The Application of Morpho-Syntactic Language Processing to Effective Information Retrieval

---

A Dissertation presented in fulfilment

of the requirement for the M.Sc Degree

in Computer Applications

June 1991

Paraic Sheridan B.Sc.

School of Computer Applications

Dublin City University

Dublin 9.

Supervisor: Dr Alan F. Smeaton.

Declaration.

This dissertation is based on the author's own work. It has not previously been submitted for a degree at any academic institution.

Paraic Sheridan
June 1991.

Acknowledgements.

The work reported in this thesis was done while I was working as a Research Assistant on the ESPRIT II project SIMPR. I am very grateful to the CEC for the generous funding that was provided to me while I worked on this project. I would also like to acknowledge the help and guidance I received from the members of the SIMPR project, particularly the team members at the University of Strathclyde and at the University of Helsinki, with whom I worked most closely.

I would like to thank the staff members in the School of Computer Applications for their help and support over the six years that I've been here. Thanks to Jim Doyle for help with hardware and to Heather Ruskin for advice and help with the statistical aspects of my work. I would also like to express my thanks to all those who took the time to provide the data for the experiments reported in this thesis. I realize the considerable effort that was required to complete the task of ranking the phrases (becuase I also had to do it myself!).

The fact that I seem to have accomplished so much in the short two years that I have been working towards this thesis is due to my supervisor Dr. Alan Smeaton (the ultimate information retrieval machine!). He has never failed to provide exaclty the right piece of information at exactly the right time and has always guided me in the right direction. I also appreciate the working environment that he has created for me while working on the SIMPR project. He has sheltered me from the administrative and political aspects of a large research project and has allowed me to just get on with the work at hand. Finally I would like to thank him for much help with the proof-reading and correction of this thesis. He has been very patient and has given me much encouragement this past few weeks. Thank you Alan!

If my supervisor has created the perfect work environment, then my parents and family have, over the past 23 years, created the perfect environment for me outside work. Despite my many long years at school and University, my parents have been the principal educators in my life. They have taught me more than they will ever realize. Rather than do the injustice of trying to express in words my immense gratitude and love for them, I would simply like to dedicate this work to them.

Abstract

The fundamental function of an information retrieval system is to retrieve texts or documents from a database in response to a user's request for information, such that the content of the retrieved documents will be relevant to the user's original information need. This is accomplished through matching the user's information request against the texts in the database in order to estimate which texts are relevant. This thesis proposes a method for using current natural language processing techniques for the construction of a text representation to be used in an information retrieval system. In order to support this proposal, a matching algorithm was designed specifically for performing the retrieval task of matching user queries against texts in a database, using the proposed text representation.

The effectiveness of a prototype implementation of the matching algorithm was then evaluated through a formal experiment. This experiment involved the use of standard statistical methods to compare the phrase matching capabilities of the matching algorithm to a sample of information retrieval system users performing the same task. The results of this experiment are presented and are analysed in some detail. These experimental results are not only useful for evaluating the effectiveness of the proposed text representation and matching algorihtm, but more generally, they give some indication as to the usefulness of incorporating natural language processing techniques into information retrieval systems.

Table of Contents.

List of Figures

List of Tables

Chapter_1.

Introduction

Over the past number of decades the computer has permeated just about every working environment. More and more tasks have become automated and more and more information is now being stored in electronic form. This is reflected in the fact that we are often said to be living in the "information age". The advent of the personal computer and the wordprocessor has led to a complete change in the modern office environment. The old scenario of pens, paper, index cards and rows of filing cabinets has been replaced with personal computers and small boxes of floppy disks. Most office documentation, such as letters or memoranda, is now produced in electronic form and only reproduced on paper when necessary. The development of computer networks and electronic mail systems has meant that it is often possible for people to communicate using electronic means only.

This "information explosion" has not been limited to the modern office. At the other end of the scale, there are environments in which the amounts of information involved are orders of magnitute greater than the one or two page office documents. As more and more complex machinery and systems are designed and built in a variety of application areas, there is a corresponding increase in the amount of documentation needed to record how they work and to explain how they can be operated and maintained (Consider, for example, the documentation involved with explaining the operation and maintenance of a modern jumbo jet). As more and more research is being done in all scientific disciplines, from artificial computer intelligence to microbiology, there has been a corresponding increase in the amount of technical research reports being produced. These are just two examples of cases where the amounts of information, usually in electronic form, have increased greatly. In order to keep pace with this explosion in the amount of information in the world, there have been many technological advances in electronic data storage media and in computer processing speeds. It is now possible to store enormous quantities of information in quite compact form. A recent example of technological advance in this area is the use of compact disc technology for storing computer data. Using this technology, up to 600 Megabytes of computer data can be stored on a small disk of 12cm diameter. To put these figures in perspective, this 200 page thesis, including all figures, tables and appendices amounts to only a half of one Megabyte of data. A compact disc could therefore store more than 240,000 pages of textual information.

Despite the huge explosion in the amount of electronic data, and the techological advances in computer hardware for storing this information, it has been widely acknowledged that there have not been the corresponding advances in computer software for managing information of this nature. The research discipline of **information retrieval** is concerned with researching and developing exactly this type of computer software; for managing huge databases of textual information and for facilitating the retrieval of information from those databases. The distinction must be made here between the different types of information that can be stored electronically. The explicit mention of the term *textual information* when defining the task of information retrieval research was deliberate. Researchers of information retrieval are concerned primarily with the problem of retrieving documents or text from large databases of textual data. An example of such a database would be one consisting of all research articles published in a particular computer journal over the past twenty years. If the data or information were more structured in nature, for example a database of train schedules or information relating to the customers of a particular manufacturing firm, then that information could be stored in a structured format using current database technology and it could be retrieved from the database using standard structured database querying languages. On the other hand, if the data is not only textual but also contains digitized pictures and possibly some digitized speech patterns (so that the computer can display pictures and can synthesize speech) then this information is said to be multimedia in nature and there exists a seperate research discipline which is interested in the particular problems associated with managing and retrieving such information.

Many of the problems encountered by researchers of information retrieval have been due to the textual nature of the information they are dealing with, and the way in which users usually want to retrieve information from these text databases. Users who want to retrieve information from a structured database usually have a request that can be specified very precisely using a structured database query language (like SQL, for example). In a large text database however, the user's information need is more often quite vague and not easily expressed precisely. Much research has been directed at how users of information retreival systems express their information needs and how a system should go about verifying that it has interpreted a search request correctly. A more difficult problem presents itself after the retrieval query has been entered to the computer. It is the task

of comparing that query against all of the information in the textbase to ascertain what pieces of text may provide the information to satisfy the user's request. The information retrieval system must decide what the user is looking for and must decide what each text is about and then must match one against the other to decide which texts are about the topic that the user is looking for information on. For example, if an aircraft mechanic uses an information retrieval system to search for information in the aircraft maintenance manual about how to replace the main fuel line to engine number 3, then the system must match that information need against each chapter or page or paragraph of the maintenance manual and decide which sections provide the best information about how to replace the main fuel line to engine number 3. It may turn out that the best it can do is find information about replacing fuel lines or repairing engines, but the task is to decide *what is the most relevant information.*

Many of the problems associated with processing textual information, for any purpose, are associated with the fact that the text is usually written using free-form natural human language. This type of language can often be difficult to understand, even by the human reader. The problems encountered by a computer system trying to understand it are therefore much more complex. When a person reads a piece of text they use all of their accumulated knowledge about the world in the understanding process. It is usually the knowledge relating to the particular subject that is used most, but it is also possible for a person to use knowledge of unrelated but similar topics to draw analogies in order to help understanding. Without this huge amount of world knowledge it can often be difficult to understand texts. A simple example of this is the difference in the levels of understanding achieved by a person from a non-technical background reading this thesis and an information retrieval researcher reading this thesis. This difference is due to the amount of background knowledge about information retrieval that the researcher of that discipline will have.

The task of encoding the amounts of world knowledge necessary for a computer system to achieve a respectable level of understanding of general texts has proven to be enormous. One attempt to do this has been going on at MCC in the US for the past seven years [Lenat *et al.* 1990]. In the absence of such levels of knowledge, researchers interested in the task of **natural language processing** have tried to develop other methods of

understanding natural language texts (or at least methods for deciding what a text is about). Of course, because these methods operate without any real knowledge, they can often become confused and don't always provide the single correct analysis of a given sentence or text, especially when there is more than one possible interpretation there. Despite these shortcomings, some researchers of information retrieval have recently begun to use these natural language processing techniques to try and solve some of the problems associated with their discipline; in particular to help identify the content or subject of texts and to decide what the user's information queries are about so that the two can be matched.

The research work reported in this thesis was directed at exactly that problem. It represents an attempt to use current natural language processing techniques to construct a representation of texts which captures the subject matter of those texts, and the development of a computer algorithm capable of matching these text represenations in order to perform the information retrieval task of matching a user retrieval query against the information in a large database. I will start off in the next chapter then, by first describing the problems of information retrieval and how researchers have previously attempted to overcome them in the absence of any natural language processing techniques. I will also describe how a concentrated research effort over the past number of decades has led to the development of robust natural language processing techniques and I will then discuss how researchers of information retrieval have recently been using those techniques in an attempt to gain some improvement in the effectiveness of information retrieval systems.

In the chapters following I will then describe in detail the natural language processing techniques that have been made available to me by the Research Unit for Computational Linguistics at the University of Helsinki, and how I have used those techniques in the research and development of a computer-internal representation of text designed specifically for use in the storage of text representatives in an information retrieval system. I will also describe how this text representation can be powerfully used for the retrieval of texts from a large database in response to a user's information need by building a representation of the user's query and matching this representation against the text representations in the database, with the result of generating a list of texts ranked according to their estimated relevance to the user's original search query.

In the final chapters I will describe an experimental evaluation of the matching algorithm that I have designed. I will describe the experimental design, the statistical tests used in the calculation of results and I will present the results of the statistical analysis and comment in some detail on their implications. This will lead me into my final conclusions and I will finish by suggesting some possible directions for future research.

Chapter 2.

Information Retrieval and Natural Language Processing

2.1 Information Retrieval.

The fundamental function of an information retrieval system is to retrieve documents from a database in response to a user's request for information, such that the content of the retrieved documents will be relevant to the user's original information need. Any information retrieval system can be divided naturally into two parts. The main function, retrieval, is to search for and retrieve documents from a database in response to a search query, as mentioned above, but prior to that the documents must be entered into the system and stored in such a way that this search and retrieval is possible. This process is concerned with establishing the subject matter and information content of a document and representing it in such a way that it can be used to establish the relevance of that document to a user's information need. In any document or text only a small subset of the language or words used are necessarily relevant to the information content of that document. The process of extracting this subset is known as *indexing* and the process of establishing the subject area of a text is known as *classification.*

The standard information retrieval scenario is that texts entered to the database are *indexed* to produce a *text representative* which encapsulates the information content of that text. The texts can also be *classified* to establish subject matter. The text representatives are then stored along with the original text in the text database. At retrieval time the user's information need, as expressed via a query, is taken by the system and transformed in the same manner as the texts into the same type of representation. The query representative is then matched against the text representatives for each text in the database in order to establish which texts are most relevant to the user's query. This basic scenario is illustrated in Figure 2.1 below.

It would seem intuitive that in order to establish the information content of a piece of text the text would have to be read and understood. In order then for a computer to be able to perform the indexing task it would ideally have to be able to read and understand the texts entered. This fact was recognised by early researchers of information retrieval but at that time methods of automatic natural language processing were not well developed. Researchers therefore sought to discover what constituted a good text representative and

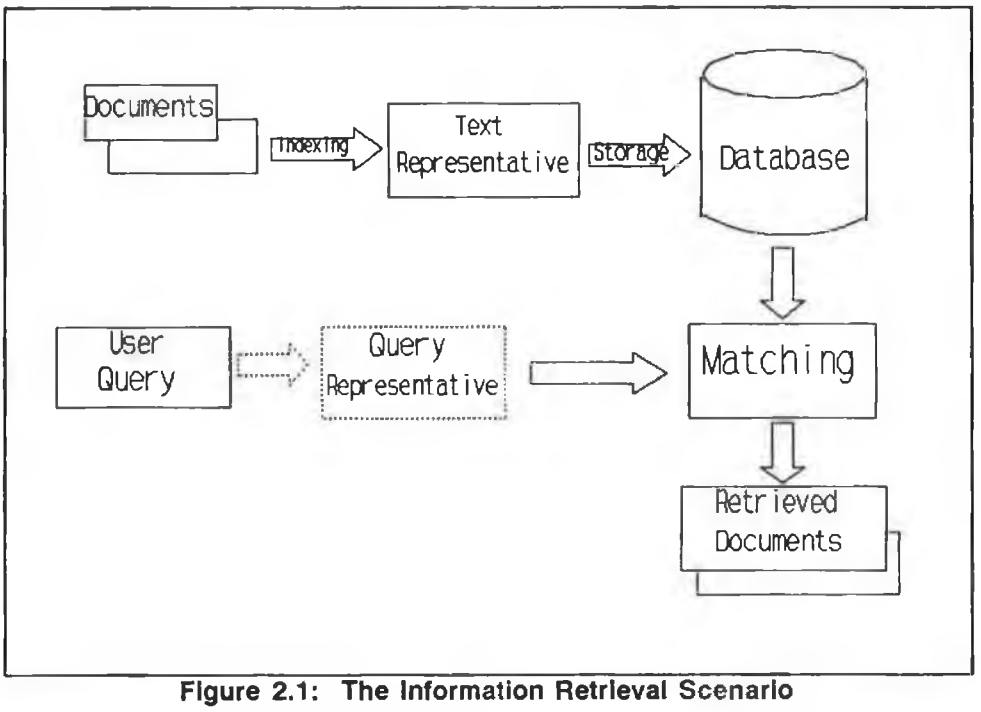


Figure 2.1: The Information Retrieval Scenario

how such a representative could be extracted from text without actually "understanding" the text.

The obvious initial choice of text representatives were the individual words in the texts. The extraction of text representatives based on single words is known as *term indexing* and the text representatives (words) extracted are called *terms*. One of the most common and successful term indexing methodologies extracts terms based on statistical information. During text processing a record is kept of the number of occurances of each word in each text in the database. Term extraction is then based on the hypothesis that words which occur with medium to high frequency in a particular document but with low frequency across the rest of the document set are the best representatives for that document. Other methods of term indexing use complex mathematical and probabilistic models of the text database to estimate good index terms for particular documents.

An undesirable characteristic of index terms as a representative of document content is that they have a highly variable specificity; some words are highly specific in that they describe only a single concept and other words are very general in that they can describe

a whole range of different concepts depending on the context in which they are used. In order to eliminate this characteristic, information retrieval researchers turned to the use of *phrases* as text representatives, where a phrase could consist of two or more words. The use of phrases helped generalise the highly specific term indexes and helped specify the highly general terms. Statistical phrase indexing methods were developed similar to those for term indexing but using word co-occurance statistics as well as single term occurances. More complex mathematical and probabilistic models were also developed for constructing phrase indexes.

2.2 Natural Language Processing.

While researchers of information retrieval were investigating indexing methods based on statistics or probabilities, other research communities were investigating the automatic processing of nautural language text for tasks like story understanding, text generation, speech recognition and automatic translation. The early problems associated with automatic text processing have now been well researched and many different methodologies and strategies have been proposed for solving them. Effective and robust language processing systems have existed now for quite some time.

Automatic processing of spoken natural langauge can take place at several different levels, as illustrated in Table 2.1

Table 2.1: Levels of Language Processing

Phonological
Morphological
Lexical
Syntactic
Semantic
Pragmatic/Discourse

At the lowest level, phonological processing deals with the individual sound elements that go to make up a spoken word. This level of processing is used mainly in speech recognition and speech synthesis systems and is not applicable to the information retrieval domain. The morphological level is concerned with word particles. For example, a morphological analyser would discover that the word "*preprocessing*" was made up of the base word "*process*" with the prefix "*pre*" and the suffix "*ing*". At the lexical level the full word is analysed as a single unit to establish grammatical category, singularity, case etc. For example, the word "*text*" in the sentence "*The new text is first preprocessed...*" would be analysed as a singular noun in the nominative case. Syntactic level processing is then concerned with the relationship between words in sentences. Valid relationships between words in a language have been defined in the form of grammars by linguists for many years. For example a grammar would specify the syntactic rule that you cannot have an adjective directly before a preposition ("*large from*"). Syntactic analysis of text would be able to indicate that there was a different relationship between the words "*graph*" and "*searching*" in the sentences:

"*After searching the database the results are plotted on a graph*"
"*An algorithm for searching highly structured three-dimensional graphs*"

Once word relationships have been identified at the syntactic level, semantic processing attempts to identify the *meaning* of text. This is a very complex task and must rely on large quantities of domain specific world knowledge. Many formalisms have been proposed for the representation of such knowledge and many methods of semantic analysis have been researched but no effective robust method presently exists for semantic analysis of large-scale general texts.

A pragmatic/discourse analysis operates at an even higer level and is more complex again. At this level an attempt is made to construct a model of the context surrounding the events described in the text. For example, in a text describing two people meeting, a pragmatic and discourse level analysis would try to infer information about the meeting scene (meeting at a job interview is different from meeting in a bar) and the types of people who were meeting (shy, outgoing, do they know each other already?). Methods for this type

of language processing are still at the research stage.

A system which performs automatic language processing will perform processing at only a subset of these levels. An important consideration in deciding on which levels of processing to include is the range of text domains that are likely to be processed. If a system is to be designed so that it can process any text dealing with any topic then it will not be possible to perform processing above the syntactic level without an enormous knowledge base like, for example, the one being created at MCC [Lenat *et al.* 1990]. If a system will only process texts in a fairly restricted domain then it may be possible to construct a knowledge base containing all the world knowledge relevant to that domain so that semantic level processing can be performed to establish the meaning of sentences.

At the lower levels of natural language processing (e.g. lexical and syntactic) there exist many different phenomena or characteristics of language that present problems for any system performing automatic processing. These problems can usually only be handled through recourse to higher level processing. For example, at the lexical level many problems arise becuase of homonymity (words having more than one meaning) and lexical ambiguity (words that can be a noun and a verb, for example). The word "*bar*" has many different meanings ("a long thin piece of rigid material", "counter across which drinks are served", "place in lawcourt where prisoner stands", "fasten with bar, bolt etc", "exclude from consideration" etc..) and can also act as either a noun or as a verb. The word "*mine*" can be a noun, verb or pronoun. Usually these types of ambiguity can be resolved at the level of syntactic processing. For example, the word "*mine*" is obviously a noun in the sentence "*He works long hours in the mine*" and the word "*bar*" is a noun in the setnence "*He stood quietly at the bar*". However one must invoke some semantic level processing in order to establish whether the word "*bar*" refers in this latter case to a counter across which drinks are served or a place in lawcourt where a prisoner stands. In order to establish the correct meaning, a system would need to gather some information relating to the context of the above statement; whether it was dealing with a criminal at court or somebody going for a drink.

There are also many problems associated with ambiguous relationships betwen words at the syntactic level. The classic example of this is the sentence "*I saw the man on the hill with*

the telescope", in which it is impossible to decide the relationship between the viewer, the hill, the man, and the telescope. Any computer system which attempts to provide automatic processing of natural language must take account of these problems and make some effort to overcome them as far as possible. I will describe in more detail later how I have taken account of these phenomena in my own work.

2.3 Natural Language Processing for Information Retrieval.

At this point in time there exist many commercially successful natural language processing systems for tasks such as automatic machine translation (METAL, ALPS), database querying (INTELLECT, Q&A) and grammer and style checking (CRITIQUE). These systems are robust and efficient and are testimony to the advances that have been made in natural language processing techniques.

Once researchers of information retrieval became aware of the developments that had been made in natural language processing, some of them directed their attention to researching methods of using language processing in the indexing and retrieval of information. There emerged two different schools of thought about how information retrieval systems should use language processing techniques. The first school of thought believes that in order to retrieve information about a given search request, the system has to understand the request and understand each text in the database. Such information retrieval systems would obviously have to include processing at the semantic level in order to achieve the required level of understanding. A result of this level of processing is that such systems can reply to a user's request by actually *answering* the query. This has been referred to as "*a qualitative difference between document retrieval ... and que stion answering*" [Salton & McGill 1983] and such systems have been described as performing *conceptual information retrieval* [Smeaton 1989]. One drawback of such systems however is the need for semantic level processing. As mentioned above, semantic processing requires large amounts of pre-coded knowledge. Most research in conceptual information retrieval to date (e.g. [Spark Jones & Tait 84], [Tong et al. 1987], [Brajnik *et al.* 1988], [Wood &Sommerville 1988]) has

therefore concentrated on systems which operate only in restricted domains.

The second school of thought in information retrieval believe that in order to retrieve documents dealing with a particular concept the system does not have to understand what that concept is but only has to know where to locate documents which are likely to deal with that concept. These documents are retrieved and the user can then judge which documents best answer the information need. These types of information retrieval system usually try to minimise the use of semantic level processing in the interest of keeping the system as domain independant as possible. Much of the research within this school of thought, which has used natural language processing techniques, has therefore been concentrated on the application of syntactic level processing to information retrieval.

Early attempts at incorporating syntactic processing in information retrieval were made by [Salton 1968], [Dillon & Gray 1983] and [Smeaton & van Rijsbergen 1988]. None of these attempts however led to any significant improvements in retrieval effectiveness and some researchers argued [Lewis & Croft 1989] that the use of purely syntactic level processing for constructing phrase indexes held little promise. A major piece of research in this area was undertaken by Joel Fagan [Fagan 1987]. He used a syntactic analysis to identify the modification relationships betwen words in sentences. This involved identifying the head word of each clause and then categorising the relationship of each of the other words to that head. An example analysis might be:

"*automatic extraction of index terms*"

Pre-modifier:	automatic
Head:	extraction
Post-modifier:	of index terms

A set of rules was then used by Fagan to extract useful index phrases. A simple example of such a rule would be to extract all heads plus each of their pre-modifiers in turn. This would extract the index phrase "*automatic extraction*" from the above phrase. Fagan's system (and the others developed at that time) extracted only two- or three-word index phrases. The index phrases were all stored in a normalized form. For example, the term

"*automatic extraction*" might be stored in the form "*extraction: automatic*". A user query would then undergo the same processing and the normalized query would be matched directly against all of the normalized index phrases in the database. It was concluded that this method was comparable to statistical methods but Salton stated that "*given the equivalent phrase precision for statistical and syntactic processes, the advantage must rest with the simpler, statistical methodolgy*" [Salton *et al.* 1989]. Further experiments on the use of syntactic processing for phrase indexing have been carried out by [Lewis & Croft 1990] and [Sacks-Davis *et al.* 1990].

More recently there has been a shift in emphasis in the use of syntactic processing for information retrieval. Rather than using syntactic information for simply deciding which index phrases to extract, some researchers are now aiming to incorporate as much syntactic information as possible into the text representative. This gives rise to a kind of structured representative of texts rather than just terms or phrases. This is the approach being taken at the Siemens/Nixdorf research centre in Germany [Schwarz 1990] as part of the TINA project. They use shallow syntactic analysis to construct dependency trees for noun phrases (structures that explicitly represent the dependency or modification relationships between words). These dependency relationships are represented by setting up directed links between words. At retrieval time the matching mechanism would not only compare the words in the query and index but would also search for a match in the dependency links. The TINA software has been tested at both the US Patent and Trademark Office and the German Patent Office.

A second example of work which builds a structured represenation of text from syntactic analysis is that of Doug Metzler at the University of Pittsburgh [Metzler & Haas 1989] which is based on the hypothesis that "*the aspect of syntactic description that is most relevant to the semantic composition of larger linguistic entities is the hierarchical structure of these entities*". Metzler therefore works with the hierarchical structure of dependency relationships. His system produces tree structures that highlight the modification relationships between words. The trees are binary and at each level the dominant branch is marked with an asterisk (the dominant branch being the one leading to the node which contains the head at that level). This type of structure is built both for text indexes and queries and the retrieval process then becomes a task of matching these tree structures.

Although this system appears quite powerful there has been no proper evaluation on any large-scale dataset to date.

In his paper [Metzler & Haas 1989], Metzler states that:

> *"A major reason for the rather limited success enjoyed by previous attempts to use syntactic information to improve information retrieval performance is that they have not utilized the appropriate aspects of syntactic description"*

I would reinforce this statement by saying that not only has previous work not used the appropriate aspects of syntactic description but researchers (including Metzler) have neglected to take full advantage of the wealth of information that can be obtained from a syntactic analysis of text. **My thesis is that the best improvements in performance are not to be found by discovering the "appropriate aspects of syntactic description to utilize" but rather in discovering a way of utilizing all aspects of syntactic description.** The work reported here is an attempt to define a text representative that incorporates as much syntactic information as possible so as to provide as rich an indexing of text as possible (at the syntactic level) and to design a matching algorithm that can utilize all of this information to the full at retrieval time.

Chapter 3.

The General Architecture of an Information Retrieval System

I have already outlined some of the general tasks associated with the problem of information retrieval and I have discussed how natural language processing techniques may be used to assist in some of these tasks. I have also mentioned that my research has been concerned with using natural language techniques to construct a structured representation of text, and with developing a matching algorithm that can use this text representation for information retrieval. I now want to put this research in perspective by sketching, in a little more detail, a possible architecture for an information retrieval system. The reason for doing this here is that the work reported in this thesis is concerned only with one particular task within the overall problem of information retrieval, and it is therefore useful to know how this task integrates with the other tasks in an information retrieval system so that one can visualize how the results from this research can be incorporated into a general information retrieval system and hopefully lead to an improvement in the overall retrieval effectiveness of that system. A general picture of the proposed information retrieval system architecture is presented in Figure 3.1. The bolded sections are those with which I have been particularly concerned in my research.

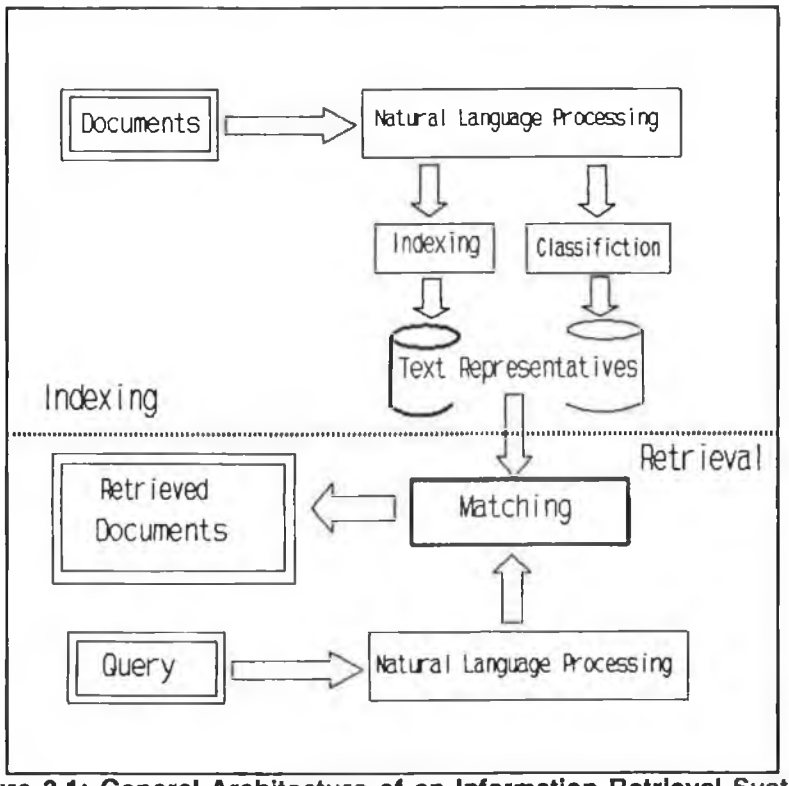


Figure 3.1: General Architecture of an Information Retrieval System.

The top half of the figure illustrates the indexing process and the bottom illustrates retrieval. Each text or document entered into an information retrieval system has to undergo some sort of processing before it is stored in the database. When natural language techniques are used, the text first undergoes full linguistic processing at a level which depends on the technique used. The research presented here was directed at an information retrieval system which performs linguistic processing at the morphological/lexical and sytnactic levels. This processing provides information about the grammatical category and syntactic function of each word in the text. The text may then be processed with a view to identifying and classifying the subject area of the text.

An information retrieval system that performed classification of texts would have to maintain some sort of classification schema which would represent the relationships between different subject areas. For example this thesis would be classified under the subject areas of information retrieval and computational linguistics, with a relationship to the area of artificial intelligence and a broader connection to computer science. This classification schema then provides this general information for matching at retrieval time. This type of information is useful in databases where there is a large collection of texts relating to different subjects. For example classification systems are widely used in libraries. The classification schema would be used for retrieval when the user's search request relates to a general topic rather than a specific item or area within a given topic, like for example, the case of a user looking for texts about artificial intelligence, as opposed to a researcher looking for information about the type of rule-base used in a particular expert system.

In order to be able to respond to more detailed requests for information (like the one looking for information on the rule-base used in a particular expert system), an information retrieval system would have to have some facility for establishing the actual *content* of each text. This is achieved through the process of indexing. This indexing process has two purposes. The first purpose is to identify the particular text segments or sequences which are considered to pertain to the actual subject or content of the text and the second purpose is to convert these text sequences into some suitable form for storage in the text index and for subsequent matching during the retrieval phase. Indexing provides much more detailed information about the text than classification. The first part of my research

has been directed at the second half of the indexing task; the conversion of content-bearing text sequences[1] to a suitable representation for storage and subsequent matching. I have therefore assumed that there exists some way of identifying and extracting these analytics. Given that there exists a full linguistic description of the text, including morphological and syntactic information, it is possible to use this information in the analytic extraction process. The approach that was taken by Fagan, as described earlier [Fagan 1987], used these syntactic descriptions of text to extract index phrases. It is possible to define a grammar specifying what sequences of syntactic functions usually constitute good content-bearing text segments. This grammer could then be specified and implemented using a set of augmented transition networks or some other suitable formalism.

It is important that the analytics extracted during the first stage of the indexing process be of good quality. Since they are to be used as representatives of the text for retrieval purposes, it is necessary that they reflect accurately the actual information content of the text. Just as important however, is the quality of the representation used to store these analytics and the quality of the algorithm used to match those representations to the user query at retrieval time. It is no use having a high quality set of analytics for a text if a lot of the information associated with them is lost or not used during storage and matching. I am proposing a structured representation of analytics which is based on the full morpho-syntactic description of the text sequences and which encodes all of this information in the representation. The proposed scenario is that the analytic extraction process should identify large segments of content-bearing text sequences, which may be as large as full sentence clauses or whole sentences, and that these text sequences are then transformed into the proposed representation for storage and subsequent matching.

As described in section 2.1, the central function of an information retrieval system is the search and retrieval of information in response to user search queries. This involves the matching of the query against the representatives constructed for each text in the database, with the objective of establishing which texts are most likely to contain the information to satisfy the user's information need. Given that a certain quality of text representative has been extracted, this task of matching user queries to text representatives is central to the

[1] In this thesis the content-bearing text sequences extracted during indexing will be referred to as ***analytics***.

search requests. It may also be beneficial to compile frequency statistics about words or phrases in texts. This has a proven record of performance in information retrieval. It may be useful to use this approach even though the type of linguistically based representation and matching algorithm that I am proposing is an attempt to improve on this performance. In short, the best retrieval results are probably to be found by using a whole selection of retrieval strategies. It has been noted previously that, using two different retrieval strategies to search a given database for a given search request, the results produced are often not the same [McCall & Willett 1986]. It is possible then, that the best set of documents to be presented to the user would be composed of those documents that lie in the intersection of the sets of documents retrieved using different retrieval strategies.

One information retrieval system that is being developed along these general lines is the SIMPR project (Structured Information Processing Management and Retrieval). SIMPR is a 64 man-year project which is being funded by the Comission of European Communities under the ESPRIT II programme. The SIMPR project is aiming to achieve advances in information retrieval and includes research on language analysis, automatic indexing, techniques of automatic classification, machine learning and domain and task modelling. The project is designing and implementing an indexing module based on the type of analytic extraction process I have outlined above and it is investigating ways of combining multiple retreival strategies to provide an overall *best* set of retrieved texts. This combination of strategies would include the use of the type of matching algorithm that I am proposing here.

The SIMPR system has been prototyped on SUN 3/60s in SUN Common LISP and will be developed in C++ on SUN SPARC workstations. A good account of the SIMPR project can be found in [Smeaton 1990].

Chapter 4.

Linguistic Processing for Information Retrieval

4.1 Introduction.

As mentioned previously, the natural language processing software used for this research has been researched and developed at the Research Unit for Computational Linguistics at the University of Helsinki. This software analyses text at the morphological/lexical and syntactic levels. The morphological description of language used is based on Koskenniemi's two-level model [Koskenniemi 1983] and the basic ideas of the syntactic description are based on Karlsson's Constraint Grammar framework [Karlsson 1990]. The whole language formalism is language independent. Most work to date has been done on the analysis of English but work on the Finnish language is well underway and work on other languages (Russian, Danish, Dutch etc.) is anticipated.

On the whole, the parsing mechanism is quite modular in nature. It consists of interactions between the lexicon, disambiguation, clause boundary determination and assignment of syntactic functions. The whole linguistic process can be represented by the sequence of operations illustrated in Table 4.1.

Table 4.1: Sequence of Linguistic Processes

Morphological Analysis
Elimination of contextually impossible morphological readings
Determination of sentence-internal clause boundaries
Assignment of syntactic function labels

The morphological analysis module performs processing at both the morphological and lexical level. The first stage uses a lexicon to decompose the word into a morphological base form plus whatever series of prefixes and suffixes are present. Using lexicon information, an enumeration of all possible lexical categories of the word is produced. Most of these possible categories will be irrelevant or wrong when the context of the word

in the sentence is examined. During the second stage of processing, a disambiguation rule set is used to examine context and to discard as many of the irrelevant word readings as possible. These disambiguation rules also make it possible to identify some of the clause boundaries at this stage. Whatever boundaries remain are then explicitly identified in the third stage. This is done on the principle that knowing the positions of clause boundaries eases the task of assigning syntactic function labels to the words. This assignment of syntactic function labels is done at the final stage using a set of mapping rules which assign syntactic labels based on a mapping from the morphological level. I will now describe each of these stages in more detail.

4.2 Morphological Analysis: The Lexicon.

The master lexicon for English used here is based on Koskenniemi's Two-Level Model [Koskenniemi 1983]. The advantages of this formalism are economy of representation (most lexical entries represent more than one actual word) and the facility for attaching "useful information" to any lexical entry. The master lexicon is made up of many so-called mini-lexicons. One of these mini-lexicons contains the lexical entries for the morphological base forms of words and pointers to other mini-lexicons which contain the information about legitimate word endings for those words. The categorisation of the lexical entries is based on the syntactic properties of words. The basic distinctive factor is part-of-speech: Noun, Verb, Adjective etc. A word or *lexeme* which can possibly have more than one part-of-speech is coded with several lexical entries. For example, the word "*bottle*" can be either a noun or a verb so it must have two lexical entries:

bottle N;
bottl Ve-0;

The second parts (N, Ve-0) are pointers to other mini-lexicons. For example the mini-lexicon "N" contains endings for nouns of this type: nominative singular (), nominative plural (-s), genitive singular (-'s), genitive plural (-s') and pointers to some other mini-lexicons containing derivational suffixes (*-like, -ness, -less* etc.). The same is true for the

verb entries. For example the mini-lexicon for verb endings would include appropriate information about endings such as *-ing* (participle), *-ed* (past tense and past participle), *-able, -ably, -ability* (derivational suffixes resulting in change of part-of-speech), and some other endings.

One of the advantages of the lexicon fomalism mentioned above was the facility for attaching "useful information" to lexical entries. An example of this would be if we wanted to record that the verb form of "*bottle*" is a verb that takes both a subject and an object as arguments. This can be encoded in the lexical entry as follows:

```
bottl Ve-0 "= <SVO>";
```

The output of the morphological analysis then includes this information just after the base form of the word. The output also includes the lexical information dealing with the part-of-speech, case, and other information gathered from the lexicon. The output for each word in the RUCL language analysis consists of the text form of the word on the first line followed by one or more lines of analyses, one for each possible morphological and lexical reading of the word. The analysis of the word "*bottles*" is therefore:

```
bottles
    bottle " N NOM PL"
    bottle " <SVO> V PRES -SG3 @+FMAINV"
```

Each analysis line consists of two parts: the base form of the word and the morphological analysis. The first analysis line here says in effect that "*bottles*" is a nominative plural form of the noun "*bottle*" and the second line says that "*bottles*" is the present tense third person singular form of the transitive verb "*bottle*". The existence of more than one analysis line for this word-form indicates that the word-form is regarded as lexically ambiguous.

The task of morphological analysis is to analyse individual words, defined as strings with no embedded blanks. There exist however, some word-like units that contain embedded blanks but which should be analysed morphologically as single units. Examples are idioms, complex prepositions (e.g. "*in spite of*"), certain adverbials and some other categories, and nominal compounds (e.g. "*tea time*"). The approach to this taken here is to identify such

units prior to the morphological analysis process and to convert them into one single string. This is done during a preprocessing stage and units such as *"in spite of"* are converted to strings like *"in=spite=of"* which can be analysed as a single unit. The list of compound expressions to be dealt with in this manner contain some 6,000 compound expressions from the Collins Cobuild Dictionary [Collins 1987] that can be trusted to be compounds in all contexts, and not strings which are accidentally adjacent to each other as parts of different structures. In addition, there are about 200 idioms like *"in spite of"*. Most of the analysis of compounds is left to the syntactic analysis stage as compounding is seen mainly as a syntactic phenomenon.

The description of the morpho-syntactic labels used in the English Master Lexicon is based almost entirely on [Quirk *et al.* 1985]. Every word-form in the lexicon has a part-of-speech label. The list of possilbe part-of-speech labels is included in Table 4.2.

Table 4.2: Valid Part-of-Speech Labels

A	Adjective
ABBR	Abbreviation
ADV	Adverb
CC	Co-ordinating conjunction
CS	Sub-ordinating conjunction
DET	Determiner (*the, any,...*)
@INFMARK	Infinitive marker (*to*)
INTER	Interjection
N	Noun
NEG-PART	Negative particle (*not*)
NUM	Numeral
PCP1	Present participle (*walking*)
PCP2	Past participle (*walked*)
PREP	Preposition
PRON	Pronoun
V	Verb

In the output of the morphological analysis, the part-of-speech label is found immediately to the right of the morphological base form of the word, except when other "useful information" has been included in the lexicon (enclosed in angle brackets "< >"). After

the part-of-speech, is printed any other morphological or lexical information that is available. A more detailed description of feature patterns and labels can be found in [Heikkil 1990].

4.3 Disambiguation.

As a consequence of the fact that morphological analysis works only at the word level and produces all possible analyses of each word-form, most of the analyses are inappropriate and about 50% are contextually impossible. The task of the disambiguation module is to eliminate as many inappropriate morphological analyses as possible. After disambiguation, only the appropriate analyses and the least detectable of the inapproprate ones should be left. Disambiguation is based on the Constraint Grammar formalism developed by Karlsson [Karlsson 1990], the main features of which are presented here.

The disamiguation and syntactic analysis modules process the morphologically analysed text in sentence-sized chunks. Disambiguation is carried out according to a set of constraint rules which are written as LISP expressions with the following format:

("object_of_rule" operation_type "ipret" (context_1) (context_2) ... (context_n))

...where...

Object_of_rule can be a word or a feature.
Operation_type is one of the following:

"=!!" X is the only correct interpretation for object_of_rule Y in context Z, and for Y X is impossible in any other context.

"=!" X is the only correct interpretation for object_of_rule Y in context Z.

"=0" X is impossible for object_of_rule Y in context Z.

Ipret can be a real feature, like "NOM", or it can be a set of features, like "A/PCP2", which is declared elsewhere as a set consisting of the name of the set and the features and/or words in the set. If ipret is a set then a rule of type "=0" can eliminate more than one interpretation line.

Context_1 to n consists of one or more context-conditions which are predicates examining positional features of words in sentence chunks.

An example of such a constraint rule is

(@w =! VFIN (NOT *-1 VFIN) (NOT *1 VFIN) **CLB-C)

which states that a verbal finite reading (VFIN) is correct if there are no other verbal finite readings in the *span* from the current word up to the next clause boundary (including both sentence beginning and end) in each direction.

One important property of the present disambiguation formalism is that the constraint rules are independent of each other in the sense that a rule refers only to the input and never to another rule. The fact that the input may change as a result of the operation by other rules is in principle accidental to the function of a given rule. This lack of rule order necessitates another requirement on the rules: each rule must be true. Rule overlap is not fatal in itself but it can become fatal if overlapping rules are in conflict with each other.

In the present formalism, clause boundary detection is not a goal in itself but rather is incidental to the main task of disambiguation. Clause boundary detection is necessary purely because of its usefulness for the other components in the syntactic analysis. Input with clearly marked clause boundaries is simply easier to analyse than input without marked boundaries. Clause boundary detection is mostly done within disambiguation but there are cases where almost any word can mark the clause boundary and this can't be handled within disambiguation. Relative clauses with no relative pronoun and the endings of subordinate clauses preceeding their mother clause (e.g. "*If you go the rest will too.*") are fairly frequent examples. So far, no uniform approach has been found for clause boundary detection in

these cases. This may however, be of little importance if it eventually seems possible to successfully assign syntactic functions without all of the clause boundaries being explicitly marked.

4.4 Assignment of Syntactic Functions.

The objective of the syntactic processing is to assign to each word an unambiguous syntactic label based on the so-called functional dependency syntax. The main idea is to indicate whether a word is functioning as a head or as a modifier in a clause. This is the same approach to syntax that was taken by Fagan [Fagan 1987], as discussed earlier. To repeat the example, the phrase "*automatic extraction of index terms*" is made up of the head "*extraction*" pre-modified by the word "*automatic*" and post-modified by the prepositional construct "*of index terms*" which consisits of a preposition, "*of*", and a prepositional complement (which is the head of the prepositional construct), "*terms*" which is pre-modified by the word "*index*".

The syntax here is surface-near; i.e. no structures are postulated that are more abstract than being in direct linear correspondence with the real word-forms occuring in the sentences to be described. For all words with modifier function, the syntactic function label assigned also gives some indication of the head which is being modified. The direction of the head is indicated using an angle bracket ("<" for left; ">" for right) and the category is indicated, for example by indicating that the closest member of category X is the head of modifier Y. All syntax labels are prefixed by the symbol "@" for recognition purposes. An example of a modifier label is "@*AN*>", which describes an adjective ("*A*") modifying a noun ("*N*") that is to the right (">") in the sentence. It need not be stressed that the present collection of codes does not exhaust all intricacies of English syntax. It is however, delicate enough to capture those aspects of clause structure that are relevant for information retrieval purposes.

Assignment of syntactic functions takes as input sentence-sized chunks of text which have

been morphologically disambiguated as fully as possible. Approximately one third of words are syntactically unambiguous and they will have been assigned their syntactic label directly from the lexicon. An example of such a word is the verb form of the word "*bottles*" shown earlier, which had the syntactic function label *@+FMAINV* assigned to it in the lexicon. If these words have been fully morphologically disambiguated then no further processing is necessary at the syntactic level. For the remaining words, syntactic analysis takes place in two stages. The first stage is a direct mapping from morphology to syntax. The idea behind this is that from any morphological feature there is a direct mapping to a limited number of syntactic functions. Different morphological features have different mappings to syntactic functions. These mappings are expressed as lists consisting of

- a string of the input - usually a morphological feature or a base-form.
- a context condition. (optional)
- the syntactic function(s).

The mapping functions are listed in order of increasing specificity. The mappings to single syntactic functions only are given first, and those giving mappings to all allowable syntactic functions that a word or feature can have are specified last. An example of a specific mapping would be from N to @SUBJ in a clause-initial position:

("N" ((-2C CLB) (-1C DET/GEN) (1C VFIN)) "@SUBJ")

The context condition in this case checks that there is a clause boundary two positions to the left, (-2C CLB), the initial word (one to the left) is a determiner or is in the genitive, (-1C DET/GEN), and that the following word is a verb, (1C VFIN). In a non-restrictive mapping there would be "NIL" in place of the context conditions.

At the end of the first stage, each word will have been assigned one or more syntactic function labels depending on the specificity of the mapping used. The second phase of syntactic analysis is similar to disambiguation at the morphological level. Each word that has more than one syntactic function label is processed in the second stage to reduce the set of labels to only the appropriate one(s). This is achieved through a set of syntax rules which are expressed in basically the same way as the morphological disambiguation rules.

An example of such a rule states that a word can have the syntactic function "@I-OBJ" (indirect object) only if there is somewhere to the left a word with the label "<SVOO>" (marking a verb governing two objects and a subject):

(@w =s0 "@I-OBJ" (NOT *-1 I-OBJ))[2]

A further component of the syntax module is the operation of the so-called *Uniqueness Principle*, which states that there may be at most one occurrence of a given syntactic tag in a clause. What this in effect means is that whenever one of the functions has been assigned as the only syntactic reading of a word, no other un-coordinated instances of the same function are allowed within the same cluase. The Uniqueness Principle has been incorporated into the formalism itself so no additional rules needed to be written for this purpose. An example of the output from syntactic analysis is:

```
remove
   remove " <SVO> <SV> V IMP VFIN @+FMAINV "

the
   the " <Def> DET CENTRAL ART SG/PL @DN> "

fuel
   fuel " <-Indef> N NOM SG " @NN>

pump
   pump " N NOM SG " @OBJ

and
   and " CC @CC "

filter
   filter " N NOM SG " @OBJ
   filter " <SVO> <SV> V IMP VFIN @+FMAINV "

from
   from " PREP " @<NOM @ADVL

the
   the " <Def> DET CENTRAL ART SG/PL @DN> "

pump
   pump " N NOM SG " @NN>

unit
   unit " N NOM SG " @<P
```

[2] The set I-OBJ contains the label "<SVOO>".

We can see that in this example most of the words have been fully disambiguated. The exceptions are "*filter*" which is morphologically ambiguous between a noun and a verb (and so is therefore syntactically ambiguous between the object or a verb) and the word "*from*" which is syntactically ambiguous between a preposition and an adverbial. The label "@NN>" indicates a noun modifying another noun to the right and the label "@<P" indicates a prepositional complement (post-modifying a preposition to the left).

I have included as APPENDIX A an example of the output from each stage of the linguistic analysis of a sample sentence. For a detailed and comprehensive description of the linguistic processing described here, the reader is referred to [Karlsson 1991].

4.5 Problematic Linguistic Phenomena.

I have mentioned previously the existence of some problematic linguistic phenomena at some levels of linguistic processing and how these problems can only be overcome by recourse to higher level processing. At the morphological level, problems are caused by **lexical ambiguity** which occurs with words like "*bank*", "*glance*" and "*run*" which can be categorized as either nouns or verbs, and the word "*mine*" which could be categorized as a noun, verb or pronoun. In most cases it is possible to disambiguate these words by refering to the context in which they are used. For example in the sentence "*The athlete ran twenty miles*", "*ran*" can be correctly categorized as a verb by analysing the categories of the other words in the sentence. This disambiguation process takes place explictly in the language analysis process and has been described earlier. In some cases, however, it is impossible to disambiguate by refering only to the syntactic level and some semantic information is necessary. The sentence "*The CPU signal interrupts transfer activity*" gives an example of such a case. Here the words "*interrupts*" and "*transfer*" can both be analysed as nouns and verbs and it is impossible to know which is which: "*The CPU signal interrupts transfer activity*" or "*The CPU signal interrupts transfer activity*". In the natural language processing system that is described here, such unresolvable ambiguity is reflected in multiple

analyses of words in the output of linguistic processing (like the word *"filter"* in the example earlier).

At the higher level, problems are caused by **syntactic ambiguity** which is a result of singularly unambiguous words being combined in different ways to form ambiguous constructs. This type of ambiguity cannot be disambiguated without recourse to semantic level processing. Since there is no semantic processing in the language analysis software used here, this ambiguity must be dealt with in some other way. One possiblility is to ask the user to intervene and to use human semantic knowledge to disambiguate. I feel however, that this is undesireable and I propose that syntactic ambiguity should be retained and that it can be implicitly encoded in a syntactically-based text representation. This proposal is based on the hypothesis that, in an information retrieval context, it is not necessary to explicitly disambiguate in order to have a single correct analysis for every language construct but rather, it is sufficient to encode all possible interpretations of each construct so that each interpretation is available for matching at retrieval time. This approach to ambiguity was also taken by Metzler previously [Metzler & Haas 1989]. In order to support this proposal for handling ambiguity in structured text representations, I carried out a survey of the types and frequency of syntactic ambiguity in natural language and, in particular, I studied the following types of ambiguity which must be taken into account in any information retrieval system.

4.5.1 Distribution over Conjunction.

In many cases there is ambiguity associated with conjoined constructs. This ambiguity can arise for two reasons - the distribution of modifiers over a conjoined head construct and the analyses of words which can function as either heads or modifiers in a conjoined construct modifying a head. Examples of each of these are:

Inspect the bearing cups and cones.
@+FMAINV @NN> @OBJ @CC @OBJ

Inspect the hub and bearing components.
@+FMAINV @NN> @CC @NN> @OBJ
@OBJ

In the first case the distribution of the modifier "*bearing*" is ambiguous because the text could be dealing either with "*bearing cups*" and "*cones*" or with "*bearing cups and bearing cones*". This ambiguity is retained after the linguistic processing because the syntactic label "@NN>" refers to a noun modifiying some other noun to the right. It does not specify whether the modified head is a single noun, a nominal compound or a conjoined construct. In the second example, the syntactic function of the word "*hub*" is ambiguous between being an object or a modifier of the word "*components*": i.e. the text could refer to the "*hub*" and the "*bearing components*" or the "*hub components and the bearing components*".

Conjunction can also cause ambiguity when used to conjoin prepositional phrases. This causes particular problems becuase prepositional phrases often present ambiguity problems in themselves.

4.5.2 Prepositional Phrase Attachment.

Prepositions are probably the most frequent and complex contributors to syntactic ambiguity. The complexity of prepositions is to some degree simplified in our case because of the explicit coding of complex prepositions in the master lexicon. The fact that complex prepositions such as *along with, out of, except for, owing to, etc.* and the even more complex multi-word prepositions such as *in case of, in spite of, in accordance with, by means of, on behalf of, for the sake of, etc.* are dealt with as single word units reduces somewhat the problems that have to be dealt with at later processing stages.

The problem with prepositions does not lie in the individual words or prepositions but in the way that prepositional constructs are often combined, sometimes in very complex ways. An example of this is the unlikely, but yet grammatically correct sentence:

> *Remove the seal from the fuel pump with the red top to the right of the engine in the car with the dent in the back from a crash on the road to Dublin during the icy spell of weather in 1988.*

This sentence has no less than thirteen prepositional phrases. The complexity associated with multiple prepositional phrases is one of reference or attachment; i.e. to what is the preposition referring. If we take a simple example:

Remove the bolt with the square head.
@+FMAINV @OBJ @PREP @AN> @<P

Remove the bolt with the red screwdriver.
@+FMAINV @OBJ @PREP @AN> @<P

In the first case the prepositional phrase "*with the square head*" should be attached to the noun "*bolt*" whereas in the second case "*with the red screwdriver*" should be attached to the verb "*remove*". Both of these sentences are syntactically identical so there is no obvious way to disambiguate them. The ambiguity can become much more complicated when there are several prepositional phrases since prepositions can possibly be attached to preceeding prepositional complements as well as to the verb and object or subject of the sentnece. A much used example of prepositional ambiguity which was mentioned earlier, is

I saw the man with the telescope.
@SUBJ @+FMAINV @OBJ @PREP @<P

I saw the man on the hill with the telescope.
@SUBJ @+FMAINV @OBJ @PREP @<P @PREP @<P

In the first case here the phrase "*with the telescope*" can be attached to either "*saw*" (I was using the telescope to see the man) or "*the man*". In the second case "*with the telescope*" can again be attached to "*saw*" and "*the man*" but can also be attached to "*the hill*" (the man that I saw was on a hill that also had a telescope on it). There is no way to know with 100% confidence which attachment is correct so I would argue that the best way to handle this is to ensure that all possible attachments are somehow preserved or encoded during indexing so that during the retrieval task, each possible interpretation is available for matching and possible retrieval: i.e. one would expect the above sentence to be retrieved in response to user searches for information about both "*men with telescopes*" and "*hills with telescopes*".

As stated earlier, conjunctions can combine with prepositions to form a type of ambiguity

that I call **attachment of prepositional phrases over conjunction.** A simple example of this would be,

The nuts and bolts in the box.
@SUBJ @CONJ @SUBJ @PREP @<P

where we definitely know that the prepositional phrase "*in the box*" can be attached to "*bolts*" but we cannot be sure whether or not it should be attached across the conjunction to "*nuts*". This is the simplest case. Things can become more complicated if there are prepositional phrases on each side of the conjunction. For example

I saw a cop in a trenchcoat and a robber in the alley. [Metzler *et al.* 1990]
@SUBJ @+FMAINV @OBJ @PREP @<P @CONJ @OBJ @PREP @<P

I saw a cop in a trenchcoat and a gun in the pocket.
@SUBJ @+FMAINV @OBJ @PREP @<P @CONJ @OBJ @PREP @<P

I saw a cop in a trenchcoat and a robber from the balcony.
@SUBJ @+FMAINV @OBJ @PREP @<P @CONJ @OBJ @PREP @<P

I saw a cop in a trenchcoat and a robber in his custody.
@SUBJ @+FMAINV @OBJ @PREP @<P @CONJ @OBJ @PREP @<P

In all cases the attachment of the first prepositional phrase is straightforward since it may only be attached to either of "*saw*" or "*cop*". The attachment of the second prepositional phrase, however, is not as simple. These examples show that the second prepositional phrase may be attached to the "*robber*" and may also be attached across the conjunction to "*saw*", "*cop*" or "*trenchcoat*". The attachment is further complicated by the fact that in some cases there may be more than one valid interpretation (even when using all forms of higher level knowledge). The first case above is an example of this. The prepositional phrase "*in the alley*" can obviously be attached to the "*robber*" but it can also be validly attached to the "*cop*" (both the cop and the robber may be in the alley). Such distinctions cannot be made using just syntactic information: all of the examples above have an identical syntactic construct and can only be truely disambiguated using semantic level processing. I believe that cases such as these provide support for my proposal to somehow deal with the problems of ambiguity rather than presenting them to a user. Because of the frequency of occurance of prepositions and conjunctions and the complexity of ambiguity that can

result from combining them, I feel that the explicit disambiguation of such constructs would prove too great a cognitive task for the average information retrieval system user.

4.5.3 Adverbial Ambiguity.

The problem of adverbial ambiguity is like that of prepositions, one of reference or attachment. Adverbs are words like *quickly, slowly, possibly, probably etc.*. Not all adverbs are ambiguous. For example in "*the robber quickly ran into the woods*", we know that the text is dealing with a robber who "***ran quickly*** *into the woods*". This may seem obvious but in the sentence "*the robber probably ran into the woods*", we cannot be sure whether "*the robber* ***probably ran*** *into the woods*" or "*the robber* ***probably*** *ran* ***into the woods***".

Although the problem of adverbial ambiguity is similar in nature to that of prepositions, it is nowhere near as common. The statistics presented later show how few words belong to the adverb category in the sample texts used. Of particular importance to the application under consideration here is the probability of an adverb being used in a user retrieval query. For example, a user is unlikely to request texts dealing with "*robbers who probably ran into woods*". It is possible however, that a user would specify a retrieval request containing an adjective which should match the use of the base form of the adjective in an adverbial context. For example it would not be unlikely for a user to expect an information retrieval system to match the phrases

Efficient algorithms.
@AN> @NPHR

An algorithm for efficiently sorting arrays.
@SUBJ @PREP @ADVL @+FMAINV @OBJ

I would claim that this also provides support for the idea of building text representatives based on rich syntactic information and using complex matching techniqes at retrieval time to facilitate the matching and retrieval of such constructs.

4.5.4 Ellipses.

A good account of ellipses is given in [Metzler *et al.*, 1990]. They describe ellipses as a problem of conjunction where pieces of conjuncts are omitted. They reference a paper by Huang [Huang 1984] which classifies ellipsis into three categories :

Gapping: When a verb, and possibly its object, are omitted from a conjunct (but not from the leftmost one). For example, "*John went to Scotland and Jim to England*".

Right node raising: When the object is omitted from a conjunct (but not the rightmost one). For example, "*I saw and Joe heard the shooting*". Further complexity and ambiguity can be introduced if the first verb is optionally transitive. For example, "*I ate and you drank everything they brought*" [Metzler 1989], as in this case I could have eaten either something that is not referred to or everything they brought.

Reduced conjunction: When conjuncts contain incomplete constituents. For example, "*The professor read through and graded the exam scripts.*" where the prepositional phrase "*through the exam scripts*" is incomplete in the first conjunct.

Traditional methods of processing elliptical constructs involve identifying the omission (usually using the complete cojnuct as a pattern) and filling it by copying the appropriate filler from the complete conjunct. For example, in "*John went to Scotland and Jim to England*" this would involve identifying this as a case of gapping with the verb "*went*" being omitted from the right hand conjunct and then copying (or creating an explicit link from) the verb from the left hand conjunct to fill the gap. This can be a complex procedure and in cases of right node raising with optionally transitive verbs it is impossilbe to tell whether there should be an object or not.

Again, I believe that representing elliptical constructs in a rich structure based on full syntactic information will allow a retrieval matching algorithm to *infer* implicit information and successfully match and retrieve the appropriate interpretations.

4.5.5 How Often Does Syntactic Ambiguity Occur ?

There as been no study that I am aware of that has looked at the frequencies of different types of ambiguity in texts and how it effects text retrieval. Some work on more general topics has been done, however. A group of researchers at Syracuse University in the U.S. have done work on investigating the distribution of anaphors and how anaphoric resolution would effect retrieval efficiency [Bonzi & Liddy 1989], but this linguistic phenomenon occurs at the discourse level and is beyond the scope of the work reported here.

Another group at the University of Massachusetts at Amherst has investigated how the interpretation of nominal compounds can be done using a knowledge intensive algorithm, which is important for information retrieval [Gay & Croft 1990]. Their conclusion, based on a sample of some hundreds of nominal compounds, is that it can be done automatically but they question its worth in terms of the improvements it could make on retrieval performance. In my case, the problem of ambiguity in nominal compounds is handled by the lexicon during linguistic processing, but the important point to be taken from their work is that they found nominal compounds to be very widespread, especially in user queries.

Despite the fact that there have been no studies of ambiguity in text there have been several studies of the structure of texts. Knowing the type of constructs and the classes of words that can cause ambiguity, the statistics provided by these studies should enable us to make some judgements as to how often ambiguity arises.

A comprehensive study of the syntactic structure of English text was carried out at the University of Gothenburg by Alvar Ellegaard and reported in [Ellegaard 1981]. The study was based on a subset of texts from the Standard Corpus of Present-Day American English, compiled at Brown University. This corpus is made up of 500 texts of about 2,000 words each, covering a wide range of subjects and prose styles. The subset of the Brown corpus used for the Ellegaard survey consisted of 64 texts from four different categories, Popular fiction (N), Journalism (A), Literary essays (G), and Science (J) (16 texts from each category). What Ellegaard did was assign students to do manual syntactic analyses of the

texts and turn the results into machine-readable format. They then compiled a huge amount of statistics on the syntactic structures used in the texts.

The texts that Ellegaard worked with consisted of 128,000 words grouped into 62,381 phrases, further grouped into 17,900 clauses or 7,100 sentences. Of the sentences in the collection, 29.3% contained only one clause, 49.1% contained 2 or 3 clauses, 16.9% contained 4 or 5 clauses and only 4.7% contained 6 or more clauses. On average, the sentences contained 2.5 clauses which were evenly distributed over the different text types. The average clause length was 7.1 words which were longer in the scientific texts and shorter in the popular texts. The depth of embedding of clauses in sentences was 1.9 on average. A sentence like "*He wished to leave when he was ready*" contains a depth of embedding of 3, while the sentence "*I came, I saw, I conquered*" contains a depth of 1. The average depth of embedding of clauses is often used as an indicator of the complexity of the text.

One of the first interesting points to emerge from the Ellegaard work is the fact that the different text types do not have many distinguishing characteristics as far as syntactic constructs are concerned. What distinguishes popular texts from scientific ones most is the length and complexity of the individual phrases making up the clause, but at the clause level all the texts are quite similar. Phrases are only an average of 1.7 words long in the sample of popular texts and 2.4 in scientific ones. Only 4% of phrases in popular texts are more than 3 words while the corresponding figure for scientific ones is 20%. Considering the comparatively small differences between text types, the statistical results presented by Ellegaard provide a sound basis for making assumptions about the distributions of clauses, clause types, phrase types, phrase constituents and word classes, in texts.

The statistical results presented in the Ellegaard work can be classified under the general headings; "**The structure of sentences in terms of clauses**", "**the structure of clauses in terms of constituents**" and "**word classes**". Using the statistics presented for the frequency of words from different classes, I believe it is possible to make certain predictions about the frequency of different types of ambiguity, as described in the previous sections. The statistics for word class frequencies presented in Table 4.3 have been taken from Ellegaard.

Table 4.3: Frequencies of Words of Different Categories.

	Text type				
Word class.	**N**	**A**	**G**	**J**	**ALL(%)**
Common nouns	187 180	260 254	242 217	294 261	**23.7**
Proper nouns	26 39	59 87	19 29	10 8	**3.5**
Articles	94 86	102 104	112 95	123 101	**10.3**
Numerals	7 10	23 47	12 11	28 21	**2.0**
Adjectives	50 47	60 60	84 90	94 107	**7.4**
Adverbs	66 82	31 41	53 57	41 51	**5.3**
Negations	10 10	5 3	6 9	3 4	**0.6**
Pronouns	174 162	64 72	109 120	48 67	**10.2**
Verbs	231 226	173 169	176 179	163 160	**18.5**
Prepositions	104 96	117 120	128 125	152 143	**12.3**
Infin. *to*	15 15	16 13	14 15	11 11	**1.4**
Conjunctions	53 62	43 47	68 63	49 65	**5.6**
Unclassified	0 1	0 1	0 6	9 14	**0.4**
Rest	1 1	0 1	0 1	0 1	**0.1**

The reason that there are two columns for each type of text is that the 16 texts from each category were divided into two groups of eight. This was done to obtain a measure of the homogeneity within each group. The important column (ALL %) is the rightmost one which gives, for each word class, the overall percentage of words in the texts that belonged to that class. As would be expected, nouns (27.2%) and verbs (18.5%) are the most frequent word types. Of importance to us in looking at ambiguity however, is the fact that 12.3% of the words in the texts are prepositions and this is the third most common word category. I have earlier outlined the complexity involved with the problem of prepositional phrase attachment. Combining the complexity of the problem with the frequency of occurance seen here, it is obvious that some method of automatically handling the ambiguity involved with prepositional attachment is required. I will describe later how prepositions are included in my proposed structured representation of text in such a way that no explicit disambiguation is necessary but all interpretations are preserved and available for matching at retrieval time.

The other word types that can cause the types of ambiguity we are concerned with are conjunctions (5.6%) and adverbs (5.3%) . As mentioned earlier, the problem with adverbial ambiguity is one of attatchment, like that of prepositions. The difference between the two is that adverbial ambiguity is neither as complex, since there are relatively few possible attachments that may exist for an adverb (usually only two), nor as common, since adverbs occur only half as frequently as prepositions in text. I have also mentioned earlier the fact that adverbs may be of little importance in an information retrieval environment because of the role they play in the retreival process.

Although conjunctions are not very common, they can contribute to ambiguity in many ways - distribution of heads/modifiers, attachment of prepositions, and ellipses. It would be useful to have statistics that differentiate between the different uses of conjunction in text. The text representation that I will propose is capable of representing ambiguity associated with the distribution of heads and modifiers and the distribution of prepositions over conjunction and although it is also capable of dealing with elliptical constructs, it is likely that in an information retrieval system such constructs would be disassembled into more than one analytic at indexing time.

In order to provide further argument against asking users to perform explicit disambiguation, in the sample texts of 128,000 words there are 463 possible disambiguations in the texts on Popular fiction, 399 in the Journalistic texts, 494 the Literary essays and 498 possible disambiguations in the Scientific texts. Assuming, in this case, that all occurances of these word classes are in some way ambiguous, the user would have to make and average of 400 disambiguations per 2,000 words. It is unrealistic however to assume that all occurances of words of these classes are ambiguous. Modifying the assumption then, to state that 50% of prepositions, adverbs and conjunctions are in some way ambiguous, leaves the user disambiguating 200 constructs per 2,000 words of text, or one disambiguation for every 10 words. This figure, along with the complexity of the disambiguation task for some constructs, illustrates the advantages to be gained in finding some method of automatically handling syntactic ambiguity in an information retrieval environment.

Chapter 5.

A Structured Representation of Text

5.1 Introduction.

Earlier in this thesis I identified several aims for the design of a structured representation of text for use in information retrieval. The representation should be *rich* enough to encompass all aspects of syntactic description and to make this information available to a matching algorithm for text retrieval, the representation should be *flexible* enough to encode implicitly all lexical and syntactic ambiguity remaining after the linguistic analysis and disambiguation processes so that each interpretation of an ambiguous construct is available for retrieval, and the representation should be *powerful* enough to provide more effective text retrieval than straight string matching of normalized phrase indexes.

In designing such a text representation, I had to constantly bear in mind the application for which it would be used. Of particular importance in my case was the fact that this text representation was to be used for the retrieval of texts from a database, so some complex matching algorithm would have to be designed in order to match query representations to text representations during the retrieval process. There existed the possiblity that the definintion of a structured representation of text may be in some way constrained by the fact that these structures would have to be matched with some degree of flexibility, with the aim of establishing whether two text segments were dealing with the same topic. This constraint was removed however, by surveying some current structure-matching techniques. I discovered that some very complex matching algorithms were already in use and much research had been done in making these algorithms quite efficient [Shapiro & Haralick 1979] [Haralick and Elliot 1979] [Shapiro & Haralick 1981]. In particular, one application that had already used complex matching algorithms to great effect was the matching of complex molecular structures in large chemical databases [Levinson 1984]. The discovery that such matching algorithms existed for matching these complex three-dimensional structures removed any restrictions that I may have considered to exist for the definition of a structured text representation.

Whatever the structure or complexity of the text representation that I devised, it had to be based on the morpho-syntactic description of language provided by the linguistic analysis software described in the previous chapter. Although linear in nature, this description is

quite rich in information and because it is based on the functional dependency syntax, it contains much implicit information about the hierarchic relalationships between words acting as heads and modifiers. The construction of a text representation around this analysis would consist of extracting this information and making it explicit in the structure of the representation and then augmenting this through the full compliment of morphological and lexical information provided.

In the initial stages of my research, I investigated two possible formalisms for a structured representation of text. The first was a simple linear structure based on the same linear structure of the linguistic output, and the second was a tree-structure constructed by extracting the relationships between words from the syntactic function labels of the linguistc output. It became apparent very quickly that the tree-structured representation was much more powerful than the linear one. The tree structure made it easier to encode the dependency relationships between words and it seemed likely that the tree structure would provide for the easier development of a matching algorithm since many tree searching algorithms already existed. An example of the type of tree structure that would be used to represent the phrase "*An analysis of schema for classsifying the subject of textual information*" is given in Figure 5.1 below.

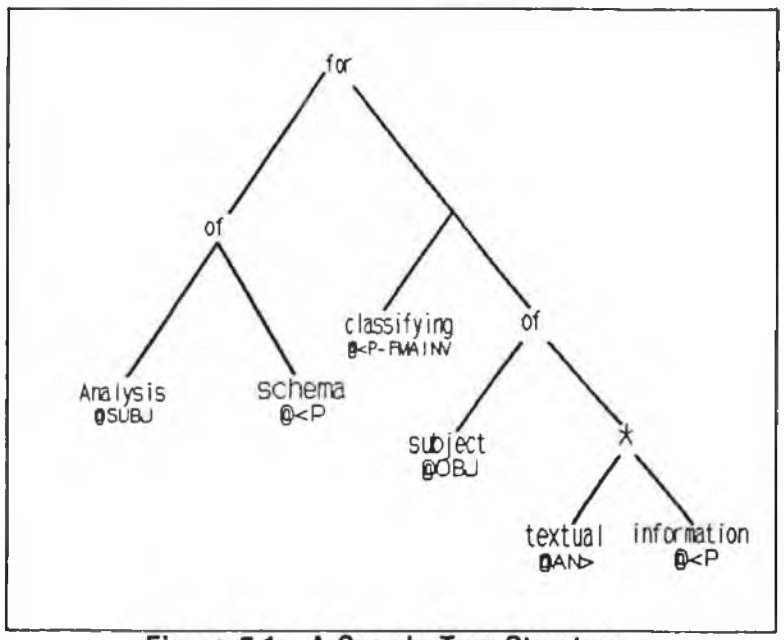


Figure 5.1: A Sample Tree Structure.

An important consideration that was also borne in mind during the initial investigations of suitable structures for text representation was the size of text segments that would be encoded in these structures. A limitation that I felt existed in previous work on using phrases as text representatives was that the phrases were usually limited to two or three words. I believed that structured text representatives should be based on anything from two or three word nominal compounds to complete sentence clauses or whole sentences. A simple linear structure would become too complex at this level whereas the hierarchic nature of the tree structure is perfectly suited to representing the relationships between clauses and clause constituents at any level.

Having decided that a tree structure was suitable for representing text segments for my purposes, I then set about organizing the structure in such a way as to achieve the aims outlined at the beginnig of this section: richness, flexibility and power. The resulting text representation has been given the name of **Tree Structured Analytic** (TSA), where an analytic is a text segment identified as being relevant to the text content and is extracted during the indexing phase of an information retrieval system. The characteristics of the TSA representation are as follows:

5.2 Tree Structured Analytics.

After much modification, I finalised a tree structure that I felt encompassed all of the desired features for the purpose of information retrieval. The main characteristics of that structure are:

- The basic structure is a **binary tree.**
- Word information is stored at the leaf nodes.
- Word information consists of the text form of the word, the morphological base form of the word and the syntactic function label of the word.
- Conjunctions are stored at the ancestor node which governs all of the conjunct structures (sub-trees).

- Prepositions are stored at the ancestor node which governs the prepositional complement construct (sub-tree).
- Direct pre-modifier/head relationships are emphasised by an asterisk (*) at the parent of the head noun.

An example TSA for the sentence "*Remove the fuel pump sediment bowl and filter from the top of the pump unit*" is given in Figure 5.2.

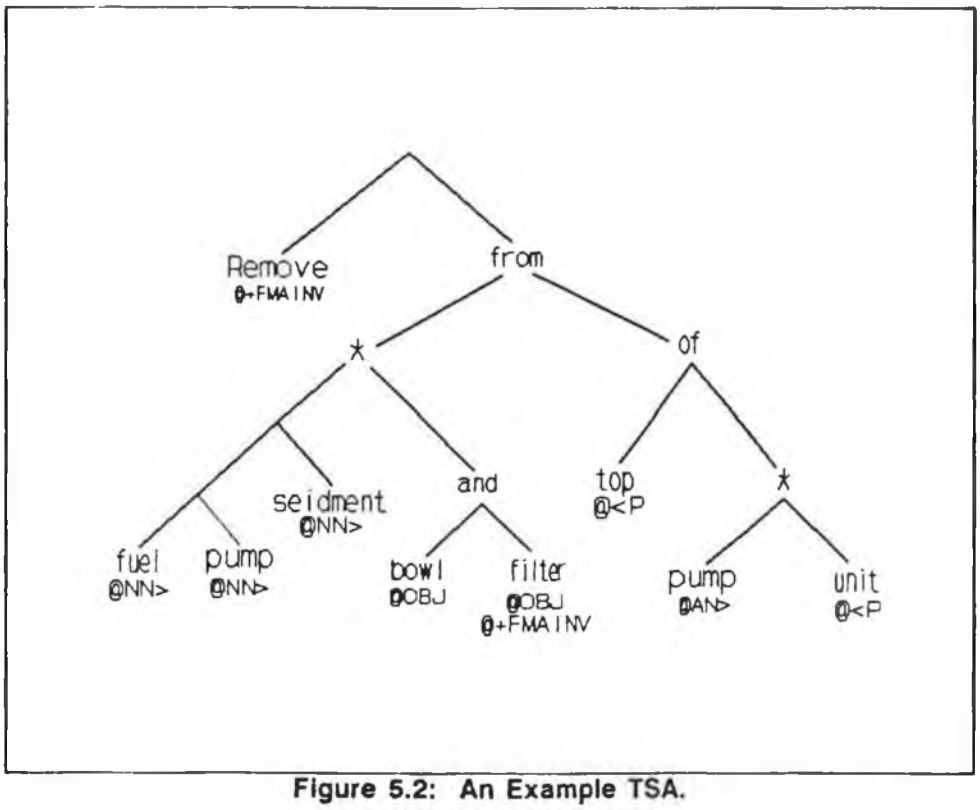


Figure 5.2: An Example TSA.

In the interest of legibility the morphological base forms of the words have been omitted and only the text forms of prepositions and conjunctions have been included in the TSAs illustrated in this thesis, but all of this extra information is stored in the actual representations. I found however, that the morphological feature information and the lexical information supplied by the lingusitic software was too detailed and too specific for the purpose of information retrieval so this is not stored. Note too, that in the above

diagram, there are two syntactic function labels present for the word "*filter*" which is lexically ambiguous in this context. This ambiguity was present after the lingusitic analysis and has been retained in the text representation with the purpose of matching against the noun or verb forms of the word at retrieval time. All ambiguities resulting from multiple lexical or syntactic analyses are stored in this fashion at the word nodes.

The organisation of the representation as binary trees also holds the key to the method of dealing with the other forms of more complex ambiguity detailed in Chapter 4. Because the representation is a tree structure, the matching algorithm is basically a tree traversal algorithm (although much enhanced). It is in the traversal of different paths through these tree structures at matching time that the algorithm can match against different interpretations of ambiguous constructs. This is best explained through the use of some examples.

5.3 Encoding Ambiguity in Tree Structured Analytics.

5.3.1 Distribution over Conjunction.

Examples were presented earlier for two different types of ambiguity due to conjunction. The first was due to the ambiguous distribution of a modifier to a head over a conjunction and the second was due to the ambiguous nature of a noun which could function as either a head or a modifier. The examples given were the phrases "*Inspect the bearing cups and cones*" and "*Inspect the hub and bearing components*". These two types of conjunctive ambiguity are dealt with in a slightly different way in the Tree Structured Analytic representation. In the second phrase most of the ambiguity is directly associated with the word "*hub*" and this is reflected in the structure by the presence of two syntactic function labels. The TSA structure for this phrase is given in Figure 5.3.

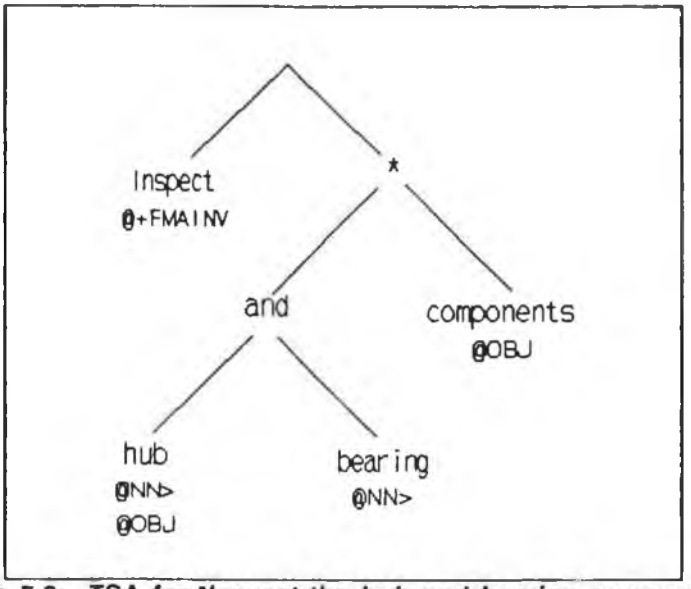


Figure 5.3: TSA for "*Inspect the hub and bearing components*"

Becuase of the fact that there is only one TSA structure built for each text sequence, the structure will always favour one interpretation of an ambiguous construct over another, even though both are encoded. In this case the favoured interpretation is that of "*Inspect the hub components and the bearing components*" because the word "*hub*" has been included under the direct pre-modifier/head relationship marked by the asterisk. The second interpretation is encoded through the second syntactic function of the word "*hub*". This means that a retrieval query relating to "*Inspecting the hub*" will score a perfect match against the above TSA becuase the matching of the word "*hub*" will take place against the *@OBJ* interpretation.

In the second example, the ambiguity is encoded entirely in the structure of the TSA. The two interpretations are found by tracing different paths through the tree structure and by using these different paths to establish different syntactic relationships between the words. The TSA for the phrase "*Inspect the bearing cups and cones*" is presented in Figure 5.4.

In the TSA illustrated, both interpretations of "*Inspect the cones*" and "*Inspect the bearing cones*" have been encoded in the structure, although the actual tree structure favours the interpretation of "*Inspect the bearing cones*" over "*Inspect the cones*". The latter interpretation can be matched by simply ignoring the part of the tree structure containing

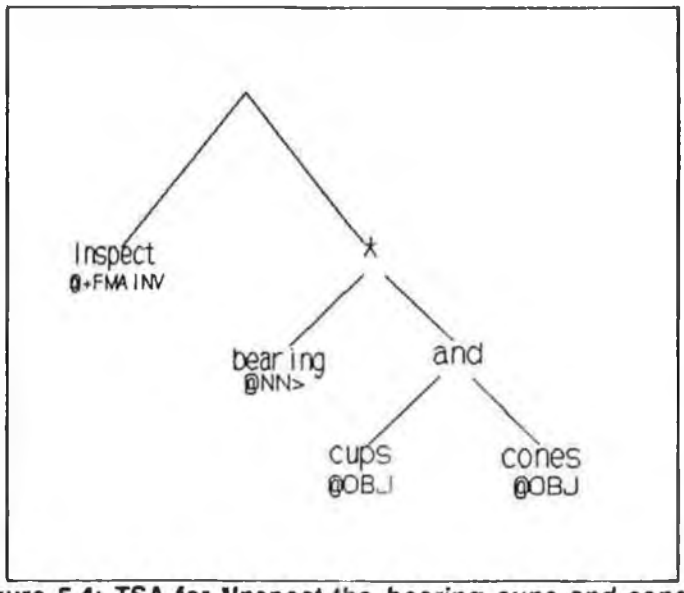


Figure 5.4: TSA for "*Inspect the bearing cups and cones*".

the modifier "*bearing*", the conjunction "*and*", and the other object "*cups*". Rather than completely ignore all of this information however, it may be very useful to use these *residual* words to identify the fact that there is ambiguity involved in the structure. In this case the words "*and*" and "*cups*" can be used to discover that the word "*cones*" is participating in an ambiguous conjunctive construct. I will describe later how the algorithm for matching these TSA structures can use this information to suitably modify the match score in order to reflect the ambiguous nature of the match that has occurred.

5.3.2 Prepositional Phrase Attachment.

The ambiguity associated with attachment of prepositional phrases is also handled by encoding multiple interpretations in the structure and then extracting those interpretations at matching time by following different paths through the tree. The TSA structure for the phrase "*I saw the man on the hill with the telescope*" is illustrated in Figure 5.5.

This TSA structure favours the interpretation of "*I saw the man on the hill with the telescope*" but the other attachments of the preopsition "*with*" are also encoded. It is

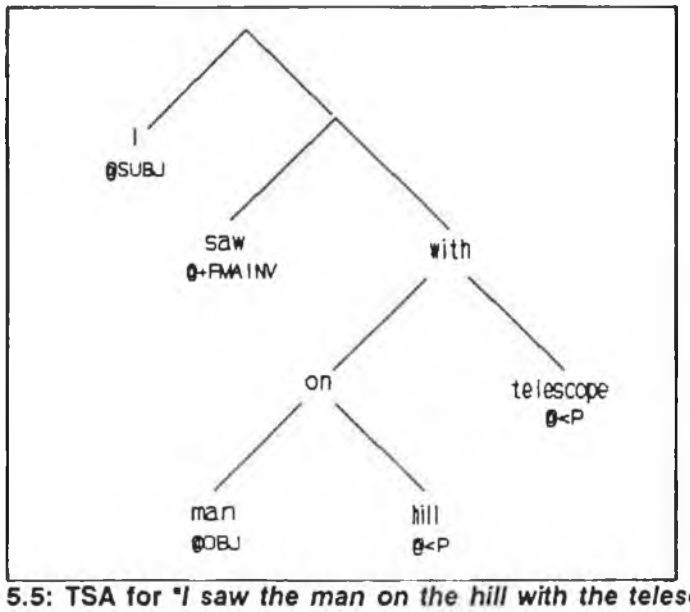


Figure 5.5: TSA for "*I saw the man on the hill with the telescope*".

possible for example, to trace a path between the words "*hill*" and "*telescope*" in order to match the interpretation relating to the "*hill with the telescope*". If this interpretation is matched then the only residual word along the path from "*hill*" to "*telescope*" is "*on*" but the fact that this is a preposition allows the matching algorithm to infer that there is ambiguity involved with the attachment of the preposition "*with*" to the noun "*hill*". The interpretation involving the "*man with the telescope*" presents a very similar case.

A more complicated case of ambiguity was described when there was an occurance of conjunction and preposition in the same sentence or analytic. One of the examples used was the phrase "*I saw a cop in a trenchcoat and a robber in the alley*". As with all other forms of syntactic ambiguity, the TSA for this phrase has the ambiguity encoded in the tree structure. The TSA is drawn in Figure 5.6.

In this case the main problem was to encode the ambiguity so that it would be possible to infer that the cop may have been in the alley. This interpretation is encoded here but it would score very poorly in a match operation. This is because the syntactic relationship between the words "*cop*" and "*alley*", as identified by their common parent, is one of conjunction rather than prepositional attachment. The match of a preposition against a conjunction would score very poorly. As will be described later, the matching algorithm

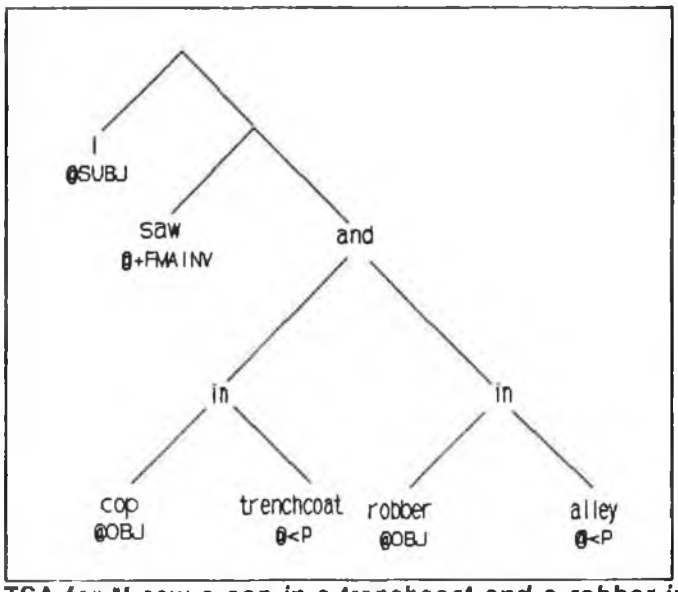


Figure 5.6: TSA for "*I saw a cop in a trenchcoat and a robber in the alley*".

would also use the residual prepositions to deduce that there was prepositional ambiguity involved in this construct.

All other forms of syntactic ambiguity are similarly encoded in the TSA structures. The main task of discovering the ambiguities is left to the matching algorithm at retrieval time. Having described here how the TSA structure makes it possible to discover and match all interpretations of ambiguous constructs, the next chapter will illustrate how the TSA matching algorithm can use residual structure analysis to discover that a match may have occured against an unintended interpretation of an ambiguous construct and can adjust the match score accordingly.

5.4 Building TSA Structures.

A prototype demonstration system for building and displaying TSA structures for input text has been implemented in Common LISP. The main TSA construction module takes the form of a rulebase in which the rules for building tree segments are based on the syntactic function labels of the words in the text sequence. The construction process works left to

right through the sequence in one pass, examining the current word and the next word adjacent to it (and in some cases further ahead), building sub-trees for these and then connecting the sub-trees into an overall TSA. Only one TSA structure exists for each text sequence. As an example of the TSA construction process, take the text sequence "*Controlled research into techniques of information indexing, classification, search and presentation.*" which has the linguistic analysis:

```
controlled
  control " := <SVO> PCP2 " @AN>

research
  research " := <SV> <SVO> <P/in> <P/on> V IMP VFIN @+FMAINV "
  research " N NOM SG " @SUBJ @OBJ

into
  into " PREP " @ADVL

techniques
  technique " N NOM PL " @<P

of
  of " PREP " @<NOM-OF

information
  information " <-Indef> N NOM SG " @NN> @<P

indexing
  index " <SVO> PCP1 " @SUBJ @<P @<NOM-FMAINV(n) @-FMAINV(n)

$COMMA

classification
  classification " N NOM SG " @SUBJ @OBJ @APP @<P

$COMMA

search
  search " N NOM SG " @OBJ @APP @NN> @<P

and
  and " CC @CC "

presentation
  presentation " N NOM SG " @OBJ @<P
```

Many of the words in this sequence are syntactically ambiguous and the word "*research*" is also lexically two-way ambiguous. The TSA construction starts off with the word "*Controlled*", discovers that this is an adjectivial modifier, and examines the next word. This

is ambiguous between a main verb, subject or object. The TSA algorithm knows to choose the noun head interpretation for representation purposes because the preceeding word was an adjectivial modifier.[3] These two words then go together to form a pre-modifier/head sub-tree parented by an asterisk (*) to mark the modification relationship explicitly. Working then with "*research*" as the current word and the preposition "*into*" as the next word the construction algorithm begins to construct a sub-tree for the rest of the sentence knowing that this sub-tree will be connected as the right branch of the tree under the word "*into*". To clarify, the tree so far is illustrated in Figure 5.7.

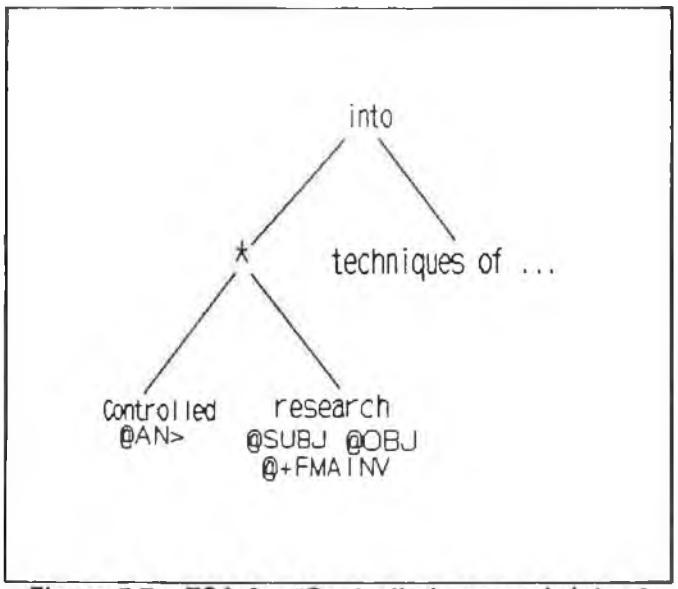


Figure 5.7: TSA for "Controlled research into..."

The first word of the new sub-tree is "*techniques*" which is a prepositional complement (post-modifier of a preposition). The next word is another preposition "*of*" so the algorithm must again "jump down" and construct a new sub-tree for the remaining sequence of words. This new sequence begins with the modifier "*information*" followed by a list of heads, joined by commas and a conjunction. The algorithm first builds the TSA for this comma-list (working left to right through the list one word at a time) and then joins this to the modifier "*information*" with an asterisk for the modification relationship. This sub-tree is illustrated in figure 5.8 (without syntactic function labels).

[3] ... although all interpretations are kept and encoded in the word information.

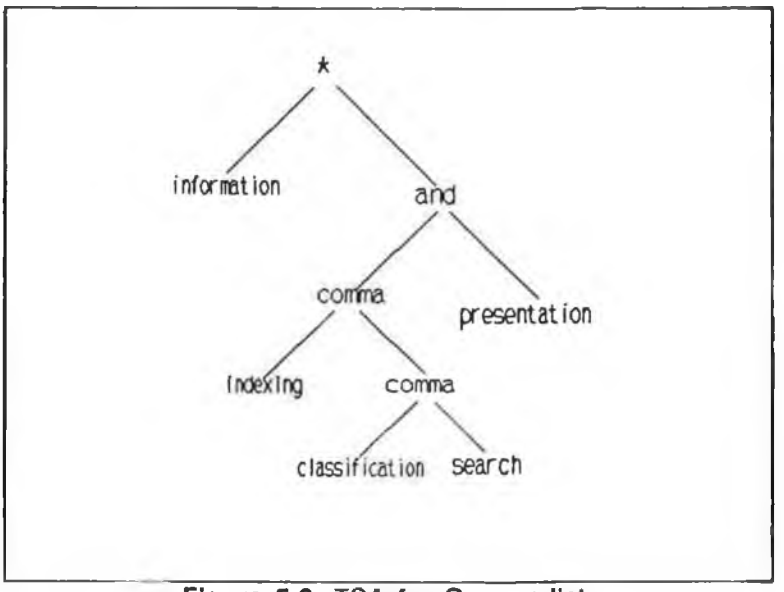


Figure 5.8: TSA for Comma-list.

This modification sub-tree, in its entirety, is the prepositonal complement for "*techniques of*" and is therefore attached to the right branch of this sub-tree and this whole tree structure is then the prepositional complement of "*research into*" and is connected as the right branch of the original tree. The final TSA for the whole text sequence is then illustrated in figure 5.9 below.

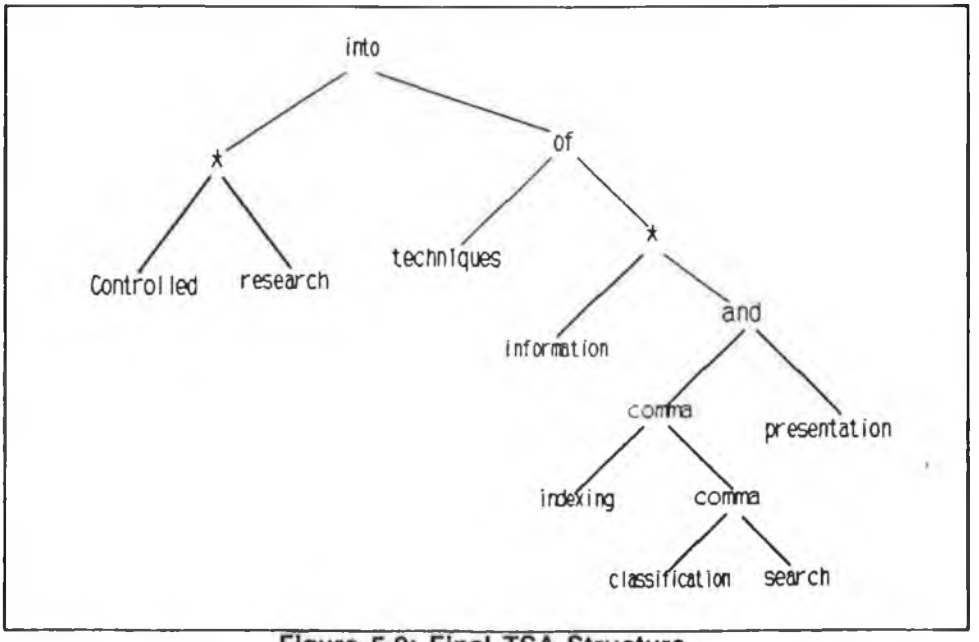


Figure 5.9: Final TSA Structure.

In an indexing environment, a TSA structure would be built and stored for each text segment or analytic extracted from each text. The TSA would then be stored in the text database with a pointer from the TSA structure to the original text(s) from which the text sequence had been extracted. It might also be desireable to construct a term index based on the TSAs with each term occuring in a TSA being stored with a link to the TSA(s) in which the word occurs. This term index could then be used as a filtering mechanism at retrieval time so that the matching algorithm would not have to compare the query TSA to each and every text TSA in the database.

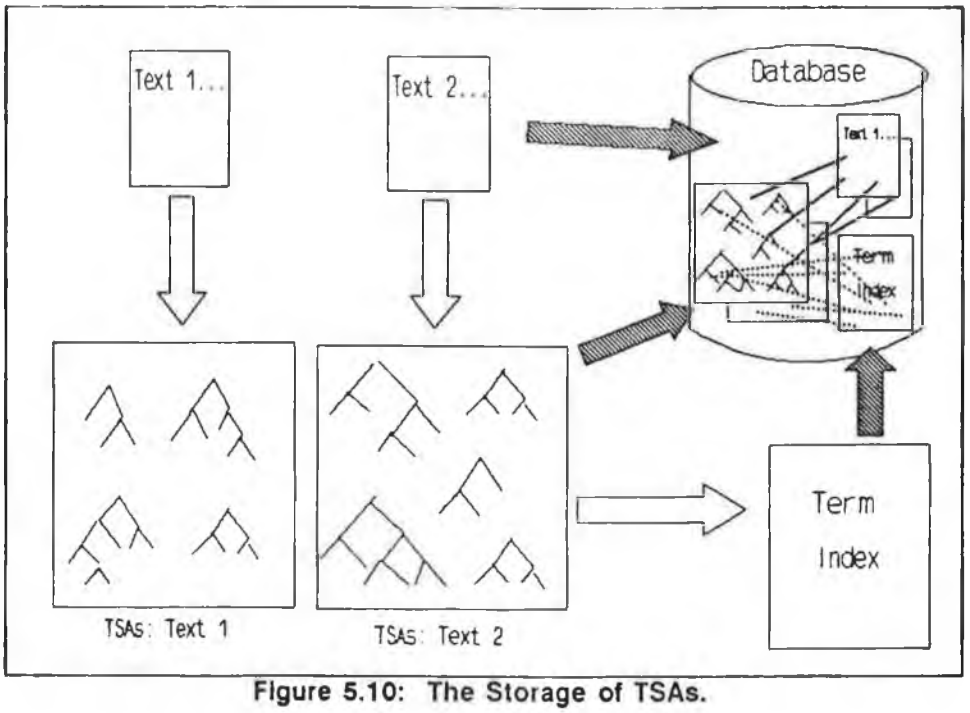


Figure 5.10: The Storage of TSAs.

This setup can be visualized with the help of Figure 5.10 above. For each text, a set of analytics are extracted for which TSAs are built. Both the TSAs and the original texts are stored in the database and for each TSA a link is maintained to the text from which that analytic was extracted so that, if that TSA is matched successfully at retrieval time, the appropriate text may be retrieved. Instead of having to match every TSA against the retrieval query, it may be beneficial to construct a term index which is done by compiling a list of all the words that occur in each TSA and storing this with a link for each word

in the list to the TSAs in which it occured. At retrieval time this index can be searched for matches against the words in the query and only the TSAs with links to some or all of those words must be matched. This results in a two-stage retrieval process: first the term index is used to locate relevant TSAs and then these TSAs are matched to find and retrieve the relevant texts.

5.5 The Representation of Uncertainty.

As the development of the TSA structure progressed, it became more and more apparent that there was a lot of uncertainty and inexactness associated with the construction and matching of the representation. For example when a TSA structure was built for ambiguous phrases such as "*Inspect the bearing cups and cones*", there is uncertainty involved because the structure will always favour one interpretation over the other, even though both are encoded. It seemed that the TSA representation should therefore incorporate some sort of uncertainty measures on the branches of the trees to reflect the strength or certainty of a relationship between two words or sub-trees. In a survey of methods that would be available for representing such confidence, certainties, probabilities, Dempster-Shafer theory and other theories were investigated, including the system of calculus used in the RUBRIC system [Tong *et al.* 1989].

It was decided that none of these formalisms suited the needs of representing uncertain interpretations in TSAs, but I also carried out some investigations as to how I could encode certain types of confidence values in the tree structures. One possibility was to *grow* the trees into graph structures by creating all possible links explicity and attaching confidence values to all uncertain links. After some further investigation however, I decided that the tree structure as described above was still the best representation and that it would be better to incorporate the confidence weights at matching time rather than trying to compute them all at indexing.

The result of this was that the TSA representation remained the same but I had gained some valuable insights into some of the necessary aspects of a matching algorithm for these

structures, namely that it must be able to manipulate weights and scores to reflect uncertainty and inexactness in the matching of text structures. This actually fitted well with my prior expectations that the algorithm would be able to *score* matches so that texts could finally be *ranked* according to their estimated relevance to the query. This necessary manipulation of confidence scores could be incorporated into the overall scoring mechanism so that there would not, in effect, be too much extra work. By the time that I had finalised the TSA representation then, there already existed an informal specification of some of the functions that must be included in an algorithm for matching these representations for information retrieval purposes.

Chapter 6.

The Matching of Structures for Information Retrieval

6.1 Necessary Functions of a Text Matching Algorithm.

Although some work had been done on structure matching for information retrieval, these efforts dealt mostly with the matching of semantic structures such as frame-based structures [Lewis *et al.* 1988] or lexical-semantic relationships [Lu 1990]. Carole Hafner also did some work on structure matching [Hafner 1989] but this was for matching concepts after linguistic analysis by a chart parser rather than for matching parse trees as in my application. In the work described earlier on applying natural language processing techniques to information retrieval for the construction of structured text representatives by Schwarz and Metzler, neither have published any details yet about how the representatives would be used at retrieval time. Schwarz has stated that using the dependency trees constructed in the TINA project, retrieval would be based on the matching of both words and word links but I believe this to be a straight matching of normalized structures rather than the application of a specialised matching algorithm. Although Metzler has published much information about the construction of trees by the Constituent Object Parser and described in detail how this formalism can handle all kinds of ambiguity and discontinuous constituents [Metzler *et al.* 1990], there has been no detailed information about how these structures would be matched at retrieval time.

The need for a specialised matching algorithm for the TSA structures comes from a different breakdown of the work loads between indexing and retrieval than is usual in information retrieval research. Most research to date has involved indexing modules which extract text representatives and store them in some normalized form. The user's search query is also transformed into this normalized form so that the matching task for retrieval is just a straight comparison. For example, the phrases "*engine removal*" and "*removal of the engine*" would both be normalized to something like "*engine: remove*" so that this could be matched against user queries for information about "*removing engines*", which would also be transformed into that normalized form. I believe that there is a certain loss of information in the process of normalization and that retrieval effectiveness can be improved by preserving the richness of the linguistic description from the indexing module and using all of this information to make judgements about the similarity of phrases or terms at

retrieval time. These similarity judgments provide naturally for a scoring mechanism to reflect the similarity of phrases and so facilitate the ranking of texts according to estimated relevance. Having these requirements in mind, and knowing that the text representatives were going to be organised in a tree structure as described in the previous chapter, I knew that the matching algorithm would have to be based on a tree traversal algorithm but that there would have to be many enhancements due to the particular application at hand.

6.1.1 Flexibility.

Because of the nature of natural language and the nature of the text segments that may be used as analytics and represented in TSA form, the matching algorithm for comparing these structures must be extremely flexible. Much simplification has been achieved in the past by limiting phrasal indexes to two- or three-word phrases. In the case of TSAs, a text representative may consist of a complete sentence clause or even a whole sentence and the tree structures can become quite complex with word relationships stretching over large sub-trees. A matching algorithm would have to be able to match each word over these quite long distances and be able to identify and validate the syntactic relationship between them. A feature of natural language text is that very often, a concept can be expressed in many different ways, either using different word forms or by combining word forms in different ways. A matching algorithm would have to identify and match similar concepts irrespective of the means used to express them. The need for this type of flexible matching gives rise to the second necessary function of such a matching algorithm.

6.1.2 Inexact Matching.

In matching similar concepts expressed using different word-forms or different syntactic structures, the matching algorithm must have some means of matching words and structures *inexactly*. The need for this inexact matching arises at two levels.

- The inexact matching of word-forms: i.e. matchng the noun "*analysis*" to the verb "*analyse*". In the Tree Structured Analytic representation this is equivalent to the inexact matching of nodes in the trees. In order to facilitate this matching the algorithm will have available all of the word information, including the text form of the word, the morphological base form and the syntactic function label.

- The inexact matching of structures or sub-tres: i.e. the matching of the sub-tree for "*engine removal*" to the sub-trees for "*removal of the engine*", "*remove the engine*" and "*removing the engine*" (or indeed, "*carefully remove the big broken engine*"). This inexact matching is a superset of the first level because it includes, first of all, the inexact matching of the word-forms at the node level and then the inexact matching of the structures reflecting the syntactic relationships between the words.

It is this inexact matching that differentiates this text matching algorithm from other ones which only perform straight comparison of normalized phrases or structures. Through this inexact matching of words and structures the matching algorithm can make judgements as to the similarity of two phrases based on the *exactness* of the matches of the constituents. A further advantage can be gained if some weighting or scoring mechanism is used to attach actual values to the levels of exactness of matches. These values can then be used to compute overall match scores for text representatives and then texts so that, at the final stage, retrieved texts can be ranked or ordered according to their estimated relevance to the initial retrieval query, based on the scores of the consituent matches. This is the third item of functionality that should be included in a sophisticated text matching algorithm.

6.1.3 A Match Scoring Mechanism.

As with the inexact matching of words and structures, the application of a scoring mechanism to the matching process can take place at several different stages and at different levels of matching.

- The scoring of word-form matches at the node level. There should be some scoring mechanism to reflect the fact that the word "*analysis*" matches the word "*analysis*" perfectly but is a variation on the word "*analyse*". This would play a role in matching the phrases "*X-ray analysis*", "*analysis of X-rays*" and "*analyse the X-rays*". The difference in score for matching these words may not be large, but it should at least reflect the fact that one is a perfect match and the other is not.

- The scoring of inexact structure matches reflecting different syntactic relationships between words. For example, such a scoring mechanism might specify that there should be a perfect or near-perfect score for matching "*engine removal*" and "*removal of the engine*" because they express the same concept, but a lesser score for a match between "*engine removal*" and "*carefully remove the big broken engine*" because the concepts expressed are not quite the same.

- The adjustment of scores associated with the interpretation of ambiguous constructs. The matching algorithm must first be capable of recognising that more than one interpretation of a construct exists and must then adjust the match score for that construct in a suitable manner (decrease). This adjustment must be effective enough so that the final ranking of that ambiguous phrase (or the text that contains that phrase) reflects the fact that the match may have occurred against some interpretation of the phrase that was not intended by the original author of the text.

- The combination of constituent scores into an overall match score for one structured representative against another: There must be some method of combining the word-form and structure scores into sub-tree scores and then combining these sub-tree scores into an overall score for the TSA. This becomes more complicated when different sub-trees of one TSA match against sections of a second TSA. For example if two words from the first TSA are found to have perfect matches in the second TSA but the syntactic relationship between the two words in the second TSA is different than that in the first, how should the scores of the two words be combined into an overal score - should they just

be added and the sum recorded, should only the score of one of the words count since they weren't in the same syntactic relationship, or should no score be recorded?

At a higher level, the match scores for the many structured representatives within a text would have to be combined in order to arrive at a complete match score for that text in relation to the retrieval search request. This is a complex problem becuase it is possible to have many structured representatives with varying degrees of match scores within different texts. One must consider whether it is better to have a few high scoring TSAs within a text or to have many lower scoring TSA matches. This type of combination of TSA scores is beyond the scope of my own work as I am only concerned with investigating a suitable structured representative for information retrieval and with developing a matching algorithm to factilitate retrieval based on those representatives. The combination of TSA match scores into overall text scores is included as a topic for future research and so will be discussed further in the final chapter of this thesis.

6.2 A Survey of Search Techniques.

Although I had used the application of searching for molecular structures in large chemistry databases to demostrate that the matching of complex structures was computationally feasible, the above requirements for a text matching algorithm eliminated any possibility of using the structure matching algorithms from the chemistry domain for the matching of structured reprsentatives in an information retrieval environment. The graph isomorphism problem that exists in matching molecular structures is an exact match problem. It involves finding structures within other structures. Text matching requires inexact matching as described above. The chemistry domain had, however, served its purpose. The initial investigation of this application led to the removal of any restrictions or limitations on the specification of the structure for the text representatives since I knew that I would not be bound by problems of computational feasibility.

Knowing that the search algorithm had to be based initially on a tree search algorithm and

that it needed to be able to perform inexact matching and have the facility for scoring matches, enabled me to perform a more detailed search for previous research that may have been of some beneifit. The purpose of this survey was twofold: first to get a feel for the type of work that had been done in that area and the types of algorithms that had been researched and proposed to solve the problems, and secondly to see if an algorithm existed that was sufficiently close to my needs that I could use this as a starting point rather than having to develop a text matching algorithm from scratch. In particular, there were three groups of researchers whose work was of interest. They were all working in different application domains but had in common the fact that they had some sort of structures for which some form of "inexact matching" was required.

6.2.1 The Universal Graph.

One of the early papers I reviewed was "A Self-Organizing Retrieval System for Graphs" by Robert Levinson [Levinson 1984]. Levinson was concerned with the development of a retrieval system for chemical reactions and molecules but decided to design "a general knowledge base for labelled graphs". In particular he wanted a system that "given a labelled-undirected graph Q and a database of labelled-undirected graphs, answer the following:

1. Is Q a member of the data base? (exact match)
2. Which members of the database contain Q as a subgraph? (supergraphs)
3. Which members of the database contain Q as a supergraph? (subgraphs)
4. Which members of the database have large subgraphs in the database in common with Q? (close matches).

These are exactly the levels of matching required for the matching of structured analytics. Obviously an exact match would be required to quickly identify analytics that were identical. One would also want to identify analytics which contained the retrieval query as a sub-graph, analytics which were subgraphs of the retrieval query and finally analytics that had large chunks in common with the retrieval query. It is also advantages to have each

type of match identified individually so that they can be appropriately scored and ranked.

An interesting concept introduced by Levinson is the idea of a "**universal graph**" of which all the graphs in the knowledge base are sub-graphs; i.e. the universal graph is a composition of all other graphs in the system (in our case it would be the composition of all structured analytics in the database). The knowledge base is then ordered in a hierarchic structure starting with the universal graph at the top, individual graphs (concepts/analytics) in the middle and at the lowest level the information from the nodes of graphs (primitives/words). Levinson presents high level algorithms for the retrieval operations specified above and also presents the advantages to be gained from the use of a universal graph. One of these that may be of particular interest in text matching is the fact that similar graphs (concepts/analytics) are located physically close together in the universal graph. A further advantage that is claimed by Levinson is that "the universal graph and hierarchic ordering are excellent aids to concept discovery". Levinson also points out that his system does have the capability of relaxed or inexact matching. In his application of chemistry this is used to specify that atoms or molecules that behave similarly in chemical reactions can be specified as matches (e.g. C-Cl11 matches C-Br11 since Cl and Br behave similarly). This type of of inexact matching has been identified above as an important and necessary aspect of a matching algorithm for information retrieval based on structured representatives.

The first impression given by this work was that Levinson's approach had many of the characteristics that I considered essential for the matching of structured analytics and also has certain characteristics thay might be considered as advantagous in that application. I have already indicated that the retrieval operations specified above are exactly those that would be required for information retrieval. Further, the universal graph may be helpful to provide assistance to a subject classification system which wanted to maintain some sort of classification schema recording the subject area of each text in the database, if such a system was being used in an information retrieval system.

6.2.2 The Quick-Look-Up Table.

The second piece of research I looked at was reported in "A Translation Aid System Using Flexible Text Retrieval Based on Syntax-Matching" by Eiichiro Sumita and Yutaka Tsutsumi from the IBM research laboratory in Tokyo. This paper describes the ETOC translation aid system for helping users in translation from Japanese to English. The aim of the system is to aid in translation by allowing a user to type a Japanese sentence and then producing several equivalent English sentences, one of which the user can select and edit as appropriate. This is accomplished by applying different levels of relaxed matching between the input sentence and the words or phrases in the dictionary. I realised however, that this matching was more of an enhanced string match than a structure matching process of the type that would be required in matching TSA structures. The different degrees of relaxed matching are achieved in ETOC by syntactically generalizing the input sentence; i.e. first look for an exact match, then replace nouns by arbitrary ones, replace pronouns by arbitray nouns etc. until finally only a syntactic skeleton remains. Some of the generalization rules given in the paper also make use of the special characteristics of the Japanese language.

One idea introduced by Sumita and Tsutsumi that may have been of use for TSA matching is what they refer to as a "**quick-look-up table**". In this table they store each word in a dictionary definition and for each of these a list of the dictionary entries in which they occur is also stored. This can then be used to restrict the search space at matching time by limiting the search to dictionary entries which contain the words in the input sentence. Therefore, for example, if an input phrase contained two words the search space would be restricted to the dictionary entries containing both of those words. This would be found by calculating the intersection of the set of dictionary entries for each of these words from the quick-look-up table. This same idea was actually mentioned by Levinson in his paper. In his case it consisted of representing the graphs in set form at a low level in the hierarchic ordering (the sets consisted of the nodes in the graphs). This greatly improved the efficiency of the exact, subgraph and supergraph matches as these could all be achieved through the set intersection operation. The set difference operator may also be of use in matching structured analytics since it would provide the residuals or leftovers of a match. These residual words or structures may play an important role in the matching of structures,

particularly if they can be used in the identification of ambiguous constructs, as mentioned earlier.

On the whole I concluded that this work was not particularly relevant to my own since it seemed to be operating at the level of string matching rather than the matching of structures. The idea of using sets and set operations to restrict the search space may have some potential however. It is interesting that this fits into the idea of a universal graph and hierarchic ordering presented by Levinson in the previous paper.

6.2.3 The Introduction of Weighted Links.

The final piece of relevant work that I came across was "Structural Descriptions and Inexact Matching" reported in [Shapiro and Haralick, 1981]. This paper is concerned with the matching of "structural descriptions of objects which consist of descriptions of the object parts and the interrelationships between these parts". Since the parts of an object can be primitive or non-primitive, this can give rise to a hierarchic description. The authors mention early on that the type of matching they are concerned with is not necessarily symmetric; i.e. *A* matches *B* does not necessarily mean that *B* matches *A*. A rigorous definition of the types of matches that are and aren't symmetric is then given (relational homomorphisms, relational monomorphisms and relational isomorphisms). The idea of an exact match is then defined in terms of the relational homomorphism (mapping the primitives of *P* to a subset of the primitives of *Q* having all the same interrelationships that the original primitives of *P* had; i.e. subgraph match - the example given is finding a chair in the structural description of an office.). The definitions of these matches and mappings are beyond the scope of this text. It may however, be enlightening to map some of the terms used in the paper to those associated with the matching of structured analytics. Shapiro and Haralick deal with the matching of a candidate structural description to prototype structural descriptions. This is analagous to the matching of a user retrieval query in structured form to a database of structured analytics. A structural description is composed of primitives and the relationships between primitives. Primitives can be compared to the words or nodes in a structured analytic and the relationships are

comparable to the links or branches in the TSA structures.

The idea of weighted descriptions and inexact matches is introduced to reflect "*random alterations in structural descriptions due to real world noise*". In the application of matching structured text representatives one can say that structures can be altered due to the use of different grammatical constructs; i.e. it is possible to say the same thing in different ways. It is then possible, using inexact matching, to match two non-identical descriptions describing the same object (the descriptions have been altered in some way due to noise). Because it is also possible for two similar structural descriptions to be describing different objects (again the descriptions would have been randomly altered), some sort of confidence measure must be introduced into the inexact matching process. The paper by Shapiro and Haralick concentrates on the inexact matching of relationships rather than primitives. A simple distance measure is used to handle the inexact matching of primitives; i.e. for each attribute there is a threshold value (not necessarily numeric and application dependant) such that the difference betwen the values of that attribute do not fall below that threshold. I have earlier drawn the analogy between primitives in structural descriptions and nodes in structured analytics. One can also compare the attributes of a primitive to the constituents at the node of an tree-structured analtyic - word, morphological base form and syntactic label. The threshold value here is used for the inexact matching of these primitives.

Since it is also desirable in inexact matching to reflect the fact that some parts are more important than others and some relationships are more important than others, "weighted structural descriptions" are introduced using "primitive-weighting functions" and "N-tuple-weighting functions" for the primitives and relationships respectively. An *e*-homomorphism is then outlined which is basically a relational homomorphism which takes into account the weighted structural descriptions. In order for a match to occur now, there must be a relational homomorphism such that the sum of the weights on the parts of the description that are NOT matched must be less than *e*. In the TSA matching application, if we speculate on the whole matching procedure, there would be importance values for each type of node and link (i.e. noun, verb or adjective nodes and preposition or plain links). In matching a query structure to a structured analytic the query may be a subtree of the analytic. In this case the sum of the importance values of the parts of the tree not

matched must be below some value *e* (again remember I am just drawing analogies). In the light of the two enhancements above, a match between a candidate object and a prototype object can only occur if:

1) each candidate primitive inexactly matches its corresponding prototype prmitives according to a threshold associated with the prototype primitives (as outlined above).

2) the sum of the weights of those prototype primitives that do not map to a candidate primitive (i.e. the sum of the distance values of the residual primitives) must not exceed another threshold.

3) that it is an *e*-homomorphism from each prototype relation to a candidate relation, where the threshold *e* is associated with the prototype relation.

The idea of a best match is mentioned but is not elaborated since they state that since there are so many error measures involved with the inexact match, the definition of a best match is not immediately obvoius (and this may have implications for combining scores!). Shapiro and Haralick also state that the definition of the best match depends on the priorities required for the matching task to be performed. Since the priorities of a TSA matching algorithm would be clearly defined, I believe that the scaling or ranking of matches in this way would be possible.

The relational homomorphism problem for 0-homomorphisms (exact matches) is then described as a special case of the "consistent labeling problem" [Shapiro and Haralick, 1979]. The definition of this problem is also beyond the scope of this text but it is significant that it can be solved by a tree search algorithm incorporating a look-ahead, forward checking, and/or relaxation operator. I investigated these tree searching algorithms in the hope that they could be adapted in some way to solve the *e*-homomrophishm problem (inexact matching) and then further enhanced to be used in the application of matching structured text representatives.

6.2.4 Tree Searching.

The consistent labelling problem can be solved by a striaghtforward tree search using backtracking but this suffers from thrashing (search failure at different points for the same reason). This could be avoided if there was some way to remember the causes of failures that have occurred and to anticipate future events and possible failures. In the case of a *e*-homomorphism the sum of the weights of the parts of the structure not matched must sum to less than *e*. At any stage in the search the sum of the weights not matched in the part of the sturcture searched so far is known. It follows then that the minimum allowable error (sum of weights not matched) from the search of the remaining parts of the structure (the "error budget") is also known. One method of limiting the thrashing is to anticipate the minimum possible error from the matching of future parts of the graphs given the present state of the search. If this anticipated minimum future error exceeds the error budget then a new path through the tree must be tried or the algorithm must backtrack. This can then be taken a step further by estimating the minimum error on future units (matches) given the current state of the search and the estimated future error on other future units; i.e not only are the missed matches so far considered but also the estimated missed matches at some stage in the future and these estimates are then used to estimate the minimum error at some point further in the future ! Since all of these error estimates are minimum values they can be used to restrict the search space by anticipating and avoiding failures (when the estimated error is greater than the allowed *e*) so improving the efficiency of the of the tree search [Haralick and Elliott, 1979].

Shapiro and Haralick [Shapiro and Haralick, 1981] go on to discuss different tree searching algorithms for finding *e*-homomorphisms. These algorithms are all defined in the paper in terms of the consistent labeling problem. I will try to describe them here in as general a way as possible. I cannot be sure, however, that my interpretation is completely accurate. It should be noted, too, that I am simply reporting these algorithms as they are given in the paper. This does not necessarily mean that they would take this form if used in the application of matching structured anlaytics. The algorithms are as follows:

Backtracking.

As the tree search proceeds it will be continuously accumulating errors (in matching structured represntatives, due to residuals in the match - subtrees, links or nodes that do not match). If the accumulated error at any stage exceeds the error budget then the tree search must try the next label (link/path) or, if there is no next label, it must backtrack. This is the striaghtforward approach that was mentioned earlier as suffering from thrashing behaviour.

Forward Checking.

This is similar to the backtracking approach except that it anticipates the minimum future error for a given path, as described in the last section. As the search proceeds it continuously broadcasts the current accumulated error to all future units (i.e. further down the path). The assigned labels (matches so far) are also broadcast and based on this information an estimate of the minimum future error can be calculated. The forward checking mechanism involves monitoring this future error and if, at any stage, this exceeds the error budget then the next label for the current unit must be tried or, if none exists, then it must backtrack.

Looking Ahead by One.

Looking ahead by one is similar in operation to forward checking. While forward checking considers the accumulated error so far and an estimated minimum error for the future labelling given the current state of the tree, looking ahead by one also maintains an estimate of the future error due to the labelling of future units (this was also mentioned earlier). If the estimated error for a future unit due to future labellings exceeds the error budget then that future unit may be excluded from consideration in future matching. To state this in general terms; if, for some future part of a tree, the sum of the weights on the estimated non-matching part of that tree exceeds the error budget, then that subtree or

part of the tree can be excluded from the path that is presently being followed.

When the current unit has been labelled (the current link or node has been processed; i.e. either matched or discarded as a non-match) it then becomes a past unit and the future error estimate can be recalculated. If at any stage the estimated error exceeds the error budget, then looking ahead by one fails and the next label must be tried or the algorithm must backtrack. It should be noted that in a running example used by Shapiro and Haralick in their paper the tree search algorithm using look-ahead by one completely eliminated both thrashing and backtracking.

Looking Ahead by Two.

Looking ahead by two is again similar to looking ahead by one but in addition it estimates the minimum error incurred by a pair of future unit-labelled pairs as they look ahead to other future units; i.e. not only does it perform look ahead by one, which compares future units to other future units, it then takes future units in pairs and calculates estimated errors. The look ahead by two algorithm always does a look ahead by one first and only if this succeeds is the look ahead by two invoked. If the cummulative look ahead error exceeds the error budget then the pair of future units checked by the look ahead by two can be discarded.

In an evaluation of these tree searching algorithms by Shapiro and Haralick, they conclude that the foward checking algorithm is the most efficent. They go on to suggest methods of increasing the efficiency of the algorithms but state that the benefits to be gained are minimal. After some experimentation with the forward checking algorithm, mostly regarding the effects of varying the allowed error (e), they concluded that "the inexact consistent labelling problem involves much more work as e gets larger". Finally it should be noted that the last line of the paper reads, "*Our results show that the inexact consistent labeling problem, and therefore inexact matching, is a much harder problem than the exact version.*".

6.2.5 Conclusions.

Having established early on that complex graph matching was computationally feasible for exact matches (the graph isomorphism problem), I could now conclude that the inexact matching of structures, including the manipulation of weights, was also feasible (the *e*-homomorphism problem). In particular the research by Shapiro and Haralick describing all types of matching problems and presenting mathematical approaches to their solutions, gave an indication of the degree to which the area of complex structure matching had already been investigated. I didn't discover a matching algorithm that seemed suited to the application of matching TSA structures for information retrieval but I had gathered some useful ideas for tree searching algorithms and the applicablity of such things as a universal graph or quick-look-up table and, although not incorporated directly into the final algorithm for the matching of tree structure analytics, the points and ideas gained from the above survey certainly played a role in the design of the match algorithm.

6.3 The TSA Matching Algorithm.

The basic function of the TSA matching algorithm is to directly compare one TSA representation against another and to compute a score reflecting the similarity of the two. This comparison is based on inexact matching and the matching is not symmetric; i.e. the score computed for the comparison of TSA *A* to TSA *B* is not necessarily the same as the score for the comparison of TSA *B* to TSA *A*. This non-symmetry is introduced because of the application domain in which the matching algorithm is to be applied. It will always be matching a retrieval query representative against a text representative. As shall be explained, the algorithm functions by trying to find matching sub-graphs of the query representative in the text representative. The non-commutativity of the operation therefore exists because a different score is computed depending on which TSA is considered as the query representative and which is the text representative. In the normal operation of the algorithm in its intended application, this non-symmetry will not arise because the query representative and the text representatives will be clearly defined and the algorithm will always operate in only one direction.

The other necessary functions of the matching algorithm discussed earlier are flexibility, the ability to perform inexact matching, and the ability to score matches and compute overall scores for TSA comparisons. As part of this scoring mechanism the algorithm must be able to detect and provide for any ambiguity that might exist. This ambiguity can be detected through examination of the residual words or sub-trees that lie between the matched words in the text representative. This need for examination of residuals together with the fact that inexact matching and match scoring take place on two levels (word and structure) suggest that the matching algorithm should operate on three levels: word, structure, and residual structure. This was tried and was found to be an ideal approach. Some comment should be made first on the overall strategy employed by the matching algorithm.

As stated above, the algorithm works by taking a query TSA and trying to find a match for it in the particular text TSA being matched. The strategy is to start from right in the query TSA and work towards the left. The matching algorithm works only with a simple binary tree at each stage, this consisting of two child nodes connected by a parent. If either one of the child nodes in the current simple binary tree is a parent and the sub-tree under that parent has not already been matched, then that sub-tree is descended and becomes the current simple binary tree. This is a recursive process. The algorithm descends down through sub-trees until both children are leaf nodes, computes a match score for that binary tree, and then jumps back up, substituting the match score for the sub-tree as the child node which was the parent to that sub-tree. Consider the simple example of the TSA structure for the query phrase "*research in information retrieval systems*", illustrated in Figure 6.1 below. The nodes in this tree have been numbered in order to clarify the explanation. It should be noted however, that a similar numbering of nodes (a *pre-order* node numbering system) is used by the TSA matching algorithm for exactly the same purpose - to keep track of the relationships between nodes in the trees. This numbering system is also useful for identifying the common parent of two nodes which are separated in the tree structures.

The algorithm starts by trying to match the initial simple binary tree consisting of nodes 1, 2 and 3. It discovers that node 3 is in fact the parent of another sub-tree so it descends and substitues this sub-tree as the current simple binary tree. It is now concerned with matching nodes 3, 4 and 5. The algorithm computes a match for this simple binary tree

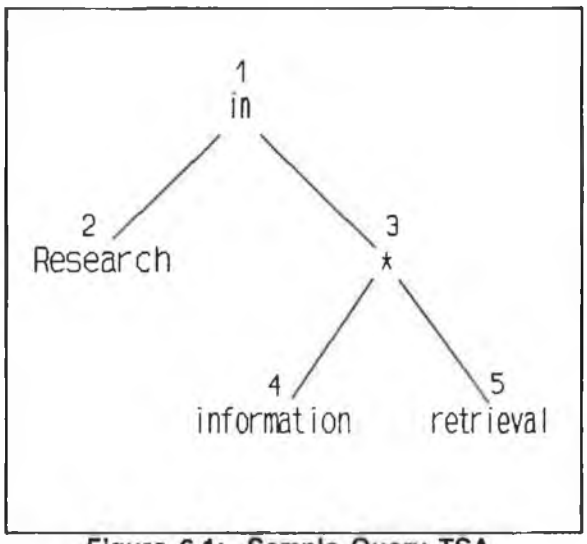


Figure 6.1: Sample Query TSA.

(since both children are leaf nodes) and then returns to the task of matching the tree 1, 2 and 3, having substitued the sub-tree match score at node 3. This tree match score is then made up of the match score for node 2 the score of the sub-tree under node 3, and the score of their parent node, node 1. This is then the final score for the whole TSA match. This basically defines a bottom-left to top-right or a *reverse post-order* tree traversal strategy.

At the level of matching simple binary (sub-)trees, the algorithm proceeds by finding a match for each of the query child nodes in the text TSA. A note is made of the position of the text TSA nodes against which the matches have occurred. A path is then traced to find the common parent of these two nodes (this is done using the node numbering system: the common parent is the first node along the ancestoral path from the right matched node which has a lower node number than the left matched node) which defines the syntactic relationship between those two nodes. This parent node is then compared against the query parent node in a similar manner as used for the matching of leaf nodes. If this simple binary tree had been a sub-tree of a higher tree (for example, if it was attached to the right child of a higher tree) then the parent that had been identified in the text TSA would be marked as the node against which the right child of the query TSA had been matched. In this way a record is always kept of the positions of the text TSA nodes against which the query TSA children have been matched.

It was mentioned earlier that a three-levelled approach to the matching operation seemed appropriate. Two of these levels have become more apparent now; the level of word/word matching and then the level of syntactic structure matching, through the matching of parent nodes. Adding the consideration of residual structures for the identification of ambiguity provides the third layer to the matching methodology. This matching methodology fits well within the tree traversal strategy outlined above and the three levels are as follows:

Node Matching.

At the lowest level, the TSA matching algorithm performs matching of individual nodes or words. This matching is inexact and is based on all of the word information stored at the node, including the text form of the word, its morphological base form and its syntactic function label. A score is assigned to each of these match constituents and it is the combination of these scores that makes up the final word/word score. For example, a word in a query representative that appears in some lexical derivativational form in a text reprsentative will not score at all for the text form match but will score perfectly on the matching of morphological base forms and will score an inexact match on the syntactic function labels (assuming that the derivational form of the word is not performing the same syntactic function in the text sequence as the word in the query phrase).

Structure Matching.

Once a match has been found for the two child nodes of a simple binary tree the next stage is to discover the syntactic relationship between the two matched nodes in the text TSA structure. This has been described above as a process involving the use of the node numbering system. The TSA structures are organised in such a way that the common parent of two nodes always defines the syntactic relationship that exists between those two nodes. The syntactic relationship can therefore be matched by comparison of the parent node of the two children in the query TSA to the parent node of the two

matched nodes in the text TSA. This matching of parent nodes is accomplished through a set of rules which specify which syntactic relationships should match; i.e. which different syntactic relationships can validly be used to express the same concept in different ways. For example there is a rule specifying that the prepositon "*of*" can validly be used to express a modification relationship and so should match perfectly with the symbol "*" which is used to mark explicitly such modification relationships. An example of a case where this rule would be used is in matching the query TSA for "*information retrieval*" to the text TSA for "*speedy retrieval of time-table information from a large database of train schedules*".

Residual Structure Analysis.

Residual structure is defined as that part of the text TSA which lies between the two matched child nodes and under the parent node but which has not been matched. The purpose of residual structure analysis is twofold. First of all the consituents of the residual structure can be used to validate the correctness of the match on the syntactic relationship performed at the parent node level. This is achieved through a small set of rules which specify certain instances of word types that, when found in a residual structure, invalidate certain syntactic relationships that may have been incorrectly identified at the second level of matching. An example of such a rule would be one which specifies that the existence of a finite main verb in the residual structure invalidates a direct pre-modifier head relationship identified at the second level. An example case for the use of this rule would be in the matching of the query phrase "*classification systems*" to the text analytic "*the development of a classification schema using library system theory*"; the possible identification of a modifier relationship between "*classification*" and "*system*" would be invlaidated by the presence of the verb "*using*" in the residual structure.

The second purpose of residual structure analysis, which has been referred to earlier, is for the detection of ambiguity. The two key word categories which signal the possible presence of ambiguity are prepositions and conjunctions. There is a small set of rules at the residual structure analysis stage which examine the residual structure, looking for

certain combinations of conjunctions and prepositions along with other word categories. An example of such a rule would state that if the matched children are in a direct pre-modifier/head relationship (identified by a "*" parent) then the presence of a conjunction and a head noun in the residual structure signals the presence of a conjunctive ambiguity, of which "*Inspect the bearing cups and cones*" was the example used earlier. Similarly if the matched children are participating in a prepositional relationship then the presence of prepositions, complete with their prepositional complements, in the residual structure signals the presence of prepositional ambiguity.

This three-pronged matching methodology, which fits into the simple-binary tree matching approach outlined earlier which, in turn is part of the reverse post-order tree traversal strategy, all go together to make up the overall TSA matching algorithm. Having been specifically designed for the purpose of matching the TSA representations, the algorithm fulfills all of the design requirements outlined earlier in this thesis, with the only undesireable characteristic being its complexity, which I think could not be avoided. A high-level specification of the complete algorithm has been inlcuded as Figure 6.2 below.

This figure presents just the core of the algorithm. Some further aspects include the decisions to be made when a match can be found for only one of the chidren of a simple tree or when both children match and the parent does not. In the present algorithm, when only one child matches, the node match score for that child counts as the score for that (sub-)tree. When both children have matched but the parent node does not, it means that the syntactic relationship between the two words in the text sequence is different from that in the query phrase. It would be wrong, therefore, to accept the score for both the words in this case (it is just as important to match the proper syntactic relationships as it is to match the words) so the algorithm uses only the score for one of the words as the score for that simple (sub-)tree. The choice of which word score to use depends on the syntactic relationship that exists between the words in the query TSA. For example, if the nodes in the query TSA are participating in a pre-modifier/head relationship like "*classification systems*" and no modification relationship can be identified in the text TSA (e.g. "*development of a classification schema using library system theory*") then the node match score for the right query node is used as the tree score, in this case the node match score for the word "*systems*".

```
Function: Match-Simple-Binary-Tree.

        If right-child-node is non-leaf then descend to right sub-tree (recursive).

        Find match for right-child-node in the Text TSA.
        Perform word/word match between the right-child-node and the Text TSA node.
        Note the match score and the current right-node position in the Text TSA.

        If the left-child-node is non-leaf then descend to left sub-tree (recursive).

        Find match for left-child-node in the Text TSA.
        Perform word/word match between the left-child-node and the Text TSA node.
        Note the match score and the current left-node position in the Text TSA.

        Trace path between the Text left-node and right-node to their common ancestor.
        Construct the residual structure, consisting of surplus nodes along this path.
        Perform a parent-node match on the Text parent and the query parent.
        Note the match score and the position of the Text ancestor node.

        Examine the residual structure to validate the syntactic relationship.
        Examine the residual structure to identify possible ambiguities.
        Adjust the match score if necessary after the residual structure analysis.

End Function.


        Divide the total tree score by number of query nodes.  (score normalization)
```

Figure 6.2: High-level Specification of Match Algorithm

As can be seen from the specification in Figure 6.2, the result of the tree matching algorithm is a normalized score (in the range 0 to 1) which reflects the similarity between the text TSA and the matched query TSA. The scoring mechanism used in the algorithm has been the subject of much investigation and should be discussed a little further.

6.4 The Scoring Mechanism.

Although it may appear to be quite complex, the scoring mechanism used in the TSA matching algorithm is, in effect, fairly straightforward. I have already identified the various

stages in the algorithm where scoring is necessary:

- During word/word matches; this consists of the matching of the constituent parts of word information.

- During the matching of parent nodes.

- Where ambiguity is present.

The scoring takes place mostly at the node level, whether it be child nodes or parent nodes. The basic constituent of the scoring mechanism then is the **perfect-node-match-score** which is awarded when two nodes are matched perfectly (usually the nodes are identical, but not necessarily). This perfect score is awarded directly to parent nodes when they provide a perfect match but can only be awarded to the matching of child nodes through the composition of perfect scores for each constituent. A child node score is composed of the following sub-scores:

identical-text-form-match-score
identical-base-form-match-score
identical-syntax-match-score

The first two of these are either awarded or not; the text forms and morphological base forms either match or they don't. In the case of the syntactic function label, the identical-syntax-match-score is awarded if a perfect match is found. If not, it is possible for an inexact match to occur between the functions of head, modifier and verb, in which case a value is awarded under the heading of **syntax-head-modf-verb-match-score**.

At the level of parent node matching, a perfect-node-match-score value is awarded if both parent nodes are identical or if they both belong either to the set [*, of, for] or the set [and or $comma] (i.e. the prepositions *of* and *for* are considered to match perfectly against modification relationships indicated by *). If this is not the case then it is possible to have an inexact match. The first level of inexact matching is for the matching of modification relationships against null relationships (denoted by *Nil* in the LISP tree structures). The

value of **modifier-specIprep-nil-score** is awarded to such inexact matches. An example of such an inexact syntactic match is in the case of matching the phrase "*information retrieval*" to the text sequence "*information retrieval systems*". In the query phrase the words "*information*" and "*retrieval*" are in a direct pre-modifier/head relationship. In the text sequence the words "*information*" and "*retrieval*" are both pre-modifiers of the word "*systems*" but there is no direct relationship explicitly identified between them. In cases like this the modification relationship in the query TSA should match against the *Nil* relationship in the text TSA.

At the lower level of inexact parent node matching, it is possible for any preposition to score an inexact match against any other preposition and against a modification or a *Nil* syntactic relationship. This matching rule is extremely "loose" but it was found that general rules like this worked better than more specific ones in TSA matching. At this level the inexact match is awarded the value of **modifier-anyprep-nil-score**.

At the stage of residual structure analysis, if a syntactic relationship is found to be invalid, then the appropriate scores are just discarded by reseting them to zero. If ambiguity is discovered in the structure then, depending on the type of ambiguity present, a suitable deduction is taken from the overall tree score accumulated at that point. These deductions are according to the values of **conj-ambiguity-deduction** and **prep-ambiguity-deduction**.

The values of these *match parameters* were originally based purely on intuition and have obviously been the subject of much empirical investigation. The values that were found to work best during my experimentation are presented in Figure 6.3.

Although these values were found to provide the best performance results during my evaluation of the algorithm, I cannot claim that they are optimal. It would take much further empirical investigation before any set of parameter values could be justified as *best* for the application.

All the individual node match scores are added together to form the overall match score for the whole TSA structure. From this are subtracted any ambiguity deductions that have been accumulated. In order to normalize all the match scores to a value in the range of

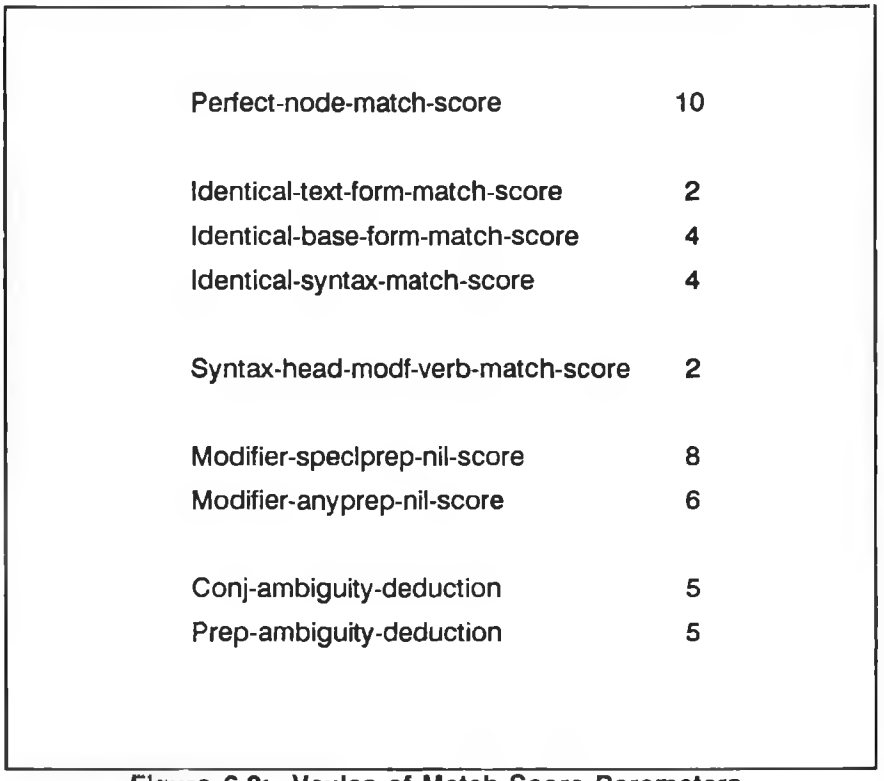


Figure 6.3: Vaules of Match Score Parameters.

zero to one, this final score is then divided by a normalizing value computed by multiplying the number of nodes in the query tree by the perfect-node-match-score. If all nodes in the query tree have been matched perfectly then dividing by this normalizing value will leave a final score of 1. Note that the final normalized score for a TSA match is an absolute value. This means that given a large set of text TSAs which have been matched against a query TSA, it is possible to rank the text TSAs in order of similarity to the query, based on the final normalized scores. This absolute ranking of results by the algorithm is one of the advantages that I would claim to have gained over previous work in this area.

6.5 An Example Match.

The best way to demonstrate all of the above aspects of the matching algorithm working together is to step through an actual example of the algorithm at work. For this purpose

I will demostrate the matching of the query "*information retrieval systems*" against the text analytic "*information retrieval using a hypertext help system*". The TSA representation of each of these phrases is given in Figure 6.4. The nodes in these TSA structures have been numbered in the same pre-order manner that they would be by the TSA construction algorithm. During this numbering procedure, the algorithm also performs a count of the number of nodes in the query structure and stores this for use at the score normalization.

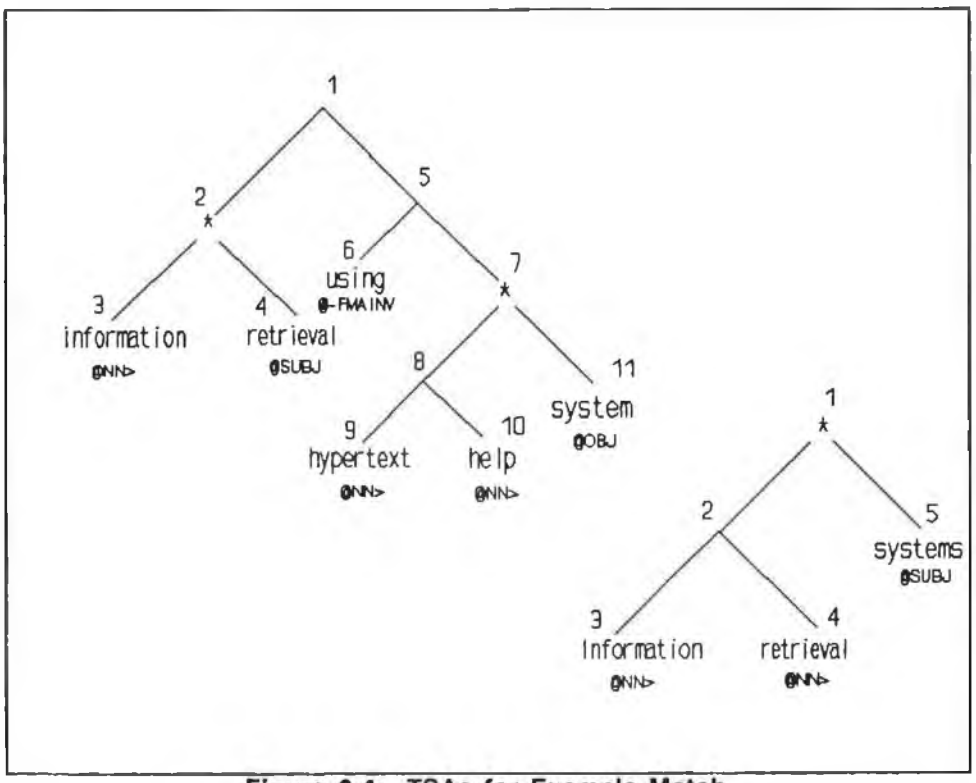


Figure 6.4: TSAs for Example Match.

The TSA matching algorithm proceeds as follows:

Start Match (1).

- Start in the query tree with the simple binary tree 1,2,5.
- Examine the leafness of the right child, node 5. It is a leaf.

- Search for matching node in the analytic tree[4]. Find *system* (text node 11).
- Perform word/word match on *system* and *systems*; score: text-form 0, base-form 4, syntax (both heads) = *8*.
- Examine the leafness of the left child, node 2. It is a parent.

Start Match (2).

- Start in the query sub-tree with the simple binary tree 3,4,5.
- Examine the leafness of the right child, node 4. It is a leaf.
- Search for matching node in the analytic tree. Find *retrieval* (text node 4).
- Perform word/word match on *retrieval* and *retrieval*; score: text-form 2, base-form 4, syntax 2 (head matching modifier) = *8*.
- Examine the leafness of the left child, node3. It is a leaf.
- Search for matching node in the analytic tree. Find *information* (text node 3).
- Perform word/word match on *information* and *information*; score: text-form 2, base-form 4, syntax 4 (identical) = *10*.
- Trace common ancestor of nodes 3 and 4 in text analytic; Find text node 2.
- Construct residual list between text nodes 3 and 4; *none*.
- Match text parent node (node 2) to query parent node (node 2); *Nil* vs *: score *8* (modifier-speclprep-nil-score).
- Verify the parent match by checking the residual structure; *none => ok!*
- Identify any possible ambiguities by checking the residual structure; *none*.
- Set node 2 to be the current position in the text TSA (needed later).
- Calculate the total score for the tree; *8+10+8 = 26*

End Match (2).

- Trace common ancestor of nodes 2 and 11 in text analytic[5]; Find text node 1.
- Construct residual list between text nodes 2 and 11; [*Nil, using, *, Nil, hypertext, help*].
- Match text parent node (node 1) to query parent node (node 1); * vs *Nil*; score *8* (modifier-speclprep-nil-score).

---

[4] This node search is based on the morhological base form of the query word.

[5] Node 2 because this was ordained the current node in the TSA at the end of the recursive sub-tree match, and node 11 because this was matched against the right child node (*system*) of the current tree.

- Verify the parent match by checking the residual structure; "*using*" => score *invalid!*
- Parent score invalid => Set parent score to 0. Set left node score to 0[6].
- Identify any possible ambiguities by checking the residual structure; *none.*
- Calculate the total score for the tree; *26+8 = 36.*

End Match (1).

- Calculate the normalization factor; *5 x 10 = 50.*
- Calculate the final normalized tree match score; *36 ÷ 50 = 0.72*

The normalized match score for matching *information retrieval systems* against *information retrieval using a hypertext help system* is **0.72.**

[6] In this case the left node score is a whole sub-tree score so it is not reset to 0. This is based on the hypothesis that since there is probably a high overlap or match between the query sub-tree and the analytic TSA then the whole score should not be just dropped.

Chapter 7.

An Experimental Evaluation of the TSA Matching Algorithm

7.1 Introduction.

Having designed a structured representation for text representatives in an information retrieval system and having defined a matching algorithm to use this representation for retrieval, the next step was to evaluate this algorithm through a formal experiment. The overall objective of this experiment was ***to investigate and evaluate the effectiveness of the TSA matching algorithm on a large test data set by using standard statistical methods for the evaluation***. Since the basic function of the matching algorithm is to provide a set of ranked analytics, ordered according to the similarity of the analytics to a user query, the experiment proposed to evaluate the effectiveness of the algorithm by comparing a set of phrase rankings provided by the TSA matching algorithm to the set of rankings provided by a sample of information retrieval system users. The higher the correlation between the matching algorithm's rankings and the sample humans' rankings, the better the performance of the algorithm. This comparison is based on an prototype implementation of the TSA construction and matching algorithms in SUN Common LISP on a SUN 3/60 workstation with 12 Megabyes of main memory and 360 Megabytes of disk space.

In discussing the details of the experiment it must be remembered that, at the present time, the TSA representation and matching algorithm still only provides a mechanism for matching and ranking text representatives against a retrieval query. It is a subject for future research to define some strategy for combining these ranked analytics in order to achieve an overall ranking for whole texts. The experimental evaluation is therefore based on a comparison of phrase rankings rather than texts. This phrase ranking does however, provide for the proper evaluation of the matching algorithm because that is exactly the task it was designed for. This matching of individual text representatives or phrases is central to the information retrieval task.

Apart from the overall aim of the experiment, to evaluate the effectiveness of the matching algorithm, there were several other incidental objectives. The first of these, which was reflected in the design of the test dataset, was to verify and evaluate one of the more powerful aspects of the matching algorithm; the ability to infer subtle differences in

semantic meaning based only on the structural aspects of the syntactic description. A second purpose of the experiment was to facilitate the iterative evaluation and optimization of the set of match score parameters. This was achieved by compiling different sets of rankings produced by the match algorithm using different sets of match score parameters and then by calculating a correlation value to reflect the effectiveness of each of the parameter sets. The idea was that by successively refining the score parameters, a final set of *best* score parameters could be found.

The basic experimental design was based on 32 sets of phrases. Each phrase set consisted of an article title from the ACM Transactions on Office Information Systems, which acted as the *base phrase*, and nine artifically generated phrases which were closely related to the base phrase in some way. The sample of users consisted of 24 people who were judged to be knowledgeable enough to judge the similarities between the phrases in the phrase sets. Using this user sample, a large collection of human ranking judgements were collected and tabulated. I also applied the TSA matching algorithm to the whole dataset to produce a ranking for each of the 32 phrase sets. The evaluation then consisted of a comparison between the human rankings and the match algorithm's rankings. This was perfomed using standard statistical tests. This basic design is illustrated graphically in APPENDIX B.

The rest of this chapter is concerned with describing in detail the experimental design, the collection of data, the statistical test used to analyse the set of rankings obtained from the sample users, the results of the statistical analysis of these rankings, and the statistical tests used to establish the correlation between these human rankings and the rankings provided by the TSA matching algorithm. The results of the comparison between the two rankings, and the evaluation of the algorithm will then be presented in the next chapter.

The first statistical test described is **Friedman's test**. This was used to examine the variability in the rankings assigned by different people in the sample, or in other words, to interpret whether people come up with the same or similar rankings of phrases. The test enables us to determine for each set of phrases whether there is a significant difference in the way different people assign rank positions. This test is not directly related to the evaluation of the TSA matching algorithm but it does provide some information about the suitability of the metric to which I am comparing the algorithm's performance; the

judgement of humans. It should be noted that in preparation for this test I had to perform some normalization of the data provided by the sample users. This involved using a standard format for the representation of tied rankings. The method of normalisation that was used is described in the third part of the next section.

Having described the workings of Friedman's test I will then present the results that were obtained when I applied it to the data collected for the ranking of phrases by humans. I will analyse the results obtained in order to establish whether or not these human rankings form a solid foundation for comparison and whether they are a suitable target for the performance of the TSA matching algorithm.

The two statistical tests described in the final sections are more directly concerned with the comparison of computer to human. They describe how, having established the human rankings as a solid base against which I can compare the performance of the matching algorithm, I can then perform this comparison and evaluate the algorithm's performance. The **Wilcoxon signed rank test** was first used to test if there were significant differences between the average human rankings and the computer rankings. The **Spearman correlations** where then used to compute an actual figure for the degree of similarity or dissimilarity between the two sets of rankings. Note that in both of these tests the results of the computer rankings are compared against the **mean** of the human rankings. The reason for using a mean ranking and the calculation of the mean is also described in the next section.

7.2 The Experimental Design.

7.2.1 The Test Dataset.

The test dataset for this experiment was based on 32 titles of articles taken from the international journal, ACM Transactions on Office Information Systems (TOIS). This journal deals with a wide variety of topics such as office automation, cooperative computing, information retrieval and electronic mail. One of the journals chosen was a special issue

on information retrieval. To take an example, the first title was "*A scheme for communicating among groups that use different type hierarchies*", and another one was "*A critical investigation of recall and precision as measures of retrieval system performance*".

Around each of these TOIS titles was constructed a further 9 phrases. This task was equally divided between myself and my research supervisor. The 9 phrases in each phrase set were specifically constructed so that they were very close in meaning to the original title. This was because I was aiming to test how useful the matching algorithm was at distinguishing between very closely related phrases. Some of the phrases were constructed by using the same words as in the title but rearranged in such a way as to have a completely different meaning, and some phrases were constructed as having identical meaning but with different words and/or syntactic structure. This was done so that it would be generally difficult for both the computer algorithm and the sample users to distinguish some of the relationships between the phrases and the title. To give an example, the phrase set constructed around the first title was

A scheme for communicating among groups that use different type hierarchies.

A comparison of type hierarchies and their use in communication.
A method for communication which is based on the use of hierarchies.
Applications of type hierarchies.
Communication among objects which is based on alternative type hierarchies.
Communication among objects.
Group communication.
Procedures for communication among groups.
Schemes for group communication.
Using type hierarchies for communication.

A full listing of the 32 TOIS titles and phrase sets is included as APPENDIX C. For the purpose of the individual tests, each of the 32 phrase sets is treated individually. When I talk about a phrase set I am referring to a TOIS journal title plus the nine phrases that were generated around that title. The example of phrase set 1 is used quite frequently in this thesis. When I refer to phrase set 1 I am referring only to the title and phrases

illustrated above, completely in isolation of all other phrase sets. It is only in the calculation of overall or average results that I will refer to the complete dataset of 32 sets of phrases.

7.2.2 The Sample Users.

In order to ascertain how users of an information retrieval system would judge the relevance of documents to a search query, I wanted to see how users would rank the relevance of the 9 phrases in each group to the original title of that group. For comparison purposes I wanted to have a sample of 10 different rankings for each set of phrases - a total of 320 rankings. Most users were asked to perform a ranking for 10 of the 32 phrase sets. Only one user ranked all 32 phrase sets and some users ranked half of the sets. Ranking of a set involved assigning to each of the phrases a rank position (a number - not necessarily unique) according to how similar in meaning the phrase was to the title. For example, in the first phrase set, if the user was of the opinion that the phrase "*schemes for group communication*" was closer in meaning to the original title than the other phrases then this would be ranked number 1. Note that ranks could be tied, so that if the person thought that "*group communication*" was just as relevant or close to the title then this could also be ranked number 1. An example ranking for phrase set 1 could be

A Scheme for communicating among groups that use different type hierarchies.

9	A comparison of type hierarchies and their use in communication.
4	A method for communication which is based on the use of hierarchies.
8	Applications of type hierarchies.
5	Communication among objects which is based on alternative type hierarchies.
6	Communication among objects.
3	Group communication.
2	Procedures for communication among groups.
1	Schemes for group communication.
7	Using type hierarchies for communication.

In order to be able to provide a ranking for such a set of phrases the sample users had to have a certain degree of knowledge of computing. The scenario which is being modelled in the experiment is a large information base of computing journals dealing with office information systems. It is fair to assume that the user of such a system would have a certain amount of knowledge about such systems and it must certainly be assumed that the user of an information retrieval system has some knowledge of the type of information they are looking for, at least to the extent of deciding whether returned documents are or are not relevant.

The participants in this experiment represent a fair sample of the type of people who would be using such an information retrieval system. The sample consisited mainly of postgraduate students of Computer Applications at Dublin City Universiy, who would have a good all-round knowledge of computing as well as a particular area of specialisation. There was also some participation from a group of postgraduate and research students at the University of Strathclyde in Glasgow, one lecturer of Computer Science at Univerisy College Dublin and a team of information retrieval researchers at Computer Resources International in Denmark. The phrases that each user ranked did not necessarily deal with topics of that persons specific interest but then, one does not only use an information retrieval system for retrieving information relevant only to one's own field of interest. On the whole, I believe that the judgement of those people who participated in the experiment forms a reasonable sample of the judgement that would be used by users of an information retrieval system in judging the relevance or otherwise of retrieved documents or titles.

7.2.3 The Normalisation of the Collected Data.

The use of Midranks.

After an initial analysis of the rankings that I collected from different people, I found that different people compiled the rankings in different ways. In particular, people handled the occurrence of ties differently. For example, if there was a set of phrases where there

was a two-way tie for rank position 1 and a three-way tie for position 4 then people generally handled this in two different ways as follows,

1	1	2	3	3	3	4	5	6
1	1	3	4	4	4	7	8	9

Upon examination I found that this was simply a matter of format. What I needed was a standard format for handling ties so that I could have all rankings in this format. It turns out that from a statistical point of view, neither of the above formats are desirable. In each of the three statistical tests that I used in the experiment, ties are handled by assigning midranks. Midranks are calculated by taking the average of the rank positions being filled by the ties. For example, if there is a two-way tie for position 1, then this fills positions 1 and 2. The midrank is the average of these values and is therefore 1.5. Similarly with the three-way tie at position 4. This accounts for positions 4, 5 and 6. The midrank is (4+5+6)/3 which is 5. The complete ranking for the above example, after midranks have been assigned, is

1.5	1.5	3	5	5	5	7	8	9

This process of assigning midranks had to be performed for all of the total of 320 phrase set rankings performed by the sample of users and also for the rankings performed by the TSA matching algorihtm. A simple C program was therefore written to complete the task.

The Calcualtion of the Mean Rank.

When I originally designed this experimental evaluation of the matching algorithm, I imagined that I could calculate the overall correlation between the human and computer rankings by comparing the matching algorihtm's rankings to each of the sample users' rankings and then calculating the average correlation from these figures. Because of the work that is involved in calculating the correlation between two sets of rankings, I instead decided to first calculate a mean human ranking and then compute a single correlation between this and the match algorithm's rankings.

Table 7.1: Illustration of mean rank calculations.

Phrase	1	2	3	4	5	6	7	8	9
Subject									
JD	7	7	7	7	4	3	2	1	7
LB	2	7	4	1	7	7	7	7	3
GP	2	1	6	7	9	8	4.5	4.5	3
PS	9	4	8	5	6	3	2	1	7
PC	9	7	6	5	4	3	1	2	8
DM	7	4	8.5	1	8.5	4	4	4	4
SP	4	1	9	2	8	7	6	5	3
PD	4.5	8	9	1	6.5	6.5	2.5	4.5	2.5
PM	7.5	7.5	7.5	4	4	4	2	1	7.5
NC	6	2.5	9	1	8	5	2.5	4	6
Mean	5.4	4.1	7	3.1	6.1	4.5	2.8	2.9	4.5
Midrank	7	4	9	3	8	5.5	1	2	5.5

The mean human ranking was quite simply calculated for each phrase by adding up the rank positions assigned to it by each of the sample users and then dividing this number by the number of people who assigned a ranking to it (always 10 in this experiment). For example, if we examine the information contained in Table 7.1, the top half of the table is the midranked data representing the rankings assigned to the phrases of set 1 by the 10 users who provided rankings for that set. This corresponds to the ranking of the phrase set illustrated earlier. The rankings provided by each user are organised across the rows. The first line of the bottom half of the table contains the mean rank positions assigned to each phrase. These are calculated by simply adding up each column and dividing by 10. For example, phrase 1 was assigned rank positions varying from 2 to 9 but the mean position is 5.4. These mean rank positions are then ranked (again using midranks) to arrive at the final mean human ranking for each phrase set. This is presented in the final row of the table. To illustrate, phrase 7 has the best mean rank of 2.8 so this is given the midrank position 1, phrase 8 has the next best mean rank at 2.9 so this is midranked position 2. So forth down to phrase number 3 which is midranked 9 because its mean rank position is the lowest at 7.

7.3 The Test of Friedman.

Friedman's test was used to test the hypothesis that "*there is no significant difference between the rankings assigned to phrases by different people*". In order to perform this test the data must be organised in the form of **randomized complete blocks**:

> "A *randomized complete block design* for the comparison of *s* treatments requires blocks of size *s* with the *s* subjects in each block being assigned to the *s* treatments at random. The blocks may consist of animals from the same litter, of agricultural plots that are close together, of students that are matched by class and past performance, or patients matched on such variables as age, sex, and severity of disease." [Lehmann 1975]

This requirement was fulfilled by my experiment. The first condition of the blocks being 'square' (i.e. *s* treatments being assigned to *s* subjects) was not applicable in this case as I was not dealing with testing how different people react to different medical treatments but with testing how different people rank a given set of phrases. The requirement of random assignment is fulfilled here since the subjects in the experiment were randomly assigned phrase sets to rank. The only exceptions to this were in the cases where one user performed a ranking of all 32 sets and two users performed rankings on 16 sets each. It should be noted, however, that this condition may also not be significant in this experiment since the phrase sets are completely independent and the particular sets of phrases assigned to a particular subject will have no effect on any particular ranking. I have also fulfilled the requirement of homogeneity as all of the sample users had a considerable backgound in computing. This was important since a high proportion of phrases had a high computing jargon content.

The randomized complete blocks for each phrase set in my experiment consisted of the rank positions assigned to each of the nine phrases of the set by each of the ten users who provided rankings for that set. For example, Table 7.2 above shows the randomized complete block design for phrase set 1. This design would be repeated for each of the 32 phrase sets in the experiment. Note that the sample users participating in the experiment

Table 7.2: The randomized complete block design.

Phrase	1	2	3	4	5	6	7	8	9
Subject									
JD	7	7	7	7	4	3	2	1	7
LB	2	7	4	1	7	7	7	7	3
GP	2	1	6	7	9	8	4.5	4.5	3
PS	9	4	8	5	6	3	2	1	7
PC	9	7	6	5	4	3	1	2	8
DM	7	4	8.5	1	8.5	4	4	4	4
SP	4	1	9	2	8	7	6	5	3
PD	4.5	8	9	1	6.5	6.5	2.5	4.5	2.5
PM	7.5	7.5	7.5	4	4	4	2	1	7.5
NC	6	2.5	9	1	8	5	2.5	4	6

are in the first column and the phrases are positioned across the top. This is because we are observing the way different users assign ranks to each phrase rather than observing how different phrases are assigned ranks by a given user.

For the purpose of this test, $N=9$ is the number of groups, or in this experiment, the number of phrases in each set, and $s=10$ is the number of sample users being observed ranking each set. The Friedman test is based on the fact that there will always be some form of ordering across the rankings provided by the s subjects. This ordering is made explicit by performing a ranking of each of the columns in Table 7.2. Each column (phrase) is taken independently and the values in that column are ranked. The results of this process are displayed in Table 7.3 below. Note that in all cases of ranking (in both Table 7.2 and Table 7.3) tied ranks are denoted by the ***midrank*** - the value lying in between the tied values.

To clarify the contents of each of these tables, Table 7.3 contains the actual data that I collected during the experiment. Each row of the table represents the rank positions of each of the nine phrases as assigned by the user named in the leftmost column. So for example, in the first case phrase number 1 was assigned rank position (joint) 7 by JD, phrase number 2 was also assigned rank position 7, phrase number 6 position 3, and so forth. As there are nine phrases to be ranked, one can see that the rank positions across the rows in Table 7.2 range from 1 to 9. Table 7.3 is then constructed as part of the Friedman test. The principle is that the rank positions assigned to each phrase by the

Table 7.3: Column rankings for the Friedman test.

Phrase	1	2	3	4	5	6	7	8	9	R_L
Subject										
JD	6.5	7	4	9.5	2	2	3	2	7.5	43.5
LB	1.5	7	1	2.5	6	10	10	10	3	51
GP	1.5	1.5	2.5	9.5	10	8.5	8	7.5	3	52
PS	9.5	4.5	6	7.5	4	2	3	2	7.5	46
PC	9.5	7	2.5	7.5	2	2	1	4	10	45.5
DM	6.5	4.5	7	2.5	9	4.5	7	5.5	5	51.5
SP	3	1.5	9	5	7.5	8.5	9	9	3	55.5
PD	4	10	9	2.5	5	7	5.5	7.5	1	51.5
PM	8	9	5	6	2	4.5	3	2	9	48.5
NC	5	3	9	2.5	7.5	6	5.5	5.5	6	50

different users can be arranged in some order. This ordering is made explicit by ranking each value in each column in ascending order. For example, the ranks of 2 assigned to phrase 1 by LB and GP are the highest rank positions for that phrase so they are given rank position 1.5 (the midrank) for phrase 1 in Table 7.3. The next highest rank position was by SP who assigned position 4 to phrase 1 so this is given rank 3 in Table 7.3. Since there were 10 subjects providing rankings for each phrase, the rank positions down the columns in Table 7.3 range from 1 to 10.

The large number of ties in the column rankings of Table 7.3 is significant and desirable. It reflects the fact that the same rank position was assigned to a given phrase by several different people. It reflects the degree of agreement between the different users as to the relevance of the given phrase to the base title.

An indication of the position of any ranking i within the overall ordering is given by the average rank calculated over the N blocks (phrases):

$$R_{i.} = \frac{R_{i1} + \ldots + R_{iN}}{N} \tag{7.1}$$

If the rankings assigned by different users to a given phrase differ widely among each other, it will be reflected in large difference among the R_i's. On the other hand, when the hypothesis that there is no significant difference between the rankings assigned by different

users is true, the R_i's tend to be close to each other. In this case the R_i's are also close to the overall average which is calculated by:

$$R_{..} = \frac{(R_{11} + \ldots + R_{s1}) + \ldots + (R_{1N} + \ldots + R_{sN})}{sN} \tag{7.2}$$

since the numerator of this expression is equal to $Ns(s+1)/2$ this means that

$$R_{..} = \frac{(s+1)}{2} \tag{7.3}$$

Friedman proposed a statistic for measuring the closeness of the R_i to this average $R_{..}$. This is given by:

$$Q = \frac{12N}{s(s+1)} \sum_{i=1}^{s} R_i^2 - 3N(s+1) \tag{7.4}$$

where Q is known as the *Friedman Statistic*.

The Friedman statistic Q will be equal to zero when the R_i are all equal and will be large when there are substantial differences among the R_i. The hypothesis of no difference in rankings is rejected when

$$Q \geq c \tag{7.5}$$

Tables of the distribution for this statistic for certain values of s and N have been constructed by Friedman. For larger values of s and N however, it is found that the X^2 (Chi Squared) distribution with $s-1$ degrees of freedom provides a good approximation. This is so in our case where $s=10$ and $N=9$, since these values are not provided in the Friedman tables and the values are so large so as to ensure that the X^2 tables provide a suitable approximation. In general then,

$$P_H(Q \geq c) \approx \phi_{s-1}(c) \tag{7.6}$$

and the critical values of ϕ (the X^2 distribution) can be found in any book of statistical tables. This will become much more clear when I work through a detailed example later.

The Presence of Midranks:

The formulae presented above are actually based on the assumption that all rank positions are unique; i.e. no ties are allowed. The TSA evaluation experiment is more complicated because phrases may be ranked equally and so ties may occur. This has been represented in the tables by assigning midrank values whenever a tie occurs. Let e_i denote the number of distinct values in block (column) i. Then d_{1i} represents the number of observations being equal to the smallest of these values, d_{2i} represents the number being equal the the next smallest, and so forth with d_{ei} being equal to the number of observations equal to the highest rank value. Taking the rankings for phrase 1 in Table 7.3, there are 7 distinct values (1.5, 3, 4, 5, 6.5, 8, 9.5) and $d_{11} = 2$, $d_{21} = 1$, $d_{31} = 1$, $d_{41} = 1$, $d_{51} = 2$, $d_{61} = 1$ and $d_{71} = 2$.

If R^*_{ij} is the midrank assigned to phrase j by subject i then the sum of midranks assigned by a given sujbect i is

$$R^*_i = R^*_{i1} + \ldots + R^*_{iN} \tag{7.7}$$

The Friedman Statistic is then defined by

$$Q^* = \frac{[12/Ns(s+1)]\Sigma R^{*2}_i - 3N(s+1)}{1 - \sum_{j=1}^{N}\sum_{i=1}^{e_j}(d_{ij}^3 - d_{ij})/Ns(s^2-1)} \tag{7.8}$$

again, the hypothesis that there is no significant difference between the rankings assigned to phrases by different subjects is rejected if

$$Q^{\bullet} \geq c \qquad (7.9)$$

The critical values *c* can be found, as before, by reading the X^2 tables with *s-1* degrees of freedom.

Example:

I will work through the calculation of the Friedman Statistic for the data collected on the ranking of phrases in phrase set 1, represented in Table 7.2. Taking, first of all the numerator of formula (7.8), [12/Ns(s+1)] = 0.012121 (N=9, s=10) and $\Sigma R^{\bullet}_i{}^2$ = 24620.5 (the values of $R^{\bullet}_i$ are included at the right of Table 7.3). Multiplying these two values gives 298.4303 and then substracting 3N(s+1) leaves the numerator to be **1.430303**.

The calculation of the denominator is not quite so simple. One way of simplifying the calculation of the sums of ties, $\Sigma\Sigma(d_{ij}^3-d_{ij})$, is to count the number of sets of ties in the whole table; i.e. count the number of unique values, the number of paris, triplets, quadruplets etc.. In Table 7.3 there are 38 unique values, 15 pairs, 6 triplets, and 1 quadruplets. The values for (d^3-d) are also calculated at each level:

$$(1^3 - 1) = 0$$
$$(2^3 - 2) = 6$$
$$(3^3 - 3) = 24$$
$$(4^3 - 4) = 60$$

The calculation then simply consists of the sum of the multiples,

$$(38*0) + (15*6) + (6*24) + (1*60) = 294$$

Dividing this by $Ns(s^2-1)$ gives 0.0329966 and subtracting this from 1 gives the denominator a value of **0.967003**. The Friedman Statistic is then 1.430303/0.967003 = **1.479108**.

Checking the X^2 tables with s-1=9 degrees of freedom, the critical value at the 1% level of significance is 21.6660. Since 1.479108 < 21.6660, ($Q^* \geq c$ is false) we must accept the hypothesis and conclude that there is no significant difference between the rankings supplied by the different users for phrase set 1. Note too that even at the 25% level of significance the critical value is 11.3888 so the hypothesis will still be accepted. It is also worth remarking on the difference between the Friedman Statistic and the critical value *c* from the tables; i.e. between 1.479108 and 21.6660. Because of the large difference one can state that, in general, there must be very little variance between the rankings assigned by different users. Of course one can see from the tables that there are certain cases where there is quite high variability in some of the phrase rankings but this result shows that in general the differences are not significant. The cases of high variability can be explained by referring to the construction of the dataset. As explained in section 7.2.1 the phrase sets were deliberately constructed in such a way so that in some cases it would be difficult for both man and machine to make fine distinctions between the levels of relevance of some of the phrases. It is in these cases that the variability in the rank positions assigned to the phrases is higher.

The general low level of variability in the rank positions assigned to the phrases in set 1 is a desirable result since it indicates that the base set of human rankings that I am using in the evaluation of the TSA matching algorithm, at least in phrase set 1, is sound and appropriate. I will analyse this further now by examining the results of the Friedman test for all 32 phrase sets of the experiment.

7.4 An Analysis of Human Rankings.

The purpose of analysing the phrase rankings provided by the sample information retrieval system users is to determine whether there is a suitably high level of agreement between

users as to the relative relevance of a given set of phrases to a particular base phrase. If there is a high level of agreement then it demonstrates that the mean rankings of sample users is a suitable benchmark against which to compare the performance of a computer algorithm designed to perform the same task. The Friedman test is used to test the hypothesis that there is no significant difference between the rankings provided by different users. I would hope that the results of the Friedman test would demonstrate that in all or most of the 32 cases there was no significant difference in the way different people ranked the phrase sets. This would show that there exists a single *best* ranking for each set of phrases and that it is therefore appropriate to use this as a *performance target* for the TSA matching algorithm. The aim of this research was to design an algorithm which ranks phrases in a manner which is acceptable to the average user of an information retrieval system - such that the phrases are ranked in a manner which is comparable to the way in which the user would rank them using his/her own judgement on their relevance to the retrieval query. If there exists a single *best* ranking for a given set of phrases then it is desirable that an information retrieval system achieves this ranking.

The results of the Friedman test for each of the 32 phrase sets are presented in Table 7.4. These were computed by performing the above calculations on the 10 rankings collected for each of the 32 sets of phrases. Note that the value for phrase set 1 in the table is the value computed during the example above. We can examine the overall variability in the ranking of humans by examining each of the Q^* values given, and by comparing the values to the critical value for the Friedman test, as described above. In the experiment there are 10 rankings for each phrase set so $s=10$. The critical value is found using the X^2 approximation with 9 degrees of freedom.

In order to see if there is a significant variance in the ranking of the sample users we compare the Friedman statistics to the critical value at the 1% level of significance. Recall from above that the hypothesis of no significant difference is rejected when

$$Q^* \geq c$$

Taking the critical value found above, the hypothesis of no difference will be rejected when,

Table 7.4:

Friedman Statistics for Human Rankings.

Phrase Set	Q*	Phrase Set	Q*
1	1.479	17	3.351
2	6.702	18	0.646
3	1.255	19	0.555
4	1.363	20	1.587
5	0.847	21	1.974
6	2.366	22	2.122
7	3.858	23	1.243
8	0.171	24	1.710
9	6.587	25	1.946
10	4.552	26	0.600
11	1.177	27	1.523
12	1.283	28	3.635
13	2.005	29	2.025
14	1.391	30	1.546
15	5.384	31	2.297
16	6.041	32	1.266

$$Q^* \geq 21.666$$

In the case at hand, this proves to be a trivial comparison. Looking at Table 7.4, the highest value of Q^* is 6.702. This is well less than the critical value of 21.666 so we can conclude that **for all 32 phrase sets, there is no significant difference in the way different subjects ranked the phrases.** We can examine the results more closely to perform a more detailed comparison to the critical value and so give a more detailed statement about what we mean by *no significant difference*. This is best achieved by a graphical display of the positions of the Friedman statistics along the chi-squared distribution. Figure 7.1 displays the chi-squared distribution with the Friedman statistic for phrase set number 1 (1.479), the

highest Friedman statistic recorded in the experiment (6.702), and the critical value which must be exceeded for the null hypothesis to be rejected.

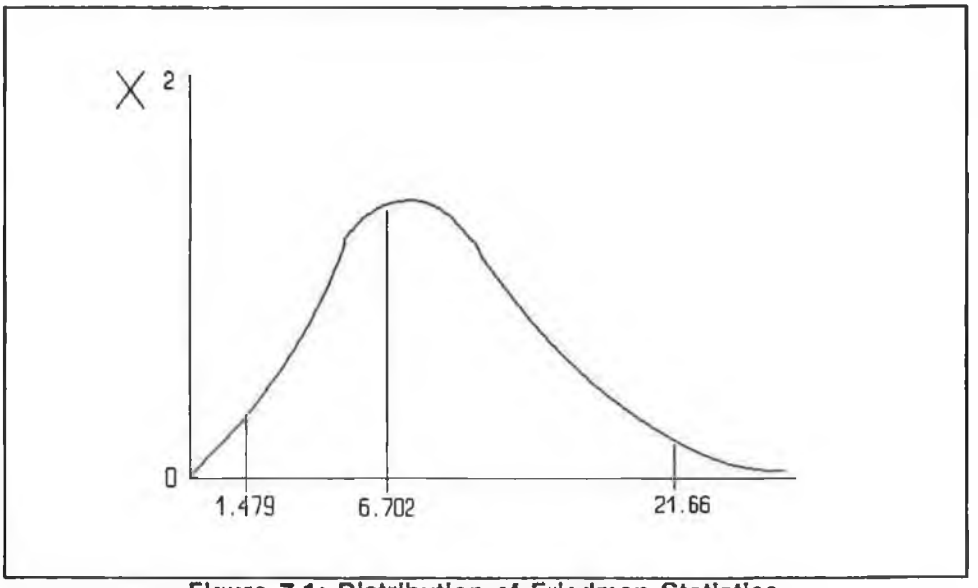


Figure 7.1: Distribution of Friedman Statistics.

This figure gives a much better impression of the spread and degree of the Friedman results. All of the results are well within the acceptance region (between 0 and 21.666) so the hypothesis that there is no significant difference between the rankings of different people is safely accepted for each of the 32 phrase sets and never even comes close to being rejected.

These results fulfill my hopes for the first part of the experiment. The results show that there is a very high level of agreement between different people as to the ranking of phrases. This is particularly noteworthy since in the experiment the relationships between phrases were made deliberately confusing by choosing some phrases that were very close in semantic content but with different syntactic structures, and choosing some which had similar structure and high overlap in word content but which had completely different meanings. Despite this we find that taking 24 different people ranking different combinations of 32 different sets of 9 phrases, there were very small variances in the resulting rankings.

This is useful for the purpose of constructing a base against which we can compare the

performance of the TSA matching algorithm. There is convincing evidence that there is a common view among different people as to the particular order into which a set of phrases can be ordered in relation to a single base phrase. More specifically, given a single search request in an information retrieval environment, there is convincing evidence that there is a common agreement among different users as to the relative relevance of document titles in the text base. This leads me to believe that if I can get the matching algorithm to produce rankings equivalent to the average rankings assigned by the sample uers in the experiment, then this is the *best* or *most appropriate* level of performance one could expect, and in general the rankings of phrases produced will correspond well to users expectations. I feel therefore, that I am justified in using the mean user rankings computed in the experiment as a target for the matching algorithm's performance.

It may also be possible to gather some useful information regarding retrieval systems in general from these results. The question has been asked as to whether ranking of retrieved documents in an information retrieval system is a good idea at all. I believe these results indicate that it is indeed desirable. I have not only shown that people can order a set of phrases in accordance with their relevance to a particular target phrase but I have also shown through a (small) sample group that users of an information retrieval system can produce remarkably similar orderings. If a retrieval system can rank retrieved documents into this ordering then there will obviously be a high level of user satisfaction. My aim was to design a retrieval mechanism that could produce such an ordering.

Given that there now existed a solid benchmark against which I could compare the performance of the TSA matching algorithm, this comparison was initially intended to be done using two different statistical tests. The first was the Wilcoxon signed rank test. This test would prove the truth or otherwise of the hypothesis that "there was no significant difference between the mean user rankings and the rankings produced by the matching algorithm. The second statistical measure was the Spearman correlation, which produces a measure reflecting the correlation between the user rankings and the algorithm's rankings. This provides a more specific comparison between the two sets of rankings and can be used in the calculation of an overall performance measure for the TSA matching algorithm.

7.5 The Wilcoxon Signed Rank Test.

The Wilcoxon test is a test used for paired comparisons in which one subject is considered to be the *control* and the other is the *treated* (These terms were assigned in the traditional context of medical/agricultural experimentation). In my experiment I assumed that the rankings provided by the human subjects were the control and the computer rankings were considered as the treated. The test is used to establish if there is a significant difference between the control and the treated. One simple method to test this hypothesis is based on the number, S_N, of pairs for which the treated subject ranks better than the control. The hypothesis of no difference would then be rejected when S_N was sufficiently large. This simple test is known as the *sign test*, and would be used in cases where it is only known which subject ranks higher, without having a quantative assessment. For example when the only information is that "*A is better than B*". In the case at hand however, I had the values of the rank positions assigned by the human subjects and the computer algorithm so we can use the *Wilcoxon signed rank test*. The signed rank test is as follows:

If there are N pairs of subjects, where each pair consists of a control subject and a treated subject, then $N_+ = n$ of the differences between the treated and control values are positive and the remaining $N_- = m = N - n$ differences are negative. Because I had actual values for the differences I could rank them according to their absolute values and then attach the sign of the difference to the ranking. This is illustrated in Table 7.5.

Note that the rank positions given for the human ranking in the Table are the mean rank positions as calculated in Table 7.1. The difference is simply computed by subtracting the computer rank position from the mean human rank position. The signed midranks are computed by first ranking the absolute values from the row of differences (e.g. the values of (-)1 for phrases 2 and 5 are the lowest so they receive joint rank position 1 - midrank 1.5) and then assigning the sign of the difference to the midrank.

If we wanted to test the hypothesis that there is no difference between the two sets of rankings against the alternative that the human rankings are better than the computer

Table 7.5: Signed Midranks of absolute value differences.

Phrase	1	2	3	4	5	6	7	8	9
Human Rank	7	4	9	3	8	5.5	1	2	5.5
TSA Rank	3	5	1.5	6	9	7.5	7.5	4	1.5
Difference	4	-1	7.5	-3	-1	-2	-6.5	-2	4
Midrank	-6.5	-1.5	9	-5	-1.5	-3.5	-8	-3.5	6.5

rankings[7], then one simple method would be to count the number of cases, $(S_1, \ldots, S_n)$, where the treated subject ranks higher than the control; i.e. count the number of positive differences. It is also relevant to note whether the differences in the rankings are larger in these positive cases than when the difference between the rankings is negative. One test statistic is based on the sum of the signed midranks which are positive:

$$V_s = S_1 + \ldots + S_n \tag{7.10}$$

and the hypothesis of no difference is rejected when

$$V_s \geq c \tag{7.11}$$

This simple statistic is based on the assumption that there are no ties in the differences and that none of the values in any pair are identical (in which case the difference is zero). It is therefore not applicable in the present case. In the presence of ties, the differences are calculated and the signed midranks are assigned as before. *After* all midranks have been calculated, the zero differences are assigned a midrank of zero. The sum of the positive midranks (V_s^*) is also calculated as before. For large values of N, the significance value for V_s^* can then be calculated using a Normal approximation as follows

$$E_H(V_s^*) = \frac{N(N+1) - d_0(d_0+1)}{4} \tag{7.12}$$

[7] Note that this alternative provides the reason for choosing the human rankings as control. We must assume that if there is a difference between the sets of rankings then the human rankings are better. It is unlikely that I could claim that the matching algorithm's rankings were better than those performed by humans.

where d_0 is the number of zero differences. The variance of V_s^+ depends on both the number of zero differences and on the number of ties among the midranks, d_1, ... ,d_e, and is given by the following formula, (7.13):

$$Var_H(V_s^+) = 1/24\ [N(N+1)(2N+1) - d_0(d_0+1)(2d_0+1)] - 1/48 \sum_{i=1}^{e} d_i(d_i-1)(d_i+1)$$

The significance value for V_s^+ is then given by

$$P(V_s^+ \geq x) \approx 1 - \Phi\left(\frac{x - E_H(V_s^+)}{\sqrt{Var_H(V_s^+)}} \right) \qquad (7.14)$$

where the $1-\Phi$ value is found in the tables for the Standardised Normal Distribution.

Example.

We will follow the example of applying the Wilcoxon signed rank test to the average human rankings and the computer rankings for phrase set 1, using the data already cotained in Table 7.4. I have already calculated the differences and assigned signed midranks. The sum of the positive midranks is, $V_s^+ = 9 + 6.5 = 15.5$. Looking at the distribution of tied midranks, $d_0=0$, $d_1=3$, $d_2=3$. Thus, $E_H(V_s^+) = 90/4 = 22.5$, and $Var_H(V_s^+) = 1710/24 - 18/48 = 70.875$. Therefore:

$$P(V_s^+ \geq 15.5) \approx 1 - \Phi\left(\frac{15.5 - 22.5}{\sqrt{70.875}} \right)$$

The entry in the Standardised Normal Tables for -0.831 is **0.79** so

$$P(V_s^+ \geq 15.5) \approx 0.79$$

Since the null hypothesis is that there is no significant difference between the mean human rankings and the computer generated rankings (i.e. H_o: $V_{Human} = V_{Computer}$) and we are not interested in the direction of any difference that may arise, a two-tailed statistical test is used. The critical values for the two-tailed standarised normal distribution at the 5% level of significance are ±1.96; i.e. in order to reject the null hypothesis the value calculated above (i.e. -0.831) must lie outside ±1.96. This will be clearer if the standardised normal distribution is displayed graphically, as below:

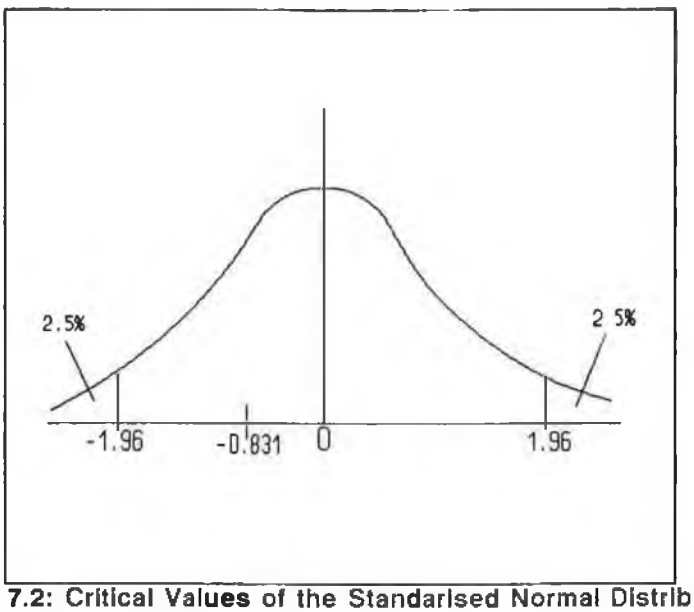


Figure 7.2: Critical Values of the Standarised Normal Distribution.

Since the Wilcoxon value of -0.831 is within the limits of ±1.96, there are no grounds to reject the null hypothesis and we can conclude that there is no significant difference between the mean human rankings and the computer ranking for phrase set 1. This is the desired result since the objective is to have a computer algorithm which ranks phrases in a comparable manner to humans.

7.6 The Spearman Correlation.

The Spearman Correlation goes one step further than the Wilcoxon Signed rank test. Rather than simply testing whether or not there exists a relationship between the human and computer rankings, the Spearman test calculates a measure of correlation which indicates the extent of the similarity or dissimilarity between the two rankings.

In order to explain the calculations involved with this measure of correlation, let us take the data for the first phrase set. Table 7.6 displays the data on the ranking of phrase set 1 by both the sample users and the TSA matching algorithn. The users' rank positions are a mean of the rankings assigned by all 10 subjects who ranked this phrase set, as explained earlier.

Table 7.6: The two sets of rankings for phrase set 1.

Phrase	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Human	7	4	9	3	8	5.5	1	2	5.5
Computer	3	5	1.5	6	9	7.5	7.5	4	1.5

We can get a clearer picture of the relationship between these two sets of rankings by arranging one of them in its natural order. If we drop the phrase numbers and sort the human rankings into order then the data is as follows:

Figure 7.7: Rankings arranged in ascending order.

Human	1	2	3	4	5.5	5.5	7	8	9
Computer	7.5	4	6	5	7.5	1.5	3	9	1.5

Let us first of all consider the case that we are wondering about whether there exists a relationship between the two sets of rankings or whether they are completely independent

of each other. Let us denote the ranks assigned by the computer algorithm as $(T_1, \ldots, T_N)$, where $N=9$. If the hypothesis is true that the ranks assigned by the computer are completely independent of those assigned by the human subjects, then all $N!$ orderings of $(T_1, \ldots, T_N)$ are equally likely. The probability of any one of these orderings occuring then, (which also represents the null distribution underlying the hypothesis) is given by

$$P_H(T_1=t_1, \ldots, T_N=t_N) = \frac{1}{N!} \tag{7.15}$$

This hypothesis, that all $N!$ orderings of ranks of the N phrases are equally likely, is also referred to as the *hypothesis of randomness*. The question that must be asked is, "For what values of $(T_1, \ldots, T_N)$ should the hypothesis of independence be rejected?". The alternative that we wish to consider in this case is that there is a *positive association* between the two sets of rankings. This would mean that high rank positions assigned by the computer algorithm would correspond to high positions assigned by the sample users and low rankings by the computer would correspond to low rankings by the users. One simple statistic for testing the correspondance of the rankings is given by $D = \Sigma(T_i-i)^2$, for which small values for D would be significant; i.e small values of D indicate that there is little difference in the two rankings and so the hypothesis of independence must be rejected.

It is also possible to calculate the statistic D without having to arrange one of the rankings in its natural order. We can look on the rankings as a matrix, with the two rank positions of the first phrase being represented by (R_1,S_1), the positions of the second phrase represented by (R_2,S_2) and so on. It is apparent, then, that the differences $(T_i - i)^2$ and $(S_i - R_i)^2$ are simply the same N numbers just arranged in different order. It follows, therefore, that

$$D = \Sigma(S_i - R_i)^2 \tag{7.16}$$

As before, small values of D are significant. Tables have been drawn up by Spearman to indicate what values of D should be considered significant and which should not. The statistic D is also related to the calculation of an overall correlation coefficient between the

two rankings. This was defined by Spearman and is known as *Spearman's rank correlation coefficient*, denoted by r_s.

$$r_s = \frac{\Sigma(R_i - \bar{R})(S_i - \bar{S})}{\sqrt{\Sigma(R_i - \bar{R})^2 \Sigma(S_i - \bar{S})^2}} \tag{7.17}$$

where $\bar{R} = \Sigma R_i/N$ and $\bar{S} = \Sigma S_i/N$. This formula, however, reduces to

$$r_s = 1 - \frac{6D}{N^3 - N} \tag{7.18}$$

Interpreting the Value of r_s

The value r_s is equivalent to the test statistic *D*, but large values of r_s are considered to be significant. The values of r_s range from -1 to 1. The correlation r_s takes on its maximum value 1 only when *D* takes on its minimum value 0. This occurs when $T_1=1$, ..., $T_N=N$. The correlation coefficient will take on its minimum value -1 when $T_1=N$, ..., $T_N=1$; i.e. the second ranking is a complete reversal of the first. The values of the coefficient are illustrated graphically in Figure 7.3 below.

We are interested, in this experiment, in finding out just how much of a correlation there is between the rankings provided by the sample users and the ranking provided by the TSA matching algorithm. One would hope therefore, that the correlation coefficients are as close to 1 as possible, as a value of 1 means that the two rankings are identical. Negative correlations would be particularly disappointing as it would mean that the matching algorithm was ranking phrases in the opposite way to the sample information retrieval system users.

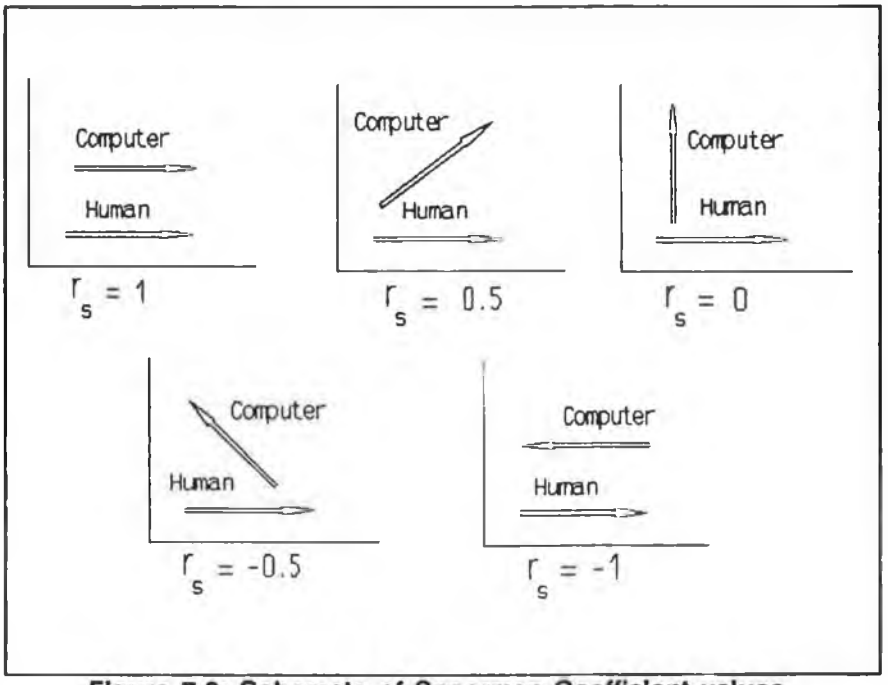


Figure 7.3: Schemata of Spearman Coefficient values.

Example.

I will continue to use phrase set 1 to illustrate the Spearman calculations. The first thing to be calculated is the test statistic D by computing the differences between the two ranking sets. This data is presented in table 7.8.

From the table we see that $D=149.5$. It has also been given earlier that $N=9$. Substituting these values into Formula (7.18) gives $r_s = 1 - 897/720 =$ **-0.24583**. Unfortunately this is not the type of result that would be hoped for. Since there is a negative value, it means that there is a negative correlation between the user rankings and the algorithm's rankings for phrase set 1, or in other words, the computer has ranked phrase set 1 in the opposite way to the human subjects. Note that the rankings are not directly opposite (as this would produce a value of -1), but this is definitely not an acceptable performance level for the matching algorithm.

Table 7.8: Calculation of Spearman Coefficient.

Phrase	Average Human Rank	Computer Rank	Difference	Squared
1	7	3	4	16
2	4	5	-1	1
3	9	1.5	7.5	56.25
4	3	6	-3	9
5	8	9	-1	1
6	5.5	7.5	-2	4
7	1	7.5	-6.5	42.5
8	2	4	-2	4
9	5.5	1.5	4	16
			Total (D) =	149.5

7.7 Conclusion.

Quite a bit of time was spent in choosing the three statistical tests described above. Because of the nature of the experiment (the fact that it was concerned wholly with the comparison of rankings) it meant that I had to use non-parametric statistics. I have already described how I used the Friedman test to good effect in illustrating the low level of variation in the rankings of different users. I have proven the basis of human rankings to be most appropriate both as a basis for comparison and as a target for the performance of an optimal matching algorithm. I believe that in the Wilcoxon and Spearman tests there is enough information to ascertain the power of the TSA matching algorithm by comparing it to the mean of the sample user rankings. The objective is obviously to achieve an overall average Spearman corelation coefficient as near to 1 as possible.

As mentioned earlier in this chapter, one of the secondary objectives of this evaluation experiment was to provide a mechanism whereby I could iteratively optimize the performance of the matching algorithm by performing several iterations of a modify/re-evaluate cycle. The first application of the Wilcoxon and Spearman tests were therefore

on an initial version of the matching algorithm and the results were obviously not expected to be optimal. The hope was that the initial evaluation would indicate that the matching algorithm was indeed "useful" and that, after several iterations, the final evaluation results would prove the matching algorithm's worth.

In the next chapter I will begin by presenting the initial results of the application of the Wilcoxon and Spearman tests to the TSA matching algorithm. I will then present the intermediate results from the modification/re-evaluation stage and I will finally present the final results that were considered to be "optimal" in some sense.

Chapter 8.

Analysis of Experimental Results

8.1 Introduction.

In this chapter, the results of the statistical evaluation of the TSA matching algorithm are presented and analysed in some detail. The evaluation of the matching algorithm is based on a comparison of rankings of phrases performed by the matching algorithm to rankings of the same phrases by a sample of information retrieval system users. The statistical tests used for this comparison are the Wilcoxon Signed Rank test and the Spearman Correlation, as described in the previous chapter.

It was also mentioned in the previous chapter that there would be several stages in the evaluation of the TSA matching algorithm. The initial statistical evaluation was performed on the first prototype of the matching algorithm and therefore I did not hold high expectations for the level of results. What was sought from the first evaluation was an indication that the approach had some merit and that with some adjustment to the matching rule-set and some optimization of the scoring parameters, the TSA matching algorithm would prove itself very useful.

I will start off in this chapter then, by presenting the results of the very first evaluation of the TSA matching algorithm and I will comment on these results and, in particular, on my opinion of the appropriateness of the statistical tests used. I will then briefly describe the iterations that took place between modifying the matching rule-base and scoring mechanism and the re-evaluations using the Spearman correlation statistics. The final results of the evluation of the matching algorithm will then be presented and discussed in some detail.

Since the same dataset of 32 sets of phrases that was used in the experimental evaluation of the matching algoithm was used throughout all iterations of the evaluation, it seemed quite possible that the resulting TSA matching algorithm would be "tuned" to that specific dataset. I was therefore interested in evaluating the matching algorithm's performance on a completely different dataset from a different domain, once the algorithm had become stable. This involved repeating all of the initial tasks of the experimental design, constructing the dataset, having sample users provide rankings for the different phrase sets, calculating mean user rankings, having the TSA matching algorithm perform the same

rankings and then comparing the two using the statistical tests.

In the final parts of this chapter I will describe this second dataset and experiment and present the results achieved for that data. I will conclude by commenting on the overall performance of the matching algorithm, highlighting its particular strengths and weaknesses, and suggesting how some of the weaknesses could be overcome.

8.2 Results of the Initial Wilcoxon Test.

The Wilcoxon Signed Rank Test, as described previously, was used to determine if there was a significant difference between the mean human ranking and the computer generated ranking for each of the 32 phrase sets. I was not interested in testing the directions of differences that may occur so a two-tailed normal distribution was used in the analysis. The null hypothesis can be formulated as: $H_o \rightarrow V_{Human} = V_{Computer}$. For each of the 32 phrase sets a value of u was caluclated from the formula

$$u = \frac{x - E_H(V_S^*)}{\sqrt{Var_H(V_S^*)}}$$

The values of u represent the points along the X axis of the Standardised Normal Distribution. The probabilities which represent the proportion of the area under the curve to the right of each u point are then read from the statistical tables and are given by:

$$P(V_S^* \geq x) \approx 1 - \Phi(u)$$

To illustrate these readings on the standardised normal distribution I will use the example of phrase set 1 for which a u value of -0.831 was calculated and $P(V_S^* \geq x) \approx 0.79$. These values are illustrated in Figure 8.1. The figure also illustrates the critical values for the test at the 5% level of significance. These critical values of ± 1.96 are used to test the truth

of the null hypothesis. If any of the u values calculated for the Wilcoxon test lie outside the critical values then the null hypothesis is rejected in that case.

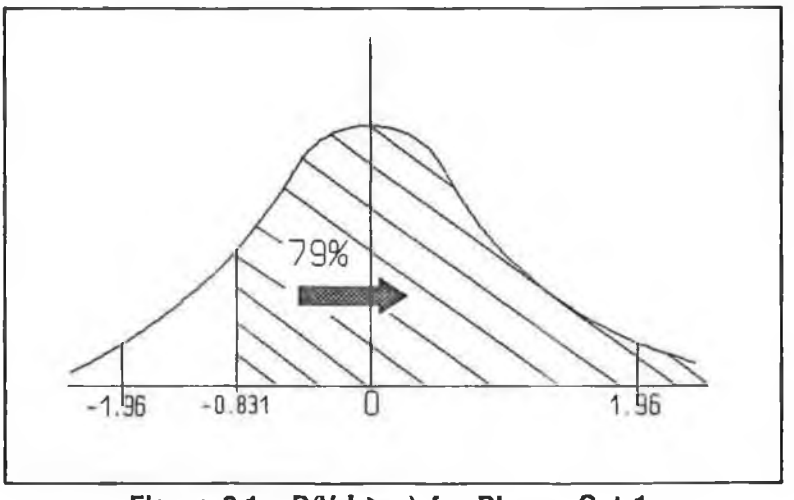


Figure 8.1: $P(V_s^+ \geq x)$ for Phrase Set 1.

Since the u value for phrase set 1 lies within the critical values, the null hypothesis is held true, that there is no significant difference in the mean human rankings and the matching algorithm's ranking for phrase set 1. The Wilcoxon signed rank test was carried out on each of the 32 phrase sets and 32 u values calculated. These are presented below in Table 8.1. In each case, in order to establish whether the null hypothesis is to be accepted or rejected, the u value must be plotted on the standardised normal distribution and compared to the critical values. The two most extreme u values from the Wilcoxon test are -0.889 for phrase set 1 and +1.304 for phrase set 31. Notice too, that below 1.304 the nearest value is 0.474. Plotting all of these values on the standardised normal distribution, as in Figure 8.1, shows that they are all well within the critical range and only the solitary value of 1.304 approaches the critical value. We can therefore conclude that **in all of the 32 phrase sets tested there was no difference between the mean human ranking and the initial version of the TSA matching ranking at the 5% level of significance.**

This is a suprisingly good result. In all cases tested the initial version of the TSA matching algorithm ranked the phrases in a comparable manner to the mean of the human rankings. This means that, according to this test, the TSA matching algorithm had fullfilled its

objective at the first attempt. Since it seems highly unlikely that the initial TSA matching algorithm is optimal (or even as good as these results would lead us to believe), these results led me to wonder about the applicability of the Wilcoxon signed rank test in this case. It would seem that this test provides results at too general a level to be useful for the purpose at hand. Of course I had to assume that, at the level of this test, the above results held true and the human and computer rankings were comparable but what I wanted was a more detailed analysis of the relationship between the two.

Table 8.1:

Results of the Initial Wilcoxon Signed Rank Test.

Phrase Set	u	Phrase Set	u
1	-0.831	17	-0.059
2	0.239	18	0.000
3	-0.119	19	0.239
4	0.237	20	-0.118
5	0.474	21	0.000
6	-0.657	22	0.059
7	0.178	23	0.000
8	0.000	24	0.237
9	-0.118	25	0.178
10	-0.059	26	-0.118
11	-0.059	27	0.059
12	0.119	28	0.000
13	-0.889	29	-0.059
14	0.237	30	-0.178
15	-0.059	31	1.304
16	0.000	32	-0.059

The Spearman correlation calculations that were also carried out on the data provide exactly that type of detailed analysis of the relationship between the mean human ranking and the rankings provided by the TSA matching algorithm for each of the 32 phrase sets analysed. Indeed it was with the specific intention of obtaining more detailed restults that

the Spearman test was used in the experiment. One can now see the justification for using both the Wilcoxon signed rank test and the Spearman correlation for testing the relationship between the human and TSA rankings.

8.3 The Initial Spearman Correlation Results.

The objective of the Spearman correlation test, as described earlier, is to compute a direct correlation measure r_s between two variables. This measure takes a value between -1 and +1 with a value of 1 being the most desirable result, meaning that the two variables are parallel (identical in our case). In the present case we are computing the correlation between the mean set of rankings provided by a sample of users and the rankings computed by the TSA matching algorithm. A value of $r_s=1$ in this test means that the TSA ranking is identical to the mean user ranking for the phrase set in question. It should be noted that negative values of r_s denote a negative or reverse correlation between the two variables. In this experiment this means that the TSA rankings would be in the opposite direction to the mean human ranking (i.e. for r_s = -1: User rankings = 1,2,3,..9, TSA rankings = 9,8,7,..1). The correlation values computed for each of the 32 phrase sets tested are displayed in Table 8.2 below.

These more detailed results give a much better impression of the relationships between the sets of rankings. The correlation values reflect precisely the relationship between the user rankings and the TSA rankings and are therefore much more desirable than the Wilcoxon values which simply allow one to state that "there is or is not a significant difference in the rankings". On analysis of the results, 22 (68%) phrase sets show positive correlation, with 9 (28%) above 0.5. Although 10 (31%) phrase sets have negative correlations, only 5 (15%) have values below -0.1. The complete spread of the correlations is illustrated in Table 8.3. An overall measure of the TSA maching algorithm's performance can be found by computing a single correlation value between the sample user rankings and the algorithm's rankings. This was easily computed by averaging the 32 individial Spearman correlation values presented in Table 8.2. Using these values, the overall correlation was

computed to be **0.224**. The fact that this average correlation was positive was encouraging. It meant that the matching algorithm had a positive correlation to the average human rankings. The ideal correlation value would have to be positive, and would optimally be a close to 1 as possible. A phrase ranking algorithm which was performing as "average" would have an overall average correlation of zero. The initial prototype of the TSA matching algorithm performed better than average, with the correlation of 0.224.

Table 8.2:

The Initial Spearman Correlation Co-efficients.

Phrase Set	r	Phrase Set	r
1	-0.245	17	-0.033
2	0.666	18	-0.125
3	0.595	19	-0.016
4	0.004	20	-0.566
5	0.233	21	-0.033
6	0.483	22	0.116
7	0.516	23	0.266
8	0.583	24	0.433
9	0.520	25	-0.250
10	-0.100	26	0.233
11	0.333	27	0.445
12	0.454	28	-0.350
13	0.125	29	0.725
14	0.033	30	0.283
15	0.562	31	0.587
16	-0.095	32	0.787

These results "*felt*" much better than those produced by the Wilcoxon test. They showed that although the algorithm had performed quite acceptably there was definitely room for improvement. This is what I expected from the initial testing of the first prototype algorithm. It would seem that these Spearman results are much more informative and reliable than those that were produced from the Wilcoxon signed rank test. But why didn't the Wilcoxon test detect differences between the rankings in the phrase sets which have

Table 8.3:

Distribution of Initial Spearman Correlations.

Range	Number	Range	Number
above 0.5	9	below 0	6
0.4 to 0.5	4	-.2 to -.3	2
0.3 to 0.4	1	-.3 to -.4	1
0.2 to 0.3	4	-.4 to -.5	0
above 0.0	4	below -.5	1
Total > 0	22	Total < 0	10

produced negative correlations? I felt it would be worthwhile to make a detailed comparison between the results presented above and the actual raw data in order to provide a manual verification of the results before proceeding with further evaluation.

8.4 A More Detailed Analysis of Initial Results.

8.4.1 The Applicability of the Wilcoxon Test.

The 32 phrase sets with the mean user rankings and the TSA rankings have been included as APPENDIX D. We can manually examine a sample of the rankings of the sets in order to validate and verify some of the above results.

Take for example phrase sets 1 and 2. Phrase set 1 has a negative Spearman correlation of -0.245 whereas phrase set 2 has quite a high positive correlation of 0.666. Upon examination of phrase set 1 we see that the highest ranked phrase by users was ranked in position 7 by the TSA match algorithm and the phrases ranked joint highest by the TSA algorithm were placed in positions 5 and 9 by the sample users. These pairings would

obviously contribute greatly to a negative correlation. On the other hand, looking at phrase set 2, the top three rank positions have been assigned to the same phrases, although in different order, by the users and the TSA matching algorithm. There has also been full agreement for positions 6 and 7. This agreement is reflected in the high positive correlation. It is also interesting to compare phrase sets 2 and 3. In phrase set 3 there is the same agreement between the top three positions and the difference in order but there is not so much agreement about the ordering of the lower rank positions. Phrase set 3 therefore has a slightly lower positive correlation than phrase set 2. This has been reflected in the Spearman correlations of 0.666 for phrase set 2 and 0.595 for phrase set 3. Taking each of the pairs of rankings for each phrase set in APPENDIX D it is possible to verify that the Spearman correlation values presented above are a precise reflection of the degree of similarity between the two rankings.

It would seem then that the results of the Wilcoxon test, although very favourable, are quite dubious, while the results of the Spearman correlations reflect exactly the relationships between the rankings. Some attempt had to be made to investigate and explain this lack of faith in the Wilcoxon results. Surely in the cases where there was a negative correlation between the mean user rankings and the TSA algorithm's rankings, the Wilcoxon test should have rejected the null hypothesis and concluded that there was a significant difference between the two.

One possible explanation for the poor performance of the Wilcoxon test (in terms of accuracy) is that the assumptions underlying the test just did not hold true in the above experiment. These assumptions are usually illustrated in the traditional context of medical experimentation so a certain degree of extrapolation and inference was used in applying them to the particular experimental design used here. The Wilcoxon rank test is used for *paired* comparisons where the subjects in the pair are highly homogeneous. The pairs here consisted of the set of mean user ranks and the TSA algorithm's ranking. I therefore made the assumption that the procedure used by the TSA matching algorithm in ranking phrases was very similar to that used by humans performing the same task. The validity of this assumption under normal conditions is debatable but it was most likely invalidated by the nature of the experiment. As far as the human subjects were concerned they were simply ranking a set of phrases according to their similarity to a given target phrase. The TSA

matching algorithm, on the other hand, was specifically designed with the information retrieval task in mind. It was designed to rank a set of analytics according to their relevance to a given query. In the evaluation experiment the phrases were such that the target phrase represented an analytic in a large text database and the nine other phrases represented user queries to the information retrieval system. The algorithm then went about computing a relevance score for the analytic against each of the nine queries and then ranking the queries according to the relative relevance of the analytic to each of them. This was still a perfectly valid way to go about testing and evaluating the matching capabilities of the TSA algorithm but it means that the assumption of homogeneity between the procedures for ranking in the humans and computer is considerably stretched. The examples used when talking about the degree of homogeneity [Lehmann 1975] are of pairs of twins or the hands of one person. It is also stated that when a natural pairing does not exist a subgroup can be constructed by "*careful matching of subjects, for example, of patients who are alike with respect to age, sex, and severity of desease, of communities that have the same size, urban-rural character, geographic location, and so forth*". I probably did not achieve such homogeneity in the pairing of sample users and the TSA matching algorithm.

Once the parings have been constructed for the Wilcoxon signed rank test, one subject is assigned at random to be the control and the other is refered to as the "*treated*". Note that it seems normal to have a series of pairings and in each one there is a random assignment of control and treated. In the TSA experiment there was only one pairing of human and computer and the mean user rankings were assumed as the control since that was the base against which I was testing the performance of the matching algorithm. In short, when "translating" the details of the Wilcoxon test from the context of medical experimentation to the experiment at hand, quite a few assumptions were made about the validity of assumptions underlying the test and about the applicability of procedures in the case of testing the rankings. It is possible then, that somewhere in the process of translating from the context of medical experimentation to the present domain, an invalid assumption was made or an invalid procedure was used which rendered the Wilcoxon test inappropriate for the purpose of comparing rankings, and hence the unreliable nature of the results.

On the other hand, it is equally possible that the translation to the TSA domain was correct and the Wilcoxon signed rank test was appropriate and was also executed correctly

but the test was such that it just did not detect the differences between the human and computer rankings to the same degree as the Spearman correlations. The Spearman correlation was always known to be more detailed than the Wilcoxon test (indeed that was why the Spearman correlations were included in the suite of statistical tests to be used). The word **significant** is of the utmost importance in the conclusions from the Wilcoxon test that *there is no **significant** differences between the human and computer rankings for all of the 32 phrase sets tested.* It is possible that at the tested level of significance the human and computer rankings are equivalent and the results and conclusions presented above are completely legitimate.

Whatever the case, whether the Wilcoxon signed rank test was correct or not, it had turned out that the results produced by it were inappropriate for the purpose of fine tuning the TSA matching algorithm. If the Wilcoxon test showed at this very initial stage that the human and computer rankings were equivalent then any improvements that might be gained from changes to the algorithm or scoring mechanism were not going to show up. It was therefore pointless to use the Wilcoxon signed rank test in the subsequent analyses of revised versions of the matching algorithm, but to restrict the testing to the more detailed results provided by the Spearman correlation co-efficients.

8.4.2 Some Comments on the Initial Spearman Correlations.

The Spearman correlation co-efficients tabulated above seem to indicate a level of performance which one would expect from an initial version of the matching algorithm and I have verified by comparison to the actual data that the correlations reflect exactly the relationships between the mean user rankings and the rankings generated by the TSA matching algorithm. I therefore concluded that the Spearman correlations are sufficiently detailed and accurate to monitor the progress of the matching algorithm as I made refinements and improvements to the matching rule-base and the scoring mechanism.

Before proceeding with the refinement and improvement of the matching algorithm, I felt that it would be beneficial to take a closer look at the initial results from the Spearman

correlation calculations to see if I could find explanations for some of the negative correlations and possibly anticipate areas for improvements in performance. For example, we can start by examining the negative correlation value for phrase set 1. The phrase that was ranked highest by the sample of users was "*procedures for communication among groups*". The TSA matching algorithm, however, had placed it in rank position 7. This can be explained by one of the algorithm's inherent limitations, the lack of information on synonymity. In this case the algorithm was unaware that one of the senses of the word "*procedure*" is synonymous with one of the senses of the word "*scheme*". The TSA matching algorithm also failed to match the words "*communication*" and "*communicating*" because they have the respective base forms of *communication* and *communicate*. The only thing that the algorithm could match was the concept of being "*among groups*".

At the other end of the scale, I could attempt to explain why the phrases ranked most highly by the TSA matching algorithm were ranked lowly by the sample users. This is attributable to the difference in the procedure for ranking phrases referred to briefly above. The users were simply ranking phrases according to similarity to a target phrase whereas the TSA matching algorithm was performing an information retrieval task and ranking queries on the relevance of the retrieved analytic. Take for example the phrase "*applications of type hierarchies*". Comparing this directly to the target phrase "*a scheme for communicating among groups that use different type hierarches*" leads one to rank it in a low position, and indeed it had rank position 9 in the mean rankings of the sample users. If, however, one treats the first phrase as a query to an information retrieval system and reasons about how relevant a document with the target phrase as title (or even just containing that phrase) would be to that query then one is led to make a completely different judgement. This is the way in which the TSA algorithm makes it's judgements about phrases. For each of the nine phrases in the set it constructs the scenario that it is a query and the target is an analytic or title. It then computes a relevance score for the match and after doing this for all nine phrases, ranks the results. It is this difference in ranking procedure coupled with the inability to recognise synonymous words that accounts for the differences in rank positions in phrase set 1.

Phrase set 20 has the most negative correlation with a r_s value of -0.566. We can see that the phrase ranked in the highest position by the human subjects was placed in the lowest

position by the TSA matching algorithm. In this case we cannot explain the difference through the algorithm's inability to deal with synonyms or its approach to ranking. It seems that the algorithm was simply confused or inhibited by the complex syntactic structure of the phrase "*the consideration that must be given to encryption and compression in systems which retrieve text*". All of the words in this phrase appear in the same form as in the target phrase (with the exception of *retrieval*) and even looking at the phrase as a query to an information retrieval system containing the target phrase as an analytic, one would expect it to receive a high relevance score. It seems that in this case the algorithm has simply failed to attribute an appropriate ranking to the phrase. One factor that would have reduced the match score for the above phrase is the number of redundant words in the phrase. For example in expressing the simple concept of "*encryption and compression considerations*" the phrase uses 9 words; "*consideration that must be given to encryption and compression*". The reason that this fact has an effect on the overall score is because at normalization time, the normalization factor is based on the number of nodes in the query TSA. If there are a lot of redundant words in the query then there are also a lot of superfluous nodes in the TSA so the denominator in the normalization calculation is higher, resulting in a lower score. It is obvious that the phrases ranked numbers 1 and 2 by the TSA matching algorithm were so ranked because in the scenario of an information retrieval system they are the queries to which the target phrase would be most relevant (although the ranking of "*string text compression systems*" at position 3 by the TSA algorithm is questionable).

On the brighter side of initial TSA matching performance, phrase set 32 had the most positive correlation with an r_s value at 0.787. There is complete agreement at 4 of the rank positions including the last three and there is quite high agreement at the other positions (e.g. 1-3, 3-5, 5-6, 6-4). Looking at the phrases we can see that in this set, all of the nine phrases have relatively simple syntactic structure and all use the same words as in the target phrase. The phrases in the top four rank positions are all extremely close in meaning to the target phrase so one would accept some difference in the ordering of the individual phrases between the sample of users and the TSA matching algorithm. The TSA matching algorithm has performed very well in ranking this relatively simple set of phrases and a correlation of 0.787 is certainly an acceptable performance level.

Through performing this type of failure analysis on each of the 32 phrase sets, and in particular the ones for which the TSA matching algorithm had performed poorly, I was able to draw up a list of sources of possible improvement in performance. These related mostly to a less general set of matching rules in the algorithm. Having computed an initial performance measure for the TSA matching algorithm and identified potential sources of performance improvement, the next task was to enter an iterative loop of implementing the changes to the matching algorithm and repeating the statistical evaluation in order to judge the effect of the changes on the algorithm's performance.

8.5 The Iterative Re-evaluation of the Algorithm.

As stated above, the implementation of a more specific matching rule-set was identified as the main source of possible improvement after the analysis of the results of the initial evaluation experiment. Recall from chapter 6, that one of the rules for matching parent nodes states that a preposition can match any other preposition and can match against modificiaton relationships. This rule in particular was one that I felt needed to be made more specific by limiting it to only certain prepositions, particularly for matching against modification relationships. A further possible source of performance improvement were the rules for residual structure analysis. These rules were also quite general in the initial version of the algorithm and I identified several cases where I felt more specific rules would be beneficial.

One new factor that was introduced to the TSA matching algorithm, over which I had no control, was the release of a new version of the linguistic analysis software. This updated version included a new master lexicon containing more entries, and a new version of the syntactic disambiguation rules. It was very likely that this change in the linguistic software would have some sort of effect on the performance of the matching algorithm. Indeed, since the new release was in effect an improved version, it seemed fair to assume that the inclusion of this new software in the matching algorithm would lead to an improvement in performance due to a more accurate syntactic analysis.

The first iteration of the optimization of the performance of the matching algorithm therefore consisted of the inclusion of the new linguistic analysis software and the implementation of much more specific rules for parent matching and for residual structure analysis. For example, I introduced rules into the residual structure analysis module which stated that the occurance of a main verb plus a head or a preposition plus a head in the residual structure invalidated any syntactic relationship that had been identified during the second phase of matching. Having implemented all of the changes to the algorithm, I then used the new version to provide new rankings for all of the 32 phrase sets. Using these new rankings I recalculated the Spearman correlation coefficients for each phrase set and computed the average correlation. The new average correlation figure was **0.144**, which represented quite a dramatic drop in performance from 0.224, obtained in the first iteration.

In an attempt to explain this unexpected drop in performance, I again set about performing a failure analysis on the rankings produced for certain phrase sets, particularly those which had low correlation coefficients. It seemed that, although the changes to the rule sets had resulted in performance improvements for certain phrase sets (usually those at which the rule changes were specifically aimed), the overall effect was a negative one. In the ensuing analyses and re-evaluations I performed a further five iterations of the cycle, trying to arrive at an optimal rule-set which gave the best overall performance. The average overall correlation coefficients calculated at each iteration of the evaluation experiment are presented in Table 8.4 below.

Having achieved an initial performance level of 0.224, and then having a drop in performance to 0.144, the following five iterations (iterations 3 to 7) were based on changes to the rule-sets both for parent node matching and residual structure analysis. The changes resulted in rule-sets of varying specificity but in all cases the rules used were more specific than those used in the very first version of the matching algorithm. As can be seen from Table 8.4, there was no overall improvement to be found in the performance of the matching algorithm using these more specific rule sets. It seemed that for the best overall performance the very general matching rules, as used initially, were the best.

Table 8.4: Average Correlation Coefficients for each Iteration.

Iteration	Average Correlation	
1	0.224	(initial version)
2	0.144	(new language
3	0.176	software plus
4	0.172	changes to the
5	0.168	matching rule-
6	0.187	base)
7	0.096	
8	0.212	(changes to the
9	0.217	match score
10	0.207	parameters)

Having then concluded that the general matching rules were the best, iteration 8 of the evaluation used the same version of the rule-set as was used in the initial matching algorithm. This evaluation led to an average correlation of 0.212. The reason for the loss of 0.012 can be explained through the use of the new version of the natural language processing software. At this stage I was satisfied that no improvements to this value could be achieved through modifications to the rule-sets used in the algorithm.

The second possible source of improvement in performance was the scoring mechanism. I have already outlined in chapter 6 that there is a set of parameters to this mechanism which represent the scores to be assigned to the varying degrees of inexact matches. One of the objectives of the iterative evaluation of the algorihtm was to try and optimize these values. The iterations of the experiment up to iteration 8 were all based on the initial score parameters which were based on a maximum perfect-match-score of 15. I then performed two furter iterations of the experiment, the first based on a maximum score of 10 and the second based on a maximum of 30. The parameter file which used a maximum perfect node score of 10 provided the best overall correlation of 0.217. The use of different score parameters led to quite small variations in performance so I was satisfied that there was not much further improvement to be gained through further experimentation. The score parameters which resulted in the best overall performance were included earlier in Figure 6.3. Since iteration 9 of the experiment was the based on

the version of the experiment which is now considered to be the "best", I will now present the results from that iteration in more detail.

8.6 The Final Spearman Correlation Coefficients.

A complete listing of the Spearman correlation coefficients for each of the 32 phrase sets based on the comparison between the mean rankings of the sample users and the rankings produced by the *best* version of the TSA matching algorithm is presented in Table 8.4 below. Upon analysis, one can see that there have been improvements in the correlation in some of the phrase sets and a drop in the correlation of some of the other phrase sets. On the whole, however this is the best correlation that has been achieved. This is reflected both in the average correlation figure of **0.217** and in the distribution of the correlation values.

The distribution of the Spearman correlation values for this final evaluation of the TSA matching algorithm is presented in Table 8.6 below. In this final evaluation there are only 7 (21%) of the phrase sets which exhibit a negative correlation. This is one source of improvement over the initial evaluation since the minimization of the number of negative correlation values is an important aspect in the evaluation of the TSA matching algorithm. The number of phrase sets with positive correlations has risen from 22 (68%) to 25 (78%) although the number of correlation values above 0.5 has dropped from 9 (28%) to 7 (21%).

Phrase set number 20 still has the most negative correlation coefficient, and indeed the value has even decreased to -0.629 in this final evaluation, from a value of -0.566 in the initial experiment. I have already described how this negative correlation can be attributed to the nature of the experiment and the difference in ranking procedures used by the sample of users and the TSA matching algorithm, as well as to the fact that the algorithm simply performed badly on ranking some of the phrases. I will provide a more detailed comment on these final experimental results later, but first I will describe a second experiment that was used as a verification of the TSA matching algorithm's performance.

Table 8.5:

The Final Spearman Correlation Co-efficients.

Phrase Set	r	Phrase Set	r
1	-0.104	17	0.116
2	0.716	18	0.100
3	0.662	19	-0.033
4	-0.320	20	-0.629
5	0.704	21	0.358
6	0.500	22	0.166
7	0.600	23	0.066
8	0.479	24	0.166
9	0.408	25	0.179
10	0.304	26	-0.083
11	0.470	27	0.279
12	0.050	28	0.020
13	0.091	29	0.683
14	-0.233	30	-0.400
15	0.408	31	0.079
16	0.604	32	0.529

Table 8.6:

Distribution of Final Spearman Correlations.

Range	Number	Range	Number
above 0.5	7	below 0	3
0.4 to 0.5	5	-.2 to -.3	1
0.3 to 0.4	2	-.3 to -.4	1
0.2 to 0.3	1	-.4 to -.5	1
above 0.0	10	below -.5	1
Total > 0	25	Total < 0	7

I have mentioned in the introduction to this chapter the possibility that after using one experimental dataset for the tunning of the algorithm and also for the evaluation of the final matching algorithm, it was possible that the final algorithm would in fact be specifically tuned for the particular dataset that was used and that it might not therefore function as effectively on a different dataset. In order to test this hypothesis, and indeed hopefully to prove it to be false, I performed a second experimental evaluation of the TSA matching algorithm based on a completely different dataset from a different domain.

8.7 A Second Experimental Evaluation.

The basic design of the second experiment was identical to that of the first experiment. It involved the construction of a test dataset consisting of sets of phrsaes made up of a target phrase and nine semantically similar phrases. These phrase sets were then ranked by a sample of information retrieval system users and also by the TSA matching algorithm. The Spearman correlation test was then used to compute a correlation measure for each of the phrase sets and an overall average correlation measure, as described earlier.

In order to ensure that the dataset for the experiment was independent of the TSA matching algorithm and to ensure that the generated phrases were not biased toward the way in which the matching algorithm ranked phrases, the dataset for this experiment was constructed wholly by my research supervisor. The dataset consisted of 24 phrase sets built around the titles of articles from computing journals such as "*IEEE Computer*", "*IEEE Transactions on Data and Knowledge Engineering*" and "*Communications of the ACM*". The articles chosen were related to database systems, artificial intelligence and expert systems. As before, each phrase set consisted of the title of one journal article and nine phrases generated around that title. The phrases were again generated so that there were often only very subtle differences between them and they were generally quite difficult to rank. The 24 phrase sets of this second dataset are included in APPENDIX E.

Since the dataset related again to topics of computing, the sample of users involved postgraduate students of Computer Applications at Dublin City University. The sample

consisted of 18 students who ranked varying numbers of phrase sets so that there was finally a collection of 10 user rankings for each phrase set (240 user rankings). It may be worth noting that many of the users commented that they found it much more difficult to perform the rankings on this second dataset than for the first dataset because of the nature of the phrases.

The data provided by the sample of users was then processed in the same way as in the first experiment to produce a midranked mean ranking for each phrase set. The TSA matching algorithm was also used to produce a ranking of phrases in each set. The first application of the matching algorithm to the second dataset was done when the matching rule-base had been stabilised. This corresponded to iteration 8 of the first experiment in which the score parameters were based on a maximum node score of 15. The matching algorithm was also applied to the second dataset during iterations 9 and 10 of the first experiment, with score parameters based on maximum scores of 10 and 30 respectively. At each of these three iterations the Spearman correlation coefficients were calculated for each of the 24 phrase sets and an overall correlation value was computed. These average Spearman correlation values are presented in Table 8.7 below.

Table 8.7: Average Correlation Values In Experiment 2.

Iteration	Correlation Dataset 2	Correlation Dataset 1
2.1	0.400	(0.212)
2.2	0.402	(0.217)
2.3	0.392	(0.207)

The second iteration of the application of the TSA matching algorithm to the second dataset produced the best result. It was this version of the matching algorithm and score parameters that also produced the best correlation results for the first dataset. This version has therefore been designated as the *final* or *best* version of the TSA matching algorithm.

The average correlation coefficients for this second dataset are much better than those for the first dataset. A more detailed analysis of the Spearman correlation coefficients for the

individual phrase sets also demonstrates how much better the performance has been for this dataset. The individual correlations are presented in Table 8.8 and the distribution of the coefficients is illustrated in Table 8.9.

Table 8.8:

The Spearman Correlation Co-efficients: Dataset 2.

Phrase Set	r	Phrase Set	r
1	0.633	13	0.433
2	0.800	14	0.679
3	0.112	15	0.345
4	0.616	16	0.379
5	0.375	17	-0.054
6	0.287	18	0.395
7	0.395	19	0.433
8	0.700	20	0.200
9	0.387	21	0.312
10	0.595	22	0.216
11	0.266	23	-0.145
12	0.479	24	0.800

One can see from the individual correlation coefficients presented in Table 8.8 that there are quite a few high values and there are only two negative correlations. The distribution figures in Table 8.9 bear this through. There are 22 (92%) positive correlation coefficients with 7 of those (29%) above 0.5 and 17 (71%) above 0.3.

These results indicate a much better level of performance from the TSA matching algorithm on this second dataset. The hope for the evaluation was to have an average correlation as close to 1 as possible, but I was initially unsure about how high a correlation I could expect to achieve. This value of 0.402 is, I believe, sufficiently high so as to demonstrate the usefulness of the TSA matching algorithm.

Table 8.9:

Distribution of Spearman Correlations: Dataset 2

Range	Number	Range	Number
above 0.5	7	below 0	2
0.4 to 0.5	3	-.2 to -.3	0
0.3 to 0.4	7	-.3 to -.4	0
0.2 to 0.3	3	-.4 to -.5	0
above 0.0	2	below -.5	0
Total > 0	22	Total < 0	2

The overall usefulness or power of the TSA matching algorithm can then be judged from an analysis of the results of the evaluation experiment on both of the datasets. In particular, it is worth explaining the reason for the difference in performance between the first and second datasets.

8.8 Analysis of the Final Results.

The final results of the evaluation of the TSA matching algorithm are based on the averages of individual Spearman correlation measures for each of 32 phrase sets and 24 phrase sets in the first and second datasets respectively. The average correlation coefficient over 32 phrase sets in the first dataset was **0.217** and the average correlation over 24 phrase sets in the second dataset was **0.402**. These correlation coefficients represent values in the range -1 to 1, where a coefficient of 1 is the optimal value.

The evaluation of the TSA matching algorithm on the second dataset has, I believe, proven the power of the algorithm. The value of 0.402 is quite high, especially when put in perspective by comparison to the result of 0 that would be obtained if rankings were assigned randomly. It should also be noted here, in putting the evaluation in perspective,

that the results produced by this experimental evaluation represent an **underestimate** of the TSA matching algorithm's effectiveness. This is because of the design of the experiment. I have referred earlier to the fact that, in these experiments, there was a difference in the acutal tasks being performed by the sample of users and by the TSA matching algorithm. The sample users were simply ranking phrases according to their similarity to a base phrase whereas the TSA matching algorithm was ranking phrases by treating them as queries to an information retrieval system and estimating the relevance of a target phrase to those queries. This difference in ranking strategy has been described earlier in Sections 4.1 and 4.2, and some examples have been given there. This "mismatch" in the experimental design was due mainly to a lack of resources. The design of the present experiment meant that it was only necessary for the sample users to be able to reason about the semantic similarity of a set of phrases. The only requirement for the sample then was that the users have sufficient knowledge of the domain of the dataset to be able to make these judgments. If the experiment had involved making relevance judgments about retrieval queries and a large database of document titles, then it would have either required a sample of users with not only knowledge of the domain of the dataset but also with a much more detailed knowledge of information retrieval systems, or it would have placed a much greater cognitive load on the sample of users that was used in the experiment described here.

What is important to understand at this stage is that the experimental evaluation of the TSA matching algorithm outlined here was perfectly valid because the TSA algorithm is presently a basic phrase matching algorithm, but it represents the lower bound of the algorithm's performance rather than the actual level of performance that could be expected in the natural information retrieval environment. The reason that I can state that it is the lower bound of effectiveness that is demonstrated by the experiment rather than an upper bound is that in some of the cases where the TSA matching algorithm has ranked phrases differently to the sample of users, due to the difference in the ranking strategy, this has resulted in a loss of correlation but in fact in an information retrieval environment, the TSA algorithm's ranking would be considered more apporpriate and would therefore have a higher correlation with the **actual** desired ranking in that case. I can therefore conclude that **the average correlation between the rankings of a sample of information retrieval system users and the rankings of the TSA matching algorithm for a dataset of 24 phrase sets is 0.402 and this represents the lower bound of the algorithm's effectiveness at**

ranking those phrases.

Similarly for the first dataset, the average correlation between the rankings of a sample of information retrieval system users and the rankings of the TSA matching algorithm is 0.217 and this represents the lower bound of the algorithm's effectiveness at ranking those phrases. This average correlation of 0.217 is lower than that achieved for the second dataset. Some attempt should therefore be made to explain this difference and to speculate as to which of these levels of effectiveness is more likely in reality.

I stated earlier that in the construction of the dataset for the second evaluation experiment, the phrase sets were all devised by my supervisor in order to avoid any bias that would mean that the phrase sets were actually suited to the way in which the TSA matching algorithm ranked phrases. The construction of the first dataset however, was divided approximately equally between myself and my supervisor. He constructed the first 16 phrase sets and I constructed the phrases in the second 16 phrase sets. I believe that the fact that I was invovled in the construction of the dataset did introduce some bias into the experiment, but it was in fact bias of a negative nature. In the construction of the phrase sets, I wanted to test to the full the ability of the matching algorithm to detect and infer subtle differences in meaning from syntactic level analysis. Being fully aware of the intrinsic details of the TSA matching algorithm, I constructed the phrase sets in such a way as to make this inference process as difficult as possible for the matching algorithm. Some of the phrase sets were even specifically designed to highlight some of the weaknesses of the TSA matching algorithm. This was in contrast to the first 16 phrases sets of the dataset, which were constructed without detailed knowledge of the operation of the matching algorithm and were designed according to the objective of producing a set of semantically similar phrases which would be useful for comparing the ranking capabilities of a computer algorithm to those of an average information retrieval system user.

This difference in the approach to constructing the dataset has shown itself quite clearly in the experimental results and I believe it provides the explanation for the relatively poor performance of the TSA matching algorithm on the first dataset. This can be best seen by analysing the results of the Spearman correlation test for each half of the first dataset. These results were presented earlier in Table 8.5. Taking these correlation coefficients it

is possible to construct a distribution table for each half of the dataset. This is illustrated in Table 8.10.

Table 8.10: Distribution of correlation over both halves of dataset 1

Distribution of Spearman Correlations.

1st Half (His)		2nd Half (Mine)	
Range	**Number**	**Range**	**Number**
above 0.5	5	above 0.5	2
0.4 to 0.5	5	0.4 to 0.5	0
0.3 to 0.4	1	0.3 to 0.4	1
0.2 to 0.3	0	0.2 to 0.3	1
above 0	2	above 0	8
below 0	3	below 0	4
Total	16	Total	16
Avg Corr.	0.333	Avg. Corr.	0.099

On analyis, this data shows the difference in performance of the algorithm over each half of the dataset. The first 16 phrase sets (illustrated on the left hand side of Table 8.10), which was constructed independently of the matching algorithm, has 11 (69%) correlation coefficients above 0.3, whereas in the second 16 phrase sets only 3 (19%) have a correlation above 0.3. If average correlation coefficients are computed for each half of the dataset then the first 16 phrase sets show an average correlation of 0.333 and the second 16 sets have an average correlation coefficient of 0.099. This demonstrates that there is clearly a difference in performance over each half of the dataset and indicates that the latter 16 phrase sets certainly did prove difficult for the TSA matching algorithm, just as they were designed to do. These figures and insights into the construction of the dataset for the first evaluation experiment also provide clear explanation for the somewhat poorer result of 0.217 for this first experiment.

This explanation also provided a good indication of the level of performance that can be

expected from the TSA matching algorithm. Given that the first 16 phrase sets in the first experiment and the whole dataset of the second experiment were constructed in isolation of the techniques used in the TSA matching algorithm, they are therefore a good reflection of the types of data that would usually be subject to ranking by the TSA matching algorithm. It is fair then to use the results from this data in the final evaluation of the algorithm. Using the average correlation coefficients of 0.333 form the first half of the first dataset and 0.402 from the second dataset, we can conclude that **under normal circumstances, the TSA matching algorithm can be expected to produce phrase rankings with an average correlation to the rankings that would be assigned by the users of information retrieval systems of at least 0.3 or 0.4.**

I have already presented one reason for qualifying the experimental results achieved here as lower bounds of the TSA matching algorithm's effectiveness. One must also remember however, that even the phrase sets constructed in isolation of the matching algorithm, although not designed to try to "trick" the matching algorithm in any way, were designed to be "difficult" to rank. The phrase sets were constructed by building phrases using the same words as the target phrase but in different syntactic relationships and by building some phrases with identical semantic content expressed in a completely different way. These phrase sets were not just difficult to rank for the TSA algorithm but also for the sample users, but it did mean that there was less likelihood of having a high correlation between the two sets of rankings. In a normal information retrieval environment there would be a much lower percentage of such semantically similar phrases. There would of course be many occurances of similar concepts expressed in different ways but these would be natural expressions rather than ones artificially created to be deliberately complex. Indeed the TSA matching algorithm is specifically designed to handle such natural re-phrasing of concepts in different ways. I belive this less complex, natural environment would also contribute to the more effective performance of the TSA matching algorithm.

I have referred on one or two occas ions previously to some of the inherrent limitations of the TSA matching algorithm. In particular, I mentioned how I had designed the latter half of the first experimental dataset to highlight these limitations. It is worth explaning these limitations here because they can be quite easily overcome and in a natural information retrieval environment will probably not arise.

The main limitation of the TSA matching algorithm in its present form is its inability to match synonyms. Matching takes place only on text forms, base forms and syntactic labels. No facilities are presently available which would make synonymy information available to the algorithm. In an overall information retrieval environment this information could be supplied by either of two sources. If the information retrieval system maintaned a classification hierarchy relating to the texts in the database, then this would be one source of synonymy information. In such a classification hierarcy, synonymous words would be linked in some way as they would each apply to the same subject area. A second possible source of such information on synonyms is of course a standard thesaurus which is specifically designed to provide such information. A thesaurus would therefore probably provide a more comprehensive and complete description of synonyms than would be provided by a classification system. Whatever the source, it would be quite an easy task to modify the matching algorithm so that it could also take account of synonyms. It would simply involve the addition of a fourth level to the node (word) matching procedure. If no match had occured on either the text form or the base form of the word then the next step would be to check if the current word in the analytic TSA was a described as a synonym to the current word in the query TSA. If so then an inexact match would occur and the algorithm could then proceed as usual.

A second limitation of the TSA matching algorihtm that I outlined earlier, was that it performed poorly or seemed to get confused when matching long complex phrases. The example under discussion at the time was the matching of "*Compression and encyrption considerations of string text retrieval systems*" and "*The consideration that must be given to encryption and compression in systems which retrieve text*". It is unlikely however, that such long and complex phrases would occur in a natural information retrieval environment, especially the latter phrase above. If such a complex phrase did occur in a text, it is most likely that it would be divided into more than one analytic during the indexing stage, before it was to be represented in TSA form. If however, a phrase such as the first one above occurred as an analytic and a user of the system entered a search query of the very complex nature of the second phrase, then TSA matching algorithm's poor performance at matching these would remain as a limitation of the algorithm.

8.9 Summary.

I have described in some length in this chapter the evaluation of the TSA matching algorithm for matching text representatives in an information retrieval environment. I have presented in detail the results of an evaluation experiment which was specifically designed to test the particular powers of the matching algorithm. Because of the difficult nature of the tasks involved in the experiment, and for several other reasons outlined earlier, I have concluded that the actual results obtained through the experimental evaluation of the TSA matching algorithm represent an under-estimate of the algorithm's likely effectiveness in a natural information retrieval environment. Given these experimental conditions, it is possilbe to draw two main conclusions from the experimental evaluation results presented above.

The TSA matching algorithm does indeed have the power to detect and infer subtle differences in phrase meanings and so rank those phrases accordingly. This is one of the main advantages of the TSA matching approach over former approaches to information retrieval. The desire to prove and demonstrate this advantage provided the reason for testing this aspect of the matching algorithm explicitly. In the experiments described above, the TSA matching algorithm achieved average correlations of 0.217 and 0.402 in ranking a total of 56 sets of phrases, where each set of phrases comprised of nine semantically similar phrases and therefore represented a very difficult ranking task. I believe that it is very fair to assume that if the TSA matching can perform so well in ranking very difficult and similar phrases that it will perform at least as well when ranking simpler phrases. For example, if the TSA matching is capable of correctly ranking the TSA for the analytic "*The relationship between compression, information theory and grammars*" higher than the TSA for "*A theory of information theory and grammars*" in relation to the target phrase "*A unified approach to comression, information theory, and grammars*" then it will not have a problem with ranking either of these two TSAs higher than the TSA for "*A theory of unified grammars*" in relation to the same base phrase.

The TSA matching algorithm can be expected to produce phrase rankings with an average correlation to the rankings that would be assigned by the users of information retrieval systems of <u>at least</u> 0.3 or 0.4. I think that this is a very acceptable level of performance

for the TSA matching algorithm in its present state and I would even forecast that in an information retrieval environment or even in an experiment in which the phrase sets were not deliberately constructed to be difficult to rank, an average correlation of nearer to 0.6 or 0.7 could be achieved by the the TSA matching algorithm. I am also quite confident that, at such a level of matching effectiveness, the TSA matching algorithm would provide an overall improvement in the retrieval effectiveness of an information retrieval system. The realization of such an improvement however, can only come after the TSA matching results have been combined in some way to provide an overall ranking for retrieved texts. This is a topic for further research and will discussed a little more in the final chapter. At the level of phrase or analytic matching however, I believe that I have shown the TSA matching approach to be very effective, and on a wider scale, I believe that I have demonstrated the usefulness of using natural language processing techniques effectively for some information retrieval tasks.

Chapter 9.

Conclusions and Directions for Future Research

9.1 Conclusions.

In this thesis, I have presented a method for using natural language processing techniques in performing information retrieval tasks. I have described a structured text representation which can be used for the representation of content-bearing text segments, or analytics, in an information retrieval environment. I have also described how these Tree Structured Analytics can be effectively used for retrieval through matching against a structured representation of the user's retrieval query and I have presented a matching algorithm which has been designed for this purpose. This matching algorithm has then been evaluated using standard statistical methods to compare its performance against a sample of information retrieval system users performing the same task. The results of this evaluation have been presented and analysed in some detail. The main conclusions that can be drawn from all of this work are discussed below.

The main thesis underlying this work was that "**the best improvements in retrieval performance are not to be found by discovering the appropriate aspects of syntactic description to utilize, but rather in discovering a way of utilizing all aspects of syntactic description**". I have proposed here a structured representation of text based on a very rich morpho-syntactic description of language which utilizes all aspects of the syntactic description. I claim that this text representation is rich enough to encompass all aspects of syntactic description and to make all of this information available to a matching algorithm at retrieval time, flexible enough to encode implictily all lexical and syntactic ambiguity remaining after the linguistic analysis and disambiguation processes, and powerful enough to provide more effective text retrieval than straight string matching of normalized phrase indexes. I have substantiated this claim by implementing a prototype system capable of building these text representations and by designing and implementing a matching algorithm for matching these structured representations and providing rankings for the retrieved text sequences, based on estimated relevance to the retrieval query.

In evaluation experiments which compared sets of phrase rankings provided by the TSA matching algorithm to the same sets of phrase rankings provided by a sample of information retrieval users, I computed average correlation figures of 0.333 (over 16 phrase sets) and

0.402 (over 24 phrase sets). The overall average correlation between the TSA matching algorithm rankings and the sample users rankings over 40 phrase sets was therefore 0.374. I can conclude from this that **the TSA matching algorithm can be expected to produce phrase rankings with an average correlation to the rankings that would be assigned by the users of information retrieval systems of at least 0.374.** This is an absolute minimum figure for the average correlation and I believe that in reality the actual correlation would be much higher than that. Even at the current level of effectiveness however, I believe that this correlation is high enough to demonstrate that the use of syntactically based structures as text representatives in an information retrieval system provides effective text retrieval.

The datasets in the experimental evaluation of the TSA matching algorithm were specifically designed to test one of the claimed advantages of the approach. I think that the performance of the matching algorithm in these experiments was good enough to justify the conclusion that **because of the richness and flexibility of the tree structured representation of text, the TSA matching algorithm can detect and infer a certain amount of semantic information from the syntactic description of the text.** What is important is that the amount of semantic information that can be gathered from the syntactic description is sufficient to allow the TSA matching algorithm to make fine distinctions between the meanings of different phrases and so rank those phrases accordingly. This ability to infer semantic information without actually performing any semantic level language processing, and to use this information in the matching process to provide rankings of retrieved analytics, represents a major advantage of the TSA approach over previous approaches to the task of matching queries to analytics in information retrieval systems.

Based on the effectivenes and power of the TSA matching algorithm, I would also conclude that the use of syntactic level natural language processing in information retrieval systems does provide an avenue for improvements in retrieval effectiveness. Unlike the statistical approaches to information retrieval, all possibilities for the application of syntactic level natural language processing techniques to information retrieval have not been exhausted. The use of structured text representatives based on syntactic description is just one of these possiblities and the work reported here has, I believe, indicated that this is a very useful approach to take. I think that, building on this approach and using other approaches to

incorporating syntactic level natural language processing in the information retrieval scenario, we will see the realization of many improvements in retrieval effectiveness over the coming decades.

9.2 Directions for Future Research

The main limitation of the current version of the TSA matching algorithm, which I discussed in section 8.7, was its inability to match synonyms. I also described there how this limitation could be addressed through the inclusion of either a classification hierarchy or a standard thesaurus. This expansion of the matching algorithm to include a facility for matching synonyms is one direction for future research that should not prove difficult.

The TSA matching algorithm, as described here, provides a mechanism for matching a retrieval query against individual text representatives in a database. The result of this matching process in an information retrieval environment would be a ranked list of text representatives or analytics taken from various texts. It is likely that for any given text there will be several retrieved analytics. In order to complete the information retrieval task, some method of combining these individually retrieved analytics for the retrieval of complete texts is required. Such a method for retrieving and ranking texts based on ranked analytics would have to take account of both the number of analytics retrieved for a given text and the match scores for those analytics. This combination of ranked analytics for text retrieval is the subject of a separate research initiative here at Dublin City University, which is also looking at methods of combining the retrieval results of different retrieval strategies to provide an overall best set of retrieved texts.

One undesirable aspect of the TSA approach to information retrieval which was mentioned earlier was the complexity of the matching algorithm. Due to the nature of the problem such complexity was unavoidable but now that a basic solution to the problem has been found this complexity can be addressed through clever and efficient implementations of the algorithm. At present, the approach to TSA matching is to perform a match for the query TSA against every text TSA in the database. This is obviously wasteful since the majority

of text analytics will always be irrelevant to a given query. A simple way to reduce the amount of matching required is to provide a term index based on the words that occur in the text analytics (as illustrated in Figure 5.10). This would provide a filtering mechanism so that only those text analytics which had words (or synomyms) in common with the query would have to be matched. Since the matching algorithm used here was only a prototype version developed in LISP, some of its complexity can be addressed during a re-implementation in a more efficient language such as C or C++. It might also be possible to address some of the complexity involved with the tree structured representation by, for example, designing a text signature similar to those currently used in information retrieval but which would also include the coding of the extra linguistic and structural aspects of the representation. The investigation of methods for addressing the complexity of the problem of matching sturctured text representatives for information retrieval provides many possiblities for future research.

Given the need for a method of applying the TSA matching approach to the retrieval of complete texts and the desire to address the complexity of the TSA matching algorithm, it may be possible to research ways of doing both of these things at once. Given that the representation of individual text representatives is based on a tree structure then it may be possible to combine these individual tree structured analytics into one single graphical representation of the complete text. The inspiration for this line of thinking came from the universal graph structure described in section 6.2.1 [Levinson 1984]. If the text representative were to take this form then the matching algorithm would have to be based on some form of graph searching. Some recent work on graph or network searching for information retrieval has been done by Turtle [Turtle & Croft 1990] using inference networks and by Kwok [Kwok 1989] using neural netorks. The use of neural networks for information retrieval has also been investigated by [Belew 1989] and [Hingston & Wilkinson 1990] and for other relevant applications such as representing context and performing word sense disambiguation by [Gallant 1990]. It seems that the searching mechanism of neural networks would be particularly suited to searching a large graph-structured text representative composed of individual TSA structures. I think that this application of natural language processing to the information retrieval task would provide a particularly interesting direction for future research.

Given that I have shown the TSA matching approach to be an effective method of applying morpho-syntactic language processing techniques to information retrieval tasks, these possibilities for future research, any or all of which may lead to improvements in retrieval effectiveness, lend further support to my final conclusion that the application of morpho-syntactic language processing to information retrieval will result in many improvements in the retrieval effectiveness of future information retrieval systems.

Bibliography

[Belew 1989]
"Adaptive Information Retrieval: Using a connectionist representation to retrieve and learn about documents", R. Belew, *Proceedings of the 12th International ACM SIGIR Conference*, Cambridge Massachussetts, 1989, pp11-20.

[Bonzi & Liddy 1989]
"The Use of Anaphoric Resolution for Document Descriptions in Information Retrieval", S. Bonzi and E. Liddy, *Information Processing and Management*, 25(4), 1989, pp429-441. (also in *Proceedings of the 11th ACM SIGIR Conference*, Grenoble, 1988, pp53-66.)

[Brajnik *et al.* 1988]
"IR-NLI II: Applying Man-Machine Interaction and AI Concepts to Information Retrieval", G. Brajnik et al., *Proceedings of 11th ACM SIGIR Conference*, Grenoble, 1988, pp387-399.

[Collins 1987]
"Collins Cobuild English Language Dictionary", Collins: London and Glasgow, 1987.

[Dillon & Gray 1983]
"FASIT: A Fully Automatic, Syntactically Based Indexing System", M. Dillon and A.S. Gray, *Journal of the ASIS*, 34(2), 1983, pp99-108.

[Ellegaard 198?]
"The Syntactic Structure of English Texts: A Computer-Based Study of Four Kinds of Text in the Brown Corpus", A. Ellegaard, *ACTA Uinversitatis Gothoburgensis, ISBN 91-7346-051-6.*

[Fagan 1987]
"Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods", J.L. Fagan, *Technical Report* 87-868, (Ph.D. Thesis), *Department of Computer Science, Cornell University*, September 1987.

[Gallant 1990]
"A Practical Approach for Representing Context and for Performing Word Sense Disambiguation using Neural Networks", S.I. Gallant, *Technical Report NU-CCS-90-5, College of Computer Science, Northeastern University*, 1990.

[Gay & Croft 1990]
"Interpreting Nominal Compounds for Informaton Retrieval", L.S. Gay and W.B. Croft, *Information Processing and Management*, 26(1), January 1990, pp21-38.

[Hafner 1989]
"A Robust Method for Context Analysis of Natural Language Text", C.D. Hafner, *Technical Report NU-CCS-89-25, College of Computer Sciece, Northeasern University, Boston*, 1989.

[Haralick and Elliot 1979]
"Increasing Tree Search Efficiency for Constraint Satisfaction Problems", R.M. Haralick and G.L. Elliot, *Proceedings of the 6th Joint Conference on Artificial Intelligence*, 1979.

[Heikkil 1990]
"Morphological Features for English", J. Heikkil, *SIMPR Document, SIMPR-RUCL-1990-13.2e*, 1990.

[Hingston & Wilkinson 1990]
"Document Retrieval Using a Neural Network", P. Hingston and R Wilkinson, *Technical Report #29, RMIT and the University of Melbourne*, 1990.

[Huang 1984]
"Dealing with Conjunctions in a Machine Translation Environment", X. Huang, *Proceedings of the 11th International Conference on Computational Linguistics: COLING-84*, 1984, pp243-246.

[Karlsson 1990]
"Parsing and Constraint Grammar", F. Karlsson, *Unpublished paper. Research Unit for Computational Linguistics, University of Helsinki*, 1990.

[Karlsson 1991]
"Natural Language Processing for Information Retrieval Purposes", F. Karlsson, A. Voutilainen, J. Heikkila and A. Anttila, *SIMPR Document SIMPR-RUCL-1991-13.4e* (220 pages), 1991.

[Koskenniemi 1983]
"Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production", K. Koskenniemi, *Publications of the Dept. of General Linguistics 11, University of Helsinki*, 1983.

[Kwok 1989]
"A Neural Network for Probabilistic Information Retrieval", K.L. Kwok, *Proceedings of the 12th International ACM SIGIR Conference*, Cambridge Massachussetts, 1989, pp21-30.

[Lehmann 1975]
"Nonparametrics: Statistical Methods based on Ranks", E.L. Lehmann, Mc Graw Hill, *ISBN 0-07-037073-7*, 1975.

[Lenat *et al.* 1990]
"CYC: Toward Programs with Common Sense", Douglas B. Lenat *et al.*, *Communications of the ACM*, 33(8), August 1990, pp30-49.

[Levinson 1984]
"A Self-Orgainising Retrieval System for Graphs", R. Levinson, *Proceedings of AAAI-84*, Also *MSc Thesis - University of Texas at Austin, Tech report AI-85-05*, 1984.

[Lewis *et al.* 1988]
"An Algorithm for Text Content Matching", D. Lewis, N. Bhandaru and W.B. Croft, *University of Massachusetts, Private Communication*, 1988.

[Lewis & Croft 1989]
"Language Processing in Information Retrieval", D.D. Lewis and W.B. Croft, *University of Massachusetts, Private Communication*, 1989.

[Lewis & Croft 1990]
"Term Clustering of Syntactic Phrases", D.D. Lewis and W.B. Croft, *Proceedings of the 13th ACM SIGIR Conference*, Brussels, 1990, pp385-404.

[Lu 1990]
"Document Retrieval: A Structural Approach", Xin Lu, *Information Processing and Management*, 26(2), 1990, pp209-218.

[McCall & Willett 1986]
"Criteria for the Selection of Search Strategies in Best-match Document Retrieval Systems", *International Journal on Man-Machine Studies*, 25, 1986, pp317-326.

[Metzler & Haas 1989]
"The Constituent Object Parser: Syntactic Structure Matching for Information Retrieval", D.P. Metzler and S.W. Hass, *Proceedings of the 12th ACM SIGIR Conference*, Cambridge Massachussetts, 1989, pp117-126.

[Metzler *et al.* 1990]
"Conjunctions, Ellipses and Other Discontinuous Constituents in the Constituent Object Parser", D. Metzler *et al., Information Processing and Management*, 26(1), January 1990, pp53-72.

[Quirk *et al.* 1985]
"A Comprehensive Grammar of the English Language", R. Quirk, Greenbaum, Leech and Svartvik, *Longman: London and New York*, 1983.

[Sacks-Davis *et al.* 1990]
"Using Syntactic Analysis in a Document Retrieval System that uses Signature Files", R. Sacks-Davis, P. Wallis and R. Wilkinson, *Proceedings of the 13th ACM SIGIR Conference*, Brussels, 1990, pp179-192.

[Salton 1968]
"Automatic Information Organisation and Retrieval", G. Salton, McGraw-Hill, 1968.

[Salton & McGill 1983]
"Introduction to Modern Information Retrieval", G. Salton and M. McGill, McGraw-Hill, 1983.

[Salton *et al.* 1989]
"On the Application of Syntactic Methodologies in Automatic Text Analysis", G. Salton, C. Buckley and M. Smith, *Proceedings of the 12th ACM SIGIR Conference*, Cambridge Massachussetts, 1989, pp137-150.

[Schwarz 1989]
"Content Based Text Handling", C. Schwarz, *Information Processing and Management*, 26(2), 1990, pp219-226.

[Shapiro & Haralick 1979]
"The Consistent Labeling Problem: Part 1", L. Shapiro and R. Haralick, *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1(4)*, April 1979, pp173-184.

[Shapiro & Haralick 1981]
"Structural Descriptions and Inexact Matching", L. Shapiro and R. Haralick, *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-3(5)*, September 1981, pp504-519.

[Smeaton & van Rijsbergen 1988]
"Experiments on Including Syntactic Processing of User Queries into a Document Retrieval System", Alan F. Smeaton and C.J. van Rijsbergen, *Proceedings of the 11th ACM SIGIR Conference*, Grenoble, 1988, pp31-51.

[Smeaton 1989]
"Information Retrieval and Natural Language Processing", Alan F. Smeaton, *Keynote Address presented at INFORMATICS 10*, University of York, April 1989.

[Smeaton 1990]
"SIMPR: Using Natural Language Processing Techniques for Information Retrieval", Alan F. Smeaton, *Presented at the 1990 British Computer Society Research Colloqium on Research and Development in Information Retrieval*, 1990.

[Spark Jones & Tait 1984]
"Automatic Search Term Variant Generation", Karen Spark Jones and J.I. Tait, *Journal of Documentation*, 40, 1984, pp50-66.

[Sumita and Tsutsumi 1988]
"A Translation Aid System Using Flexible Text Retrieval Based on Syntax-Matching", E. Sumita and Y. Tsutsumi, *TRL Research Report TR97-1019,* 1988.

[Tong *et al.* 1987]
"Conceptual Information Retrieval Using RUBRIC", R.M. Tong *et al*, *Proceedings of 10th ACM SIGIR Conference*, New Orleans, 1987.

[Tong *et al.* 1989]
"A Knowledge Representation for Conceptual Information Retrieval", R.M. Tong *et al.*, *International Journal of Intelligent Systems,* 4(3), 1989, pp259-283.

[Turtle & Croft 1990]
"Inference Networks for Document Retrieval", H. Turlte and W.B. Croft, *Proceedings of the 13th ACM SIGIR Conference*, Brussels, 1990, pp1-24.

[Wood & Sommerville 1988]
"An Information Retrieval System for Software Components", M. Wood and I. Sommerville, *ACM SIGIR Forum*, 22(3/4), 1988, pp11-28.

APPENDIX A.

Output from Linguistic Processing

Output from Morphological Analysis.

```
s-startdoc
 s-startdoc " S-LIM @GML "

s-startdochead
 s-startdochead " S-LIM @GML "

s-starttitle
 s-starttitle " S-LIM @GML "

the
 the " <Def> DET CENTRAL ART SG/PL @DN> "

application
 application " N NOM SG "

of
 of " PREP "

syntactic
 syntactic " <DER:ic> A ABS "

level
 level " := <SVO> <SV> <P/with> V SUBJUNCTIVE VFIN @+FMAINV "
 level " := <SVO> <SV> <P/with> V IMP VFIN @+FMAINV "
 level " := <SVO> <SV> <P/with> V INF "
 level " := <SVO> <SV> <P/with> V PRES -SG3 VFIN @+FMAINV "
 level " N NOM SG "
 level " A ABS "

natural
 natural " A ABS "

language
 language " N NOM SG "

processing
 process " :+ion <SVO> <SV> PCP1 "
```

```
techniques
 technique " N NOM PL "

to
 to " PREP "
 to " INFMARK> @INFMARK> "

effective
 effective " <DER:ive> A ABS "

information
 information " <-Indef> N NOM SG "

retrieval
 retrieval " <-Indef> N NOM SG "

$.

s-endtitle
 s-endtitle " S-LIM @GML "

s-enddochead
 s-enddochead " S-LIM @GML "

s-enddoc
 s-enddoc " S-LIM @GML "
```

Output after Disambiguation and Syntactic Analysis.

```
the
  the " <Def> DET CENTRAL ART SG/PL @DN> "

application
  application " N NOM SG " @SUBJ

of
  of " PREP " @<NOM-OF

syntactic
  syntactic " <DER:ic> A ABS " @AN>

level
  level " A ABS " @AN>

natural
  natural " A ABS " @AN>

language
  language " N NOM SG " @NN> @<P

processing
  process " :+ion <SVO> <SV> PCP1 " @NN> @<P @<NOM-FMAINV(n)
@-FMAINV(n) @AN>

techniques
  technique " N NOM PL " @OBJ @<P

to
  to " PREP " @<NOM @ADVL

effective
  effective " <DER:ive> A ABS " @AN>

information
  information " <-Indef> N NOM SG " @NN> @<P

retrieval
  retrieval " <-Indef> N NOM SG " @<P
```

APPENDIX B.

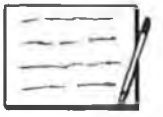
The Experimental Design

Stage 1.

Knowledge based search tactics .....

A user interface to relational databases .....

Tools for communicating in a hypertext .....



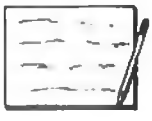
Generation of 9 close variants

32 TOIS titles ⟶ **32 Sets of 9 phrases**

Stage 2.

Knowledge based search tactics...

A user interface to relatioal databases...



Tools for communicating in a hypertext...

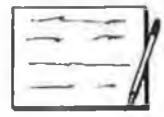
Sample users performing rankings of phrases according to similarity to title

32 Sets of 9 phrases ⟶ **10 Human rankings for each of 32 phrase sets**

Stage 3.

Knowledge based search tactics...

A user interface to relatioal databases...

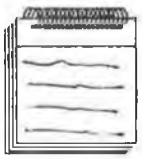
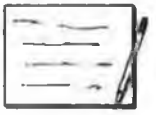


Tools for communicating in a hypertext...

TSA matching algorithm performing ranking of each set of phrases

32 Sets of 9 Phrases ⟶ **A Computer Ranking for each of 32 phrase sets**

Stage 4.



r_s



r_s



r_s

.
.
.

.
.
.

.
.
.

Mean Human Ranking + **Computer Rankings** → **Correlation Measures**

÷ 32

Average Correlation

APPENDIX C.

The First Experimental Dataset

A scheme for communicating among groups that use different type hierarchies.

A comparison of type hierarchies and their use in communication.
A method for communication which is based on the use of hierarchies.
Applications of type hierarchies.
Communication among objects which is based on alternative type hierarchies.
Communication among objects.
Group communication.
Procedures for communication among groups.
Schemes for group communication.
Using type hierarchies for communication.

A unified approach to compression, information theory, and grammars.

A single approach to information theory and grammars.
A theory of compression.
A theory of information theory and grammars.
A unified theory of computing problems.
Compressing information using a grammar of information content.
Information on grammatical formalisms.
The potential application of grammars to data compression.
The relationship between compression, information theory and grammars.
Uniting the compression of information with a theory of information.

Distributed form management.

Applications for distributed data.
Forming strategies for control of information.
How to control information which is distributed in nature.
Managing forms and other user interface metaphors.
Managing distributed forms.
Partitioned and replicated information.
Strategies formed by management for controlling distribution of resources.
The management of forms.
The management of distributed data.

Current attitudes and future prospects for work at home for computer professionals.

Attitudes to programmers working at home.
Current attitudes of professionals to working at home.
Professional workers who work out of the office.
Professional home computers.
Prospects for home computers in the future.
Prospects for working at home.
The attitude of computer professionals.
The liklihood of professional programmers working at their residences.
The prospects for home computers working in professional environments.

An extensible object oriented information management environment.

Extended information management.
Extensible environments.
Extensible environments for managing information of an object oriented nature.
Managing object oriented information.
Object oriented design.
Object oriented information.
Object oriented programming environments.
Orienting managers with the objective of improving worker environments.
The object of environment management.

An automatic tool for office system conceptual design.

Automatic design of office systems.
Automatic design tools.
Concepts used in the design of office tools.
Designing office systems using computer tools.
Office tools.
Office design.
System design.
Tools for automatically building systems for use in office environments.
Tools for design of office systems.

A hypertext tool for exploratory policy discussion.

A discussion on hypertext systems.
A tool for building hypertext systems.
A discussion of policies between individuals which is facilitated by a hypertext tool.
A policy for the use of hypertext systems.
Exploring the policies used in human-computer discussion.
Hypertext tools.
Hypertext systems used in the discussion of company policies.
Policy discussion using hypertext tools.
Policy discussion.

A spreadsheet for cooperative work.

A spreadsheet which works on personal computers.
Cooperative work.
Cooperative work using spreadsheets.
How spreadsheets work.
Spreadsheets for group work.
The efficiency of spreadsheets in a cooperative work environment.
Using spreadsheets.
Using spreadsheets efficiently in group work.
Working with spreadsheets.

A field experiment into work group structures and computer support.

A group of workers with computer support.
Computer support of work groups.
Experimental structures for supporting field work.
Experiments on computer support.
Experiments on computer support in work groups.
Field computer support.
Grouping experiments into structures by computer.
Supporting field experiments using working computers.
The structure of work groups.

A preliminary inquiry into diversity in the use of electronic mail.

An investigation into different applications of electronic mail.
An inquiry into the use of mail.
Diverse uses of electronic mail.
Electronic mail systems.
How users use computer mailing systems.
How users use computer mail.
Mailing diverse reports by electronic means.
Using electronic mail.
Using electronic mail systems.

Tools for communicating in a hypertext environment.

Communication environments using hypertext technology.
Communication tools.
Communication tools in a hypertext environment.
Hypertext communications.
Hypertext systems and communication tools.
Hypertext systems used in communications.
Hypertext environment tools.
Hypertext tools.
Hypertext communication tool.

Document structure with browsing semantics.

Browsing semantics.
Browsing through document structures.
Browsing through documents.
Browsing the semantics of document structures.
Semantic browsing using document structure.
Structured semantics with application to browsing systems.
Structured browsing through document texts.
The structure and semantics of documents.
The semantics of structure.

Context and orientation in hypermedia networks.

A network of hypermedia systems.
Hypermedia networks.
Hypermedia tools for orienting the end user.
Hypermedia browsing and orientation tools.
Hypermedia orientation tools.
Maintaining the context information in a hypermedia system.
Orientation tools in hypermedia systems.
Orientation in hypermedia systems.
User orientation in hypertext browsing tools.

A data model for flexible hypertext database systems.

Data models.
Flexible hypertext systems.
Flexible modelling.
Flexible hypertext databases.
Hypertext data models.
Hypertext database systems.
Modelling hypertext database systems.
Models for database systems.
Models of hypertext systems.

Object specialisation.

Design of objects.
Object design.
Objective specialisation.
Specialisation of systems.
Specialised objectives.
System specialisation.
The objective of specialisation.
The specialisation of objects.
The objective of specialisation in system design.

An algebra for structured office documents.

Algebraic documents.
An algebra for document structure.
Office document algebra.
Office documents.
Structured documents.
Structured algebras.
Structured office document algebra.
Structured office documents.
Structured algebras for office documents.

Design issues and performance evaluation of partitioned signature files.

A file of design and performance issues.
Designing signature files.
Designing partitioned signature files.
Evaluation of signature files.
Partitioned signature files.
Partitioned file design and evaluation.
Partitioned signature file performance.
Performance issues and evaluation of signature files.
Signature file performance and design.

Optimum polynomial retrieval functions based on the probability ranking principle.

Optimum ranking retrieval functions.
Optimum principles for probabilistic retrieval functions.
Optimum retrieval principles based on probabilistic ranking.
Polynomial retrieval functions.
Probabilistic retrieval functions.
Probabilistically based retrieval principles.
Retrieval functions.
The principle of ranking in document retrieval systems.
The probability ranking principle and retrieval functions.

A critical investigation of recall and precision as measures of retrieval system performance.

A critical look at measures of performance for computer systems.
A critical investigation of precision for measuring performance of retrieval systems.
A look at the measures of recall and precision for different retrieval systems.
An investigation of recall as a measure of retrieval performance.
An investigation of measures used in evaluating the performance of retrieval systems.
An investigation into the precision of information retrieval systems.
Investigating the performance of retrieval systems using measures of recall and precision.
The performance of precision retrieval systems.
The precision of measures used in investigating information retrieval systems.

Compression and encryption considerations of string text retrieval systems.

Compression considerations of retrieval systems for encrypted text.
Considering the compression and encryption of strings for text retrieval systems.
String text compression systems.
String compression considerations of text retrieval systems.
String text retrieval systems.
Systems for retrieval of compressed and encrypted text strings.
Text retrieval with consideration of string encryption and compression.
The consideration that must be given to encryption and compression in systems which retrieve text.
The compression and encryption of string text retrieval systems.

Knowledge based search tactics for an intelligent intermediary system.

An intelligent intermediary system with knowledge based search tactics.
Intelligent tactics for intermediary searches on knowledge based systems.
Intermediary search tactics based on knowledge of intelligent systems.
Knowledge based systems for performing intermediary searches.
Knowledge based intelligent systems.
Search techniques based on semantic knowledge for use with an intermediary system based on artificial intelligence.
Search tactics for intelligent systems.
Tactics for searching intelligent systems.
Tactics for searching knowledge bases.

Information retrieval using a hypertext based help system.

A system to help with retrieval of hypertext information.
An information retrieval system that uses hypertext technology for helping users.
Hypertext help systems.
Information retrieval systems.
Information retrieval help systems organized as hypertext.
Retrieval of information from hypertext systems.
The use of hypertext based systems in information retrieval.
The use of hypertext systems in providing helpful retrieval information.
The use of hypertext systems in aiding users to retrieve information from databases.

Syntactic structure matching for information retrieval.

Information systems that use syntactic structure matching for retrieval.
Information retrieval based on the structured matching of syntactic information.
Retrieval of information relating to the syntactic structure of texts.
Retrieval of structured information.
Retrieval of syntactic information for use in structure matching.
Retrieval of information by matching syntactic structures.
The use of structure matching for retrieval of syntactic information.
The matching of syntactic structures for retrieval of information.
The use of syntactic level language processing to construct structures for use in matching queries to documents in information retrieval.

A user interface to relational databases that permits vague queries.

A database that permits vague queries.
A relational database with a user interface which allows users to enter vague queries.
A database interface for users with vague queries.
A user interface to a computer system.
An interface that permits vague queries.
How users with vague queries interface with relational databases.
Relational databases with vague user interfaces.
The special requirements of user interfaces to relational databases.
User queries to relational databases.

Knowledge based tools to promote shared goals and terminology between interface designers.

Knowledge based tools for promoting the terminology of interface designers.
Knowledge based tools for promoting the design of shared interfaces.
The use of knowledge based tools in the design of computer interfaces.
The goals of interface designers.
The design of interfaces for knowledge based systems.
The promotion of shared goals for designers of computer interfaces.
The use of knowledge based tools for the promotion of a shared terminology by the designers of computer interfaces.
Tools for standardising interface design through the promotion of shared goals and terminology.
Tools for promoting the use of standard terminology between interface designers.

A rule based message filtering system.

A rule based mail message filtering system.
A water filtering system.
A computer system for filtering rule based messages.
A system of rules for filtering messages.
A rule based system for discarding invalid messages in an object oriented environment.
An expert system which discards unwanted mail messages.
Rules for the construction of message filtering systems.
The use of a set of rules for filtering system messages in a computer environment.
The use of rule based message filtering systems in tools for automatically creating object oriented programs.

Access to, usage of and outcomes from an electronic messaging system.

Access to, usage of and outcomes from a system which sends and receives electronic messages.
Access to the outcomes of usage of an electronic messaging system.
Access to, usage of or outcomes from electronic messaging systems.
Access to the usage statistics of an electronic messaging system.
An electronic messaging system.
The usage of electronic access controls in messaging systems.
The analysis of access and usage statistics to determine the outcomes from use of messaging systems.
The usage of electronic messaging in illegeally accessing computer systems.
Usage of and access to electronic systems.

A new language for the study and implementation of coordination.

A new language for the study of the implementation of coordination.
A study and implementation of human cognitive coordination through the introduction of a new human language.
A new language for studying and implementing coordination in robots.
A new implementation of the coordination of the study of language.
A study of the coordination of the implementation of programs in a new computer language.
New languages for the study and implementation of computer programs for coordination of civil engineering projects.
The study and implementation of coordination.
The implementation of a new language for the study of coordination.
The coordination of a study of language.

Computer systems and the design of organisational interaction .

Computer systems and organisational interaction.
Computer systems and organisational design.
Computer designed organisational interaction.
Organisational constraints in the interactive design of computer systems.
Systems for aiding in the design of organisational interaction.
The design of interactive computer systems for business organisations.
The use of computer systems in the interactive design of organisational structures.
The interaction between computer systems in an organisation.
The organisational interaction necessary for the design of effective computer systems.

Query processing in a multimedia document system.

Document processing in a multimedia environment.
Document retrieval systems which are multimedia in nature.
Multimedia systems for processing documents.
Multimedia systems and their strategies for processing queries.
Processing queries in information systems.
Query systems for processing multimedia documents.
The processing of search requests in an information retrieval system.
The systematic processing of queries to multimedia documents.
User query systems.

Implementing ranking strategies using text signatures.

A strategy for signing texts based on an implementation of ranking.
Implementing signature-based retrieval.
Ranked retrieval on textual data.
Signature implementations of text ranking strategies.
Signature retrieval on text databases.
Strategies for ranking output in document retrieval.
The implementation of superimposed coding and signature files which generate ranked output.
The use of text signatures in the ranking of retrieved documents.
The signature of military documents by highly ranked authors.

An experimental multimedia mail system.

Experimental systems for multimedia mail.
Experimental mail systems.
Experiments on multimedia systems.
Experiments on mail systems.
Mail systems in a multimedia environment.
Mailing experimental results using conventional multimedia delivery systems.
Multimedia systems for experimental mail re-routing.
Multimedia mail.
Using multimedia systems for experimenting with mail tools.

APPENDIX D.

Rankings of the First Experimental Dataset

RANKINGS:

HUMAN	COMPUTER	
		A scheme for communicating among groups that use different type hierarchies.
1	7	Procedures for communication among groups.
2	4	Schemes for group communication.
3	6	Communication among objects which is based on alternative type hierarchies.
4	5	A method for communication which is based on the use of hierarchies.
5	7	Group communication.
5	1	Using type hierarchies for communication.
7	3	A comparison of type hierarchies and their use in communication.
8	9	Communication among objects.
9	1	Applications of type hierarchies.
		A unified approach to compression, information theory, and grammars.
1	3	A single approach to information theory and grammars.
2	1	The relationship between compression, information theory and grammars.
3	2	A theory of information theory and grammars.
4	8	A unified theory of computing problems.
5	4	Uniting the compression of information with a theory of information.
6	6	A theory of compression.
7	7	Information on grammatical formalisms.
8	9	The potential application of grammars to data compression.
9	5	Compressing information using a grammar of information content.
		Distributed form management.
1	3	Managing distributed forms.
2	1	The management of forms.
3	2	The management of distributed data.
4	6	How to control information which is distributed in nature.
5	8	Managing forms and other user interface metaphors.
6	4	Applications for distributed data.
6	9	Partitioned and replicated information.
8	7	Forming strategies for control of information.
9	5	Strategies formed by management for controlling distribution of resources.

RANKINGS:

HUMAN	COMPUTER	
		Current attitues and future prospects for work at home for computer professionals.
1	9	The liklihood of professional programmers working at their residences.
2	3	Attitudes to programmers working at home.
3	4	Current attitudes of professionals to working at home.
4	2	Prospects for working at home.
5	7	Professional workers who work out of the office.
6	1	The attitude of computer professionals.
7	6	The prospects for home computers working in professional environments.
7	7	Prospects for home computers in the future.
9	5	Professional home computers.
		An extensible object oriented information management environment.
1	7	Extensible environments for managing information of an object oriented nature.
2	2	Managing object oriented information.
3	3	Object oriented programming environments.
4	9	Extensible environments.
5	1	Object oriented information.
6	5	Extended information management.
7	4	Object oriented design.
8	6	The object of environment management.
9	8	Orienting managers with the objective of improving worker environments.
		An automatic tool for office system conceptual design.
1	2	Tools for design of office systems.
2	4	Designing office systems using computer tools.
3	3	Automatic design of office systems.
4	5	Tools for automatically building systems for use in office environments.
5	8	Automatic design tools.
6	7	Office design.
7	1	System design.
8	9	Office tools.
9	6	Concepts used in the design of office tools.

RANKINGS:

HUMAN	COMPUTER	
		A hypertext tool for exploratory policy discussion.
1	5	A discussion of policies between individuals which is facilitated by a hypertext tool.
2	3	Policy discussion using hypertext tools.
3	7	Hypertext systems used in the discussion of company policies.
4	2	Hypertext tools.
5	1	Policy discussion.
6	4	A discussion on hypertext systems.
7	6	A tool for building hypertext systems.
8	8	Exploring the policies used in human-computer discussion.
9	9	A policy for the use of hypertext systems.

		A spreadsheet for cooperative work.
1	4	Spreadsheets for group work.
2	2	Cooperative work using spreadsheets.
3	5	Using spreadsheets efficiently in group work.
4	6	The efficiency of spreadsheets in a cooperative work environment.
5	1	Cooperative work.
6	3	Working with spreadsheets.
7	9	Using spreadsheets.
8	8	A spreadsheet which works on personal computers.
9	7	How spreadsheets work.

		A field experiment into work group structures and computer support.
1	4	Experiments on computer support in work groups.
2	2	Computer support of work groups.
3	5	Experiments on computer support.
4	6	A group of workers with computer support.
5	1	The structure of work groups.
6	8	Supporting field experiments using working computers.
7	3	Field computer support.
7	7	Grouping experiments into structures by computer.
9	9	Experimental structures for supporting field work.

RANKINGS:

HUMAN	COMPUTER	
		A preliminary inquiry into diversity in the use of electronic mail.
1	3	Diverse uses of electronic mail.
2	6	An investigation into different applications of electronic mail.
3	8	How users use computer mail.
4	9	How users use computer mailing systems.
5	2	Using electronic mail.
6	4	Using electronic mail systems.
7	1	An inquiry into the use of mail.
8	5	Electronic mail systems.
9	7	Mailing diverse reports by electronic means.
		Tools for communicating in a hypertext environment.
1	2	Communication tools in a hypertext environment.
2	7	Hypertext systems and communication tools.
3	4	Hypertext tools.
4	5	Hypertext communications.
5	3	Hypertext communication tool.
6	1	Hypertext environment tools.
7	8	Communication environments using hypertext technology.
8	5	Communication tools.
9	9	Hypertext systems used in communications.
		Document structure with browsing semantics.
1	2	Browsing through document structures.
2	1	Browsing semantics.
3	7	Browsing through documents.
4	6	Semantic browsing using document structure.
5	9	The structure and semantics of documents.
6	5	Structured browsing through document texts.
7	3	Browsing the semantics of document structures.
8	8	Structured semantics with application to browsing systems.
9	4	The semantics of structure.

RANKINGS:

HUMAN	COMPUTER	
		Context and orientation in hypermedia networks.
1	2	Orientation in hypermedia systems.
2	4	Orientation tools in hypermedia systems.
3	6	Hypermedia orientation tools.
3	5	Maintaining the context information in a hypermedia system.
5	7	Hypermedia browsing and orientation tools.
6	9	Hypermedia tools for orienting the end user.
7	8	User orientation in hypertext browsing tools.
7	1	Hypermedia networks.
9	3	A network of hypermedia systems.
		A data model for flexible hypertext database systems.
1	8	Hypertext data models.
2	5	Modelling hypertext database systems.
3	3	Models of hypertext systems.
4	7	Flexible hypertext databases.
5	1	Hypertext database systems.
6	1	Flexible hypertext systems.
7	3	Models for database systems.
8	6	Data models.
9	9	Flexible modelling.
		Object specialisation.
1	1	The specialisation of objects.
2	7	Design of objects.
2	2	Object design.
4	2	Specialisation of systems.
5	2	System specialisation.
6	8	The objective of specialisation in system design.
7	2	Objective specialisation.
8	2	The objective of specialisation.
9	9	Specialised objectives.

RANKINGS:

HUMAN	COMPUTER	
		An algebra for structured office documents.
1	6	Structured office document algebra.
2	8	An algebra for document structure.
3	7	Office document algebra.
4	1	Structured office documents.
5	4	Structured algebras for office documents.
6	1	Office documents.
6	1	Structured documents.
8	5	Structured algebras.
9	9	Algebraic documents.
		Design issues and performance evaluation of partitioned signature files.
1	8	Signature file performance and design.
2	5	Designing partitioned signature files.
3	3	Performance issues and evaluation of signature files.
4	4	Partitioned signature file performance.
5	7	Designing signature files.
6	2	Evaluation of signature files.
7	1	Partitioned signature files.
8	9	Partitioned file design and evaluation.
9	6	A file of design and performance issues.
		Optimum polynomial retrieval functions based on the probability ranking principle.
1	4	Optimum retrieval principles based on probabilistic ranking.
2	6	The probability ranking principle and retrieval functions.
3	9	Probabilistically based retrieval principles.
4	1	Polynomial retrieval functions.
5	3	Optimum ranking retrieval functions.
6	4	Probabilistic retrieval functions.
7	8	Optimum principles for probabilistic retrieval functions.
8	7	The principle of ranking in document retrieval systems.
9	1	Retrieval functions.

RANKINGS:

HUMAN	COMPUTER	
		A critical investigation of recall and precision as measures of retrieval system performance.
1	4	A critical investigation of precision for measuring performance of retrieval systems.
2	5	Investigating the performance of retrieval systems using measures of recall and precision.
3	9	A look at the measures of recall and precision for different retrieval systems.
4	1	An investigation of recall as a measure of retrieval performance.
5	6	An investigation of measures used in evaluating the performance of retrieval systems.
6	3	An investigation into the precision of information retrieval systems.
7	7	A critical look at measures of performance for computer systems.
8	8	The precision of measures used in investigating information retrieval systems.
9	2	The performance of precision retrieval systems.
		Compression and encryption considerations of string text retrieval systems.
1	9	The consideration that must be given to encryption and compression in systems which retrieve text.
2	7	Considering the compression and encryption of strings for text retrieval systems.
3	5	Text retrieval with consideration of string encryption and compression.
4	4	String compression considerations of text retrieval systems.
5	2	The compression and encryption of string text retrieval systems.
6	6	Compression considerations of retrieval systems for encrypted text.
7	8	Systems for retrieval of compressed and encrypted text strings.
8	1	String text retrieval systems.
9	3	String text compression systems.

RANKINGS:

HUMAN	COMPUTER	
		Knowledge based search tactics for an intelligent intermediary system.
1	3	An intelligent intermediary system with knowledge based search tactics.
2	9	Search techniques based on semantic knowledge for use with an intermediary system based on artificial intelligence.
3	1	Search tactics for intelligent systems.
4	4	Tactics for searching intelligent systems.
5	6	Intermediary search tactics based on knowledge of intelligent systems.
5	7	Knowledge based systems for performing intermediary searches.
7	8	Intelligent tactics for intermediary searches on knowledge based systems.
8	5	Tactics for searching knowledge bases.
9	1	Knowledge based intelligent systems.
		Information retrieval using a hypertext based help system.
1	5	An information retrieval system that uses hypertext technology for helping users.
2	3	Information retrieval help systems organized as hypertext.
3	1	The uses of hypertext based systems in information retrieval.
4	9	The use of hypertext systems in aiding users to retrieve information from databases.
5	8	The use of hypertext systems in providing helpful retrieval information.
6	6	Hypertext help systems.
7	2	Information retrieval systems.
8	7	Retrieval of information from hypertext systems.
9	4	A system to help with retrieval of hypertext information.
		Syntactic structure matching for information retrieval.
1	1	Retrieval of information by matching syntactic structures.
2	2	The matching of syntactic structures for retrieval of information.
3	6	Information systems that use syntactic structure matching for retrieval.
4	9	The use of syntactic level language processing to construct structures for use in matching queries to documents in information retrieval.
5	8	Information retrieval based on the structured matching of syntactic information.
6	4	Retrieval of information relating to the syntactic structure of texts.
7	5	The use of structure matching for retrieval of syntactic information.
8	7	Retrieval of syntactic information for use in structure matching.
9	3	Retrieval of structured information.

RANKINGS:

HUMAN	COMPUTER	
		A user interface to relational databases that permits vague queries.
1	7	A relational database with a user interface which allows users to enter vague queries.
2	4	A database interface for users with vague queries.
3	2	A database that permits vague queries.
4	3	How users with vague queries interface with relational databases.
5	1	An interface that permits vague queries.
6	6	User queries to relational databases.
7	8	The special requirements of user interfaces to relational dtabases.
8	5	Relational databases with vague user interfaces.
9	9	A user interface to a computer system.
		Knowledge based tools to promote shared goals and terminology between interface designers.
1	8	The use of knowledge based tools for the promotion of a shared terminology by the designers of computer interfaces.
2	5	Tools for standardising interface design through the promotion of shared goals and terminology.
3	4	Tools for promoting the use of standard terminology between interface designers.
4	5	The promotion of shared goals for designers of computer interfaces.
5	1	Knowledge based tools for promoting the terminology of interface designers.
6	9	The use of knowledge based tools in the design of computer interfaces.
7	3	Knowledge based tools for promoting the design of shared interfaces.
8	7	The design of interfaces for knowledge based systems.
8	2	The goals of interface designers.
		A rule based message filtering system.
1	1	A rule based mail message filtering system.
2	3	A system of rules for filtering messages.
3	9	An expert system which discard unwanted mail messages.
4	7	A rule based system for discarding invalid messages in an object oriented environment.
5	6	The use of a set of rules for filtering system messages in a computer environment.
6	4	Rules for the construction of message filtering systems.
7	2	A computer system for filtering rule based messages.
8	8	The use of rule based message filtering systems in tools automatically creating object oriented programs.
9	5	A water filtering system.

RANKINGS:
HUMAN COMPUTER
----- --------

Access to, usage of and outcomes from an electronic messaging system.

1	5	Access to, usage of and outcomes from a system which sends and receives electronic messages.
2	2	Access to, usage of or outcomes from electronic messaging systems.
3	3	Access to the outcomes of usage of an electronic messaging system.
4	3	Access to the usage statistics of an electronic messaging system.
5	9	The analysis of access and usage statistics to determine the outcomes from use of messaging systems.
6	1	An electronic messaging system.
7	8	The usage of electronic access controls in messaging systems.
8	6	Usage of and access to electronic systems.
9	7	The usage of electronic messaging in illegeally accessing computer systems.

A new language for the study and implementation of coordination.

1	7	A study and implementation of human cognitive coordination through the introduction of a new human language.
2	6	A new language for studying and implementing coordination in robots.
3	1	The study and implementation of coordination.
4	9	The implementation of a new language for the study of coordination.
5	5	A new language for the study of the implementation of coordination.
6	8	New languages for the study and implementation of computer programs for coordination of civil engineering projects.
7	2	A new implementation of the coordination of the study of language.
8	4	A study of the coordination of the implementation of programs in a new computer language.
9	3	The coordination of a study of language.

RANKINGS:

HUMAN	COMPUTER	
		Computer systems and the design of organisational interaction.
1	2	Computer designed organisational interaction.
1	3	Systems for aiding in the design of organisational interaction.
3	1	Computer systems and organisational interaction.
4	7	The use of computer systems in the interactive design of organisational structures.
5	6	Computer systems and organisational design.
6	5	The organisational interaction necessary for the design of effective computer systems.
7	4	The interaction between computer systems in an organisation.
7	9	The design of interactive computer systems for business organisations.
9	8	Organisational constraints in the interactive design of computer systems.
		Query processing in a multimedia document system.
1	4	Multimedia systems and their strategies for processing queries.
2	3	The systematic processing of queries to multimedia documents.
3	1	Query systems for processing multimedia documents.
4	7	Document retrieval systems which are multimedia in nature.
5	6	Processing queries in information systems.
6	9	The processing of search requests in an information retrieval system.
7	8	Document processing in a multimedia environment.
8	2	Multimedia systems for processing documents.
9	5	User query systems.

RANKINGS:

HUMAN	COMPUTER	
		Implementing ranking strategies using text signatures.
1	1	The use of text signatures in the ranking of retrieved documents.
2	7	The implementation of superimposed coding and signature files which generate ranked output.
3	2	Signature implementations of text ranking strategies.
4	5	Signature retrieval on text databases.
4	3	Implementing signature-based retrieval.
6	6	Strategies for ranking output in document retrieval.
7	9	Ranked retrieval on textual data.
8	4	A strategy for signing texts based on an implementation of ranking.
9	8	The signature of military documents by highly ranked authors.

		An experimental multimedia mail system.
1	3	Experimental systems for multimedia mail.
2	2	Multimedia mail.
3	5	Mail systems in a multimedia environment.
4	1	Experimental mail systems.
5	6	Experiments on mail systems.
6	4	Multimedia systems for experimental mail re-routing.
6	6	Experiments on multimedia systems.
8	8	Using multimedia systems for experimenting with mail tools.
9	9	Mailing experimental results using conventional multimedia delivery systems.

APPENDIX E.

The Second Experimental Dataset

Algebraic Support for Complex Objects in Database Systems

Complex Object Database Systems
Complex Objects in Database Systems
Database Algebras Supporting Objects
Database Algebras
Database Support of Complex Objects
Database Systems with Complex Objects
Supporting Complex Algebras by Database Systems
Supporting Complex Objects in Databases
System Support for Complex Databases

Update Propagation and Security Enforcement in Distributed Database Systems

Database Security
Distributed System Security
Distributed Database Security Enforcement
Distributed Update Propagation
Distributed Security Enforcement
Enforcing the Distribution of Databases
Enforcing Security and Propagating Updates in Databases
The Propagation of Updates in Distributed Databases
Updating Distributed Security

A General Framework for Large Scale Constraint Based Optimisation

A General Framework for Constraint Optimisation
A General Optimisation Framework
A Large Optimisation Framework for General Constraints
A Large set of Constraints for General Optimisation
Constraints on Large Scale Frameworks
General Constraint Optimisation
Large Frameworks Based on Scaled Constraints
Optimising Constraints on a Large Scale
The Optimisation of Constraints on a Large Scale

User Models in Dialog Systems

A Dialog System for Users
A System for Modelling User Dialog
A System User Model
A System of User Models
A Model for Users in Dialog Systems
A System for User Modelling
Dialog with System Users
Modelling Users in Systems
User Dialog Modelling

A Gentle Introduction to Symbolic Processing in Common Lisp

A Common Lisp Introduction
A Common Introduction to Symbols
A Gentle Introduction to Symbols in Lisp
An Introduction to Common Lisp
An Introduction to Symbolic Processing
Common Lisp Processing
Lisp Processing of Symbols
Processing Gentle Symbols in Lisp
Symbolic Processing and Common Lisp

Knowledge Representation and Defeasible Reasoning in Artificial Intelligence

A Reason for Knowledge Representation
A Knowledge of Artificial Intelligence
An Intelligent Reason for Representing Knowledge
An Intelligent Representation of Artificial Reasoning
Artificial Reasoning and the Representation of Intelligence
Artificial Representation of Reasoning Processes
Artificial Knowledge and Defeasible Reasoning
Artificial Reasoning with Knowledge
The Representation of Knowledge in Artificial Intelligence

Encoding Techniques for Complex Information Structures in Connectionist Systems

A Connectionist System for Complex Structures
A System for Encoding Complex Information
Complex Structures and Encoding Techniques
Complex Information Systems
Information on Techniques for Encoding Complex Systems
Structured Information Techniques and Information Systems
System Information and the Structure of Complex Encoding Techniques
Techniques for Encoding Complex Structures
The Structure of Complex Systems

Data Aquisition for Natural Language Processing

Data Processing
Natural Aquisition of Language Data
Natural Language Processing
Natural Data Processing
Natural Language Aquisition
Processing in Data Acquisition
The Natural Processing of Data
The Natural Aquisition and of Data
The Aquisition of Natural Language

The Representation of Cognitive Structure in Expert Knowledge

A Knowledge of Expert Representation Methods
An Expert on the Representation of Cognitive Structure
Cognitive Experts and Knowledge Representation
Expert Knowledge Representation
Representation of Expert Knowledge
Representation of Structure by Experts
Representing Cognitive Experts
The Structure of Expert Knowledge
The Structure of Cognitive Expert Knowledge

Incomplete Information in a Deductive Database

Database Information
Deductive Information
Deductive Incomplete Information
Incomplete Information
Incomplete Deductive Databases
Incomplete Databases
Information on Deductive Databases
Information on Databases
Information on Incomplete Databases

The Construction of Dynamic Models of Security Using Knowledge Based Techniques

A Security Model for Dynamic Knowledge
A Model of Security in Dynamic Systems
A Security Model Dynamically Constructed
Dynamic Security Model Construction
Dynamic Knowledge Model Construction
Knowledge of Dynamic Techniques for Constructing Security Models
Knowledge Based Methods for Constructing Dynamic Security Models
Model Construction using Knowledge Based Techniques
Techniques for Dynamically Constructing Models of Security

A Planning Approach for the Software Development Process

A Development Process for Planning Software
An Approach to Software Processes
Development Approaches to Plan Processing
Development Processes for Software Plans
Planning from a Software Approach
Planning Software Processes
Planning Software Development
Plans for Software Development Processes
Processing Plans for Software

An Artificial Neural Network for Handling Uncertainty in Expert Systems

A Network of Artificial Expert Systems
A System for Uncertainty Handling Using Neural Networks
A Neural Network for Expert Systems
An Expert System for handling Uncertainty
Artificial Systems and Networks of Uncertainty
Network Uncertainty and Expert Systems
Network Experts and Artificial Systems
Uncertainty about Neural Networks
Uncertainty in Artificial Expert Systems

Evaluating Design Using Knowledge of Purpose and Structure

Design Evaluation using Knowledge
Knowledge of Structure on Design EvaluationR
Knowledge of Purpose in Evaluation of Design
Knowledge of Design Evaluation
The Purpose of Evaluating Design
The Purpose of Designing Structure
The use of Purpose and Structure in Evaluating Design
The Purpose of Design Evaluation
Using Knowledge for Design Evaluation

Computer Based Medical Systems Using Artificial Intelligence

Artificial Intelligence in Medical Systems
Artificial Intelligence in Computer Systems
Artificial Medical Systems using Computer Intelligence
Artificial Medical Systems
Artificial intelligence in Medical Computer Systems
Computer Intelligence in Medical Systems
Computer Systems in Medical Applications
Medical Systems Using Artificial Intelligence
Medical Computer Systems using Artificial Intelligence

Automated Reasoning Tools for Information Management Systems

A System for Managing Information Automatically
Automated System Management Tools
Automated System Tools for Management of Information
Information on the Management of Automatic Reasoning Tools
Reasoning and Management of Information Systems
Reasons for Automating Information Management
Tools for Management Reasoning
Tools and Systems for Automatic Reasoning
Tools and Systems for Information Management

A Comparative Evaluation of Expert System Building Tools

A Tool for Building Expert Systems
A Tool for Expert Evaluation of Systems
A Comparison of Tools for Building Expert Systems
Building Expert System Tools
Building Evaluations of Expert Systems
Building Systems with Expert Tools
Evaluating Tools used to Build Expert Systems
Evaluation of Expert System Tools
The Evaluation of Expert System Tools

Building and Using a Highly Parallel Programmable Logic Array

Building Logic Arrays and Highly Programmable Parallel Uses
Building and Using Parallel Programs
Building Highly Logical Parallel Arrays
Building Highly Logical Parallel Programs
Logical Uses of Highly Programmable Arrays
Parallel Use of Logic Arrays
Programming and Using Logic Arrays in Parallel
The Use of Highly Parallel Logic Arrays
Using Highly Logical Arrays in Parallel Programs

Designing the System Architecture of a Parallel Computer

Computer System Architecture and Parallel Design
Design of Parallel Computer Architectures
Parallel Computer Architecture Design
Parallel Computer System Design
Parallel Computer System Architecture
Parallel Design of Computer Architectures
System Design for Parallel Computers
System Architecture of a Parallel Computer
The Architecture of a Parallel Computer

Compiling Scientific Code Using Partial Evaluation

A Partial Evaluation of Scientific Code Compilers
A Partical Compiler for Scientific Code
A Code Compiler for Scientific Systems
An Evaluation of Partial Code Compilers
Compiling Partial Evaluations of Scientific Code
Evaluating Code Using Partial Compilation
Evaluating Partial Compilation of Code
Partial Evaluation of Scientific Code
Partial Code Compilers for Scientific Data

Experimental Research and Development of Computer Architecture

Computer Architecture Developments and Research
Computer Experiments in Research and Development
Experimental Computer Architecture and Research
Experimental Computer Architectures and their Development
Experiments into the Development of Computer Research
Research and Development of Experimental Computer Architecture
Research and Experiments in Computer Architecture
The Architecture and Development of Computer Research
The Development of Computer Architecture Experiments

Comprehensive Support for Highly Interactive Graphical User Interfaces

Comprehensive User Support for Graphics
Graphical User Interface Support and Design
High Support for Interactive Interfaces
Interactive Graphics and User Interfaces
Interactive Interfaces for Graphical Support of Users
Interactive User Interfaces and their Support
Interface Support for Interactive Graphics
Support for Comprehensive User Interfaces
User Support for Graphical Interfaces

Multiprocessor Algorithms for Relational Database Operations on Hypercube Systems

Algorithms for Database Operations on Hypercubes
Database Systems an Multiprocessor Hypercubes
Hypercube Database Systems and Multiprocessor Operations
Hypercube Relational Database Operations
Hypercube Systems and Algorithms for Database Operations
Multiprocessor Operations and Algorithms for Database Systems
Multiprocessor Relational Database Operations
Multiprocessor Hypercube Systems and Algorithms for Relational Operations
Multiprocessor Hypercube Relational Database Algorithms

Fault Tolerant Clock Synchronisation in Distributed System

Clock Synchronisation Tolerances in Faulty Systems
Clocking Faults in Distributed Systems
Distributed System Clock Synchronisation
Distributed Clock Synchronisation Systems
Distributed Synchronisation of Faulty System Clocks
Distributing Faults in Synchronised Systems
Faults in Clock Synchronisation in Distributed Systems
Faulty System Clocks and Synchronisation
Synchronising Faults in Distributed Systems