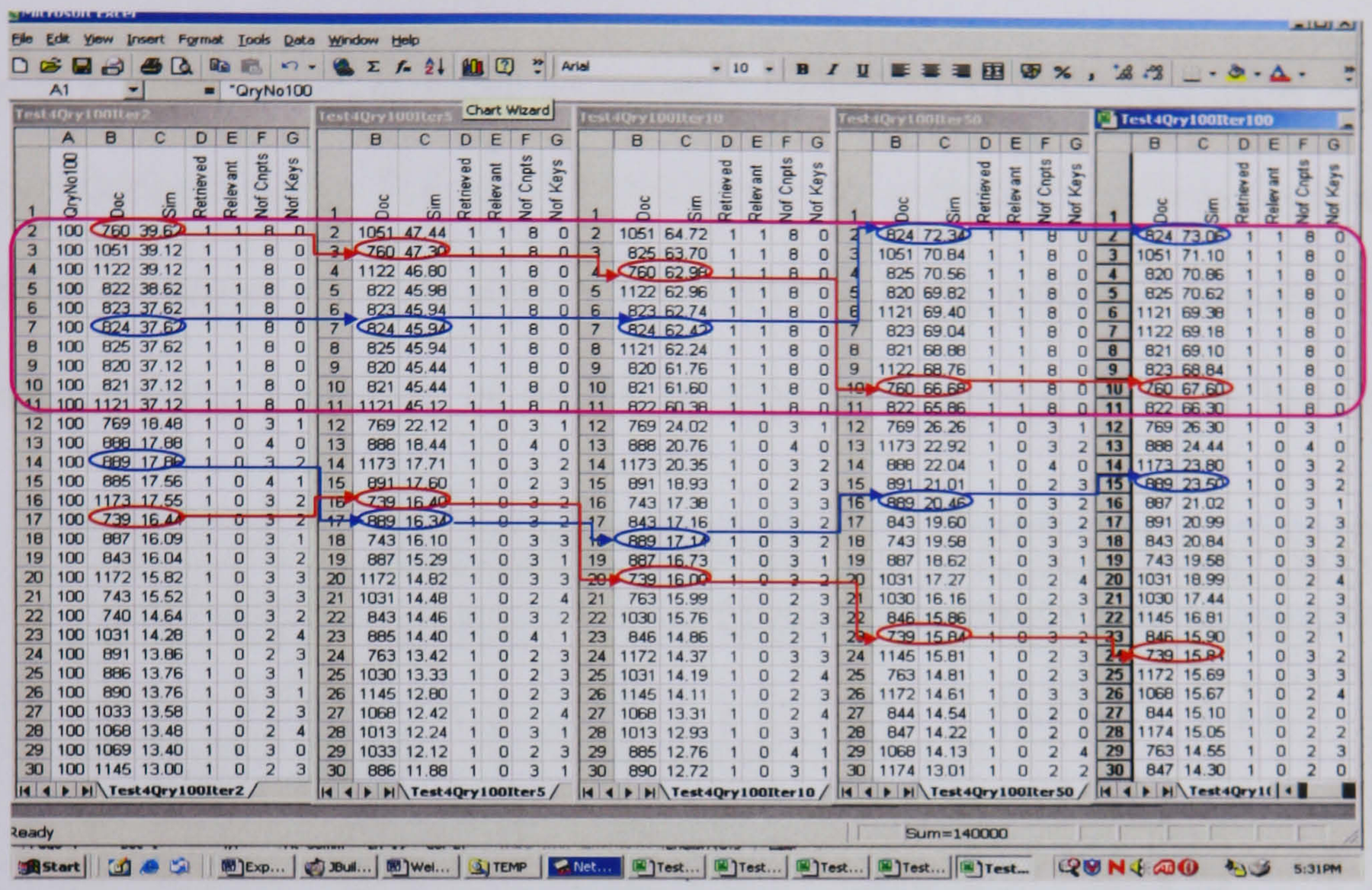


Figure 9.20 : Results of Query#100 (a Seen Query) at Different Training Iterations (Results of Test23)

In contrast to the unseen query testing, the performance of the system on seen query testing does not show a drop in performance over training iterations (Figure 9.19). It shows a similar big jump at the first iterations, followed by little increases thereafter.

This is because all the relevant documents have already been retrieved and updated with the corresponding query concepts at the early iterations (in the case of seen query testing). Therefore, the number of concept matches with relevant documents is higher, and as a result those reinforced documents have much higher similarity scores than the non-relevant ones (see Figure 9.20). As the training proceeds, the similarity scores of relevant documents get higher and higher, making them (relevant documents) clearly separate from the non-relevant documents (Figure 9.20). Although this does not help in increasing performance figures (as the documents at the top are the same), it helps in changing the rank order among the relevant documents (and also among the non-relevant documents) by ranking the best-match documents at the top. For example, see the result of testing Query

number 100 in the Cranfield collection given above in Figure 9.20. It shows clearly how the similarities of the relevant documents keep increasing, making the block of relevant documents separate from the block of non-relevant documents, and how the order of documents within each block changes as weights are learnt over iterations.

Figure 9.21 shows an example of testing Query#1 in Test20. This query has not been shown during training. The rank of certain relevant documents, such as documents 14 and 858, which started at a low position, moved up to higher positions as a result of concept weight learning. Note that, no additional concept or keyword matches have taken place

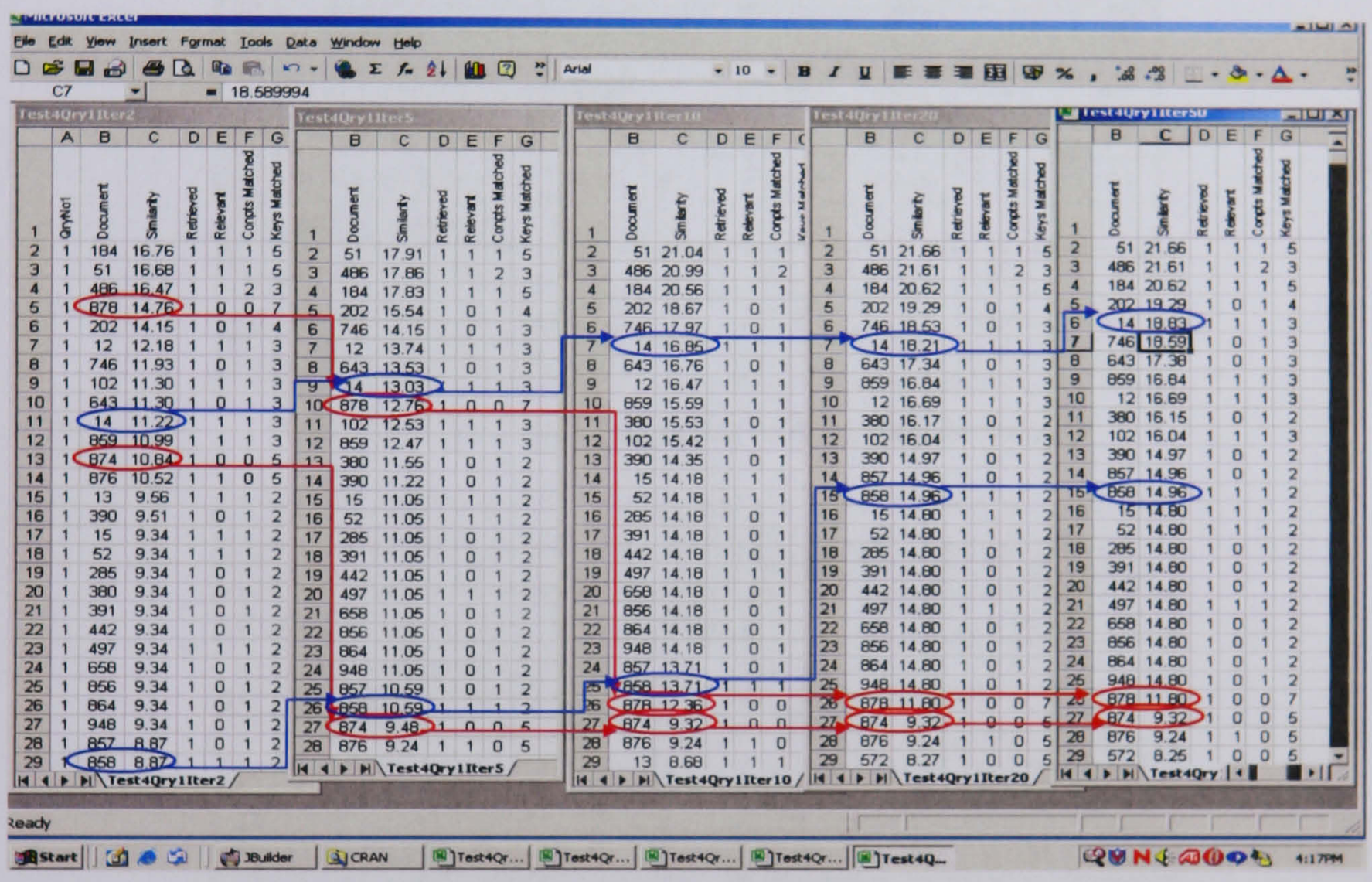


Figure 9.21 : Results of Query#1 (an Unseen Query) at Different Training Iterations (Results of Test20)

between iterations (i.e. unit-concept and keyword matches are the same). On the other hand, the ranks of certain other non-relevant documents, such as documents 878 and 874, have lowered over the iterations. The documents whose ranks are lowered over iterations tend to be the ones that have been retrieved by keyword matching only, while the documents whose ranks are heightened tend to be the ones having at least one good (highly-weighted) matching unit-concept (see the keyword and unit-concept counts). There are also certain documents (such as document 843 in Figure 9.20 and document 864 in Figure 9.21) that show mixed behaviour, i.e. increased rankings at certain iterations and decreased rankings at certain other iterations. These are the ones with concepts that conflict

with training queries, i.e. certain queries increase their weights, while certain other queries decrease them.

9.5.9 Comparison with Published Results

Table 9.23 and Table 9.24 list average precisions of our system obtained on the CACM and CISI collections (results of Test18 and Test19) alongside the results published by Carpineto and Romano [Carpineto & Romano 2000]. In addition, the average precisions of our system on the Cranfield collection (results of Test20) are listed in Table 9.25.

CACM				
	BMR	**HCR**	**CLR**	**Ours**
Avg. Precision (non-interpolated)	0.320	0.231	0.253	0.224
11-point precision Average	0.340	0.257	0.281	0.203
Precision at retrieval Point 5	0.346	0.342	0.412	0.265
Precision at retrieval Point 10	0.304	0.298	0.240	0.204
Precision at retrieval Point 20	0.238	0.202	0.164	0.172
Recall at retrieval point 5	0.227	0.136	0.228	0.134
Recall at retrieval point 10	0.297	0.224	0.266	0.187
Recall at retrieval point 20	0.428	0.323	0.319	0.289

Table 9.23 : Comparison Figures with Published Results on CACM

CISI				
	BMR	**HCR**	**CLR**	**Ours**
Avg. Precision (non-interpolated)	0.164	0.127	0.162	0.165
11-point precision Average	0.183	0.153	0.185	0.149
Precision at retrieval Point 5	0.269	0.280	0.337	0.274
Precision at retrieval Point 10	0.266	0.254	0.286	0.217
Precision at retrieval Point 20	0.239	0.209	0.234	0.179
Recall at retrieval point 5	0.027	0.042	0.043	0.063
Recall at retrieval point 10	0.060	0.066	0.095	0.077
Recall at retrieval point 20	0.107	0.103	0.139	0.113

Table 9.24 : Comparison Figures with Published Results on CISI

Cranfield	
Avg. Precision (non-interpolated)	0.371
11-point precision Average	0.349
Precision at retrieval Point 5	0.386
Precision at retrieval Point 10	0.279
Precision at retrieval Point 20	0.186
Recall at retrieval point 5	0.295
Recall at retrieval point 10	0.396
Recall at retrieval point 20	0.505

Table 9.25 : Our Results on Cranfield

Observations

The results of our system, given above in Tables 9.23 through 9.25, are the test results on novel (unseen) queries. As can be seen, the performance results of our system are as good as the published results (by Carpineto and Romano) on both the CACM and CISI collections. However, as with the published results of the three models (BMR, HCR and CLR), our model also has shown better figures on CACM than on CISI. The comparative performance given by our model on these two collections (CACM and CISI), despite their inadequate desired properties of the two collections for our learning strategy to make documents learn from experience (described in Section 9.1.1), is encouraging. Note that the removal of author and journal information from the document collections (of CACM and CISI) might have caused a small (negative) affect on the performance figures of our system on these two collections.

We expect our model to perform better on a test collection with more cross relations and other desired features. For instance, the results on the Cranfield collection (Table 9.25), though not comparable with the results obtained on the other two collections, show higher performance figures. This may be because Cranfield is richer in terms of the desired properties that help learning in our system. Also, see the P-R curves on the seen query testing given in Figure 9.17. The performances of the model on all three collections are the same (near optimal) up until the 60% recall level on seen query testing. This result suggests that, if there were more overlaps between queries (i.e. if the system were trained with more similar queries for each information need), better results could be obtained.

9.5.10 Impact of Learning on the Size of the Documents

Our reinforcement learning strategy causes the documents (representations) to grow as they learn. The addition of query concepts to the documents is the major cause of the growth, as changing the values of weights does not change the size of documents. In the

following table (Table 9.26), the sizes of the document collection(s) before and after learning are tabulated to give an idea of how much learning affects the collection sizes.

Test Collection	Original Size (Before Learning)	After Learning	Percentage Increase
Cranfield	4.31 MB (4,523,439 bytes)	5.09 MB (5,342,208 bytes)	18%
CACM	5.25 MB (5,514,593 bytes)	5.84 MB (6,132,496 bytes)	11.23%
CISI	4.13 MB (4,336,061 bytes)	4.63 MB (4,856,162 bytes)	12.10%

Table 9.26 : Growth of Test Collections

The sizes reported here under "After Learning" are the sizes of test collections after the training of Test21, Test22 and Test23 were completed. In each of these tests, all queries were trained for **100** iterations. Note that our aim here is not to draw any conclusions about the growth of test collections, but to show some statistics for the sake of completeness of our evaluation. However, the percentage increments indicate that Cranfield has learnt more than the other two and that the CISI and CACM have learnt roughly the same.

9.6 SUMMARY

This chapter covered a detailed evaluation of our implementation. Test methodologies used and tests conducted (23 in total) under each test methodology were described. The results of the tests were plotted in a number of different charts and the effects of different aspects of our system were observed. It was first shown that the performance of the system in its full capacity improves over training queries (experience) (Figure 9.3), and then evidence was given for the conclusion that this performance improvement is a collective result of all the components (2 matching entities and 3 learning components). A detailed analysis of performance dynamics was given and reasons for certain observations such as the big performance jump at the early iterations and slight drop at the later iterations were discussed. The results obtained on CISI and CACM were compared (Table 9.23 and Table 9.24) against the results reported by Carpineto and Romano. Finally, statistics of the collection growth sizes were given to show how much concept addition affects the size of the document collections. The next chapter uses the test results and the observations made in this chapter to draw final conclusions on our work.

CHAPTER 10 - CONCLUSIONS AND FUTURE WORK

This chapter begins with a summarised discussion of our research objectives and the approach taken towards achieving them, followed by a discussion of the strengths and weaknesses of the proposed model. The environments in which it would be likely to be successful are also indicated. Next, conclusions are drawn based on the experimental results reported in the previous chapter. Finally, this thesis is concluded with a discussion of possible directions for improvements and fruitful future extensions.

10.1 DISCUSSION

10.1.1 A Summary of Our Approach

The main goal of this research was to investigate the use of a more elaborate construct as the basic unit for matching queries and documents. A secondary goal was to investigate the role of interactive reinforcement learning, based on relevance feedback information, in this task.

In our work, an emphasis was placed on the extraction and formulation of concepts from document contents and the creation of document (query) representations in the form of concept lattices. This supported the production of an IR model that is based on (formal) concept matching rather than keyword matching. We used an object-attribute pair (i.e. the smallest construct of a formal concept), that we referred to as a *unit-concept*, as the basic representation (hence the comparison) unit. In particular, we attempted to match the most specific (formal) concepts between queries and documents whenever possible, to help retrieval based on the most specific concepts or ideas rather than more generic or common ideas that might present in documents. The use of unit-concepts allowed us to perform partial matching between two formal concepts. In addition, matches of the constituent elements of unit-concepts (i.e. object or attribute matches, refered to as keyword matches)

were also considered in the absence of unit-concept matches, in order not to miss out any contribution that such a feature common to a query-document pair could make on retrieval.

Significances of unit-concepts and keywords were modelled with weights with respect to the document in which they appeared. The allocation of weights to unit-concepts made it possible to compute a degree of similarity between two formal concepts based on the degree of importance of matching unit-concepts between them.

A reinforcement learning strategy based on relevance feedback information was employed for achieving our secondary goal - interactive learning. It has two major components: (1) weight learning/tuning; and (2) concept addition. The weight learning component deals with learning weights of both unit-concepts and keywords, and the concept addition component with updating relevant documents with the concepts of their respective queries. Learning takes place on document representations. Documents retrieved and found to be useful for a given query are positively reinforced and retrieved documents that were not found to be useful are negatively reinforced. During positive reinforcements, all the query concepts that are not present in documents relevant to the query are added into the document representations and the weights of matching unit-concepts (and matching keywords) in the relevant documents are rewarded (increased). During negative reinforcements, the weights of matching unit-concepts (and keywords) of those documents that are retrieved but not found to be very useful are penalised (decreased). Through these steps, documents that are relevant to the query are made more similar to the query and the retrieved documents that are not relevant are made less similar to the query.

Another very important feature of our implementation is the use of BAM structures for embedding concept lattices. We consider this to be a significant contribution to the use of concept lattices in IR as a whole, as well as individually to the applications of BAMs (adding to the few applications of BAMs to be found in literature). Also, this work is

significant as it may be the first attempt to employ Bělohlávek's discovery [Bělohlávek 2000] in a practical application.

Through the use of BAMs we avoid the need to use complex lattice building algorithms, which makes the use of concept lattices easier and more efficient. The process of setting up a BAM with a concept lattice is a one-off learning task, and once a concept lattice is set up in a BAM, it can be easily updated with additional concepts (as required by our concept learning process) by simply adding nodes to its object and/or attribute layers as appropriate and re-computing the connections (link weights) between nodes (Section 6.8). Furthermore, a BAM allows fast access to the most specific (or most generic) concept in its learned lattice for a given set of objects (attributes) (Section 6.6.3) without having to use complex algorithms to traverse the lattice structure.

10.1.2 Strengths and Differences to Keyword-Based Models

Our model is different to keyword-based approaches in a number of ways, and it is these differences that give it its strength. The first difference is the use of object-attribute pairs (unit-concepts) to match between queries and documents, and the allocation of weights to unit-concepts rather than to individual keywords. Though concept matching may result in some relevant documents being missed out, it helps to reduce false hits, thus helping to improve precision. In particular, the multi-constituent structure of the comparison unit (unit-concept) helps in solving the polesemy (or homonymous) problem to a greater extent, and the synonymy problem to some extent. A homonymous term that is used in one of its several meanings is unlikely to possess the same properties/attributes (if it appears as an object) or is unlikely to be possessed by the same objects (if it is an attribute) as when used in one of its other meanings. Thus two concepts formed by the same homonymous term but with two different meanings are unlikely to match together. On the other hand, concept matching may help in identifying where a synonymy problem occurs, as two concepts formed by two synonymous terms (objects) are likely to share the same attributes (if the

synonym words are objects) or are likely to be possessed by the same objects (if they are attributes).

In addition, maintaining weights with respect to documents independently of the other documents in the collection gives greater freedom and flexibility in concept weighting. In particular, it helps the same concept to have different weights in different documents independent of each other. Although systems based on *tf-idf* weighting schemes do allocate different weights to the same keywords in different documents, they are to some extent constrained to the frequency of term appearances within the document as well as within the document collection. As a result, the qualities of those weighting schemes are heavily dependent on the quality of the documents (writing). Deliberate repetition of keywords, the use of synonyms, homonyms, anaphors and the use of examples are a few well-known problems in those weighting schemes. In contrast, we do not use frequency statistics of any kind at all. Instead, weights are learned based on user interactions. Therefore, the quality of concept weights in documents in our case is attributable to the quality of user queries and relevance judgements. However, as with others, our system can also be fooled by being given wrong user judgements deliberately (Section 7.2.2.3).

Additionally, we attempted to use the sub-super concept relationship information that is already available in concept lattice representations of documents and queries. The idea was to focus on matching more specific concepts between queries and documents. However, the lack of adequate information to build sufficiently informative concept lattices, in particular for queries, made concept lattices of queries less useful in this particular feature.

Furthermore, the document representations which were created using only the local information present in the individual documents may make them initially more local rather than global. However, the balance between locality and globality will be achieved through learning via user interactions. During learning, concepts that are capable of identifying a document for retrieval tend to emerge as a result of their weights being tuned to have

higher values. At present no stopwords are used; verbs only are stemmed and plural nouns are converted to singular. It would be useful to have a set of "stop-unit-concepts", and the effect of stemming all the terms remains to be investigated. However, we presume that stemming will not help to improve the effectiveness of the system, simply because it removes the semantics of terms.

Another interesting property related to the use of local information for document characterisation and weight learning is that it helps greatly towards making the system domain independent. Unlike in *tf-idf* based schemes, each document is characterised independent of the others. The later addition of a document to the collection affects neither the representation of other already present documents nor their retrieval. Though *tf-idf* based indexing schemes can be applied in documents from any domain, they are not as flexible as ours since the term weights depend on frequency statistics of the collection.

In addition, learning in our model helps enhance document representations according to user decisions, by adding concepts in the query to the relevant documents, and by tuning the weights of concepts. The weight tuning is engineered so that the concepts which specifically help identify a document (i.e. specific unit-concepts) gain higher weights and the concepts that are common to many documents and therefore are not useful for the correct retrieval of a document end up with smaller weights. The addition of query concepts to document representations and tuning the concept weights of document representations as described above results in the document representations becoming more customised according to the personal views of its users. As a consequence, better performance of the system can be expected in a more personalised environment, where one or more users with similar interests interact with the system.

An interesting by-product of our learning strategy is that it helps the system to cope with the document ageing problem. As new documents with later and more informative contents are made available, documents that were useful to an information need in the past become

obsolete. Users are likely to pick newer documents with the latest information instead of obsolete ones when a list of retrieved documents is presented to them. As a result, those newer documents get positively reinforced and older documents that were rejected by the users get negatively reinforced. This makes the similarity scores of newer documents become greater and those of older documents become lower, in time causing newer documents to be ranked above the older documents. As more documents with up-to-date information are made available, the older documents will eventually disappear from the retrieved list. However, the classic documents which remain important despite their age will not be affected as they are unlikely to be rejected by all users and thus will be positively reinforced as they are picked up. This is a very useful feature that no system with a static representation scheme could provide. However, we only describe the theoretical aspect of this without empirical evidence, as the system was not tested for this property simply because this was not the main objective of our work. Also, at present, no test methodology and supporting test environment (document collection) is available for the evaluation of this property.

10.1.3 Strengths and Differences to Other FCA-Based Models

The application of concept lattices to information retrieval tasks has usually focused on interactive browsing (e.g. TOSCANA [Becker et al. 2002], CEM [Cole et al. 2000]). These approaches create a large single concept lattice to represent the whole document collection (e.g. Carpineto and Romano's model [Carpineto & Romano 2000], TOSCANA, CEM) and it is this concept lattice that serves as the search space for users to explore. Typically, a visualisation of the full or part of the full concept lattice is provided at the beginning and the user may then navigate through the nodes in the lattice to find the documents contained in each node. Some approaches use conceptual scaling to help visualisation of the concept lattice in a structured manner [e.g. TOSCANA and CEM]. Usually the top level of the structure is displayed first and the user navigates the lattice by expanding (unfolding) the parts/nodes of interest until the desired information/documents

are found. More recently, the basic navigational framework has been extended to accommodate Boolean queries and full-text indexing. Carpineto and Romano's model [Carpineto & Romano 2000] may be one of the pioneering piece of work that used concept lattices to document retrieval without browsing. However, once created, the concept lattice (i.e. search space) is fixed in all these models and therefore they lack adaptivity.

All previous attempts formulate a concept with the document IDs as the objects in its extent, and keywords extracted from the contents of the document as the attributes or properties in its intent. This is a rather unrealistic concept formulation compared to how a human might formulate concepts. The criterion for the construction of the lattice is based simply on the presence of keywords. A node in the lattice does not represent a natural concept or idea, nor does the hierarchical order represent a concept hierarchy. Such a representation only creates categories of documents sharing the same keywords, and therefore is not very different to keyword-based document categorisation technique(s).

Our approach is different to the present FCA approaches found in literature in at least four ways. Firstly, a concept in our work consists of objects stated in the content of the document (i.e. textual labels of objects) rather than document IDs. Properties of the objects stated in the text are the attributes in the intents. Therefore, matching such object-attribute pairs may leads to concept matching in the same sense as how the human brain might work. Secondly, it uses separate concept lattices for representing each individual document rather than using one large concept lattice. Thirdly, it performs document retrieval without browsing. Fourthly, it uses interactive learning of document representations (i.e. dynamic lattices).

The above mentioned features, in particular the use of one lattice per document, give our model a greater flexibility to operate on large document collections with full-text indexing rather than being constrained to smaller document collections and to a limited number of

terms or keywords (as with Carpineto and Romano's experiments [Carpineto & Romano 2000]).

10.1.4 Strengths and Differences to Other Conceptual Knowledge based Models

Models that make use of more elaborate conceptual knowledge representations, such as semantic networks and conceptual graphs, use an extensive amount of detailed information (knowledge) in characterisation (indexing) of text items and as a result suffer from heavy processing requirements. The acquisition of such knowledge requires consultation of domain experts and as such is expensive and sometimes impossible. In addition, storage of such a large amount of knowledge is expensive and inefficient to access. In contrast, the use of concept lattices in our model makes it lighter in terms of the amount of knowledge necessary for the characterisation, and the use of BAMs makes it more efficient in accessing the stored knowledge.

10.1.5 Drawbacks of the Current Implementation

Lack of sufficient information within the contents of documents and queries to create representations with adequate knowledge to well-represent them is a major problem in IR. The difficulty of extracting information already available in the text in the form used by the system (i.e. object-attribute pairs) for creating representations causes further limitations, in particular to the models that operate on more elegant knowledge structures. Ours is not an exemption. An additional concern with our learning strategy is the possible risk involved in adding query concepts into the document representations. This may cause the original concepts or the original purpose of the documents to change in the long term. Also, it may result in the representations of similar documents being made identical. The overheads involved in extracting candidate node pairs from the query and document concept lattices to match between and the unnecessary concepts creeping into the document representations through learning are another two problems. A "stop-concept" list can be used to control unnecessary concepts entering the documents. Periodic pruning of

document representations to remove poorly weighted concepts will also help to dispose of unnecessary concepts. This can be restricted to remove only the concepts that were added through learning in order to keep the original content of the document intact. These problems are further discussed in Section 10.3 under future work.

10.1.6 Potential Environments in Which Our Model is Likely to Perform Well

A main characteristic of an adaptive system is that the system becomes more and more tuned to its environment, or strictly speaking to its inputs, as it learns. Consistency of training examples, or in our case consistency of users in terms of the use of vocabulary in query formulation and making relevance assessments, is essential for such a system to converge or become better tuned for its inputs. Consistency is maximised in single-user environments. Essentially, this makes the system better adapted to its only user, thus making it strictly personalised.

On the other hand, learning in multi-user environments helps with learning more exhaustive and better-generalised document representations. It helps the system to learn the different possible ways that different users formulate similar queries, perhaps using different vocabularies, but targeting the same documents. However, consistency among the users in making relevance assessments is essential for convergence. In an environment with more inconsistent users, the system dynamics would rapidly vary with time, resulting in unreliable system responses to an average user. For instance, in such an environment, the user is not guaranteed to get the same (relevant) documents back for the same query at different attempts. Therefore, the system is likely to perform better in more personalised environments, i.e. in a single user environment or in multi-user environments with similar or consistent users. Indeed it has the potential to outperform conventional keyword-based systems in such environments.

In addition, the fact that there is nothing central in the document representations in our model makes it suitable for distributed environments. No global knowledge of the document collection is used in document characterisation, and the document representations are local and independent of each other. Therefore, documents distributed in a network can be processed locally, including their characterisations, and the results integrated. An interesting extension to this work would therefore be to investigate the potential of the system in a distributed set-up, possibly implemented in a multi-agent environment. This is further discussed under future work given below in Section 10.3.5.

10.2 Results and Conclusions

Firstly, we have shown a way of using more elaborate and true concepts for creating more meaningful representations of textual material and using them for explicit concept matching. Secondly, a radically different approach to using concept lattices in IR was developed and its feasibility was evaluated. Thirdly, the importance of an interactive learning strategy and the effectiveness of retaining the learnt knowledge were demonstrated. Finally, the advantage of a hybrid approach that uses both concept matching and keyword matching, together with concept addition and weight learning mechanisms, was justified.

The empirical results given in Section 9.5 show that without learning, performance of the system is poor and static. Addition of each component of our learning strategy (i.e. concept learning, keyword learning and concept addition) shows improvements, with the system gaining experience as more queries are encountered. This is true for all cases of matching entities, i.e. for "concept matching only", "keyword matching only" and a combination of the two. Moreover, the use of keywords for matching in the absence of matching unit-concepts helps in improving retrieval performance. As a result, the system in its full capacity (i.e. with both keyword matching and concept matching with all three learning components) shows a significant improvement with experience (Figure 9.3). The results

were found to be consistent at different retrieval points (charts in Figure 9.7) as well. The P-R curves given in Figure 9.5 further confirm that the performance of the system improves as more queries are encountered (i.e. as more experience is gained). These results support our interactive learning strategy in its ability to improve document retrieval by enhancing their representations with additional concepts and learning (tuning) concept weights.

Mixed results were observed when concept addition is considered (Figure 9.10) in which concept matching was outperformed by keyword matching, as the system learns. This result, and the improved performance shown by the system when both concept matching and keyword matching were considered, demonstrates the importance of using keywords as a matching unit in addition to unit-concepts.

Results of repeated testing of the system with its full capacity show only marginal performance differences (Figure 9.16) over the presentation order. The variations between the graphs at the middle part of Figure 9.16 were mainly due to the differences of the contributions made by the randomly picked queries during learning. Note that the set of queries picked for training at its sub-training sessions are different (with overlaps) as they are picked randomly from the full training set. However, the set of queries at the final training sessions are the same for all tests.

The analysis into numbers of concept and keyword matches given in Tables 9.18 through Table 9.22 shows fewer concept matches with relevant documents compared to keyword matches. However, more concept matches have taken place with Cranfield compared to concept matches on CISI and CACM (the ratio of concept matches to keyword matches is 0.5365 on Cranfield, 0.1877 on CACM and 0.112 on CISI (See Table 9.22). This result is consistent with the performance of the system on Cranfield as can be seen in the P-R curves in Figure 9.17, suggesting that more concept matches make the system perform better. This is further confirmed by the results on known queries (i.e. testing on the queries

used for training), which shows more concept matches on all three test collections compared to the results on novel queries (unseen query testing), and also much better (near perfect) performance shown by the P-R curves in Figure 9.17. Therefore, we can conclude without doubt that it is the occurrence of more concept matches that have made the system perform better on the Cranfield collection, thus supporting our initial hypothesis that more elaborate constructs of concepts might help improve effectiveness of IR.

The fact that the system makes more concept matches on the Cranfield collection than on the CISI and CACM collections depends on the nature and the quality of queries, documents and relevant assessments in the collection. The main requirements for better learning are well-expressed queries and documents in natural language and adequate overlaps in queries and documents to produce sufficient cross relationships in relevance assessments. Table 9.2 gives some statistics of the cross relations and Table 9.3 gives the average lengths of queries in terms of "number of words", which gives an impression of the expressiveness of queries. In addition to these, the availability of more queries in the Cranfield collection (225 compared to 64 in CACM and 52 in CISI) helps the system to learn better on the Cranfield collection than on the other two collections.

Furthermore, the number of concept matches with non-relevant documents is much smaller compared to the number of keyword matches on all three test collections. This proves that concept matches are more precise compared to keyword matches, thus helping to improve the precision of retrieval results.

Learning has helped the system to make significantly more concept matches with known queries (testing on seen queries) than with novel queries (testing on unseen queries), as can be seen in respective figures given in Table 9.18 and Table 9.19. In practice, when the system is in operation and has adequately learnt from past queries, a mixture of seen and novel queries can be expected. Thus better performance can be expected in a real situation

than in the case of completely unseen query testing reported in Table 9.18, Table 9.19, Table 9.23, Table 9.24 and Table 9.23.

The performance of the system over training iterations (performance dynamics), given in Figure 9.18 and Figure 9.19, shows that learning is rapid in the first few epochs and then tends to saturate. It is during the first epoch that the system picks up the most new relevant documents (for the shown queries) and therefore this is when most of the updatings of the document representations occurs. Hence the performance gain is greater at the first epoch. The concept additions that take place at the early epochs during training trigger those documents to be picked up by different queries at subsequent epochs. The recall of new documents at subsequent iterations becomes less and less as more relevant documents are recalled over training epochs, and finally comes to its limit when either all the relevant documents for the query are recalled or no more new relevant documents can be recalled by further training epochs. This behaviour is clearly reflected in Figure 9.20 for the seen query testing in which all the relevant recalled documents are ranked above the non-relevant documents due to their high similarity (all query concepts are present in the relevant documents in this case).

In the case of unseen query testing, the same dynamics are shown at the early epochs, but a slight drop in average precision is observed during the subsequent iterations. This drop could be due to the low number of concept matches that have taken place with relevant documents during unseen query testing. These concepts, though not guaranteed to be the best candidates for representing the document, are equally weighted at the beginning. Therefore, the better performance shown at early epochs (epochs 1, 2, 3 see Figure 9.18) is purely based on the number of concept matches, regardless of the significances of the matching concepts. Since more concept matches seem to occur with relevant documents than with non-relevant documents, on average, more concept matches favour retrieval performance. However, as the weights are tuned during the subsequent learning epochs,

weights of certain concepts in certain documents that helped their retrieval to certain queries are decreased, and as a result, the rank of those documents within the retrieval results will be degraded, causing a drop in performance.

Another reason for this drop could be the result of "over-fitting", i.e. differences between testing and training queries adversely affect performance on test queries as the system is tuned more and more for the queries in the training set. However, as can be seen in Figure 9.18, this drop is very small and the performance curve shows a tendency towards a "convergence". A sub-conclusion that can be drawn from this observation is that the number of matches that has occurred with insignificant concepts is fewer compared to the number of matches with important concepts, as otherwise the drop in performance would be greater. Note also that when the number of concept matches is smaller, as with the unseen query testing, keyword matches tend to dominate retrieval.

10.2.1 Summary of Final Conclusions:

1. Comparable performance figures given by our system on CACM and CISI to published results (of Carpineto's) indicate the viability of our approach (i.e. concept matching within the FCA framework) to document retrieval. Results on the Cranfield collection show its potential for outperforming keyword-based systems.

   The Cranfield collection has been one of the main document collections used for the evaluation of IR systems by many researchers, including Fabio Crestani and van Rijsbergen [Crestani & Rijsbergen 1994, Crestani 1995] on a neural network-based relevance feedback model; W.B. Croft [Croft et al. 1989, Croft 1980] on an inference model and a cluster-based classification model; Susan Dumais [Dumais et al. 1991] on the latent semantic model; Gerad Salton [Salton & Allan 1994] on a text structuring model; K. L. Kwok [Kwok 1995] on a network approach to probabilistic IR and Thomas Hofmann [Hofmann 1999] on a probabilistic LSI model to name a few. However, despite the myriad IR models

evaluated on the Cranfield collection by a vast number of IR researchers, we were unable to find performance figures for a model similar to ours. In order to make a fair comparison, evaluation results obtained on the same collection by a similar system (i.e. one stemmed on FCA-based concept matching) is required.

2. The system's performance improves significantly as it gains experiences. The point at which this rising trend would achieve its maximum and saturate or go flat is not known. This, in fact, is dependent mainly on the user interactions and therefore it is impossible to give an exact figure. A more practical figure could be obtained if the system was set to operate in a practical environment for a longer period and its performance measured. Such an evaluation is time-consuming and so is avoided during this research.
3. Concept matching seems to produce fewer false hits than keyword matching (hence improved precision), as fewer concept matches occur with non-relevant documents compared to keyword matches.
4. The performance of the system is shown to be superior for the queries seen before or for queries that are similar to the seen ones (i.e. queries similarly formulated with the same vocabulary). This gives the system the potential to perform successfully in a more personalised situation and/or in an environment in which the majority of the users share common interests and the same depth of knowledge about the topics/subjects on which queries are made, and hence expect a similar level of depth from the contents of the documents.

10.2.2 Recommendation

Finally, this research is concluded with the following recommendation. Our objective was to make the use of concepts as similar as possible to the formulation of concepts in the human brain. However, this research was far from achieving this objective, but only a first step towards it. The difficulty of automatic concept extraction from text and inadequate background information available in the contents of documents for building more complete

concept hierarchies caused major difficulties for an investigation of the full potential of concept matching within the framework of FCA. However, given these difficulties, the comparative performance results obtained are impressive and encouraging. More than outperforming existing models, it shows a way of performing true concept matching and the feasibility of using FCA in a different and more advantageous way to that of existing FCA approaches. We are optimistic of the potential of the FCA framework to deliver better performance, provided that better and more meaningful document and query representations can be created through the incorporation of background knowledge, and through advances in NLP technology for the extraction of objects and related attributes for formulating more meaningful concepts.

10.3 Future Work

This work can be extended and improved at least in the following five directions.

1. by introducing in external knowledge source(s) to enhance the document and query lattices with background knowledge
2. by incorporating a query reformulation mechanism.
3. by allowing documents to learn from each other.
4. by improving concept extraction for creating more meaningful concepts and concept hierarchies
5. by employing a more efficient mechanism for extracting candidate matching concepts (nodes) pairs from concept lattice representations of queries and documents.

10.3.1 Using External Knowledge Sources

External knowledge sources can be used for two purposes: (1) for enhancing document representations with background knowledge; and (2) for reformulating queries. Both of these help to minimise ambiguities in the meanings of concepts and reduce imprecision involved with their (document and query) representations.

The essence of the whole exercise of using more elegant constructs of concepts (i.e. formal concepts), structuring them hierarchically in concept lattices, and allowing document representations to learn from experience etc. is to make the IRS better understand the

contents of documents (and queries) and make use of such knowledge for the retrieval of documents by concept matching. Understanding the contents of documents has been identified as a way forward for achieving significant improvements for retrieval effectiveness. However, the nature of documents is such that they are not self-contained, meaning that understanding the content of a document needs a great deal of background knowledge about the subject. This has caused serious problems for machine understanding of documents based only on their contents. As a result, representations created by extracting (local) information from the contents of documents are incomplete. This affects shorter documents (queries in particular) more severely as they contain less detail and hence less background information than longer documents. Consequently, this makes lattice representations of short and less informative documents (and queries) less effective and the matching of more specific concepts between queries and documents a more difficult task to achieve.

The use of domain knowledge is a major approach that has been applied to understanding documents in IR. For this purpose the external sources such as ontologies, thesauri and dictionaries have been used in IR applications. The use of these knowledge sources in the past has been mainly to reformulate queries by adding new terms or phrases. In particular, dictionaries and lexical databases such as WordNet (*http://ww.cogsci.princeton.edu/~wn/*) have been used to add synonymous words to the query, mainly to alleviate the synonymy problem. In addition, hierarchical representations of terms or phrases in knowledge sources allow the enhancement of queries with broader, narrower or related terms (to the existing terms). An application of a thesaurus in creating document and query representations in a FCA-based approach can be found in [Cole et al. 1997].

Alternatively, the use of learning mechanisms allows systems to learn knowledge from experience. In contrast to the use of external knowledge sources, learning has been used to capture underlying associations between terms (concepts) and also between documents

based on term co-occurrences and relevance feedback. The knowledge of those associations has been used: (1) to enhance the query by adding terms (concepts) that are strongly associated with the terms (concepts) already present in the query; and (2) for retrieving documents that are strongly associated with documents that match best with query terms (concepts). This particular case allows the retrieval of documents that have no matching concepts with the query, yet contain useful information which fulfils the information need at hand. However, external knowledge resources are always superior in terms of their rich structure, in-depth knowledge and reliability to the knowledge that can be gained through learning. Nevertheless, unavailability of external knowledge bases to cover the domain knowledge required for the collection of documents under consideration has severely restricted the use of external knowledge sources in IR. The worst case to mention is the retrieval of domain independent free text documents. In this case, a large number of knowledge bases would be needed. Such an effort is bound to integration difficulties and overheads caused by structural differences between the knowledge bases.

In addition, incorporation of background knowledge from existing knowledge bases into lattice representations of document contents requires derivation of more sophisticated knowledge (in the form of object-attribute pairs) from those existing knowledge sources. In the worst case, one might think of creating knowledge bases from scratch, with the necessary background knowledge in the form of formal concepts or object-attribute pairs.

Alternatively, mechanisms similar to those of keyword-based approaches could be employed in our case too, to enhance query and document representations with broader, narrower or related (formal) concepts. The task is however more complicated than in the case of keyword-based approaches, as we need to deal with object-attribute pairs rather than with simple keywords. The simplest form of an extension would be to resolve concept mismatches caused by synonymous words. A lexical database such as WordNet (*http://ww.cogsci.princeton.edu/~wn/*) could be used to verify whether synonyms of any

object or attribute of a query concept match with a document concept, whenever a match occurs only on one side between a query and a document concept pair. This process not only allows concept matches to take place inspite of the use of synonymous words in their representations, but also helps to avoide the number of keyword matches, causing the retrieval results to be based more on concept matches than on keyword matches.

10.3.2 Knowledge Exchange Between Documents

Another useful extension for the model would be to let the documents (retrieved and relevant to a given query) interact with each other and learn from each other by sharing their knowledge based on the co-occurrences of concepts. As a starting point, we can experiment by adding the significant concepts of each document to other documents that were judged by the user as relevant to the same query. However, this should be done in a very controlled manner, as it tends to make similar documents more and more similar and therefore might eventually lead to them all becoming identical. Also, we can look for the conceptswhich are common to all those similar documents and reward their weights. This will help in reinforcing certain important concepts which are otherwise not reinforced simply because the queries do not possess them.

10.3.3 Tools for Extraction of Better Concepts

Concept extraction from free text is an unsolved problem in NLU. A major problem with the identification of a term or a phrase as an object is the dual role a particular term or phrase could play depending on the purpose of its use. Keith Devlin [Devlin 1991] describes this as "*it's always a matter of purpose what parts of the world are individualised as objects*". Only a simple set of rules were used in our work to extract a basic set of objects and related attributes from natural language text to form concepts, and to create representations of documents and queries. Examination of the representations of documents reveals that certain formal concepts thus extracted were not accurate and meaningful in the human sense. This seriously affects the performance of the system as it

depends heavily on concept matching. Therefore, it would be interesting to see how the model performs when it uses more meaningful concepts similar to those that humans might use. As a starting point, one can manually index documents and queries with meaningful concepts and try the system on them. This is only possible with a small collection of documents. A complete solution for this problem requires the development of better methodologies and tools for automatic concept extraction - a breakthrough in NLU.

10.3.4 Efficient Way of Detecting Nodes to Match Between

Detecting which node pair(s) to match between lattice representations of queries and documents is a difficult and expensive process. The algorithm we developed (Figure 8.8) works by extracting object concepts (and attribute concepts) from both the query and document for each object/attribute present in the query representation. This works fairly well with smaller lattices, but may become more expensive (time consuming) with larger lattices, particularly if background knowledge is incorporated using external knowledge resources. Therefore, a better and more efficient way of detecting nodes to match between is desirable.

10.3.5 Application to an Agent-Based Distributed Environment

Given the prospective properties of our model to operate in a distributed environment (as detailed in Section 10.1.6) it will be interesting to see the potential of the model in such an environment by extending our implementation to an agent-based distributed environment. Distributed storage of documents is a likely feature in large sources of documents, including the World Wide Web, and processing them locally would be a useful alternative to maintaining large indexes. Such an approach has the potential to scale up the system's ability to operate on large collections of documents.

Appendix A - Concept Extraction Rules

A description of the concept extraction process and the data structures used were given in Section 8.2 and Section 8.3 without the detailed rules used. The rules used are detailed below under the same categories that they were described in Chapter 8. Note that, the LTChunk [http://www.ltg.ed.ac.uk/software/chunk/] software tool was used for tagging and chunking text, before concepts are extracted, and that this particular software uses the *Penn Treebank* tag set for tagging. Those tags are used below at various places in describing the rules. Also note that, the abbreviated notations NG, VG and OG are used to refer into noun, verb and other (ungrouped) groups/chunks respectively. Text in NGs are enclosed within double square-brackets (i.e. [[<*term1*> <*term2*> <*term3*>]]) and terms in VGs in double round-brackets (i.e. ((<*term1*> <*tem2*> <*term3*>))).

As already mentioned in Chapter 8, rules given below are not complete and may be not consistent by any means. Our objective was to extract a reasonably representative set of object-attribute pairs to create initial document/query representations/surrogates as a starting point. Different rules may extract the same concept (object-attribute pair) more than once or different concepts using the same terms from the same piece of text (sentence).

Note that, the examples given below, under each rule, were taken from documents in the Cranfield collection, and therefore they might look technical and unfamiliar.

Rules for Extracting Concepts based on Syntactic Structure within Noun Groups

1. If a possessive relationship between terms as indicated by **'s** or **s'** appears (with the tag *_POS*) in a noun group, a concept is formed with the term having the POS tag as the object and all the terms that follows it as attributes.

   e.g.: [[*Seattle_NNP's_POS business_NN district_NN*]] => {*Seattle*} → {*business district*}

After extracting the concept the *_POS* tag is removed together with *'s* or *s'* leaving the noun group as [[*Seattle_NNP business_NN district_NN*]]. This allows the other rules (given below) to extract some concepts (object-attribute pairs) if the updated/modified noun group satisfies those rules.

2. Noun groups with leading modifiers such as adjective and adverb terms (e.g. terms with the tags *_JJ, _VBN, VBG etc.*) (i.e. terms having tags other than *_NN*, *_NNS* and *_DT*) and with one or more trailing noun terms are processed to create concepts with all the trailing noun terms (i.e. headnoun of the noun group) as an object and each leading non-noun term (modifier) as an individual attribute of the object. This can be written in a syntactic form as a rule using the tags as given below. If the noun group in its syntactic form is of [[*JJ1 JJ2 NNP NN*]], a concept of the form {*NNP+NN*} → {*JJ1, JJ2*} is created.

   e.g: [[*The_DT famed_JJ London_NNP Bridge_NNP*]] creates:
   {*London Bridge*} → {*famed*}

3. Also noun groups with more than one trailing noun term are further processed to create concepts of the following form. If the noun group is of the form [[*JJ NN1 NN2 NN3*]] then concepts of the form {*NN3*} → {*NN2*}; {*NN3*} → {*NN1+NN2*} and {*NN2+NN3*} → {*NN1*} are created.

   e.g 1: [[*The_DT famed_JJ London_NNP Bridge_NNP*]] creates:
   {*Bridge*} → {*London*}

   e.g 2: [[*Seattle_NNP business_NN district_NN*]] creates {*district*} → {*business*} *and* {*district*} → {*Seattle business*} and {*business district*} → {*Seattle*}

4. Finally, for noun groups with only noun terms in it, a self-concept is created using all the noun terms in it as a single object as well as a single attribute, i.e. [[*NN1 NN2 NN3*]] => {*NN1+NN2+NN3*} → {*NN1+NN2+NN3*}. This is to take into account important phrases with only noun terms. Such phrases will not be taken into account (otherwise) if they are not connected to a second noun group by one of the connectors we use for concept extraction.

   e.g.: [[*London_NNP bridge_NNP*]] creates:
   {*London Bridge*} → {*London Bridge*}

Note that, this is pruned out if the phrase "London Bridge" is present as a single constitute element in another (other than a self-concept) concept.

Rules for Extracting Concepts based on Verb Groups

5. For a verb group with a single-word "be-verb" (i.e. is, are, was, were, has, had etc.) which lies in between two noun groups (i.e. in the form *NG1 VG NG2*), head nouns of the two noun groups are used for creating a concept, i.e. {HeadNoun(*NG1*)} → {HeadNoun(*NG2*)}. These are similar to *is-a* type of relationships.

   e.g. 1: [[*John_NN*]] [[***is***_*VBZ*]] [[*fat_NN*]] => {*John*} → { *fat*}.

   e.g. 2: [[*the_DT free_JJ stream_NN*]] ((***has***_*VBZ*)) [[*a_DT constant_JJ velocity_NN*]] creates: {*stream*} → {*velocity*}

6. For a verb group with a single-word non-be-verb which lies in between two noun groups (i.e. in the form *NG1 VG NG2*):
   If the verb begins with "***include***" then create:

   {HeadNoun(*NG1*)} → {HeadNoun(*NG2*)}

   e.g.: [[*the_DT reduced_VBN equations_NN*]] ((***includes***_*VBZ*)) [[*various_JJ terms_NNS*]] creates: {*equation*} → {*terms*}

   For all other single-term non-be verbs create:

   {<*Vterm*>} → {HeadNoun(*NG1*), HeadNoun(*NG2*)}

7. For a Verb group with a multi-word verb which lies in between two noun groups (i.e. in the form *NG1 VG NG2*):
   If the last word of the verb group is a form of a verb word (i.e. having a tag starting with VB, let's write this as Vterm _VB?) then create:

   {<*Vterm*>} → {HeadNoun(*NG1*), HeadNoun(*NG2*)}

   e.g.: [[*engineers_NNS*]] ((*to_TO meet_VB*)) [[*the_DT demands_NNS*]] creates: {*meet*} → {*engineers, demand*}

8. For verb groups that are not in between two noun groups (i.e. not of the form *NG VG NG*), but having previous and next noun groups at distance:
   If the verb group is having a single-word non-be-verb, then create:

   {<*Vterm*>} → {HeadNoun(*NG1*), HeadNoun(*NG2*)}

e.g. [[*the_DT propeller_NN axis_NN*]] ((***undergoes_VBZ***)) *pitching_VBZ* [[*vibrations_NNS*]] creates: {*undergo*} → {*propeller axis*, *vibration*}

If the verb group is having a multi-word verb and the last term of it is of a verb form (i.e. Vterm_VB?) then create:

{<*Vterm*>} → {HeadNoun(*NG1*), HeadNoun(*NG2*)}

e.g. [[*a_DT solution_NN*]] ((***is_VBZ found_VBN***)) *for_IN* [[*the_DT temperature_NN radient_NN*]] creates:
{*found*} → {*solution, temperature radient*}

If a verb group contains a form of "include" (with -ing, -ed forms etc.), then create {HeadNoun(PreviousNG)} → {HeadNoun(NextNG)}. There can be anything (any other groups, other than NGs) in between

e.g. [[*Appendixes_NNS*]] ((***include_VBP***)) [[*the_DT questionnaire_NN*]] ... creates:
{*appendix*} → {*questionnaire*}

Rules for Extracting Concepts based on Prepositional Words (Connectors) between Noun/Verb chunks

Prepositional words are not usually placed within noun or verb chunks (by the chunker LtChunk), but are left ungrouped. As already mentioned such ungrouped pieces of text are treated as belonging to a third kind of a group denoted by "Other Group" (*OG*). The rules given below are to extract concepts based on such prepositional words that belong to Other Groups in a chunked sentence.

Full set of prepositional words (connectors) used:

{by, at, as, after, into, for, of, in, on, to, through, over, among, between, toward, like, upon, along, from, under, within}

This set of connectors is grouped into five overlapping groups as given below.

Connector Set1 : {by, at, as, after, into, for, of, in, on, to, through, over, among, between, toward, like, upon, along}

Connector Set2 : {by, at, as, after, into, from, of, in, on, under, within, over, among, toward, like, upon}

Connector Set3 : {by, at, as, into, from, in, under, through, between, toward}

Connector Set4 : {by, at, as, into, through, within, among, toward, upon, along}

Connector Set5 : {by, at, as, into, through, toward, along}

Generic Rules:

9. If a starting group of a sentence is an *OG* and if it is of the form *<By+Vterm_VB?>*, then {*<Vterm>*} → {*NG1*}. i.e. if *NN1 = [[JJ NN1 NN2]]* then create:

{*<Vterm>*} → {*NN2*}, {<Vterm>} → {NN1+NN2} and {<Vterm>} → {JJ+NN1+NN2}

e.g.: ***By_IN applying_VBG*** [[*the_DT recommended_VBN scaling_NN*]] ... creates: {*apply*} → {*scaling*}

10. If OG contains only one-word connector and if it is in the middle of two NGs (i.e. of the form NG1 OG NG2) and also if the <connector word> ∈ Connector Set1 then create {HeadNouns(NG1)} → {HeadNouns(NG2)}. (Note that repeated connectors are handled by a separate specific rule (see Rules 18 & 19))

e.g.: [[*algorithm_NN*]] ***for_IN*** [[*analysing_NNP logical_NNP statements_NNP*]] creates: {*algorithm*} → {*analysing logical statements*}

11. If *OG* is a single-term connector and is of the form *NG1 VG OG NG2* and *VG* is a one-word verb and the verb is not a be-form word (i.e. not of the type *is*, *are*, *was*, ***were***, *has*, *had*, *will* , *would* etc) and the <*connector term*> ∈ *Connector Set2* then create {HeadNoun(*NG1*)} → {HeadNoun(*NG2*) , <*verbterm*>}

e.g. [[*this_DT method_NN*]] ((***differ***_*VBP*)) ***from_IN*** [[*the_DT experimental_JJ results_NNS*]] creates: {*method*} →{*results, differ*}

12. If *OG* is a single-term connector and is of the form *NG1 VG OG NG2* and *VG* is a one-word verb and the verb is not a be-form word (i.e. not of the type *is*, *are*, *was*, *were*, *has*, *had*, *will* , *would* etc) and the *connecter term* ∈ *Connector Set3* then:

{<*Vterm*>} → {HeadNoun(*NG1*), HeadNoun(*NG2*)}

e.g1: [[*this_DT method_NN*]] ((***differ***_*VBP*)) ***from_IN*** [[*the_DT experimental_JJ results_NNS*]] creates {*differ*} → {*method, results*}

e.g2: [[*air_NN*]] ((***decelerates***_*VBZ*)) ***through_IN*** [[*the_DT speed_NN*]] *of* ... creates: {*decelerate*} → {*air, speed*}

13. *NG1* ***such_JJ+as_IN*** *NG2 ,_, NG3,_, ... and/or_CC NGn* then create:

{HeadNoun(*NG1*)} → {HeadNoun(*NG2*), HeadNoun(*NG3*), ..., HeadNoun(*NGn*)}

e.g1: [[*external_JJ constraints_NNS*]] ***such_JJ as_IN*** [[*precedence_NN*]] ,_, [[*urgency_NN*]] ,_, etc. creates {*constraint*} →{ *precedence, urgency*}

e.g2: [[*individuals_NNS*]] ***such_JJ as_IN*** [[*researchers_NNS*]] ***and_CC*** [[*clinicians*]] ... creates {*individuals*} →{*researchers, clinicians*}

14. If *OG* is the two-word connector "***as of***" and lies between two noun groups (i.e. of the form *NG1 as_IN of_IN NG2*) then create:

{HeadNoun(*NG1*)} → {HeadNoun(*NG2*)}

e.g.: ... [[*the_DT Federal_NNP Government_NNP*]] ,_, ***as_IN of_IN***
[[*December_NNP*]] ... creates {*Federal Government*} → {*December*}

15. If *OG* contains two words and is of the form *NG1 OG NG2* and the 1st term of *OG* is "*toward*" or "*along*" and 2nd term is a verb term (*Vterm_VB?*) then create:

{HeadNoun(*NG1*)} → {HeadNoun(*NG2*), <*Vterm*>}

e.g1: [[*a_DT first_JJ step_NN*]] ***towards_IN*** *developing_VBG* [[*a_DT costing_VBG method_NN*]] creates {*step*} →{*method, developing*}
e.g2: [[*items_NNS*]] ***along_IN*** *interacting_VBG* [[*channels_NNS*]] creates {*items*} → {*channels, interacting*}

16. If *OG* contains two words and is of the form *NG1 OG NG2* and the 1st term of *OG* is a verb term (*Vterm_VB?*) and the 2nd term ∈ *Connector Set4* then create:

{HeadNoun(*NG1*)} → {HeadNoun(*NG2*), <*Vterm*>}

e.g.: [[*solutions_NNS*]] ***given_VBN by_IN*** [[*Wassermann_NN*]] creates:
{*solution*} → {*Wassermann, given*}

17. If *OG* contains two words and is of the form *NG1 OG NG2* and the 1st term of *OG* is a verb term (*Vterm_VB?*) and the 2nd term ∈ *Connector Set5* then create:

{<*Vterm*>} → {HeadNoun(*NG1*), HeadNoun(*NG2*)}

e.g.: [[the_DT solution_NN]] **measured**_VBN **along**_IN [[the_DT surface_NN]] creates {measure} → {solution, surface}

Specific Rules:

18. If three noun groups are connected by one of the connectors "*in*", "*on*", "*at*", "*by*" (i.e. of the form *NG1 CON NG2 CON NG3*) the create:

    {HeadNoun(*NG1*)} → {HeadNoun(*NG2*), HeadNoun(*NG3*)}

    e.g1: [[*a_DT flat_JJ plate_NN*]] ***in_IN*** [[*an_DT incompressible_JJ fluid_NN*]] ***of_IN*** [[*small_JJ viscosity_NN*]] creates {*plate*} → {*fluid, viscosity*}

    e.g2: [[slipstream_NN]] **at**_IN [[different_JJ angles_NNS]] **of**_IN [[attack_NN]] creates {slipstream} → {angle, attack}

19. If three noun groups are connected by the connector "of" (i.e. of the form *NG1* of_IN *NG2* of_IN *NG3*) then create :

    {HeadNoun(*NG1*)+"of"+HeadNoun(*NG2*)} → {HeadNoun(*NG3*)}

    {HeadNoun(*NG1*)} → {HeadNoun(*NG2*)+"of"+HeadNoun(*NG3*)}

    e.g.: [[*A_DT new_NNP method_NNP*]] ***of_IN*** [[*computation_NNP*]] ***of_IN*** [[*square_NNP Roots_NNP*]] … creates:

    {*new method of computation*}→ {*square root*} and

    {*new method*} → {*computation of square root*}

20. If two noun groups are connected by one of the connectors "*in*" ", "*on*", "*at*", "*by*" and to a third noun group by "*and*" or "*or*" (i.e. of the form *NG1 CON NG2 and/or_CC NG3*) then create:

    {HeadNoun(*NG1*)} → {HeadNoun(*NG2*), HeadNoun(*NG3*)}

    e.g.: [[*the_DT rotation_NN term_NN*]] ***on_IN*** [[*pressure_NN distribution_NN*]] ***and_CC*** [[*drag_NN*]] creates:

    {rotation term} → {pressure distribution, drag}

21. If two noun groups are connected by "***and***" (i.e. of the form *NG1 and_CC NG2*) then for any concept formed taking *NG1* as an object/attribute (by other rules), another concept is created with *NG2* with the same partner that *NG1* creates a concept.

    e.g.: [[*temperature_NN differences_NNS*]] ***between_IN*** [[*the_DT wall_NN*]] ***and_CC*** [[*the_DT free_JJ stream_NN*]] creates:

    {temperature difference} →{wall, stream}

22. If two noun groups are connected by "***or***" (i.e. of the form *NG1 or_CC NG2*) then for any concept created with one of the noun groups (by other rules), an alternative concept is created with the other.

    If a concept {NG1} → {NG3} is created then {NG2} → {NG3} is also created.

    e.g.: [[*the_DT immediate_JJ vicinity_NN*]] *of_IN* [[*the_DT wall_NN*]] ***or***_*CC* [[*the_DT laminator_JJ sublayer_NN*]] creates:
    {*vicinity*} → {*wall, sublayer*}

23. Remove any ***or***_*CC* or ***and***_*CC* present within a noun group in between modifiers (i.e between adjectives (*_JJ*), adverbs *_VBN*, *_VBG* etc.).

    e.g1: [[*traditional_JJ* ***and***_*CC current_JJ professional_JJ ideas_NNS*]] is turned into [[*traditional_JJ current_JJ professional_JJ ideas_NNS*]]
    e.g2: [[*annular_JJ* ***or***_*CC side_JJ air_NN intake_NNS*]] is changed to [[*annular_JJ side_JJ air_NN intake_NNS*]]

24. If ***or***_*CC* or ***and***_*CC* is present in a noun group (*NG*) but not between two modifiers, then it may be talking about two aspects of the same noun/topic/object. Therefore two concepts are created for each of those aspects whenever a concept is created with this noun group (*NG*). (This rule was not implemented.)

25. If an Other Group (*OG*) is of the form ***to***_*TO+<Vterm_VB?>* and lies between two noun groups (i.e. *NG1 to_TO+Vterm_VB? NG2*) then create :

    {*<Vterm>*} → {HeadNoun(*NG1*), HeadNoun(*NG2*)}

    e.g: [[*a_DT computer_NN*]] ***to***_*TO translate_VB* [[*simple_JJ algebraic_JJ formulas_NNS*] creates {*translate*} →{*computer, formula*}

26. If an Other Group (*OG*) which lies in between two noun groups and starts with "*based on*" or "*based upon*" or *based either upon*" (i.e. *NG1 based_VB? + on/upon/either upon_IN NG2*) then create:

    {HeadNoun(*NG1*)} → {HeadNoun(*NG2*)}

e.g.: [[*Relevance_NN figures_NNS*]] ***based_VBN upon_IN*** [[*the_DT response_NN*]] .... creates: {*relevance figures*} → { *response* }

27. If an Other Group (OG) is of the form ***toward_IN+Vterm_VB?*** and that it lies in between two noun groups (i.e. *NG1 toward_IN+Vterm_VB? NG2*) the create :

{HeadNoun(*NG1*)} → {<*Vterm*>, <*Vterm*> + HeadNoun(*NG2*)}

e.g.: [[*a_DT first_JJ step_NN*]] ***towards_IN*** *developing_VBG* [[*a_DT costing_VBG method_NN*]] creates:

{*step*} → {*developing, developing method*}

Appendix B – Concept Matching between Concept Lattices (A Transcript)

Given below is the detail of Query #51 and Document #528 of the Cranfield collection. This document is assessed as relevant to the query. The document context and the corresponding document concept lattice given below in Figure B.2 are the ones obtained after training the document collection for 20 epochs with all queries except query #51.

Note that, "O" followed by a serial number identifies an object, and "A" followed by a serial number identifies an attribute. The software tools "ConImp" and "Diagram" (*http://www.mathematik.tu-darmstadt.de/ags/ag1/Software/software_en.html*) both developed at Darmstadt, Germany were used for producing the lattice diagrams given below.

Cranfield Query #51

Objects and Related Attributes:

```
O1  : A2,A1
O2  : A3
O3  : A6,A4
O4  : A5
O5  : A9,A8,A7
O6  : A10,A9
O7  : A10
O8  : A11
O9  : A12
O10 : A13
```

Details of Objects and Attributes:

Objects	Attributes
O1: what	A1. what
O2: information	A2. information
O3. Boundary layer	A3. available
O4. layer	A4. Boundary layer
O5. body	A5. boundary
O6. revolution	A6. body
O7. Continuum flow	A7. very
O8. flow	A8. slender
O9. Curvature effect	A9. revolution
O10. effect	A10. Continuum flow
	A11. continuum
	A12. transverse
	A13. curvature

Concept ID	Concept
C1	<all objects> → {}
C2	{Effect} → {curvature}
C3	{Curvature effect} → {transverse}
C4	{flow} → {continuum}
C5	{Revolution,continuum flow} → {continuum flow}
C6	{Body, revolution} → {revolution}
C7	{revolution} → {revolution, continuum flow}
C8	{body} → {very,slender,revolution}
C9	{layer} → {boundary}
C10	{boundary layer} → {body, boundarylayer}
C11	{information} → {available}
C12	{what} → {what, information}
C13	{} → <all attributes>

Table B.1 : Objects, Attributes and Concepts Extracted from Query#51 of Cranfield

Query Concept Lattice

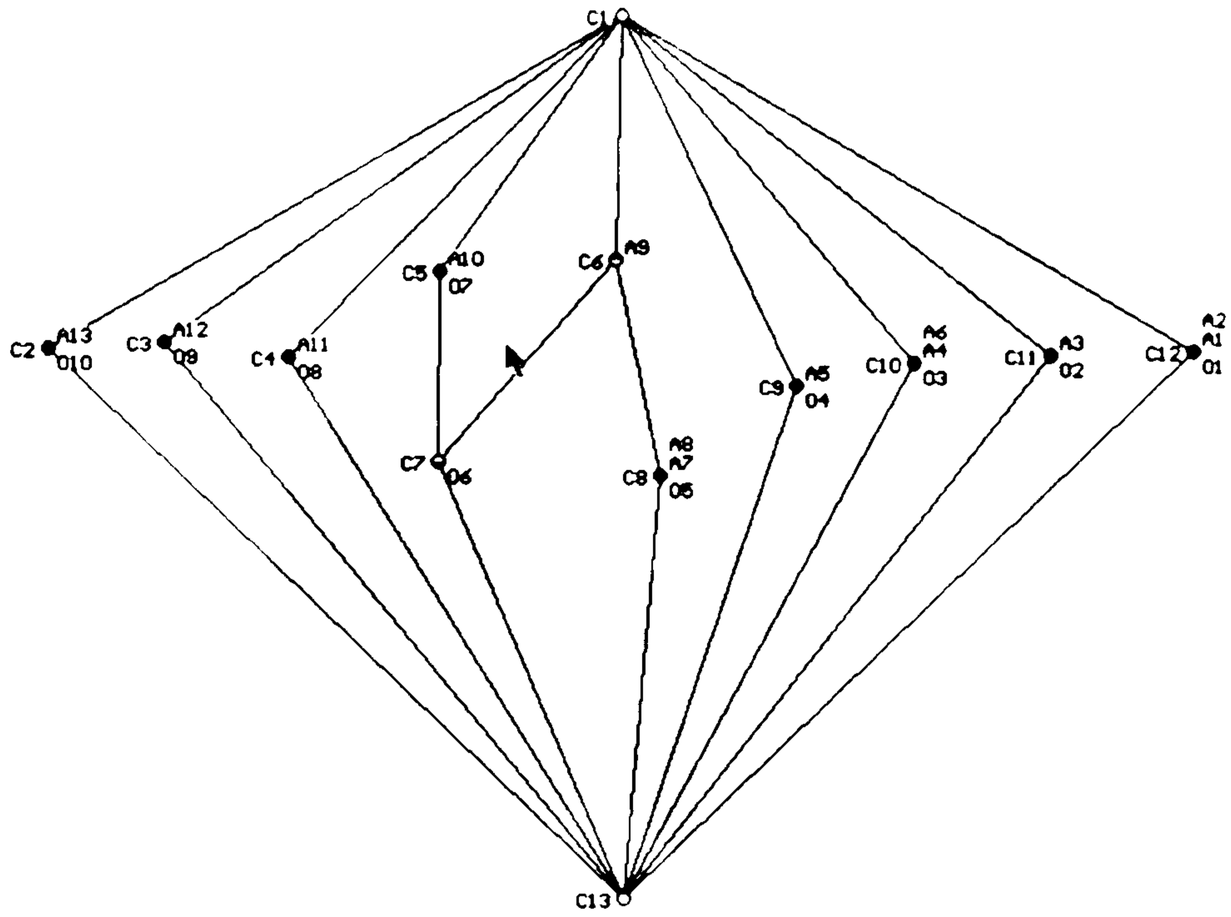


Figure B.1 : Concept Lattice of the Query#51 of Cranfield Collection

Cranfield Document #528

Given below are the objects, their attributes and the weights of unit-concepts (i.e. weights between object-attribute pairs) of Document#528. Note that O1:A3 2.5; A1 2.5 means Object O1 possesses Attributes A3 and A1, and the weights of the unit concepts, {O1} → {A3} and {O1} → {A1} are 2.5 each.

O1 : A3 2.5; A1 2.5
O2 : A54 4.5077567; A43 2.6; A46 2.6; A2 2.5; A45 4.7884655
O3 : A6 2.5; A17 2.0286775; A4 2.1744843
O4 : A5 1.9842464
O5 : A8 4.4555316; A7 4.4786963
O6 : A8 2.5
O7 : A9 2.5
O8 : A10 1.9244329
O9 : A11 2.5
O10 : A29 2.5; A12 2.5
O11 : A28 2.5; A15 2.5; A14 2.5; A13 2.5
O12 : A14 2.5; A13 2.5
O13 : A14 2.5
O14 : A16 2.5
O15 : A18 2.5
O16 : A20 2.5; A19 2.5
O17 : A1 2.5
O18 : A2 2.5
O19 : A21 2.5
O20 : A22 2.5; A21 2.5
O21 : A22 2.5; A28 2.5; A26 2.5; A25 2.5; A24 2.5; A23 2.5
O22 : A27 2.5; A25 2.5; A2 2.5
O23 : A26 2.5
O24 : A28 2.5
O25 : A30 2.5
O26 : A31 2.5
O27 : A32 2.5
O28 : A33 2.5; A32 2.5
O29 : A33 2.5
O30 : A35 2.5; A34 2.5
O31 : A37 2.5; A36 2.5
O32 : A6 2.5
O33 : A37 2.5
O34 : A38 2.5
O35 : A6 2.3866334
O36 : A39 2.5
O37 : A40 2.5; A28 2.5
O38 : A40 2.5
O39 : A42 2.5; A41 2.5
O40 : A42 2.5
O41 : A44 2.6; A43 3.2063825; A52 4.5077567
O42 : A47 4.7884655
O43 : A48 4.5077567; A47 2.6
O44 : A49 4.7884655; A47 2.6
O45 : A55 4.7884655; A50 2.6; A56 4.5077567
O46 : A49 2.6; A5 4.5077567
O47 : A51 4.5077567; A6 4.5077567; A5 2.6
O48 : A6 4.5077567; A5 4.2040606
O49 : A53 4.490081
O50 : A55 2.6
O51 : A51 2.6; A3 2.060822
O52 : A5 2.060822
O53 : A2 2.6

Details of Objects and Attributes

Object ID	Objects	Orig./ Updtd	Keyword Weight
O1	Slip effect	1	2.6
O2	Effect	1	1.3298341
O3	Boundary layer	1	2.4816785
O4	layer	1	2.519406
O5	Body	1	2.0646727
O6	revolution	1	2.4823744
O7	Pressure gradient	1	2.6
O8	gradient	1	2.6
O9	Reference 1	1	2.6
O10	1	1	2.6
O11	analysis	1	1.8698725
O12	given	1	2.4823744
O13	probstein	1	2.6
O14	elliott	1	2.6
O15	curvature	1	2.6
O16	extended	1	2.6
O17	Slip flow.	1	2.6
O18	flow.	1	2.6
O19	extension	1	2.6
O20	based	1	2.6
O21	expansion	1	2.5404787
O22	parameter	1	2.4823744
O23	Slip parameter	1	2.6
O24	ref. 1	1	2.6
O25	Wall temperature	1	2.6
O26	temperature	1	1.8698725
O27	e	1	2.6
O28	varies	1	2.4823744
O29	x	1	2.6
O30	dependence	1	2.6
O31	Body radius	1	2.6
O32	radius	1	2.6
O33	x.	1	2.6
O34	Body shape	1	2.6
O35	shape	1	2.3162313
O36	note	1	2.6
O37	re-examined	1	2.6
O38	account	1	2.6
O39	variation	1	2.2289894
O40	e.	1	2.6
O41	what	2	0.7851298
O42	Effect of amount	2	2.6
O43	amount	2	2.5404787
O44	Gas rarefaction	2	2.6
O45	rarefaction	2	2.6959999
O46	characteristic	2	1.9873333
O47	boundary	2	1.4164754
O48	layers	2	2.6
O49	information	2	2.1125894
O50	Boundary layer flow	2	2.5364094
O51	flow	2	0.837419
O52	Layer flow	2	2.6
O53	slip	2	3.6044445

Attrib. ID	Attribute	Orig./ Updtd	Keyword Weight
A1	first-order	1	2.6
A2	slip	1	3.6044445
A3	Boundary layer	1	2.4816785
A4	laminar	1	1.9423987
A5	boundary	1	1.4164754
A6	body	1	2.0646727
A7	slender	1	2.3702834
A8	revolution	1	2.4823744
A9	zero	1	2.3162313
A10	pressure	1	1.2197144
A11	Reference 1	1	2.6
A12	reference	1	2.4823744
A13	analysis	1	1.8698725
A14	probstein	1	2.6
A15	given	1	2.4823744
A16	elliott	1	2.6
A17	compressible	1	2.2119575
A18	transverse	1	2.4823744
A19	curvature	1	2.6
A20	Slip flow.	1	2.6
A21	extension	1	2.6
A22	expansion	1	2.5404787
A23	double	1	2.6
A24	asymptotic	1	2.4823744
A25	parameter	1	2.4823744
A26	Slip parameter	1	2.6
A27	transverse-curvature	1	2.6
A28	ref. 1	1	2.6
A29	ref.	1	2.6
A30	constant	1	2.6
A31	wall	1	2.4823744
A32	e	1	2.6
A33	x	1	2.6
A34	dependence	1	2.6
A35	Body radius	1	2.6
A36	local	1	1.9572567
A37	x.	1	2.6
A38	arbitrary	1	2.2809136
A39	present	1	2.6
A40	account	1	2.6
A41	variation	1	2.2289894
A42	e.	1	2.6
A43	effect	2	1.3298341
A44	what	2	0.7851298
A45	Amount of gas rarefaction	2	2.6
A46	amount	2	2.5404787
A47	Gas rarefaction	2	2.6
A48	small	2	2.1616747
A49	characteristic	2	1.9873333
A50	gas	2	2.2634664
A51	layer	2	2.519406
A52	information	2	2.1125894
A53	available	2	2.4823744
A54	rarefaction	2	2.6959999
A55	Boundary layer flow	2	2.5364094
A56	slight	2	2.6

Table B.2 : Objects and Attributes Extracted from Document#528 of Cranfield

Note that, the value "1" in the Orig./Updtd. columns in Table B.2 indicates objects/attributes originally extracted from the source document, and "2" indicates the ones that were added during learning.

Document Concept Lattice

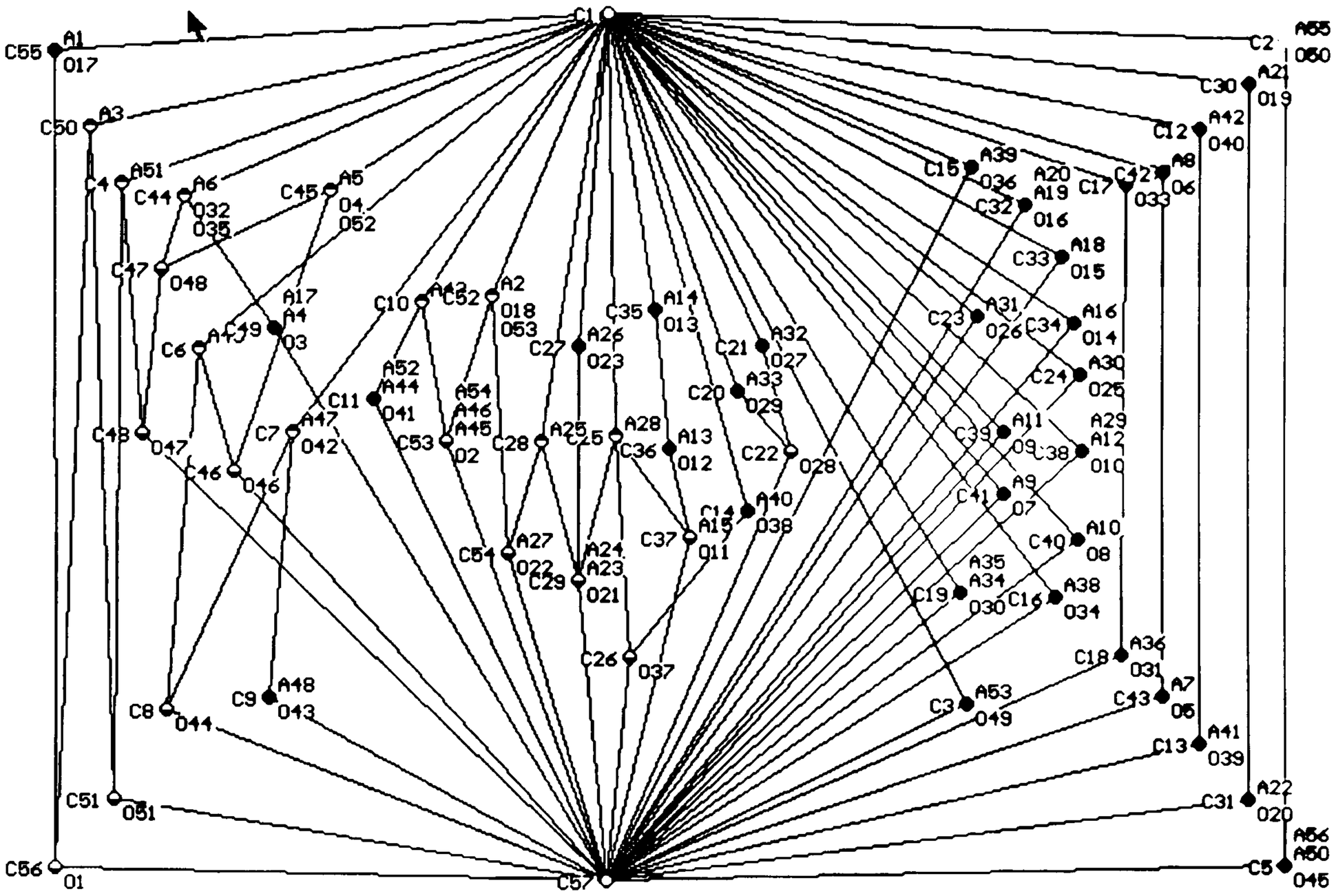


Figure B.2 : Concept Lattice of the Document#528 of Cranfield Collection

Candidate Keywords/Keyphrases (i.e. Keywords/Keyphrases Common to the Query and the Document

what	layer	revolution
information	boundary	flow
available	body	transverse
boundary layer	slender	effect
		curvature

Candidate Concept Pairs Extracted to Match Between the Query and the Document

1. Qry Cnpt : {O1} -> {A1, A2} WITH Doc Cnpt : {O41} -> {A43, A44, A52}
2. Qry Cnpt : {O2} -> {A3} WITH Doc Cnpt : {O49} -> {A53}
3. Qry Cnpt : {O3} -> {A4, A6} WITH Doc Cnpt : {O3} -> {A4, A6, A17}
4. Qry Cnpt : {O4} -> {A5} WITH Doc Cnpt : {O4, O46, O47, O48, O52} -> {A5}
5. Qry Cnpt : {O4} -> {A5} WITH Doc Cnpt : {O47} -> {A5, A6, A51}
6. Qry Cnpt : {O5} -> {A7, A8, A9} WITH Doc Cnpt : {O5} -> {A7, A8}
7. Qry Cnpt : {O6} -> {A9,A10} WITH Doc Cnpt : {O5,O6} -> {A8}
8. Qry Cnpt : {O8} -> {A11} WITH Doc Cnpt : {O51} -> {A3, A51}
9. Qry Cnpt : {O9} -> {A12} WITH Doc Cnpt : {O15} -> {A18}
10. Qry Cnpt : {O10} -> {A13} WITH Doc Cnpt : {O2} -> {A2, A43, A45, A46, A54}
11. Qry Cnpt : {O10} -> {A13} WITH Doc Cnpt : {O15} -> {A18}
12. Qry Cnpt : {O10} -> {A13} WITH Doc Cnpt : {O16} -> {A19, A20}

The following are the same candidate concept pairs, but written with actual textual labels of objects and attributes instead of their IDs. Concept/Node ID numbers used in the concept lattice diagrams are also indicated (circled). Matching Object-Attribute pairs are also indicated.

Query Nodes/Concepts **Document Nodes/Concepts**

1. C12 {what} → {what, information} WITH C11 {what} → {effect, what, information}

2. C11 {information} → {available} WITH C3 {information} → {available}

3. C10 {boundary layer} → {body, boundary layer} WITH C49 {boundary layer} → {laminar, body, compressible}

4. C9 {layer} → {boundary} WITH C45 {layer, layers, characteristic, boundary, layer flow} → {boundary}

5. C9 {layer} → {boundary} WITH C48 {boundary} → {boundary, body, layer}

6. C8 {body} → {very, slender, revolution} WITH C43 {body} → {slender, revolution}

7. C7 {revolution} → {revolution, continuum flow} WITH C42 {body, revolution} → {revolution}

8. C4 {flow} → {continuum} WITH C51 {flow} → {boundary layer, layer}

9. C3 {curvature effect} → {transverse} WITH C33 {curvature} → {transverse}

10. C2 {effect} → {curvature} WITH C53 {effect} → {slip, effect, amount of gas rarefaction, amount, rarefaction}

11. C2 {effect} → {curvature} WITH C33 {curvature} → {transverse}

12. C2 {effect} → {curvature} WITH C32 {extended} → {curvature, slipflow}

Query Document Node Matching

Figure B.3 given below illustrates (in blue lines) the same, i.e. nodes that are compared between the query and document lattices in order to compute a similarity score.

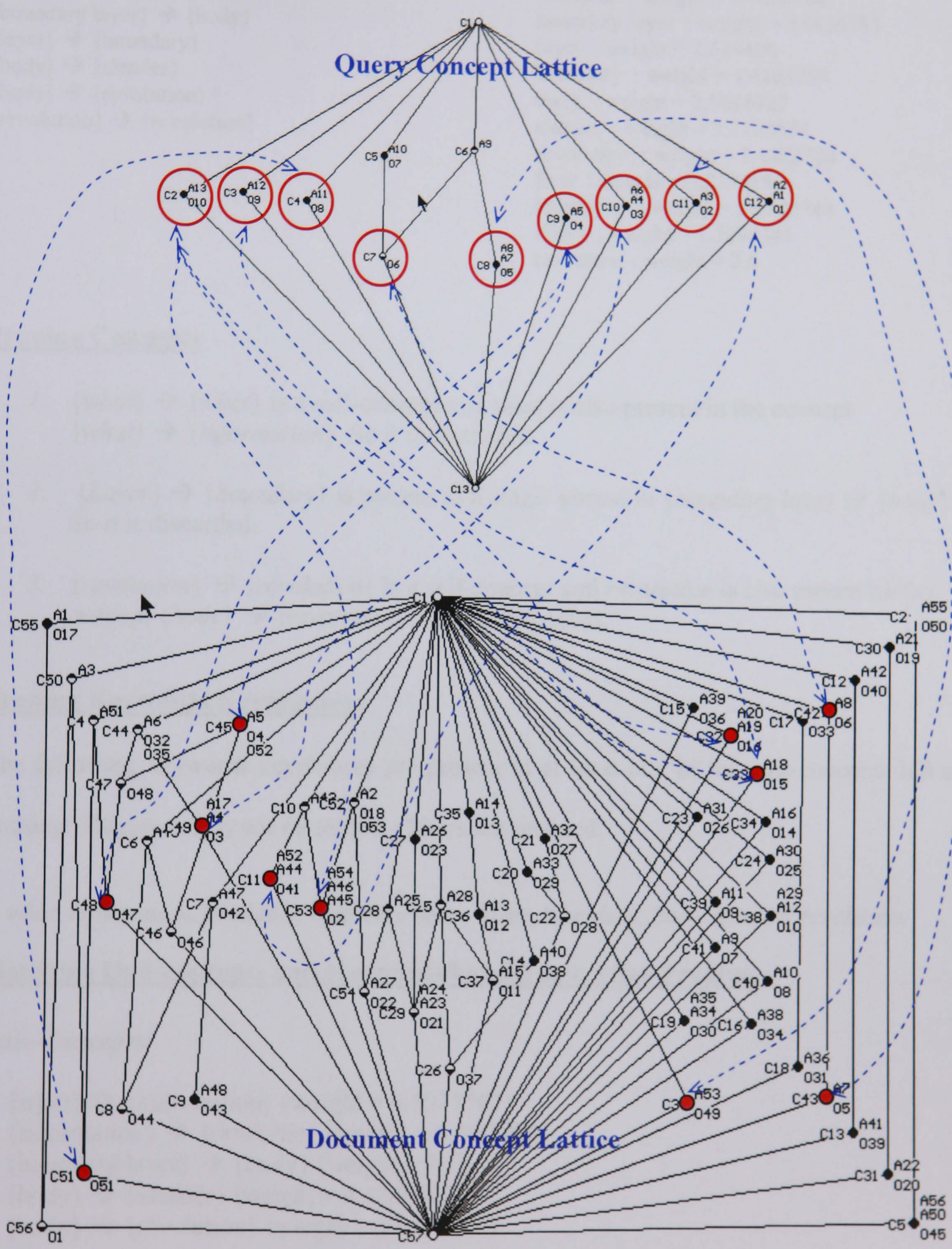


Figure B.3 : Concept Matching between Query#51 and Document#528 of Cranfield Collection

Following are the matching object-attribute pairs (unit-concepts) and keywords given by these candidate concept/node pairs. Note that they are subject to pruning to remove duplicates.

Matching Unit-concepts and Keywords Before Pruning

Unit-Concepts

{what} → {what}
{what} → {information}
{information} → {available}
{boundary layer} → {body}
{layer} → {boundary}
{body} → {slender}
{body} → {revolution}
{revolution} → {revolution}

Keywords/Keyphrases

what : weight = 0.7851298
information : weight = 2.1125894
available : weight = 2.4823744
boundary layer : weight = 2.4816785
layer : weight = 2.519406
boundary : weight = 1.4164754
body : weight = 2.0646727
slender : weight = 2.3702834
revolution : weight = 2.4823744
flow : weight = 0.837419
transverse : weight = 2.4823744
effect : weight = 1.3298341
curvature : weight = 2.6

Pruning Concepts

1. {*what*} → {*what*} is a self-concept and *what* is also present in the concept {*what*} → {*information*}. So it is discarded.

2. {*Layer*} → {*boundary*} is present as a single phrase in {*boundary laye*} → {*body*}. So it is discarded.

3. {*revolution*} → {*revolution*} is a self-concept and *revolution* is also present in the concept {*body*} → {*revolution*}. So it is discarded.

Pruning Keywords/Keyphrases

The following keywords/keyphrases are present in at least one of the unit-concepts left after pruning. Therefore they are discarded. Others are retained.

what, information, available, boundary layer, layer, boundary, body, slender, revolution.

Matching Unit-concepts and Keywords/keyphrases After Pruning

Unit-Concepts:

1. {what} → {information} (weight = 4.5077567)
2. {information} → {available} (weight = 4.490081)
3. {boundary layer} → {body} (weight = 2.5)
4. {body} → {slender} (weight = 4.4786963)
5. {body} → {revolution} (weight = 4.4555316)

Keywords/Keyphrases:

6. flow (weight = 0.837419)
7. transverse (weight = 2.4823744)
8. effect (weight = 1.3298341)
9. curvature (weight = 2.6)

Similarity Score (RSV) Calculation

The weights of the remaining (i.e. left after pruning) unit-concepts and keywords/keyphrases are used for calculating a similarity score for the query-document pair. Recall that the weights are further weighted for their informative-ness. The re-weighting criteria and the weighting factors are given below.

1. Unit-concepts in which at least one of its constituent (object or attribute) is a multi-term entity are multiplied by 2.5 (weighting factor).

2. Single-term unit-concepts are multiplied by 2.0.

3. Multi-term keyword/keypharse is multiplied by 1.5

4. Single-term keyword is left un-weighted (weighting factor is 1)

Similarity measure = $(4.5077567 \times 2) + (4.490081 \times 2) + (2.5 \times 2.5) + (4.4786963 \times 2)$
$+ (4.4555316 \times 2) + 0.837419 + 2.4823744 + 1.3298341 + 2.6$
$= 49.363758$

Threshold Calculation :

Threshold = Base Threshold × Total number of (Query) unit-concepts extracted as candidates to match/compare with the document.

$= 1.3 \times 14$

$= 18.2$

Since the similarity measure (RSV) of Document#528 to Query#51 (49.363758) exceeds the threshold values (18.2), computed for the same query-document pair, Document#528 is retrieved for the Query#51. It indeed is a document that is assessed as relevant to the query in the relevance assessments list (qrels) of the collection (Cranfield).

References

[**Abedin & Ahson 1993**] Abedin M.J and Ahson S.I., *A New Coding Procedure for Bidirectional Associative Memory*; Journal of Microcomputer Applications Vol.16, 1993, pp189-195

[**Allen 1988**] Allen J., *Natural Language Understanding;* Benjamin/Cummings Publishing Company, 1988

[**Amari 1972**] Amari S., *Learning Patterns and Pattern Sequences by Self-Organizing Nets of Thresholding Elements*; IEEE Transactions on Computers, Vol.21(11), November 1972, pp.1197-1206

[**Amati & Crestani 1999**] Amati G. and Crestani F., *Probabilistic Learning for Selective Dissemination of Information;* Information Processing and Management, 35(5), 1999, pp.633-654

[**Appel et al. 1988**] Appel R.D., Komorowski H.J., Barr C.E., Greenes R.A., *Intelligent Focusing in Knowledge Indexing and Retrieval - The Relatedness Tool;* Proceedings of the 12th Annual Symposium on Computer Applications in Medical Care, November 1988, pp.152-157

[**Arampatzis et al. 1998**] Arampatzis A.T., Tsoris T., Koster C.H.A. and Van Der Weide T.H.P., *Phase-Based Information Retrieval;* Information Processing and Management Vol.34, No.6, 1998, pp.693-707

[**Baldwin 1896**] : Baldwin M.J., *A New Factor in Evolution,* The American Naturalist, pp.441-451, 30th June 1896 (also available on http://members.aol.com/jorolat/baldwin2.html). This paper is reprinted in *Adaptive Individuals in Evolving Populations: Models and Algorithms*, edited by R. K. Belew and M. Mitchell (SFI Studies in the Sciences of Complexity, Proc. Vol. XXVI, Addison-Wesley, Reading, MA, 1996).

[**Bavarian 1988**] Bavarian B., *Introduction to Neural Networks for Intelligent Control*; IEEE Control Systems Magazine, April 1988, pp.3-7

[**Bayer et al. 1996**] Bayer T., Renz I., Stein M. and Kressel U., *Domain and Language Independent Feature Extraction for Statistical Text Categorization*; In Proceedings of the Workshop on Language Engineering for Document Analysis and Recognition, Sussex, United Kingdom

[**Becker 1991**] Becker S., *Unsupervised Learning Procedures for Neural Networks*; International Journal of Neural Systems Vol.2, Nos. 1 & 2, 1991, pp.17-33

[**Becker et al. 2002**] Becker P., Hereth J. and Stumme G., *ToscanaJ - An Open Source Tool for Qualitative Data Analysis;* In Workshop proceedings of FCA KDD workshop of the 15th European Conference on Artificial Intelligence (ECAI'02), July 21-26 2002, Lyon, France (2002)

[**Bein & Smolensky 1988**] Bein J. and Smolensky P., *Application of the Interactive Activation Model to Document Retrieval*; Technical Report CU-CS-405-88, Dept. Computer Science, University Colorado, Boulder, CO 80309

[**Belew 2000**] Belew R.K., *Finding out About: A Cognitive Perspective on Search Engine Technology and the WWW;* Cambridge University Press, ISBN 0-521-63028-2

[**Bělohlávek 2000**] Bělohlávek R., *Representation of Concept Lattices by Bidirectional Associative Memories*; Neural Computation Vol.12, No.10, October 2000, pp2279-2290

[**Berry et al. 1995**] Berry M.W., Dumais S.T. and Letsche T.A., *Computational Methods for Intelligent Information Access;* In Proceedings of Supercomputing'95, December 3-8, 1995, San Diago, Canada

[**Bharat & Henzinger 1998**] Bharat K., Henzinger M., *Improved Algorithms for Topic Distillation in Hyperlinked Environments;* In Proceedings of ACM-SIGIR'98, Melbourne (Australia), August 1998, pp.104-111

[**Boons 2000**] Boon, G.N., *Extreme Dimensionality Reduction for Text Learning: Cluster Generated Feature Spaces;* PhD Thesis, Georgia Institute of Technology, August 2000

[**Bordogna & Pasi 2000**] Bordogna G. and Pasi G., *Modeling Vagueness in Information Retrieval*; In Lectures on Information Retrieval, Agosti,M., Crestani, F. and Pasi, G. (Eds.), Third European Summer-School, ESSIR 2000, Springer-Verlag, ISABN 3-540-41933-0, ISSN 0302-9743.

[**Borlund 2003**] Borlund P., *The IIR Evaluation Model: a Framework for Evaluation of Interactive Information Retrieval Systems;* Information Research, 8(3), April 2003 paper no.152

[**Boughanem et al. 2000**] Boughanem M., Chrisment C., Mothe J., Soule-Dupuy C. and Tamine L., *Connectionist and Genetic Approaches for Information Retrieval*; In "Soft Computing in Information Retrieval" by Fabio Crestani & Gabriella Passi (Eds.), ISBN 3-7908-1299-4

[**Brin & Page 1998**] Brin S., and Page L., *The Anatomy of a Large-scale Hypertextual Web Search Engine;* In proceedings of the 7th World Wide Web Conference (WWW7), Brisbane(Australia), April 1998, pp.107-117

[**Burmeister 1998**] Burmeister P., *Formal Concept Analysis with ConImp: Introduction to the basic features*; Technical Report, Technische Hochschule Darmstadt, Germany, 1998

[**Callan et al. 1992**] Calla J.P, Croft W.B. and Harding S.M., *The INQUERY Retrieval System*; Third International Conference on Databases and Expert Systems Applications, Vol.11(2), pp.78-83, Springer-Verlag

[**Carpenter & Grossberg 1987**] Carpenter G.A. and Grossberg S., *ART 2: Self-organization of Stable Pattern Recognition Codes for Analog Input Patterns;* Applied Optics, 1987, Vol.26, pp.4919 - 4930

[**Carpineto & Romano 1996**] Carpineto C. and Romano G., *A Lattice Conceptual Clustering System and Its Application to Browsing Retrieval*; Machine Learning 1996, Vol.24(2), pp.95-122

[**Carpineto & Romano 1998**] Carpineto C. and Romano G., *Effective Reformulation of Boolean Queries with Concept Lattices*; In Proceedings of the 3rd International Conference on Flexible Query Answering Systems, 1998, Roskilde DK, pp.83-94

[**Carpineto & Romano 2000**] Carpineto C. and Romano G., *Order-theoretical Ranking*; JASIS, 2000, Vol.51, No.7, pp.587-601

[**Chen 1995**] Chen H., *Machine Learning for Information Retrieval: Neural Networks, Symbolic Learning and Genetic Algorithms*; Journal of the American Society for Information Science, April 1995, Vol.46, No.3, pp.194-216

[**Chen & Dhar 1991**] Chen H. and Dhar V., *Cognitive Process as a Basis for Intelligent Retrieval System Design;* Information Processing & Management, 1991, Vol.27, No.5, pp.405-432

[**Chen & Kim 1994**] Chen H. and Kim J., *GANNET: A Machine Learning Approach to Document Retrieval;* Journal of Management Information Systems, Armonk, Winter 1994/1995

[**Chen et al. 1993**] Chen H., Lynch K.J., Basu K. and Dorbin T., *Generating, Integrating and Activating Thesauri for Concept-Based Document Retrieval;* IEEE, April 1993

[**Chen et al. 1999**] Chen Z., Meng X., Fox R. and Folwer R., *Applications of Multilayer Feedforward Networks on WWW Document Search;* IJCNN'99

[**Chen et al. 2000**] Chen H., Ramsey M. and Li P., *The Java Search Agent Workshop*; In Soft Computing in Information Retrieval (Eds.) Fabio Crestani & Gabriella Passi, ISBN 3-7908-1299-4, pp.122-140

[**Cleverdon 1991**] Cleverdon C.W., *The Significance of the Cranfield Tests on Index Languages;* In Proceedings of the 14th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Chicago, IL, October 13-16, 1991, pp.3-12

[**Cleverdon et al. 1966**] Cleverdon C.W., Don C.W., Mills J. and Keen E.M., *Factors Determining the Performance of Indexing Systems;* Cranfield, UK: Aslib Cranfield Research Project, College of Aeronautics; 1966. 2 volumes (Volume 1: Design; Volume 2: Results). LC:67-81732;OCLC:3911240, http://www.itl.nist.gov/iad/894.02/projects/irlib/pubs/cranv1p1/cranv1p1_index/cranv1p1.html and http://www-nlpir.nist.gov/projects/irlib/pubs/cranv2/cranv2_index/cranv2.html

[**Cohen & Kjeldsen 1987**] Kjeldsen R. and Cohen P.R., *The Evolution and Performance of the GRANT System*; IEEE Expert, Summer 197, Vol.2 No.2, pp.73-79

[Cole & Eklund 1993] Cole R. and Eklund P.W., *Scalability of Formal Concept Analysis*; Computational Intelligence, 1993, Vol.2, No.5

[Cole & Eklund 1996] Cole R.J. and Eklund P.K., *Application of Formal Concept Analysis to Information Retrieval using a Hierarchically Structured Thesaurus*; International Conference on Conceptual Graphs, ICCS '96, Sydney, 1996, pp.1-12

[Cole & Eklund 1999] Cole R.J. and Eklund P.W., *Analysing an Email Collection Using Formal Concept Analysis;* European Conference on Knowledge and Data Discovery, PKDD'99, pp.309-315, LNAI 1704, 1999

[Cole & Eklund 2001] Cole R.J. and Eklund P.K., *Structured Ontology and IR for Email Search and Discovery;* Australian Document Computing Symposium (ADCS01), University of Sydney, Basser Department of Computer Science, 2001, pp.5-9,

[Cole et al. 1997] Cole R., Eklund P. and Groh B., *Dealing with Large Contexts in Formal Concept Analysis; A Case Study Using Medical Texts*; In Proceedings of the Second International Symposium on Knowledge Retrieval, Use and Storage for Efficiency, Vancouver, 1997

[Cole et al. 2000] Cole R., Eklund P. and Stumme G., *CEM-A Program for Visualization and Discovery in Email*; In D.A. Zighed, J. Komorowski, J. Zytkow (Eds), Proceedings of PKDD 2000

[Cole et al. 2003] Cole R., Eklund P. and Stumme G., *Document Retrieval for Email Search and Discovery using Formal Concept Analysis;* Journal of Applied Artificial Intelligence (AAI), Vol.17, No.3

[Crestani 1993] : Crestani, F. (1993) *Learning Strategies for an Adaptive Information Retrieval System using Neural Networks*, In Proceedings of the IEEE International Conference on Neural Networks (ICNN'93), San Francisco. California

[Crestani 1994] Crestani F., *Domain Knowledge Acquisition for Information Retrieval using Neural Networks*; International Journal of Applied Expert Systems, 1994, Vol.2(2), pp.101-115

[Crestani 1995] Crestani F., *Implementation and Evaluation of a Relevance feedback device based on Neural Networks*; International workshop on Artificial Neural Networks, Mira J. and Cabastany J. (Ed.), Springer Verlag, pp.597-604

[Crestani 1997] Crestani F., *Applications of Spreading Activation Techniques in Information Retrieval;* Artificial Intelligence Review, 1997, Vol.11(6), pp.453-482

[Crestani & Lalmas 2000] Crestani F. and Lalmas M., *Logic and Uncertainty in Information Retrieval*; In "Lectures on Information Retrieval" Agosti M., Crestani F. and Pasi G. (Eds.) Third European Summer-School, ESSIR 2000, Springer-Verlag ISABN 3-540-41933-0, ISSN 0302-9743

[**Crestani & Pasi 1999**] Crestani F. and Pasi G., *Soft Information Retrieval: Applications of Fuzzy Set Theory and Neural Networks*; In N. Kasabov and R. Kozma, editors, *Neuro-Fuzzy Techniques for Intelligent Information Systems*, Physica Verlag (Springer Verlag), Heidelberg, Germany, 1999, pp.287-315

[**Crestani & Pasi 2000**] Crestani F. and Pasi G., editors, *Soft Computing in Information Retrieval: Techniques and Applications;* Physica Verlag (Springer Verlag), Heidelberg Germany, 2000

[**Crestani & Rijsbergen 1994**] Crestani F. and van Rijsbergen C.J., *Comparing Probabilistic and Neural Relevance Feedback in an Interactive Information Retrieval System;* In Proceedings of the 1994 IEEE International Conference on Neural Networks, Orlando, Florida, USA, June 1994, pp.3426-3430

[**Crestani & Rijsbergen 1997**] Crestani F. and van Rijsbergen C.J., *A Model for Adaptive Information Retrieval*; JIIS, 1997, Vol.8(1), pp.29-56

[**Crestani et al. 1998**] Crestani F., Lalmas M. van Rijsbergen C.J. and Campbell I., *Is this Document Relevant ? ...Probably;* A survey of probabilistic models in Information Retrieval, ACM Computing Surveys, 1998, Vol.30(4), pp.528-552

[**Croft 1980**] Croft W.B., *A Model of Cluster Searching Based on Classification;* Information Systems, 1980, Vol.5, No.3, pp.189-195

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

[**Croft 1993**] Croft W.B., *Knowledge-Based and Statistical Approaches to Text Retrieval;* IEEE Expert, April 1993, pp.8-12

[**Croft 1995**] Croft B., *Effective Text Retrieval Based on Combining Evidence from the Corpus and Users*; IEEE Expert/Intelligent Systems & their Application

[**Croft & Thompson 1987**] Croft W. B. and Thompson R.H., *I3R: A New Approach to the Design of Document Retrieval System;* Journal of the American Society for Information Science, 1987, Vol.38, No.6, pp.389-404

[**Croft & Turtle 1992**] Croft W.B. and Turtle H.R., *Text Retrieval and Inference;* In "Text-Based Intelligent Systems" edited by Paul S. Jacob, Lawrence Erlbaum Associates Publishers, 1992, pp.3426-3430

[**Croft et al. 1989**] Croft W, Lucia T, Crigean J. and Willet P., *Retrieving Documents by Plausible Inference: An Experimental Study*; Information Processing and Management, 1989, Vol.25(6), pp.599-614

[**Davis et al. 1993**] Davis R., Schrobe H. and Szolovits P., *What is Knowledge Representation;* AI Magazine, 1993, Vol.14(1), pp.17-33

[**de Vries 2000**] de Vries D.H.A.P., *Relating the New Language Models of Information Retrieval to the Traditional Retrieval Models* ; TR-CTIT-00-09, June 2000

[**Dean & Henzinger 1999**] Dean J. and Henzinger M.R., *Finding Related Pages in the World Wide Web*; In Proceedings of the Eighth International World-Wide Web Conference (WWW9), 1999

[**Deerwester et al. 1990**] Deerwester S., Dumais S.T., Furnas G.W., Landauer T.K. and Harshman R., *Indexing by Latent Semantic Analysis;* Journal of the American Society for Information Science, 1990, Vol.41(6), pp.391-407

[**Devlin 1991**] Devlin K., *Logic and Information*; Cambridge University Press, 1991, ISBN 0-521-41030-4

[**Dumais et al. 1991**] Dumais S.T., *Enhancing Performance in Latent Semantic Indexing Retrieva*l; Behavior Research Methods, Instruments and Computers, 1991, Vol.23(2), pp.229-236

[**Dunbar 1999**] Dunbar G., *The clustering of Natural Terms: an Adaptive Resonance Theory Model*; International Joint Conference on Neural Networks (IJCNN'99), Washington DC, July 10-16 1999

[**Evans & Zhai 1996**] Evans D.A. and Zhai C., *Noun-Phrase Analysis in Unrestricted Text for Information Retrieval*; In Proceedings, 34th Annual Meeting of the Association for Computational Linguistics (ACL'96), Santa Cruz, CA, pp.17-24

[**Fagin 2000**] Fagin R., *Allowing Users to Weight Search Terms;* In Proceedings of RIAO (Recherche d'Informations Assistee par Ordinateur = Computer Assisted Information Retrieval), 2000, pp.682-700

[**Fahlman et al. 1981**] Fahlman S, Touretzky D, and van Roggen W., *Cancellation in a Parallel Semantic Network*; International Joint Conference on AI, 1981, pp.257-263

[**Fauconner 1990**] Fauconner, G., *Domains and connections;* Cognitive Linguistics, 1990, Vol.1, No.1, pp.151-174

[**Fellbaum 1998**] Fellbaum C., editors, *An Electronic Lexical Database*, MIT Press, May 1998 ISBN 0-262-06197-X

[**Fishler & Firschein 1987**] Fishler M.A. and Firschein O., *Intelligence: The Eye, the Brain, and the Computer;* Addison-Wesley, 1987

[**Fox 1987**] Fox E. A., *Development of the CODER System: A Testbed for Artificial Intelligence Methods in Information Retrieval;* Information Processing & Management, 1987, Vol.23, No.4, pp. 341-366

[**Frieder et al. 1999**] Frieder O., Grossman D.A., Chowdhury A. and Frieder G., *Efficiency Considerations for Scalable Information Retrieval Servers*; Journal of Digital Information, Peer reviewed paper Vol.1, No.5, December 1999

[**Fuhr & Buckley 1991**] Fuhr N. and Buckley B., *A Probabilistic Learning Approach for Document Indexing;* ACM Transaction on Information Systems, July 1991, Vol.9(3), pp.223-248

[**Fuhr 1992**] Fuhr N., *Probabilistic Models in IR*; The Computer Journal, 1992, pp.243-255

[**Furnas 1986**] Furnas G.W., *Generalized Fish Eye Views;* In ACM Conference on Human Factors in Computing Systems and Graphic Interfaces (CHI'86), Boston, May1996, pp16-23

[**Ganter & Wille 1999**] Ganter B. and Wille R., *Formal Concept Analysis: Mathematical Foundations*; ISBN 3-540-627771-5 Springer-Verlag Berlin Heidelberg, 1999

[**Ginsberg 1993**] Ginsberg A., *A Unified Approach to Automatic Indexing and Information Retrieval*; IEEE AI in Text-Based Information Retrieval, 1993

[**Godin et al. 1993**] Godin R. Missaoui R. and April A., *Experimental Comparison of Navigation in a Galois Lattice with Conventional Information Retrieval Methods*; International Journal of Man-Machine Studies, 1993, Vol.38, pp.747-767

[**Godin et al. 1995**] Godin R., Missaoui R. and Alaoui H., *Incremental Concept Formation Algorithms Based on Galois (concept) Lattices;* Computational Intelligence, 1995, Vol.11(2), pp.246-267

[**Goldberg 1989**] Godsburg D.E., *Genetic Algorithms in Search Optimization & Machine Learning*; Addison Wesley Longman Inc., 1989, ISBN 0-201-15767-5

[**Good 1965**] Good I.J., *The Estimation of Probabilities: An Essay on Modern Bayesian Methods;* MIT Press

[**Gordon 1988**] Gordon M., *Probabilistic and Genetic Algorithms for Document Retrieval;* Communications of the ACM, October 1988, Vol.31(10), pp.1208-1218

[**Greiff et al. 1997**] Greiff W.R, Croft W.B. and Turtle H.R., *Computationally Tractable Probabilistic Modelling of Boolean operators;* In Proceedings of the 20th ACM SIGIR Conference on Research and Development in IR (SIGIR'97)), 1997, pp.119-128

[**Harter & Hert 1997**] Harter S.P. and Hert C.A., *Evaluation of Information Retrieval Systems: Approaches, Issues, and Methods*; Annual Review of Information Science & Technology, 1997, Vol.32

[**Hemkemeier & Vallentin 1998**] Hemkemeier B. and Vallentin F., *On the Decomposition of* Lattices; Electronic Colloquium on Computational Complexity (ECCC), Report No. TR98-052, 1998

[**Hofmann 1999**] Hofmann T., *Probabilistic Latent Semantic Indexing*, 22nd International Conference on Research and Development in IR (SIGIR'99), August 1999, pp.50-57

[**Holland 1975**] Holland J.H., *Adaptation in Natural and Artificial Systems*; The University of Michigan, 1975

[**Honkela et al. 1998**] Honkela T., Kaski S., Lagus K. and Kohonen T., *WEBSOM - Self-Organising Maps of Document Collections*; In Proceedings Workshop on Self-Organizing Maps, Espoo, Finland Also appeared in Neurocomputing, 1998, Vol21, pp101-117

[**Hopfield 1984**] Hopfield J.J., *Neurons with Graded Response have Collective Computational Properties Like Those of Two-state Neurons;* In Proceedings of National. Academy of Science U.S.A., Vol.81, pp.3088-3092

[**Jacobs 1993**] Jacobs P.S., *Using Statistical Methods to Improve Knowledge-based News Categorization*; IEEE Expert, April 1993, pp.13-23

[**Jansen et al. 1998**] Jansen B.J., Spink A., Bateman J. and Saracevic T., *Real Life Information Retrieval: A Study of User Queries on the Web*, SIGIR Forum, Vol.32 No.1, pp.5-17

[**Jitender et al. 1998**] Deogun J.S., Raghavan V.V. and Sever H., *Association Mining and Formal Concept Analysis*; In JCIS Proceedings 98: RSDMGrC98- Sixth International Workshop

[**Jung & Raghavan 1990**] Jung G.S. and Raghavan V.V., *Connectionist Learning in Constructing Thesaurus Like Knowledge Structure;* AAAI Spring Simposium on Text-based Intelligent Systems. Working notes, March 1990

[**Karlsson et al. 1990**] Karlsson F., Voutilainen A. Heikkilä J. and Anttila A., *Natural Language Processing for Information Retrieval Purposes;* SIMPR Document No. SIMPR-RUCL-1990-13.4e, 1990

[**Karlsson et al. 1991**] Karlsson F., Voutilainen A. Heikkilä J. and Anttila A., *Constraint Grammar: a Language-Independent System for Parsing Unrestricted Text, with an Application to English, Natural Language Text Retrieval*; Workshop Notes from the Ninth National Conference on Artificial Intelligence (AAAI-91), Anaheim, California, July 15, 1991

[**Kim & Compton 2001**] Kim M. and Compton P., *A Web-based Browsing Mechanism Based on Conceptual Structures;* 9th International Conference on Conceptual Structures (ICCS 2001), Eds. *Guy W. Mineau*, California USA, 30-3 July 2001, CEUR-WS, Stanford University, pp47-60

[**Kimoto & Iwadara 1990**] Kimoto H. and Iwadera T., *Construction of a Dynamic Thesaurus and Its Use for Associative Information Retrieval*; 13th SIGIR'90, pp.227-240

[**Kjeldsen & Cohen 1987**] Kjeldsen R. and Cohen P.R., *The Evolution and Performance of the GRANT System*; IEEE Expert, Summer 1987, Vol.2 No.2, pp.73-79

[**Kleinberg 1998**] Kleinberg J., *Authoritatitive Sources in a Hyperlinked Environment;* Journal of the ACM, Vol.46(5), pp 604-632

[**Koczy & Gedeon 2000**] Koczy L. and Gedeon T., *A Model of Intelligent Information Retrieval using Fuzzy Tolerance Relations Based on Hierarchical Co-Occurrence of Words*; In "Soft Computing in Information Retrieval" Fabio Crestani & Gabriella Pasi (Eds.), ISBN 3-7908-1299-4

[**Kohle & Merkl 1996**] Kohle M. and Merkl D., *Visualizing Similarities in High Dimensional Input Spaces with a Growing and Splitting Neural Network; In* Proceedings of the 6th International Conference on Artificial Neural Networks (ICANN'96), Bochum Germany, July 16-19, 1996

[**Kohonen 1998**] Kohonen T., *Self-Organization of Very Large Document Collections: State of the Art*; ICANN98 Vol.1, pp65-74

[**Kosko 1987**] Kosko B., *Adaptive Bidirectional Associative Memory*; Applied Optics, Vol.26(23), pp.4947-4960

[**Kosko 1988**] Kosko B., *Bidirectional Associative Memory*; IEEE Transactions on Systems, Man and Cybernetics, Vol.18(1), pp.49-60

[**Kraft et al. 1994**] Kraft D.H., Petry F.E., Buckles B.P. and Sadasivan T., *The Use of Genetic Programming to Build Queries for Information Retrieval*; In Proceedings of IEEE Symposium on Evolutionary Computation 1994, pp.468-473

[**Krovetz & Croft 1992**] Krovetz R. and Croft W.B., *Lexical Ambiguity and Information Retrieval*; ACM Transactions on Information Systems, 1992, Vol.10(2), pp.115-141

[**Kulkarni & Yazdanpanahi 1993**] Kulkarni A.D. and Yazdanpanahi I., *Generalized Bidirectional Associative Memories for Image Processing*; ACM SIGAPP Simposium on Applied Computing,1993

[**Kuznetsov & Ob"edkov 2001**] Kuznetsov S.O. and Ob"edkjov S.A., *Comparing Performance of Algorithms for Generating Concept Lattices,* ICCS'01

[**Kwok 1995**] Kwok K.L., *A Network Approach to Probabilistic Information Retrieval*, ACM Transactions in Information Systems, Vol.13, No.3, July 1995, pp.324-353

[**Lagus et al. 1996**] Lagus K., Honkela T., Kski S. and Kohonen T., *Self-Organizing Maps of Document Collections; A New Approach to Interactive Exploration;* In Simoudis E., Han J. and Fayyad U., (eds), Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, California, 1996, pp.238-243

[**Lee 1995**] Lee J.H., *Analysing the Effectiveness of Extended Boolean Models in Information Retrieval;* Technical report TR95-1501, Cornell University, 1995, http://cs-tr.cs.cornell.edu/

[**Lee 1998**] Lee J.H., *Combine the Evidence of Different Relevance Feedback Methods for Information Retrieval;* Information Processing and Management, 1998, Vol.34, No.6, pp.681-691

[**Leung et al. 1997**] Leung C.S., Chan L.W. and Sum J., *Attraction Basin of Bidirectional Associative Memories;* International Journal of Neural Systems, 1997, Vol.7, Nos.6

[**Lewis 1992**] Lewis D.D., *Feature Selection and Feature Extraction for Text Categorization;* In Proceedings of Speech and Natural Language Workshop, San Francisco, February 1992, Defence Advanced Research Projects Agency, Morgan Kaufmann, pp.212-217

[**Lin et al. 1991**] Lin X., Soerget D. and Marchionini G., *A Self-Organising Semantic Map for Information Retrieval*; In Proceedings of ACM SIGIR, Chicago, IL, USA, 1991, pp.262-269

[**Lindig 1995**] Lindig C., *Concept-Based Component Retrieval*; Working Notes of the IJCAI-95 Workshop, Formal Approaches to the Reuse of Plans, Proofs and Programs, August 1995, pp.21-25

[**Liu 1993**] Liu M., *The Complexities of citation practice: A review of citation studies;* Journal of Documentation, 1993, Vol.49(4), pp.370-408

[**Losada & Barreiro 1999**] Losada D.E. and Barreiro A., *Using a Belief Revision Operator for Document Ranking in Extended Boolean Models*; In Proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval(SIGIR'99), 1999, pp.66-73

[**Losee 1998**] Losee R.M., *Comparing Boolean and Probabilistic Information Retrieval Systems Across Queries and Disciplines*; Journal of the American Society for Information Science, 1998, Vol.48(2), pp.143-156

[**Luger 2002**] Luger G.F., *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*; Fourth Edition, Addison-Wesley, ISBN: 0-201-64866-0

[**Luhn 1958**] Luhn H.P., *The Automatic Creation of Literature Abstracts;* IBM Journal of Research and Development, 1958, Vol.2, pp.159-165

[**Maass & Warmth 1995**] Maass W. and Warmuth M., *Efficient Learning with Virtual Threshold Gates;* In Proceedings of the 12th International Machine Learning Conference, 1995, pp.378-386

[**MacLeod & Robertson 1991**] MacLeod K.J. and Robertson W.A., *A Neural Algorithm for Document Clustering*; Information Processing and Management, 1991, Vol.27(4), pp.337-346

[**Mandl 2000**] Mandl T., *Tolerant and Adaptive Information Retrieval with Neural Networks*; In Global Dialogue, Science and Technology Thinking the Future at EXPO 2000, Hannover, www.shaping-the-future.de/pdf_www/190_paper.pdf

[**Marcus et al. 1993**] Marcus M., Santorini B. and Marcinkiewicz M.A., *Building a Large Annotated Corpus of English: the Penn Treebank;* Computational Linguistics, 193, Vol.19

[**Marcus et al. 1994**] Marcus M., Kim G., Marcinkiewicz M.A., MacIntyre R., Ferguson M., Katz K. and Schasberger B., *The Penn Treebank: Annotating Predicate Argument Structure*; In Proceedings of the ARPA Human Language Technology Workshop

[**Maron & Kuhns 1960**] Maron M.E. and Kuhns J,L., *On Relevance, Probabilistic Indexing, and Information Retrieval*; Journal of the ACM, 1960, Vol.7, pp.216-244

[**Mathai & Upadhyaya 1989**] Mathai G. and Upadhyaya B.R., *Performance Analysis and Application of the Bidirectional Associative Memory to Industrial Spectral Signatures;* In Proceedings of the International Joint Conference on Neural Networks (IJCNN'89), Washington D.C., June 18-22, 1989, pp.1-37

[**Merkl 1995a**] Merkl D., *Content-based Document Classification with Highly Compressed Input Data*; In Proceedings of the 5th International Conference on Artificial Neural Networks (ICANN'95), Paris, France, October 1995

[**Merkl 1995b**] Merkl D., *The Effect of Lateral Inhibition on Learning Speed and Precision of a Self-organizing Map*; In Proceedings of the Australian Conference on Neural Networks, Sydney, NSW

[**Merkl & Rauber 2000**] Merkl D. and Rauber A., *Document Classification with Unsupervised Artificial Neural Networks;* In "Soft Computing in Information Retrieval: Techniques and Applications, Crestani F. and Pasi G. (eds), Physica Verlag (Springer Verlag), Heidelberg, Germany, 2000, pp.102-121

[**Merwe & Kourie 2001**] van der Merwe F.J. and Kourie D.G., *A Lattice-Based Data Structure for Information Retrieval and Machine Learning;* ICCS'01 International workshop on Concept Lattices-based KDD

[**Miikkulainen 1991**] Miikkulainen R., *Self-Organizing Process Based on Lateral Inhibition and Synaptic Resource Redistribution*; In Proceedings of the International Conference on Artificial Neural Networks (ICANN'91), Espoo, Finland

[**Monarch & Carbonell 1987**] Monarch I. and Carbonell J., *CoalSORT: A Knowledge-Based Interface;* IEEE Expert, Spring 1987, Vol.2, No.1, pp.39-53, ISSN 0885-9000

[**Mozer 1984**] Mozer M.C., *Inductive Information Retrieval Using Parallel Distributed Computation*; Technical Report, Institute of Cognitive Science, University of California, San Diego, USA, June 1984

[**Oja 1982**] Oja E., *A Simplified Neuron Model as a Principal Component Analyser*; Journal of Mathematical Biology, 1982, Vol.15, pp.267-273

[**Oja 1989**] Oja E., *Neural Networks, Principal Components, and Subspaces*; International Journal of Neural Systems, 1989, Vol.1, No.1, pp.61-68

[**Osborn & Sterling 1999**] Osborn J. and Sterling L., *Automated Concept Identification within Legal Cases*; The Journal of Information, Law and Technology (JILT) 1991, issue1

[**Pasquier et al. 1999**] Pasquier N., Bastide Y., Taouil R. and Lakhal L., *Efficient Mining of Association Rules Using Closed Itemset Lattices*; Information Systems, 1999, Vol.24, No.1, pp25-46

[**Petry et al. 1993**] Petry F., Buckles B., Prabhu D. and Kraft D., *Fuzzy Information Retrieval Using Genetic Algorithms and Relevance Feedback;* In Proceedings of the ASIS Annual Meeting, 1993, pp.122-125

[**Pinker 1997**] Pinker S., *How the Mind Work;*, W.W. Norton and Company, 1997

[**Pollitt 1987**] Pollitt S., CANSEARCH: *An Expert Systems Approach to Document Retrieval*: Information Processing & Management, 1987, Vol.23, No.2, pp.119-138

[**Prediger & Wille 1999**] Prediger S. and Wille R., *The Lattice of Concept Graphs of a Relationally Scaled Context*; Knowledge Science and Engineering with Conceptual Structures, Lecture Notes in Artificial Intelligence, Springer, Berlin 1999. Eds. W. Tepfenhart

[**Preece 1981**] Preece S., *A Spreading Activation Model for Information Retrieval*; PhD Thesis, University of Illinois, Urbana - Champaign, USA

[**Priss 1997**] Priss U., *A Graphical Interface for Document Retrieval Based on Formal Concept Analysis*; In proceedings of 8th Midwest Artificial Intelligence and Cognitive Science Conference, AAAI Technical Report CF-97-01, pp.66-70

[**Priss 2000a**] Priss U., *Faceted Information Representation;* In Proceedings of the 8th International conference on Conceptual Structures, 2000, Shaker Verlag, Aachen, pp.84-94

[**Priss 2000b**] Priss U., *Lattice-based Information Retrieval;* Knowledge Organization, 2000, Vol.27, No.3, pp.132-142

[**Quillian 1968**] Quillian R., *Semantic Memory;* In Minsky M., editor, *Semantic Information Processing*, The MIT Press, Cambridge, MA, USA, pp.216-270

[**Rada & Hafedh 1989**] Rada R, Hafedh M., *A Knowledge-Intensive Learning System for Document Retrieval;* In Morik K. (ed.), "Knowledge Representation and organization in machine learning", 1989, pp.65-87

[**Ribeiro et al. 2000**] Ribeiro B.N., Silva L., Muntz R., *Bayesian Network Models for Information Retrieval*; In Soft Computing in Information Retrieval (Eds.) Fabio Crestani, 2000, ISBN 3-7908-1299-4

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

[**Rose & Belew 1991**] Rose D.E. and Belew R.K., *A Connectionist and Symbolic Hybrid for Improving Legal Research;* International journal of Man-Machine Studies, Vol.35, pp.1-33

[**Sahani et al. 1998**] Sahani M., Dumais S., Heckerman D. and Horvitz E., *A Bayesian Approach to Filtering Junk E-Mail*; AAAI Workshop on Learning for Text Categorization, July 1998, Madison, Wisconsin. AAAI Technical Report WS-98-05

[**Salton 1971**] Salton G., *The SMART Retrieval System - Experiments in Automatic Document Processing;* Printice Hall, 1971

[**Salton & Allan 1994**] Salton G. and Allan J., *Automatic Text Decomposition and Structuring*, Information Processing and Management, 1994, 32(2), pp.127-138

[**Salton & Buckley 1988**] Salton G. and Buckley C., *On the Use of Spreading Activation Methods in Automatic Information Retrieval*; In Proceedings of ACM SIGIR, Grenoble, France, 1988

[**Salton & McGill 1983**] Salton G., *Introduction to Modern Information Retrieval;* McGraw-Hill, New York, 1983

[**Salton et al. 1983**] Salton G., Fox E. and Wu U., *Extended Boolean Information Retrieval*; Communications of the ACM, 1983,Vol.26(12), pp.1022-1036

[**Sanderson & Croft 1999**] Sanderson M. and Croft B., *Deriving Concept Hierarchies from Text*; 22nd ACM SIGIR, 1999, pp.206-213

[**Savoy 1994**] Savoy J., *A Learning Scheme for Information Retrieval in Hypertext*; Information Processing & Management, 1994, Vol.30, No.4, pp.515-533

[**Savoy 1997**] Savoy J., *Statistical Inference in Retrieval Effectiveness Evaluation;* Information Processing & Management, 1997, Vol.33, No.4, pp.495-512

[**Savoy & Picard 2001**] Savoy J. and Picard J., *Retrieval Effectiveness on the Web*; Information Processing & Management, 2001, Vol.37, pp.543-569, http://www-seco.unine.ch/Info/Papers/WebIRLong.html

[**Schamber et al. 1990**] Schamber L., Eisenberg M.B. and Nilan M.S., *A Re-examination of Relevance: Toward a Dynamic, Situational Definition;* Information Processing & Management, 1990, Vol.26, No.6, pp.755-776

[**Scholtes 1991**] Scholtes J.C., *Unsupervised Learning and the Information Retrieval Problem;* In Proceedings of the International Joint Conference of Neural Networks, IJCNN'91, Piscataway, NJ, 1991, pp.95-100

[**Shi et al. 1998**] Shi H., Zhao Y. and Zhuang X., *A General Model for Bidirectional Associative Memories*; IEEE Transaction on Systems, Man, and Cybernetics, Part B Cybernetics, August 1998, Vol.28(4), pp.511-519

[**Shoval 1981**] Shoval P., *Expert/Consultation System for a Retrieval Data-base with Semantic Network of Concepts*; ACM SIGIR Conference on Research and Development in Information Retrieval, 1981, pp.145-149

[**Silverstein et al. 1998**] Silverstein C., Henzinger M., Marais H. and Moricz M., *Analysis of a very large AltaVista Query Log*, SRC Technical Note 1998-014, October 26, 1998

[**Singhal et al. 1996**] Singhal A., Buckley C. and Mitra M., *Pivoted Document Length Normalization;* In Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96), 1996, pp.21-29

[**Soderland 1997**] Soderland S., *Learning to Extract Text-based Information from the World Wide Web;* In Proceedings of Third International Conference on Knowledge Discovery and Data Mining (KDD-97), 1997, pp.251-254

[**Song et al. 1999**] Song D.W., Wong K.F., Bruza P.D. and Cheng C.H., *Towards Functional Benchmarking of Information Retrieval Models;* In Proceedings of 12th International Florida Artificial Intelligence Conference, pp.389-393

[**Sowa 1984**] Sowa J.F., *Conceptual Structures;* Information processing in Mind and Machine, Reading, MA, Addison-Wesley

[**Spark Jones 1992**] Spark Jones K., *Assumptions and Issues in Text-Based Retrieval;* In Text-Based Intelligent Systems - Current Research and Practices in Information Extraction and Retrieval, Paul S. Jacobs (eds.), Lawrence Erlbaum Associates, London, 1992

[**Spink & Xu 2000**] Spink A. and Xu J.L., *Selected Results from a Large Study of Web Searching: the Excite Study*, Information Research, 6(1), Available at: http://InformationR.net/ir/6-1/paper90.html

[**Stanfil & Kahle 1986**] Stanfil C. and Kahlle B., *Parallel Free-text Search on the Connection Machine System;* Communication of the ACM, December 1986, Vol.29(12), pp.1229-1239

[**Stumme 1996**] Stumme G., *Local Scaling in Conceptual Data Systems*; In Conceptual Structures: Knowledge Representation as Interlingua (P. Eklund, G. Ellis, and G. Mann, eds.), LNAI 1115, (Berlin), Springer Verlag, August 1996, pp. 308-320, International Conference on Conceptual Structures ICCS 1996: pp.308-320

[**Troina & Walker 1996**] Troina G. and Walker N., *Document Classification and Searching - a Neural Network Approach*; Esa Bulletin, 1996, Vol.87, pp.90-96

[**Tyrväinen 1984**] Tyrväinen P., *On Domain Modelling for Technical Documentation Retrieval;* PhD Thesis, Acta Polytechnica Scandinavia, Mathematics and Computer Science Series No.64, Helsinki 1994, Published by the Finnish Academy of Technology. ISBN 951-666-406-7. ISSN 0355-2713. UDC 681.327:159.95:519.683.5:681.326.34 http://lsi.argreenhouse.com/~remde/lsi/LSIpapers.html

[**Tzeras & Hartmann 1993**] Tzeras K. and Hartman S., *Automatic Indexing Based on Bayesian Inference Networks*; In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93), 1993, pp.22-34

[**Vlajic & Card 1999**] Vlajic N. and Card H.C., *An Adaptive Neural Network Approach to Hypertext Clustering*; IJCNN'99

[**Vrajitoru 2000**] Vrajitoru D., *Large Population or Many Generations for Genetic Algorithms Implications in Information Retrieval*; Soft Computing in Information Retrieval (Eds.) Fabio Crestani, 2000, ISBN 3-7908-1299-4

[**Wang 1996**] Wang Z., *A Bi-directional Associative Memory based on Optimal Linear Associative Memory*; IEEE Transactions on Computers, 1996, Vol.45, No.10, pp.1171-1179

[**Wang & Don 1995**] Wang C.C. and Don H.S., *An Analysis of High-capacity Discrete Exponential BAM*; IEEE Transactions on Neural Networks, 1995, Vol.6, No.2, pp.492-496

[**Weiss et al. 1997**] Weiss S.A., Kasif S. and Brill E., *Winnow vs SMART: A Comparison in the Newsgroup Classification*; April 24, 1997, http://www.cs.jhu.edu/~weiss/winnow.ps or http/citeseer.nj.nec.com/96397.html

[**Wen et al. 2002**] Wen J.-R., Nie J.-Y. and Zhang H.-J., *Query Clustering Using User Logs*, ACM Transactions on Information Systems, Vol.21, No.1, January 2002, pp.59-81

[**Wilkinson & Hingston 1991**] Wilkinson R. and Hingston P., *Using the Cosine Measure in a Neural Network for Document Retrieval*; In Proceedings of the 14th ACM SIGIR International Conference on Research and Development in Information Retrieval SIGIR'91, Chicago, IL

[**Wille 1997**] Wille R., *Conceptual Graphs and Formal Concept Analysis*. In Lukose, D. et al. (eds.): Conceptual Structures: Fulfilling Peirce's Dream, Proceedings of the ICCS'97, Springer, Berlin-New York, 1997, pp.290-303

[**Wolfram 2000**] Wolfram D., *A Query-level Examination of End-user Searching Behaviour on the Excite Search Engine*, CAIS 2000: Proceedings of the 28th Annual Conference of the Canadian Association for Information Science, June 2000

[**Wolfram et al. 2001**] Wolfram D., Spink A., Jansen B.J. and Saracevic T., *Vox Populi: The Public Searching of the Web*, Journal of the American Society of Information Sceinece and Technology, Vol.52, Issue 12, October 2001, pp.1073-1074

[**Woods 1997**] Woods W.A., *Conceptual Indexing: A Better Way to Organize Knowledge*: SUN Lab Technical Report, 1997

[**Wu et al. 1990**] Wu C.H., Wang C.J., Tai H.M. and Roland D.A., *An Investigation of High-order Bidirectional Associative Memories: Performance, Applications, and Parallel Implementations;* In Proceedings of IEEE International Symposium on Circuits and Systems,1990, pp.495-498

[**Yang 2000**] Yang Y., *An Evaluation of Statistical Approaches to Text Categorization*; Journal of Information Retrieval, 1999, Vol.1, No.1/2, pp.67-88, www.cs.cmu.edu/yiming

[**Yang & Korfhage 1993**] Yang J.-J. and Korfhage R.R., *Query Optimization in Information Retrieval Using Genetic Algorithms. Report on the Experiments of the TREC Project;* In Proceedings of TREC'1, NIST, Gaitherburgs (MD), pp.31-58

[**Yang et al. 1998**] Yang J., Honavar V., Miller L. and Wong J., *Intelligent Mobile Agents for Information Retrieval and Knowledge Discovery from Distributed Data and Knowledge Sources*; In Proceedings of the IEEE Information technology Conference, 1998

[**Yu & Liddy 1999**] Yu E.S. and Liddy E.D., *Feature Selection in Text Categorization Using the Baldwin Effect;* IJCNN'99

PUBLICATIONS

1. Rajapakse, R.K. and Denham, M. (2002) "*A Concept based Adaptive Information Retrieval Model using FCA-BAM Combination for Concept Representation*", In Proceedings of the 24th BCS-IRSG European Colloquium on IR Research (ECIR'02), March 2002 Glasgow, UK, pp 150-168

2. Rajapakse, R.K. and Denham, M. (2002) "*Information Retrieval Model Using Concept Lattices for Content Representation*", Workshop proceedings of the FCA KDD workshop of the 15th European Conference on Artificial Intelligence (ECAI'02), July 21-26 2002, Lyon, France

3. Rajapakse, R.K. and Denham M. (2003) "*A Reinforcement Learning Strategy for (Formal) Concept and Keyword Weight Learning for Adaptive Information Retrieval*" In proceedings of the 7th World Multiconference on Systemics Cybernetics and Informatics (SCI'2003) July 2003 in Orlando, Florida USA. This paper was also accepted and published in workshop proceedings of the workshop on "User Modeling, Information Retrieval and Machine Learning" of the 9th International Conference on User Modeling (UM'2003) held in June 2003 in Pittsburgh, USA.