

ASHG: Automatic Semantic Header Generator

S. Haddad B. Desai

August 29, 2001

Abstract

Indexing file systems is a powerful means of helping users locate documents, software, and other types of data among large repositories. In environments that contain many different types of data, content indexing requires type-specific processing to extract information effectively. Cover page (or Semantic Header) is a portion of each document which should contain information useful in searching for a document based on a number of commonly used criteria. The information from the semantic header could be used by various indexing schemes to help locate appropriate documents with minimum effort. In this paper, we present a model that automatically extracts the secondary or meta-information, and stores it in a Semantic Header which will be used as an index for the document, which will help users in accessing and searching for it.

Introduction

Rapid growth in data volume, user base and data diversity render Internet-accessible information increasingly difficult to use effectively. Browsing a hierarchy containing millions of directories is infeasible. Instead, there is a need for an automated search and for a system that allows easy ' search for and access to ' resources available on the Internet. This automated search has to help users in locating relevant information. In order to have this easy access and for efficiency reasons, this resource discovery problem requires indexing the available information. Thus, secondary information called meta-information must be extracted and used as an index to the available primary resource. In turn, building this effective index requires information extraction methods tailored to each specific environment. So, the semantics of the files in which the primary resource is stored will be exploited to extract and summarize the relevant information to support the resource discovery. To do this, the primary file type should be identified and then the type specific selection and extraction methods are applied to the file.

It is envisioned that regional and/or specialized databases would be created to maintain archives of the cover pages (or Semantic Header). These databases could be searched by cooperating distributed expert systems to help users locate pertinent documents. Such a system is currently under development at Concordia University and is called CINDI.

CINDI, an under development system, provides a mechanism to register, search and manage the meta-information, with the help of an easy to use graphical user interface. The entered information will be stored in a database. CINDI tries to avoid problems caused by differences in semantics and representation, incomplete and incorrect data cataloguing. It also tries to avoid the problems caused by the difference in terms naming. And this meta-information could be either entered by the primary resource provider or by the ASHG, a software that automatically generates the meta-information of the submitted document. So, if the provider is responsible for preparing the index entry, there is the potential for its accuracy to be very high. However, what ASHG is trying to accomplish is to prepare the index entry automatically and also with a high degree of accuracy.

ASHG will try to save the primary resource provider's time by automatically generating or extracting the meta-information (Semantic Header) of the document and storing the extracted info in a database.

Overview of the CINDI system

The trend in most research institutes, universities and business organization to interconnect their computing facilities using a digital network has become the accepted method of sharing resources. Such networks, in turn, are interconnected allowing information to be exchanged across networks using the TCP/IP protocol.

There is a need for the development of a system which allows easy 'search for and access to' resources available on the Internet. Solving the problem of fast, efficient and easy access to the documents can be started by building a standard index structure and building a bibliographic system using standardized control definitions and terms. Such definitions could be built into the knowledge-base of an expert system based index entry and search interfaces. The purpose of indices and bibliographies (secondary information) is to inventory the primary information and allow easy access to it.

Preparing a bibliography requires finding the primary source, identifying it as to its subject, title, author, keywords, abstract, etc. Since it is to be used by many users, it has to be accurate, easy to use and properly classified.

The problem with many indices like (WAIS, DEBR, FLET,.....) is that their selectivity of documents is often poor. The chances of getting inappropriate documents and missing relevant information because of poor choice of search terms is large. These problems are addressed by CINDI, that provides a mechanism to register , manage and search the bibliography.

The overall CINDI system uses knowledge bases and expert sub-systems to help the user in the registering and the search processes. CINDI uses some form of standardization of terms. The index generation and maintenance sub-system uses the knowledge and expertise

of the expert cataloguer to help the provider of the resource select correct terms for items such as subject, sub-subject and keywords. Similarly, another expert sub-system is used to help the user in the search for appropriate information resources [?]

The Semantic Header structure

Since the majority of searches begin with a title, name of one of the authors, subject and sub-subject. CINDI made the entry of these elements mandatory in the semantic header [?]. So, in the following lines, the fields of the semantic header, which is based on the SGML mark-up language, will be discussed:

<semhdr>

<title> required </title>

<alt-title> OPTIONAL </alt-title>

<Subject> required: a list each of which includes fields for subject and up to two levels of sub-subject: at least one entry is required </Subject>

<language> OPTIONAL: of the information resource </language>

<char-set> OPTIONAL: character set used </char-set>

<author> required: a list each of which includes role, name, organization, address, etc. of each person/institute responsible for the information resource: at least the name or the organization and address is required </author>

<Keyword> required: a list of keywords </Keyword>

<Dates>

<Created> required: </Created>

<Expiry> OPTIONAL: </Expiry>

<Updated> system generated </Updated> </Dates>

<Version> OPTIONAL: version of the resource </Version>

<Supersedes> OPTIONAL: which version is being replaced </Supersedes>

<Coverage> OPTIONAL: audience, spatial, temporal </Coverage>

<Classification> OPTIONAL: nature (legal, security level etc.) of the resource </Classification>

<Identifier> A list of domains for identifiers and the corresponding values: typical identifiers could be one of more Unique Resource Locator(URL), Call No. for the resource, unique name of the resource (URN), site where the item is to be archived: at least one required </Identifier>

<Abstract> OPTIONAL but recommended </Abstract>

<Annotation> OPTIONAL: </Annotation>

<SysReq> OPTIONAL: list of system requirements for example hardware and software: the component and the corresponding requirements are given </SysReq>

<Source> OPTIONAL: gives the source or related list of resources for each such resource it indicates a relationship and gives an identifier which includes the domain and the corresponding value </Source>

<size> size of the resource in appropriate units (e.g., bytes) </size>

<Cost> OPTIONAL: cost of accessing the resource </Cost>

<control>

<Ac> account number </Ac>

<password> required: encoded password or digital signature of provider of resource for initial entry and subsequent update
</password>

<signature> digital signature of the resource for authentication
</signature> </control>

</semhdr>

What is Information Retrieval?

Information retrieval is concerned with the representation, storage, organization and accessing of information items. Most people are faced with a need for information at some time or other. To facilitate the task of the information user in finding items of interest, libraries and information centers provide a variety of auxiliary aids. Each incoming item is analysed and appropriate descriptions are chosen to reflect the information content of the item. We will be only discussing the automatic information retrieval or as we shall describe it as the automatic document indexing or representation. This index, document's content representation, would then help the user in retrieving this document. Thus, retrieval depends on indexing, that is on some means of indicating what documents are about. Indexing is the basis for retrieving documents that are relevant to the user's need. [?]. The main aim of indexing is to increase precision. One of the main problems is to obtain an accurate representation of that document, which will be stored by our proposed CINDI system. A document representation could for example be a list of extracted words considered to be significant, if not explicitly stated, title, and abstract.

Compact descriptions of a document's index may increase the efficiency of matching and the effectiveness of classifying textual material as relevant. Thus, Document retrieval imposes conflicting normalizing and accurate demands.[?].

History of information retrieval Tests of indexing languages shown that indexing documents by individual terms corresponding to words or word stems produces results at least as good as those produced when indexing by controlled vocabularies. [?].

Luhn used frequency counts of words in the document text to determine which words were sufficiently significant to represent the document. The use of statistical information about distributions of words in documents was further exploited by Maron, Kuhn and Stiles who obtained statistical associations between keywords.

Statistical Document Retrieval methods assign higher numeric weights to terms showing evidence of being good content indicators. The number of occurrences of a term in a document as a whole may be taken into account, when computing the influence a term. Evidence also suggests that combining single terms into compound terms may be useful.[?].

Applying **Bayesian network** to Information Retrieval is a relatively new technology for probabilistic representation and inference. The advantages that Bayesian networks bring to the IR task include an intuitive representation of uncertain relationships and a set of efficient inference algorithms. Robert Fung and Brendan Del Favero have been directed at developing a probabilistic IR architecture that assists users who have stable information needs in routing large amounts of material. Towards these goals, they have developed and implemented a system that allows a user to specify the topics of interest, the quantitative and qualitative relationships among the topics, the document features and the quantitative relationships between these features and the topics. [?].

Algorithms used by the IR community

The main worry of the community is how to select significant words and phrases from a document that will best describe the document or a set of documents[?][?]. Automatic summarization challenges the field in understanding the content of the text at a fairly deep level, being able to ascertain the relative importance of the material and in generating coherent output[?]. Most of the community's was to handle search querying document database in order to get the best and most appropriate responses. They tried to find significant words in different documents and they even tried to attempt the content or meaning of the document. Although attempts have been made to utilize Natural language text condensation approaches [?], they generally required the selection of a narrow domain and the availability of domain knowledge. These shortcomings made it infeasible for generic text condensation tasks. Here are some of the main algorithms that make up the core of our system:

Luhn's ideas

He assumes that frequency data can be used to extract words and sentences to represent a document [?]. Then he ranked the words in the decreasing frequency of occurrence, rank order. He then plotted the graph of frequency related to rank and the curve was similar to the hyperbolic which demonstrates the Zipf's law that states that the product of the frequency of use of words and the rank is approximately constant. He then excludes the non-significant words and the very high frequency words. He used this method also to devise a method for automatic abstracting. He went on to develop a numerical measure of significance for sentences based on the number of significant and non-significant words in each portion of the sentence. Sentences were ranked according to their numerical score and only the highest ones would be included in the abstract.

C.J.van Rijsbergen's attempt

The document's representation that he was aiming is one consisting simply of a list of class names, each name representing a class of words occurring in the total input text [?]. A document will be indexed by a name if one of its significant words occurs as a member of that class. Such system will consist of 3 parts:

1. removal of high frequency words
2. Suffix stripping
3. detecting equivalent stems.

If two words have the same underlying stem then they refer to the same concept and should be indexed as such. It is inevitable that a processing system such as this will produce errors. Fortunately experiments have shown that the error rate tends to be of the order of 5 per cent (Andrews). Lovins using slightly different approach to stemming also quotes errors of the same order of magnitude. The final output would then be a set of classes, one for each stem detected. A class name is assigned to a document if one of its members occurs as significant word in the document.

Limitations of the Traditional Approaches

Traditional approaches to information retrieval use keyword searches and statistical techniques to retrieve relevant documents (e.g., [?] ; [?]). Statistical techniques take advantage of large document collections to automatically identify words that are useful indexing terms. However, word-based techniques have several limitations:

Synonymy Different words and phrases can express the same concept.

Polysemy Words can have multiple meanings [?].

Phrases Some words are good indexing terms only in specific phrases.

Local Context Some words and phrases are good indexing terms only in specific local contexts.

Global Context Some documents do not contain any words or phrases that are good indexing terms.

Enhancing the document representation

The conventional wisdom is that the keyword-type systems, where the information items are represented by sets of manually or automatically chosen index terms, have run their course: most keywords are believed to be ambiguous and often poorly represented by small collections of individual terms [?]. So It is widely believed that the keyword approach is not adequate for text content representation in information retrieval. By extension, the identification of text content by weighted term sets may also be unacceptable. Quoting from Blair: [?]

' No number of brute linguistic facts (word statistics) can be added up to give us the meaning of a text, where the meaning of a text would include such things as its subject, intellectual content context, use, purpose, or links to other documents.'

The available experimental evidence indicates that the use of abstracts in addition to titles brings substantial advantages in retrieval effectiveness. However the additional utilization of full texts of the documents appear to produce very little improvement over titles and abstracts alone in most subject areas [?].

Developments in Automatic Text Retrieval

In conventional information retrieval, the stored records are normally identified by sets of keywords or phrases known as index terms. The terms characterizing the stored texts may be assigned manually by trained personnel; alternatively, automatic indexing methods may be used to handle the term assignment automatically.

Alternative Retrieval Models

The Vector Model

In the vector space model, the documents are identified by sets of attributes, or terms. Instead of assuming that all terms are equally important, the system uses term weighting. The vector processing model offers simple, parallel treatments for both queries and documents. Extensions to the vector and Boolean models have been proposed notably including a generalized vector space model based on orthogonal vector space. Another common retrieval model is the extended Boolean system which accommodates term weights assigned to both query and document terms as well as strictness indicators. The extended system thus covers vector processing, Boolean, and fuzzy set retrieval in a common framework, and it produces vast improved retrieval performance over simple Boolean operations.

The Probabilistic Model

The probabilistic retrieval model differs from those previous discussed in that it represents an attempt to set the retrieval problem on firm theoretical foundation. In the classical probabilistic models, the needed term probability are estimated by accumulating a number of user queries containing a term and determining the proportion of times a document is found relevant to the respective queries. Alternatively, a fixed query is considered and an attempt is made to determine the probability that an arbitrary document containing a query term will be judged relevant.

The probabilistic retrieval approach accommodates a large number of different phenomena about terms and documents as part of the probabilistic estimation process, including term co-occurrence information, term relationships derived from dictionaries and thesauruses, and prior knowledge about the occurrence distributions of terms.

Progress of Natural Language Processing in IR

The goal of an information retrieval system is to locate relevant documents in response to a user's query. Documents are typically retrieved as a ranked list, where the ranking is based on estimations of relevance [?]. Lexical ambiguity is a pervasive problem in natural language processing, and previous literature divides it into two types: syntactic and semantic [?].

To process a sentence of a language, the tokens must be isolated and identified. For NLP, lexical processing operates at the single word level and involves identifying words and determining their grammatical classes. This usually consists of looking up a dictionary or lexicon, essentially a list of known words and their legitimate morphological variants. Ideally, lexical processing determines one base form for each word. Research into syntactic analysis of natural language has been concerned with the construction of wide coverage grammars[?].

The main sources of structural syntactic ambiguity in English are due to the attachment of prepositional phrases, the construction of nominal compounds and the scope of

coordination and conjunction. On another level, there are semantic constraints on what should make semantically sensible natural language statements. This is because a sentence may have a number of semantic interpretations, possibly arising from a number of syntactic interpretations, and they should be eliminated.

The difficulty with semantic processing is that all the properties of all objects must be known. Thus, a huge knowledge base should be built to support the former problem.

Detecting anaphora and resolving the reference would improve understanding of a text but even detecting anaphora is difficult as there are no indicator phrases or terms. Liddy [?] lists almost 150 words which could be indicators of an anaphoric construct; however, the problem of reliably resolving anaphora still remains. However, fully-fledged NLP is being used in information retrieval and has led to the emergence of the application known as conceptual information retrieval. In it, the user requests information and is given the information directly, not just a reference to where it may be found.

Lexical level language processing in information retrieval

The simplest applications of NLP to information retrieval have been at the word level by indexing based on some normalised or derived form of individual words occurring in the input. An alternative to the popular stemming and conflation algorithms is to determine the base forms of words from a lexicon lookup. Building such a lexicon is expensive however and has only given marginal improvements over mechanical stemming and for those reasons the idea has never really been pursued. However, lexical level language analysis has had a surge of interest recently with the increased availability of machine-readable dictionaries (MRDs). Its obvious use is to index by word senses rather than by word base forms. In information retrieval experiment, indexing by word senses using MRDs initially gave disappointing results in terms of retrieval effectiveness. Because of this, it is now believed by researchers that it may not be necessary to determine the single correct sense of a word but sufficient to rule out unlikely senses and weight likely senses highly. Krovetz and Croft stressed the importance of the word senses, that will provide a significant separation between relevant and nonrelevant documents [?]. They also mentioned word ambiguity and the use of related or synonym words as two problems with using words to represent the content of a document.

Syntactic level language processing in information retrieval

NLP techniques have been used to help index texts by elements more complex than word forms. Syntactic analysis can be used to analyse text in order to determine the boundaries of noun phrases which could then be used as internal representations.

Indexing texts on a noun phrase basis using NLP techniques was done in the IOTA system [?], but one of the major problems of indexing by noun phrase units is the variety of ways of representing a complex concept in natural language. To address the issue of ambiguity in syntactic analysis of texts for indexing purposes there have been three approaches tried;

ignore ambiguity, normalise the identified phrases or index by structures which incorporate the ambiguities.

Ignoring the ambiguity allows texts to be indexed by phrases taken directly from the text. A large amount of work in this area has been done by Salton and others at Cornell University.

Normalising indexing phrases from texts and from queries into some standard form is being taken by the CLARIT project at Carnegie Mellon university. A first order thesaurus for a domain, essentially a phrase list, is first generated automatically. Input texts are parsed and candidate noun phrases are identified. These are then compared to the thesaurus and classified as either exact (identical to some phrase in the list), general (terms in the list are constituents of those in the list), or novel (new terms new not on the list). This approach always uses terms from the list as the indexing units and thus always yields the same syntactic form for a concept which could have been expressed in a number of different ways [?].

Encoding the ambiguity in some structure and allowing the retrieval operation to make allowances for this, handles the syntactic ambiguity in syntactically based indexing.

Symantic level language processing in information retrieval

Any piece of text which contains information essentially consists of a description of objects and actions on those objects. There have been a number of conceptual information retrieval systems described in the literature. These include SCISOR, RESEARCHER and OpEd.

An Alternative To Traditional IR systems

There has been a great deal of work recently on knowledge-based information retrieval systems ([?];[?];[?];[?]). Knowledge based IR systems rely on an explicit knowledge base, such as rule base [?], semantic network [?], patterns [?], or case frames [?].

The SMART Retrieval System (Salton)

The SMART system is a sophisticated text retrieval tool based on storing all information terms in a vector of terms. In principle, the terms might be chosen from a controlled vocabulary list or a thesaurus.[?]. After performing the SMART experiments [?], the following conclusions were drawn:

Document length Document abstracts are more effective for automatic content analysis purposes than are document titles. Improvements appear possible if the abstracts were replaced by larger text portions, but still the improvement is not large enough to conclude that full text processing is superior to abstract processing.

Synonym Recognition Dictionaries providing synonym recognition produce statistically significant improvements in retrieval effectiveness compared with word stem matching procedures.

Order of merit Most effective: Abstract processing with phrase and synonym recognition. Next most effective: Weighted word stem matching and statistical word associations using abstracts for analysis purposes. Less effective: logical word stem matching disregarding term weights. Least effective: Title processing using only document titles for analysis purposes.

And some other points, but the main interesting thing to us are already mentioned.

Oracle ConText-Text Management System

One of the most critical challenges facing business today is managing information. Unstructured, primarily textual, becomes trapped as essentially dead fields in traditional databases. As a result, information that resides as text documents, manuals, reports, e-mail and faxes has been largely inaccessible to the corporate decision makers who need it most[?][?].

The Oracle7 ConText option is a fully integrated text management solution that enables users to process text-based information as quickly and easily as relational data. Oracle context analyzes the contents and understands the structure of the English text it reads . The Oracle Release 7.3 Context option consists of two separate, yet closely interrelated functions: a text management architecture contained entirely within Oracle7 and a text retrieval feature which uses natural language processing technology to identify themes and content in text. It is also capable of analyzing the thematic content stored text and generating automatic summaries. Context 's core is its syntax parser, a sophisticated and robust collection of lexical attributes (dictionary information) and parsing rules. The parser consists of five data collection layers. These layers are :

1. Syntax , theme and grammar which analyze documents at the sentence level and produce a structural and thematic representation of sentences.

2. Connotation and discourse which analyze documents at greater-than- sentence level and produce a representation of discourse dynamics.

In other words, these modules track the rise and decay of themes/subjects, and assign a subject matter description of sentences, paragraphs, and documents. In addition, Context's concept processing Engine represents an imposing English language knowledge base.

How Does Context Manage Text?

Understanding Text Systems that rely on simple word recognition and repetition often miss the actual meaning of a document and as a result produce an enormous amount of irrelevant output. By breaking the text down into its constituent grammatical elements and determining how these elements contribute to the overall meaning. ConText works to understand the text it processes. It then uses this knowledge to produce a database index which can identify the development of key themes and determines their relative prominence. Unlike other products that simply count words or use a hierarchical thesaurus to determine the main theme of a document, ConText parses every sentence in a document to determine the relative weight of the different themes.

The ConText Lexicon The heart of this text retrieval system is the Oracle ConText Lexicon, a vast dictionary of over 1,00,000 words and phrases as well as the linguistic rules that bind them into thematic units. The lexicon is designed to recognize the vocabulary used in over 1,00 industries and can be augmented by user dictionaries.

Text Reduction Users can create automatic summaries, outlines or specialized views of complete documents. The Context option provides automatic text reduction, which creates summaries conveying the main ideas and concepts of full documents. Text reduction condenses large documents to a manageable level.

Text Classification In Addition to text reduction, ConText contains a powerful text classification feature which categorizes documents according to linguistically identified themes rather than word frequency and statistics.

Advanced Text Retrieval Technology used by Oracle ConText

Text retrieval features include:

- Exact word/Phrase searching
- Wild card searching
- Multilingual Stemming
- Term Weighting
- Proximity searching

- Thesaurus support
- Relevancy ranking
- Fuzzy Searching
- Boolean logic
- Stop Lists

ConText's General System Description

Context's syntax parser processes each sentence in a document individually. It generates an attribute set representing the syntactic, grammatic, and thematic makeup of the sentences.

Document analysis may continue to the concept Processing Engine. That engine assigns a specific connotative sense to each main theme word. It determines which are the best interpretations of a word based on the concordance of connotations in its generalized structure. The connotation structure is referenced by each word, with each structure containing all possible connotations for the word.

ConText's main procedures

1. Sentece Level Processing
 - (a) Preparsing Stage
 - Morphological Analysis
 - Grouping words into Syntactic units.
 - Parsing order for rule sets.
 - (b) Parsing
 - Parsing Theory and practice
 - Lexical Bindings
 - Major Clause Types
 - Syntactic Descriptions
 - (c) Rule-Based Parsing
 - Normalization Routine
 - Main Syntax Parsing Routines
 - Data Processed
 - Syntax Parsing Rules
 - Phrase Formation
 - Logic Change rules
 - Normalization Rules
2. Concept Processing

- (a) Content Analysis
- (b) Content Reduction system
- (c) Discourse Tracking
 - Theme Calculations
 - Theme Types
- (d) Connotation Structure

Nordic WAIS/World Wide Web Project

Related work done on Improving Resource Discovery and Retrieval on the Internet - The Nordic WAIS/World Wide Web Project [?]. One of the most important developments of research libraries is the rapid growth of the internet and its electronic information resources. The Nordic project used an automatic classification according to UDC, English medium edition. The dynamic nature of the information sources on the net makes it necessary to have automatic tools to do the indexing and classification of material. The algorithm used is as follows:

1. From the different fields of the selected document, words are extracted into number of groups:
 - words from the description field
 - words from the subject field
 - words from the keyword-list field
 - words from the description field marked as keywords together with the name of the database.
2. A list of suggested classifications is constructed by comparing words from these groups with UDC's vocabulary. When a match between the vocabulary and a word is found the corresponding classification is added (restricted to the top 2 levels) to the list of suggested classifications with a weight that depends on which group the matching word originates from. That is, keywords in the subject field have higher weights than ordinary words in the description field.
3. From the list of suggested classifications we decide the final classifications based on the accumulated weights for each proposed classification.

They also suggest that their methodology can be used with other classification schemes like Library of Congress to produce different views of the subject trees.

Overview of Harvest's Essence

Essence's main objective is to extract indexing information from an input document. Content indexing requires type-specific processing to extract information effectively. So, by

exploiting the semantics of common file types, essence generates compact yet representative file summaries that can be used to improve both browsing and indexing in resource discovery systems [?].

Essence decomposes the information extraction problem into 4 components that are independent of how data are stored , updated or exported:

1. The type recognition step that uses various methods to determine a file's type
2. The presentation unnesting step that transforms nested files into an unnested format.
3. Candidate selection step, selects which objects are to be summarized.
4. The summarizing step, which applies a type specific extraction procedure to each selected object.

Steps used by our system Only steps 1 and 4 will be described since they are the only 2 related to the software that we are trying to develop.

Type Recognition

Essence recognizes file types using a combination of file and site naming conventions, content testing, and user defined methods. It can also use explicit file typing information for environments that contain such information. The main 2 type recognition steps are:

- Naming conventions and heuristics.
- Examining file contents in determining the file types.

Essence Summarizing Subsystems

Each summarizer is associated with a specific file type and understands the type well enough to extract summary information from the file.

Type	Sumarizer Function
Audio	Extract file name
Bibliographic	Extract author and titles
Binary	Extract meaningful strings and manual page summary
C, CHeader	Extract procedure names, included file names, and comments
Dvi	Invoke the Text summarizer on extracted ASCII text
FAQ, FullText, README	Extract all words in file
Framemaker	Up-convert to SGML and pass through SGML summarizer
Font	Extract comments
HTML	Extract anchors, hypertext links, and selected fields
LaTeX	Parse selected LaTeX fields (author, title, etc.)
Mail	Extract certain header fields
Makefile	Extract comments and target names
ManPage	Extract synopsis, author, title, etc., based on “-man” macros
News	Extract certain header fields
Object	Extract symbol table
Patch	Extract patched file names
Perl	Extract procedure names and comments
PostScript	Extract text in word processorspecific fashion, and pass through Text summarizer.
RCS, SCCS	Extract revision control summary
RTF	Up-convert to SGML and pass through SGML summarizer
SGML	Extract fields named in extraction table
ShellScript	Extract comments
SourceDistribution	Extract full text of README file and comments from Makefile and source code files, and summarize any manual pages
SymbolicLink	Extract file name, owner, and date created
Tex	Invoke the Text summarizer on extracted ASCII text
Text	Extract first 100 lines plus first sentence of each remaining paragraph
Troff	Extract author, title, etc., based on “-man”, “-ms”, “-me” macro packages, or extract section headers and topic sentences.
Unrecognized	Extract file name, owner, and date created.

Table 1: The Summarizers’ functions for each document

Automatic Sentence Extraction used in Title and Abstract selection

Emphasis was placed on the production of indicative abstracts (i.e. abstracts that allow a searcher to screen the body of the literature to decide which document deserves detailed attention. It was hypothesized that an extract of a document, that is a selection of significant sentences can serve as an abstract.

Text processing methods based on a determination of term or sentence importance have been used not only for indexing but also for automatic abstracting purposes [?]. Ideally, given a document represented as natural language text, one would like to construct a coherent well written abstract that informs the readers of the contents of the original, or at least indicates whether the full version may be of interest to the reader. In fact, most procedures carry out an extraction process in which the abstract is defined as simply as a small set of sentences pulled from the original, which are deemed to be important for purposes of content representation. The extracting methods used over the years all start with a calculation of word and sentence significance, similar in spirit to the computation of the term weights. Criteria for the selection of important terms may be positional, the place where the term is located in the document, semantic, pragmatic, such a system which would consider proper names as highly significant. Furthermore, statistical weights. Since the frequency criteria is not that reliable additional criteria should be tackled such as contextual inference (the word location or the presence of cue words), syntactic coherence criteria [?];[?];[?];[?];[?];[?];[?];[?].

Text Classification or Categorization

An important step in building up the document database of a full text retrieval system is to classify each document under one or more classes according to the topical domains that the document discusses. This is commonly referred to as classification. Automatic classification has two major components:[?]

1. The classification scheme, which defines the available classes under which a document can be classified and their inter relationships.
2. The classification algorithm, which defines the rules and procedures for assigning one or more classes.

Wong, Kan and Young presented an automatic classification approach called ACTION. The key idea behind it is a scheme for measuring the significance of each keyword in a given document. That scheme not only takes into account the occurrence frequency of a keyword, but also the logical relationship between the available classes.[?]

Text categorization systems assign predefined category labels to texts. For example, a text categorization system for computer science might use categories such as operating systems, programming languages, AI or information retrieval [?]. Text Categorization are typically applied to static databases [?].

The relevancy signatures algorithm uses linguistic phrases, the augmented relevancy signatures algorithm uses phrases and local context, and the case-based text classification algorithm uses larger pieces of context. These 3 algorithms were evaluated and the results suggested that information extraction techniques can support high-precision text classification and, in general, using more extracted information improves performance [?].

As we mentioned earlier, there has been some information retrieval based on an explicit knowledge base. Other approaches are knowledge based relying on a domain-specific dictionary to drive information extraction system [?]. Thus we classify or categorize the text after viewing the knowledge base that relates subjects to their keywords. It seems reasonable to believe that we could produce accurate classifications if we could actually understand the documents; however, it is an expensive endeavor that can strain computational resources. Thus, some researches have turned their attention to information extraction that extracts specific types of information from a document [?]. For example, in the domain of terrorism, an information extraction system might extract the names of all perpetrators, victims and weapons that were involved in a terrorist attack. The main advantage of this task is that portions of a text that are not relevant to the domain can be effectively ignored. And since the system is only concerned with the domain-specific portions of the text, some of the most difficult problems in NLP are simplified. As a result, information extraction is a practical and feasible technology that has achieved success in the last few years [?];[?];[?];[?].

While extracting the information in our system the domain is not known, thus at first the domain would be all subjects found in LCSH, for instance. Since each subject has its own specific keywords, we try to select each and every keyword that is found in both the text and in our knowledge-base. Thus ignoring terms that will be misleading and time consuming. Thus to classify a text under a certain subject, all the relevant keywords should be found in the text. The concept of relevancy is discussed in terms of text as a whole [?], while in our system we discuss the individual terms as relevant to a subject or not. Although keywords are useful as a first approximation for discriminating between relevant and irrelevant texts, they do not capture natural language context surrounding a word. But, we can truly believe that keywords found in both our knowledge base and text can lead to the relevant subject for that particular text. For example, if an author wants to describe some specific things or issues in computer science, then he should mention the words there, these words would lead us to that particular subject but not any other. Thus in our subject selection, we select the subject that has all its relevant or significant words found in the text.

Edmundson describes new methods of automatically extracting documents for screening purposes. His method describes the sentence significance and high content words previously described and three additional components: pragmatic words (cue words), title and heading words, and the structural indicators (sentence location) [?]. An attempt was made to classify eligible sentences as to qualitative degree of extract-worthiness. However, in practice it did not prove satisfactory for sentence selection. The principles he followed in devising the guide to the development of automatic extracting methods are:

1. Detect and use all content and format clues to the relative importance of sentences that were originally provided by the author, editor or printer.
2. Employ a system of reward weights for desired sentences and penalty for undesired sentences.
3. Employ a system of parameters that can be varied to permit different specifications for extracts.
4. Employ a method that is a function of several linguistic factors (syntactic, semantic, statistical locational, etc.).

Thus, the four basic methods he used in his automatic extracting system are the Cue, Key, Title and location methods.

Clearly there are extracting clues that have not been exploited-in captions of figures and tables, in footnotes and references.

Automatic News Extraction System

The ANES initial technical approach was similar to that taken by Johnson et al[?]. In addition to constructing summaries by extracting whole sentences from the source documents, Johnson used as an indicator of key content such phrases as the objective of this study... The ANES process of summary generation is made up of four major constituents: statistical corpus analysis, signature word selection by applying a term frequency and the inverted document frequency model , sentence weighting , which is computed by summing the weights of the individual signature words present in the sentence, and sentence selection[?].

The Automatic Semantic Header Generator, ASHG

We are trying to build an automatic semantic header generator for the CINDI system, that will save the provider of the primary source some time in entering the index manually. It lacks something similar to What Harvest's Essence provides. In addition to what Essence provides, there is still a need to develop a system that will extract almost all the information from a document to build the semantic header for it.

The main aim of the proposed ASHG system is to save the writer or author some work in filling out the semantic header. Thus, it should extract the required information from the submitted document and should display the result for the user to either modify or confirm. So this system is a combination of the type recognition step, Essence summarizers and some additional modifications to suit the required meta-data of the document. So to store a Semantic header into the database, the document should be submitted to ASHG. Once it is submitted, the document type is identified, and if it is one of the ASHG known types, the appropriate extractor is applied.

This automatic semantic header generator makes use of 2 databases: the subject hierarchy database and the keyword-subject database, which play a major role in identifying the right subject(s) for the document, that is submitted to the ASHG.

How does the ASHG handles the problem?

Here is a brief outline of how the system handles the problem:

1. Document type recognition
2. User validation
3. Applying the corresponding extractor or summarizer on the document, and then the user validates the resulting semantic header.

Type Recognition

The main 2 type recognition steps are:

1. Naming conventions and heuristics.
2. Examining file contents in determining the file types.

Upon submitting a document or the primary resource, the system will first try to distinguish its type through the name conventions and heuristics. If it fails the system will then examine its contents, if it fails to distinguish its type, it asks the user to enter its type; however if it does not, it asks the user to verify the result. Here is the algorithm which the system follows in recognising the file type.

```
if document.extension == .html or .HTML or .htm
then the file is an html file
    call user-verify
else if document.extension == .tex or .TEX
then the file is a latex file
    call user-verify
else if document.extension == .doc, or .txt or .info or .ascii
then the file is a text file
    call user-verify
else call bycontent
```

In User-verify, the user should either confirm or reject the result.
if he confirms that the document's type is as recognised then
call the right summariser
else call user-type

In User-type, the user enters the type of the primary source,
if the type is handled by the system then
call the right summariser
else call unknown

In unknown, the size of the file is extracted.

In bycontent, the semantics of the type structure is exploited in trying
to recognise the file type.
If the file contents match the html file semantics then
then the file is of html type
 call user-verify
else if the file contents match the latex semantics then
then the file is of latex type
 call user-verify
else call user-type

-bb-error = =

Figure 1: Document Type Recognition

ASHG's Extractors

After identifying the document's type, the corresponding extractor is applied to the document. Thus aiming to retrieve the corresponding meta-information to build its Semantic Header. This system is aware of only the following 3 types: HTML, Latex and Text. The system exploits their file semantics while extracting the needed index.

HTML-extractor

An HTML document is made up of 2 parts: the Head and the body. The Head provides information about the document. It contains the title, META, which contains the meta-information of the document, LINK, where we might have the author's address. The body contains several information, but we are only interested in the headings, definition terms, address and the mailto.

The objects that the system try to extract from an HTML document are: the title, subject, language (English), author(s), character set, keywords, dates (Created, Expiry, Updated), version number, size of the file, and finally the abstract.

1. The title could be easily extracted since it is tagged.
2. The subject is handled differently. In extracting the subject we use some expert systems and some knowledge base systems that will be described later. But to brief it up, the subject will be selected only after selecting the keywords from the documents. And these keywords will lead the system in identifying the right subject by using the Keyword-Subject database (discussed later).
3. Author(s) could be extracted from the META tag. It could be found following the content word. We could also extract the e-mail of the author following the mailto word or we could look for the pattern '@' which is the worldwide usable symbol for an e-mail.
4. Keywords can be either stated explicitly by the writer of the document or they should be extracted by the system itself. If they are stated by the writer, they are found in the META tag.. However, if they are not, the system will extract almost all the words that fall between the following tags: b.../b, dt, h1...h6, i, strong, tt, and all the href links.
5. Abstract can be either extracted from the META tag or we can extract the paragraph that follows the word abstract.
6. The Address can be extracted from the address tag.
7. The Dates are extracted by looking inside the META tag.

Latex-extractor

A Latex document is also tagged which will make it easier to extract the needed information. So the system will try to extract the title, the subject, language (which is English), character set, the author, the dates, the abstract and finally the keywords.

1. The title in latex could be extracted from what is between the braces in `title{ }`.
2. The subject extraction will depend on the keyword selection. Thus after selecting the keywords, the keyword subject database will be visited in order to select the most appropriate subject. This process will be described later in more detail in other sections of this paper.
3. The author(s) are extracted from what is between the braces in `author{.....}` . We might have more than one author and these are separated by the `and` . The address is extracted from within the author, but it is separated by the `.` . Similarly, we extract the e-mail by looking for the `@` symbol.
4. The dates are extracted from what is between the braces in `date{.....}` .
5. The abstract is extracted from either the `begin{abstract}...`
`end{abstract}` or the `paragraph{abstract}`.
6. The keywords are selected from the following tagged fields:
`chapter{...}`,
`section{.....}` ,
`subsection{.....}` ,
`subsubsection{.....}` ,
`index{.....}` ,
`part{.....}` ,
`footnote{.....}` ,
`item{.....}` , and many more.

Text-extractor

Extracting information from unstructured text is a challenging task for processing as well as a key problem. However, ASHG will try to extract the title, the abstract, the keywords, and finally the author.

1. The title; we first look for the pattern `title` and what follows if found will be assumed to be the title. However, if that pattern is not found, we then select the first sentence of the document.
2. The abstract: We look for the pattern `abstract` and if found we select everything till we meet a tab, otherwise, we will be obliged to select the first paragraph.

3. The keywords: We then try to select the first sentences of all the paragraphs. These sentences will be used as input to the word extractor.
4. The author(s): We look for the patterns: edited by, written by, author in selecting the author. We look in the first couple of sentences or the last couple of sentences. The e-mail will be extracted by the looking for the @ symbol.

All these extracted meta-information will be stored in a semantic header and will be displayed to the source provider for him to modify or validate.

How are keywords extracted?

The keywords are either explicitly stated in the document or implicitly stated. Having them explicitly stated, the system can easily extract them. However, a lot of work should be done in extracting keywords that are not in the primary source. The system, should then look for words in the title, abstract, and other tagged words. The way extracting the significant words follows this algorithm:-

1. Extract the title, abstract and other tagged words.
2. The above fields are then passed through an engine, that will filter out the noise English words that constitutes around 30 to 50document.
3. Having done this, we will pass the remaining words into the stem process that will remove all suffixes from the stem root of a term.
4. The words or terms found in the keyword database are left out, and the rest are just dropped.
5. We then assign weights to every remaining word. These weights are as follows:
 - (a) if the word appears in the abstract, its weight is 4
 - (b) if the word appears in the title, its weight is 3
 - (c) if the word appears in the other tagged words, its weight is 2

So these weights constitutes the importance of a word. Usually words found in the abstract are the most important words, cause in the abstract the author tries to convey his idea.

6. After assigning the weights to the words, we then extract the words having the highest weight (importance). So if a word is found in abstract title, and the other tagged words, it would be selected first. The selected words should have a weight greater or equal to 3.

Thus, words found in the abstract are more significant and they convey the idea of the article more than any words found in other locations [?]. Because the terms are not equally useful for content representation, it is important to introduce a term weighting system that assigns high weights for important terms and low weight for the less important. [?].

How Are the 3 level subject terms selected?

Selecting the keywords is one thing and selecting the subjects is totally different thing. To select a subject, we need to scan all the words found in the title , abstract, keywords (if explicitly stated) and other tagged words. However if the keywords are not explicitly stated, the system would use the already selected keywords. So 2 algorithms are to be followed, one if the keywords are explicitly stated and the other is when they are not.

The initial first steps

1. We will be using the same files used in keyword extraction for the title, abstract , and other words.
2. If keywords are explicitly stated, we pass them through the English noise words filter.

Algorithm for explicitly stated Keywords Having the keywords, title words, abstract words and other tagged words, will help or lead us to select the most appropriate subjects for a given document. The following algorithm is followed:-

1. For each word found in both the keyword database and the other fields, we trace the attached list of subjects (list of level0, level1 and level2), corresponding to each word and puts into another list.
2. Weights are also assigned to the subjects The weight for a subject is given according to where the word corresponding to it is found. So if it is in Keyword list it is given weight = 4 if it is in abstract it is given weight = 3 if it is in title it is given weight = 2 and finally if it is in other tagged words, it is given weight = 1.
3. We then extract subjects having weights greater or equal to the maximum weight found in a the subject list - 4.
4. After building the list for the 3 level subjects, we then select the subjects bottom-up. This means that we first try to select the level-2 subjects, move on to select level-1 subjects and then finally level-0(general subjects).
5. The outcome of step 4 will give the system several options, but still more work should be done in extracting the subjects.
6. The 3 level subjects for a document should have all its significant words in one of the title, abstract, keywords, or other tagged words.

Algorithm for implicitly extracted Keywords Having already extracted the keywords, we then follow almost the same algorithm except now that we do not have to look into the abstract, title and other tagged words. We follow this algorithm:-

1. For each word in the implicit keyword list, we construct the subject 3 level lists, by looking into the keyword-subject database.
2. Each subject is given a weight which is the same of that keyword.
3. we then follow the steps 3 to 6 of former algorithm.

Information Retrieval in Building the Semantic Header

Building an index for a document will help in retrieving it in later searches. To build an index, the system has to extract or retrieve information and store it in the index structure called the semantic header. Thus upon submitting a document the first step is to know its type and according to its type we apply the appropriate summarizer. The main aim of the summarizer is to extract the fields like title, abstract, author, keywords if explicitly stated, dates, address, etc... Upon extracting these fields, additional work is done to extract the keywords form the document if not explicitly stated and then the 3 level subjects. Here is the algorithm:-

1. Document type recognition
2. Apply the appropriate summarizer
3. If the keywords are not explicitly stated then apply the implicit keyword extraction algorithm.
4. Extract the 3 level-subjects that will best describe the document.
5. fill the index with this extracted information.

Basic steps used in our information retrieval

Steps used in extracting implicit keywords

English noise words are extracted out These words constitute usually around 30 to 50 per-cent of a document, and is called stop list by the information retrieval community. (word-extract2)

Filtering out more insignificant words These words are not found in our own keyword database.

Term Stemming Each word or term will pass this process that will remove all suffixes from the stem root of that term. So terms like cycled, cyclist, cycling and cycles will all be stemmed to the root term, cycle.

Assigning weights Then the words are assigned weights that will reflect their significance. This has to do with where that word appears in the document (title, abstract or other tagged words).

Selecting the highest weights ' words The words having the highest weights are the best candidates for being keywords of that document.

Steps used in selecting the 3 level subject hierarchy

Extracting subjects These subjects are extracted by retrieving the subjects linked to the significant words found in the title, abstract, keywords and other tagged words and in the keyword-subject database.

Assigning weights to the subjects After having 3 lists of subjects that correspond to level0, level1 and level2, we assign to these subject weights that will reflect their importance. This weight is given according to where the word corresponding to that subject was found in the document.

Filtering out some subjects After assigning weights to the subjects, we then extract the ones having the highest weights.

Bottom-Up selection We trace level-2 subject list and from it we select level-1 subjects, and then from level-1 to level-0.

Final selection The final step is to make sure that all the significant words found in the 3 level subjects are in one of the title, abstract, keyword and other tagged words.

In other papers

They even sometimes discuss selecting or building phrases having the selected words. In some other paper they mention the domain to be known. Then they try to select the words that are relevant to the domain.

Enhancement added to ASHG

There are a couple of interesting points like the Suffix stripping and the frequency that can be added to our system. The frequency part might help but it stills need to be studied carefully cause we until this day assign weights corresponding the the place of the word in the document. We might also add the abstract implicit adding to our system, by selecting the phrases where the most significant words appear. We might also add a synonym database or dictionary for the stem terms and create classes of them. So here is the algorithm that could be followed:

Enhancement on keyword extraction

- If keywords are not explicitly stated then we use the stop list to remove the noise words. remove the words not found in our keyword database. Remove the suffix and apply the word stemming. apply the synonym (if applicable) the resultant words would be selected according to their frequency and the place where they appear in the document.

Enhancement on selecting the subject

- For each word found in the subject we strip the suffix, stem it and compare it with the words thus, we add the strip and the stemming to what we already had.

Adding the implicit Abstracting

- Having selected the final words, we then select the phrases that contain the highest or most significant words. We also give priority to phrases found at the beginning and at the end of a document.

Stemming Process

Stemming consists of processing a word so that only its stem or root form is left. In our system, all the keywords stored will be in the stemmed form. It is a process of removing all suffixes from the stem root of a term.

Suppose the word impressionists is in a document, our primary source. Without stemming, this would match only the keyword, assuming it is in our keyword database, impressionists and not the singular form. Now suppose that the word impressionist was in our keyword database, then that document will miss that term and will not have it as a keyword. Following stemming, documents having the word impressionistic and impressionism will match the root term that is found in our keyword database.

Some stemming algorithm, Okapi for instance, used both weak stemming to remove plural endings and other grammatical suffixes like *ing* and *ed*, and strong stemming to remove derivational suffixes like *ent*, *ence* and *ision*.

Other stemming algorithms used truncation to find the root term. So all words sharing that same root will be selected; however, one big problem with right hand truncation is that more words will indiscriminately match that root keyword. QPATUS helps to avoid extraneous right hand truncation by automatically performing a process called stemming. First it evaluates the terms for common suffixes that indicate plurality, verb tense, etc. If it discovers these suffixes, it will strip the suffix to obtain the root word. Next, it takes the root form and using sophisticated linguistic rules, creates a set of word variants.

Two types of stemming are available in various versions of WAIS: Porter and Plural. Plural stemming attempts to identify and index the singular form of a term. Porter stemming attempts to identify and index the word *stem*. If a word and its stem are different, only the word stem is indexed.

Search Statement	No Stemming	No Stemming	Porter Stemming	Plural Stemming	Expected Results
play	play	play	play plays	play plays	play
plays	plays	plays	play plays	play plays	plays
play*	play	play plays player playful playground	player playful playground	play plays player playful playground	play plays player playful playground
plai	---	---	play plays	---	---
plai*	plain plainly	plain plainly plains plaintiff plait	play plays plain plains plainly plaintiff plait	plain plainly plains plaintiff plait	plain plainly plains plaintiff plait
plain	plain	plain	plain plains	plain plains	plain

plain*	plain	plain	plain	plain	plain
	plainly	plainly	plainly	plainly	plainly
		plains	plains	plains	plains
		plaintiff	plaintiff	plaintiff	plaintiff
pla*	play	play	play	play	play
	plain	plays	plays	plays	plays
	plainly	player	player	player	player
	plains	playful	playful	playful	playful
	plaintiff	playground	playground	playground	playground
	plait	plain	plain	plain	plain
		plainly	plainly	plainly	plainly
		plains	plains	plains	plains
		plaintiff	plaintiff	plaintiff	plaintiff
		plait	plait	plait	plait

Stemming algorithm followed by ASHG

We have mainly used the `spell` unix command in our system in extracting the root of a word. the `spell` command collects words from an input file and looks them up in a spelling list. Words that neither occur among nor are derivable (by applying certain inflections, prefixes, and/or suffixes) from words in the spelling list are printed on the standard output. Two options were used in our system:

1. `-v` All words not literally in the spelling list are printed, and plausible derivations from the words in the spelling list are indicated.
2. `-x` Every plausible stem is displayed, one per line, with `=` preceding each word.

The stemming algorithm followed in our Keyword Subject database building is as follows:

1. Sort the words found in the 3 level subject by using the `sort` unix command.
2. Make Filter out duplicate words by using the `uniq` unix command.
3. Change the words to their lower case form.
4. Apply the `spell` command with the `-x` option. Thus all the plausible stems are stored an in output file.
5. Apply the `spell` command with the `-v` option. Thus all words not found in the spelling list are stored.
6. Create a file which contains the words found in step 4 but not in step 5.

7. Apply the *spell* command with the *-v* option to each word found in the file that resulted from the previous step. If the resulting output is empty, this means that this root word is found.
8. Apply the same set of steps on each subject and check if it contains a word that has the root already found. If it does, relate that subject with the root of the word.

Comparing Porter with our algorithm

Words	Porter's algorithm	ASHG algorithm
Adventure	adventur	adventure
games	game	game
Computer	comput	compute
aided	aid	aide
engineering	engine	engineer
Transcription	transcript	transcription
Algorithms	algorithm	algorithm
Animation	animat	animate
construction	construct	construct
industries	industry	industry
Industrial	industry	industry
analogies	analogy	analogy
Mathematical	mathemat	mathematic
models	model	model
Simulation	simul	simulate
Civilization	civil	civil
COMPLEXITY	complex	complex
assisted	assist	assist
Authoring	author	author
graphics	graphic	graphic
programmers	programm	programmer
programming	program	programming
Computerized	computer	compute
Performance	perform	performance
Coding	code	code
Utilities	utility	utility
Optimization	optimiz	optimize
Combinatorics	combinator	combinatoric
Representation	represent	presentation

Table 2: The word stem results

-bb-error = =

Figure 2: New Enhanced Skelaton for Title, Abstract and Keyword Extraction

Rules used in Extraction

The automatic semantic header generator needs to extract the keywords, title, abstract and the 3-level subjects. Keywords might be either explicitly stated or they have to be implicitly extracted. Similarly for both the title and abstract. The rules that follow will govern the way our ASHG in extracting them.

1. $\langle \textit{Keywords} \rangle$:- Explicitly-Stated-Keywords
2. $\langle \textit{Keywords} \rangle$:- $\langle \textit{Implicitly-Stated-Keywords} \rangle$
3. $\langle \textit{Implicitly-Stated-Keywords} \rangle$:- $\langle \textit>Title} \rangle$ or $\langle \textit{Abstract} \rangle$ or $\langle \textit{Other-words} \rangle$
4. $\langle \textit>Title} \rangle$:- Explicitly-Stated-Title
5. $\langle \textit>Title} \rangle$:- $\langle \textit{Implicitly-Stated-Title} \rangle$
6. $\langle \textit{Implicitly-Stated-Title} \rangle$:- $\langle \textit{Keywords} \rangle$ or $\langle \textit{Abstract} \rangle$ or $\langle \textit{Other-words} \rangle$
7. $\langle \textit{Abstract} \rangle$:- Explicitly-Stated-Abstract
8. $\langle \textit{Abstract} \rangle$:- $\langle \textit{Implicitly-Stated-Abstract} \rangle$
9. $\langle \textit{Implicitly-Stated-Abstract} \rangle$:- $\langle \textit{Keywords} \rangle$ or $\langle \textit>Title} \rangle$ or $\langle \textit{Other-words} \rangle$
10. $\langle \textit{3-level-Subjects} \rangle$:- $\langle \textit{Keywords} \rangle$ or $\langle \textit>Title} \rangle$ or $\langle \textit{Abstract} \rangle$ or $\langle \textit{Other-words} \rangle$

Note The words that are found in the explicitly stated keywords, title or abstract need not be in our Keyword subject database. Thus, they will not be used in selecting the 3-level subject. Only the words that are found in our keyword subject database will be used in concluding the subjects of the primary source document.

Comparing and testing ASHG with similar systems

Comparing Nordic to our system The above mentioned technique is very similar to what we intend to produce. However our subject tree is based on LCSH and we try to select 3 subject level hierarchy for each document. Another major distinction is that the selected subject hierarchy for a document must have all the related keywords in at least one of the following fields: Title, Abstract, Keyword-List and other important or headings found in the document. So every word in these fields will be searched in the keyword-subject database to find a match and if a match is found the corresponding weight of the word is passed to the suggested subject. To limit our choices we filter out the subject heirarchy that does not have all the words in the specified fields.

Comparing Oracle ConText with Our System

We can easily notice the similarities between both techniques. We use the stop list to filter out the english noise words, that do not contribute to the meaning. We will be also using the word stem, word and phrase weighting. The thesaurus that we will be using will be mainly extracted from the LCSH. It will contain both words and subjects that relate to the words.

The Subject Database, (Thesaurus)

A thesaurus is a set of items (phrases or words) plus a set of relations between these items. The subject database, which is the thesaurus, aims at standardization of terms which is enforced by the CINDI system. Thus, this database will contain the subject control terms. So there is a need to built this database in order to solve problems where the same item or subject is differently referred or classified. This database is a knowledge base that will be created with the help of the expertise of the expert cataloger, the library of congress, the YAHOO, Infoseek, the Web Crawler and Lycos subject classifications, etc.. This database will help the provider of the primary resource or the ASHG in selecting the correct terms for the subject, sub-subjects.

Some of the classification schemes or thesauri usually contain relations between thesaurus items such as BT (Broader Term), NT (Narrow Term), UF (Used For), and RT (Related To) as in the LCSH (Library of Congress Subject Hierarchy). Thus, we try to built our semantic network automatically by exploiting this kind of relationship.

The subject database is made of 3 level hierarchy, where level-0 contains the general subject, which is the Broader Term, level-1 contains all the subjects that fall under level-0 subjects, and similarly level-2 will contain all Narrow Terms that fall under both level-0 and level-1 subjects. In our Semantic Network, we might have subjects that are related to others, and we might have terms that were used for our control terms, thus constituting the synonyms. Our database will be mainly built from the LCSH, YAHOO, and some other subject classification schemes. **put the figure here**

The Keyword-Subject Database

As we mentioned earlier, the CINDI system uses knowledge bases and expert sub-systems to help the user in the registering and search processes. One such need for an expert system is in avoiding chaos introduced by differences in perception of different indexer. Hence, some form of standardization of terms used has to be enforced.

As you might have noticed in the documents extractors, keywords should be extracted from the document, for 2 reasons: one is for selecting the right subject for the document , the other reason is to help users in searching for the document. Thus, as far as ASHG is concerned, keywords extraction will lead to identifying the subject hierarchy.

Each keyword has a list of subjects attached to it, some of these subjects are in level-0, others are in level-1 and the rest are in level-2. Thus when a keyword is extracted we look into the keyword-subject database and select the list of subjects to which it belongs. These subjects will be candidates for the subject , subsubjects selection.

This database will be automatically built, and the keywords initially will be extracted from the subjects that are already in the Subject Database. Each word in the subject other than the noise English grammatical words will be sent to the *stemming process*. The outcome of that process will be the word's root that will be stored as the keyword for that particular subject. For example, the word computer will have as level-0 subjects: computer games, computer-aided engineering, computer drawing.... It will have for level-1 subjects: computer adventure games, computer-aided design, computer-aided transcription systems... And finally will have as level-2 subjects: computer and arithmetic logic unit, computer war games.... **put the figure here**

Other Databases built

The synonym Database

This database contains the subject terms which were used before the current ones.

The Related Database

It contains the subjects and their related subjects.

Preliminary steps to Build the subject hierarchy, synonyms and related terms

In the first preliminary steps we try to exploit the Library of Congress hierarchy structure to build our hierarchy. Since our system's discipline is made up of 3 hierarchies, then we should try to look for the Narrow Terms, and Broader terms. The saved file from the LCSH is passed to another program, called *SUB-LC-EXTRACT*, that will exploit the structure of the input file and will produce 3 important files that will be the bases for building the rest of the systems hierarchy. The first file that has .sub2 extension contains all the subjects that fall in 3 level hierarchy, .syn file contains the term and its corresponding Used for terms and finally the .rel file that contains the term and its corresponding related terms.

.syn This file will be used as input to build the synonym Database.

.sub2 This file will be used to build the subject hierarchy Database.

.rel This file will be used to build the related terms Database.

Files and programs used to build the preliminary steps in building the keyword-Subject Database

In order to build the keyword-subject database, an additional work should be done. Work should be done initially on all .sub2 files. We have to break the 3 level terms found in .sub2 files into 3 different files, each file will contain the terms found in the same level. All the .sub2 files will be passed as input to producing-3h-files. This program will produce 3 level files and each level file will contain the terms or subject found in the corresponding level (level-0, level-1, level-2). Each of these files will then be passed to *keywords* which will extract every word or term found in the subject and output them in the corresponding file. The next step is to extract the english noise words from these files by the use of *word-extract2* program. The next and final step is to list each word followed by the list of subjects in which it is found. This final work is done by the *word-subject(1,2,3)*. The final output file(s) are *wordsinlevel(0,1,2)-words.sub*. The keyword-subject database will be built using these files. Each word will be sent to the *stemming process* that will give the word's root, which will be stored in the keyword subject database with its corresponding subjects.

-bb-error = =

Figure 3: Document Type Recognition

Programs to build the Databases

The following programs build the database by using the preliminary files produced.

new-subject-build.cc The aim of this program is to build the 3 level subject hierarchy. It takes as input the files having .sub2 extension. It produces the 3 level-hierarchy. Here is the algorithm followed:-

```
open database
read first line of the input file into subject0
check if it is already in database
if it is not then
    create one persistent object for it
read the next line into subject1
check and see if it is found in database and under subject0
if it is not found then
    create one persistent object and store it.
read until a flag all the line- one at a time into subject2.
    check to see if it is found in DB under both subject0 and subject1
    if it is not found then
        create an object for it.
```

add-synonym.cc This program handles building the synonym database by reading the .syn file. This program will read the input file and by checking the database, it either adds a synonym or does not. The program will follow the following algorithm:-

```
Open the database.
read first line into subject
if this subject is in level-0 then
    read every other line and check if it is already in synonym DB.
    if it is not create an object for it and add the subject for its list.
    else add the subject to its list if it is not already there.
we repeat the same things and check if that subject is in level-1 or 2.
```

related.cc This program constructs the related subjects database. It reads the .rel file as input . Here is the algorithm that this program follows:-

```
read the first line into subject
check to see if it is in one of the 3 levels
if it is not then skip this subject
else check and see if it is in related database
if it is not create an object for it
read the following lines one at a time into rel-term
add it to the corresponding list of the related term
Repeat the whole process until the end of a file.
```

build-keyword-db.cc This program will construct the keyword database by reading the files that contain a list of word and each word is followed by the subject in which it appears. These files are wordsinlevel(0,1,2)-words.sub. These files are the input for this program. Here is the algorithm:-

Open the first file as input.

Read the first line into subject.

Send it to the stemming process, that will return the word root.

Check and see if it is in the keyword database.

If it is not then

 Create an object for it.

Read the following lines into key-term one at a time.

We check to see if this key-term is in the 3 level subject hierarchy.

If it is then

 We add it to the appropriate list of the keyword if not already there.

We repeat the same thing for the remaining input files.

Augmentation of Subject hierarchy

We have extended the Subject hierarchy that is being used in CINDI. We have used the Library of Congress in building the thesaurus, which is a set of terms and a relationship between these items. We will try to extend it to contain almost all subjects, but we will concentrate on building the Computer Science hierarchy and the Electrical engineering as an initial stage.

Computer Science hierarchy

In building this hierarchy we will try to exploit the ACM, Library of Congress and the other subject hierarchy disciplines like yahoo to build our own discipline. Thus we will first try to extract all the sub-levels that fall under CS, and then move on to extend each and every sub-level, by using the above mentioned hierarchies.

Electrical Engineering

In building this hierarchy, we will attempt to look into STARR (Science and Technology Annual Reference and Review), LCSH and the other mentioned hierarchies.

References

- [1] LUHN, H. P., 'The automatic creation of literature abstracts.' IBM Journal of Research and Development, 2, 159-165 (1958).
- [2] C.J. van Rijsbergen, 'Information Retrieval', second edition, Butterworths 1979, 17-22.
- [3] SALTON G., The SMART Retrieval System, Prentice-Hall, Inc. 1971, 4-6.
- [4] SALTON G., MCGILL M. J., Introduction to Modern Information Retrieval, McGraw-Hill Book Company, 1983, 87-89.
- [5] SALTON G., ALLEN J. , BUCKLEY O. , Automatic Structuring and Retrieval of Large Text Files. Department of Computer Science, Cornell University. June 1992.
- [6] SALTON G. and LESK M. E., Computer Evaluation of Indexing and text processing, journal of ACM, Vol 25, No. 1, January 1968, pp. 8-36.
- [7] BLAIR D. C. , Language representation in Information Retrieval, Elsevier Science publishers, New York, 1990.
- [8] RILLOFF E. and HOLLAR L., Text Database and Information Retrieval, ACM computing surveys, Vol. 28, No. 1, March 1996 , pp. 133-135.
- [9] RILOFF E. and LEHNERT W., Information Extraction as a basis for High-Precision Text Classification, ACM Transactions on Information Systems, July 1994, Vol. 12, No. 3, pp. 296-333
- [10] Turtle and Croft, W. Bruce 1991. Efficient Probabilistic Inference for Text Retrieval. In Proceedings of RIAO 91. 644-661.
- [11] SALTON G. 1989. Automatic Text Processing: The Transformation, Analysis, and Retrieval of information by Computer. Addison-Welsey, Reading, MA.
- [12] MAULDIN M. 1991. Retrieval Performance in FERRET: A conceptual Information Retrieval System. In Proceedings, SGIR 1991. 347-355.
- [13] Goodman, M. 1991. Prism: A Case-Based Telex Classifier. In Proceedings of the second annual Conference on Innovation Applications of Artificial Intelligence. AAAIPress. 25-37.
- [14] Hayes, Philip J. and Weistein, Steven P. 1991. Construe-TIS: A system for content-based indexing of a database of news stories. In Proceedings of the second annual Conference on innovative Applications of Artificial Intelligence. AAAI Press. 49-64.
- [15] Rau, Lisa F. and Jacobs, Paul S. 1991. Creating Segmented Databases From Free Text for Text Retrieval. In Proceedings, SIGIR 1991. 337-346.
- [16] Lehnert, W. G. and Sundheim, B. 1991. A Performance Evaluation of Text Analysis Technologies. AI Magazine 12(3):81-94.

- [17] Proceedings of the third Message Understanding Conference (MUC-3), San Mateo, CA. Morgan Kaufmann.
- [18] Proceedings of the Fourth Message Understanding Conference (MUC-3), San Mateo, CA. Morgan Kaufmann.
- [19] Proceedings of the Fifth Message Understanding Conference (MUC-3), San Mateo, CA. Morgan Kaufmann.
- [20] H. P. Edmundson and R. E. Wyllys, Automatic Abstracting and Indexing - Survey and Recommendations Communications of ACM, 4:5, May 1961, 226-234.
- [21] H. P. Edmundson, Problems in Automatic Abstracting, Communications of the ACM, 7:4, April 1964, 259-263.
- [22] J.E. Rush, R. Salvador, and A. Zamora, Automatic Abstracting and Indexing- Production of Indicative Abstracts By Application of Contextual Inference and Syntactic Coherence Criteria, Journal of the ASIS, 22:4, July-August 1964, 260-274.
- [23] L. L. Earl, Experiments in Automatic Extracting and Indexing, Information Storage and Retrieval, 6:4, October 1970, 313-334.
- [24] P. B. Baxendale, Man made Index for Technical Literature - An Experiment, IBM Journal of Research and Development, 2:4, 1958, 354-361.
- [25] C. D. Paice, Automatic Generation of Literature Abstracts - An Approach Based on the identification of self indicating phrases, in information retrieval research, R.N. Oddy, S.E. Robertson, C.J. van Rijsbergen and P.W. Williams, editors, Butterworths, London, 1981, 172-191.
- [26] C. D. Paice, Constructing Literature Abstracts by Computer: Techniques and Prospects, Information Processing and Management, 26:1, 1990, 171-186.
- [27] Automatic indexing and classification of the WAIS databases <http://www.ub2.lu.se/autoclass.html>.
- [28] <http://www.oracle.com.sg/products/oracle7/oracle7.3/html/conTxtDS.html>.
- [29] http://www.oracle.com.sg/products/oracle7/oracle7.3/html/context_seybold.html.
- [30] Salton G., Allan J. , Buckley C., and Singhal A. , Automatic Analysis, Them Generation, and Summarization of MachineReadable Texts, Science, Vol264, June 1994, pp. 1421-1426.
- [31] Lewis D. D. , Jones K. , Natural Language processing for information Retrieval, Communications of the ACM, Vol 39, January 1996, pp. 92-101.
- [32] Fung R. and Del Favero B. , Applying Baysian Networks to Information Retrieval, Communications of the ACM, Vol 38, No. 3, March 1995, pp. 42-57.

- [33] E. DuRoss Liddy, Anaphora in natural language processing and information retrieval. *Information Processing and Management*, Vol. 26, No. 1, 1990, pp. 39-52.
- [34] A. F. Smeaton, Progress in the Application of Natural Language Processing to Information Retrieval tasks, the computer journal, Vol. 35, No. 3, 1992, pp. 268-271.
- [35] Y. Chiamarella et al. , IOTA: a full text information retrieval system. In proceedings of the ACM conference on research and Development in information retrieval, edited F. Rabitti, Pisa (1987), pp. 207-213.
- [36] D. A. Evans, Concept management in text via natural language processing: the CLARIT approach. In working notes for the AAAI spring symposium on Text-based intelligent systems. Stanford (1990).
- [37] B. C. Desai, Cover page aka Semantic Header, July 1994, revised version, August 1994, <http://www.cs.concordia.ca/faculty/bcdesai/semantic-header.html>
- [38] Desai, Bipin C., Department of Computer Science, Concordia University. The Semantic Header Indexing and Searching on the internet.
- [39] Darren R. Hardy, Michael F. Shwartz. Customized Information Extraction as a Basis for Resource Discovery. Department of Computer Science, University of Colorado. March 1994; Revised February 1995.
- [40] Robert Krovetz, W. Bruce Croft. Lexical ambiguity and information retrieval. *ACM Transactions on information systems*, Vol. 10, No. 2, April 1992, pp. 115-141.
- [41] Belkin N., Croft W. B. Retrieval techniques. *Annual review information science and technology (ARIST)*, 22, (1987), 109-145.
- [42] Jacqueline W. T. Wong, W. K. Kan, Gilbert Young, ACTION: Automatic Classification for full-text documents, Department of Computer science and engineering, The Chinese University of Hong Kong, *ACM Transactions on information systems*, 1997, pp. 26-41.
- [43] H. P. Edmeundson, New methods in Automatic Extracting, University of Maryland, college park, Maryland, *Journal of the Association for computing machinery*, Vol. 16, No. 2, April 1969, pp. 264-285.
- [44] Brandow R. , Mitze K., Rau L. F., Automatic condensation of electronic publications by sentence selection, *Information Processing and management*, Vol. 31, No. 5., pp. 675-685, 1995.
- [45] Rau, L.F., Jacobs, P.S. and Zernik, U., Information extraction and text summarization using linguistic knowledge acquisition. *Information processing and management*, Vol. 25, No. 4, 1989, pp. 419-428
- [46] Johnson F. C., Paice C. D., Black W.J, Neal A. P. The application of linguistic processing to automatic abstract generation. *Journal of Documentation and Text management*, 1993.