

Text Mining Technology

Turning Information Into Knowledge

A White Paper from IBM

editor: Daniel Tkach
IBM Software Solutions
email: tkach@us.ibm.com

February 17, 1998

Mining Text

Most organizations have large and increasing numbers of online documents which contain information of great potential value. Examples are:

- ▼ Electronic mail from customers, containing feedback about products and services
- ▼ Intranet documents such as memos and presentations which embody corporate expertise
- ▼ Technical reports which describe new technology
- ▼ News wires carrying information about the business environment and the activities of competitors.

The IBM Intelligent Miner for Text deals with these and many other kinds of text documents. It contains tools to

- ▼ Extract key information from text
- ▼ Organize documents by subject
- ▼ Find the predominant themes in a collection of documents
- ▼ Search for relevant documents using powerful and flexible queries

The need for tools to deal with online documents is already large (we need only mention the internet) and is growing larger. Forrester Research[1] has predicted that unstructured data (such as text) will become the predominant data type stored online. This implies a huge opportunity: to make more effective use of repositories of business communications, and other unstructured data, by using computer analysis. But the problem with text is that it is not designed to be used by computers. Unlike the tabular information typically stored in databases today, documents have only limited internal structure, if any. Furthermore, the important information they contain is not explicit but is implicit: buried in the text. Hence the “mining” metaphor -- the computer rediscovers information that was encoded in the text by its author.

Many of the tools in Intelligent Miner for Text can be seen as information extractors which enrich documents with information about their contents. This information can be used as metadata about the documents. Metadata is structured data which can be stored in a database and could be used in turn for Data Mining. This process of annotating documents with metadata is shown schematically in figure 1.

The tools in the Intelligent Miner for Text are designed to be used in building applications that deal with text. Often the first step is to extract key features from texts, to act as “handles” in further processing. Examples of

[1] *Coping with Complex Data*, The Forrester Report, April 1995











	CATEGORY	KEYWORD_1	KEYWORD_2	KEYWORD_3	MAX_AMNT
 	Intranet applications	employee productivity	web applications	network availability	\$20,500
 	Personnel policy	maternity leave	health benefit	parental leave	\$
 	Lotus Notes information	database replication	email	collaboration applications	\$45
 	Commuter information	bus line	telecommuting options	railroad station	\$1.25
 	Office ergonomics	wrist rest	spinal curvature	voice recognition	\$99

Figure 1: Annotating documents with metadata. Schematic showing how tools in the Intelligent Miner for Text can be used to populate database rows with information about documents -- in this case the categories in a cataloging scheme they belong to, the three most significant technical terms in each document, and the largest money amount they refer to.

features are the language of the text, or company names or dates mentioned. After feature extraction, the next step may be to assign the documents to subjects from a cataloging scheme, then to index them for ad-hoc searching. The tools in the Intelligent Miner for Text which support these and other functions will be described in the rest of this White Paper. A section on scenarios describes ways in which the tools can be used together

Text Analysis Functions

Functions in this grouping analyze text to select features for further processing. They can be used by application builders.

Language Identification

The language identification tool can automatically discover the language(s) in which a document is written. (See figure 2) It uses clues in the document's contents to identify the languages, and if the document is written in two languages it determines the approximate proportion of each one. The determination is based on a set of training documents in the languages. Its accuracy as shipped in the tool suite is usually close to 100% even for short text. The tool can be extended to cover additional languages or it can be trained for a completely new set of languages. Its accuracy in this case can be easily higher than 90%, even with limited training data. Applications of the

language identification tool include: automating the process of organizing collections of indexable data by language; restricting search results by language; or routing documents to language translators.

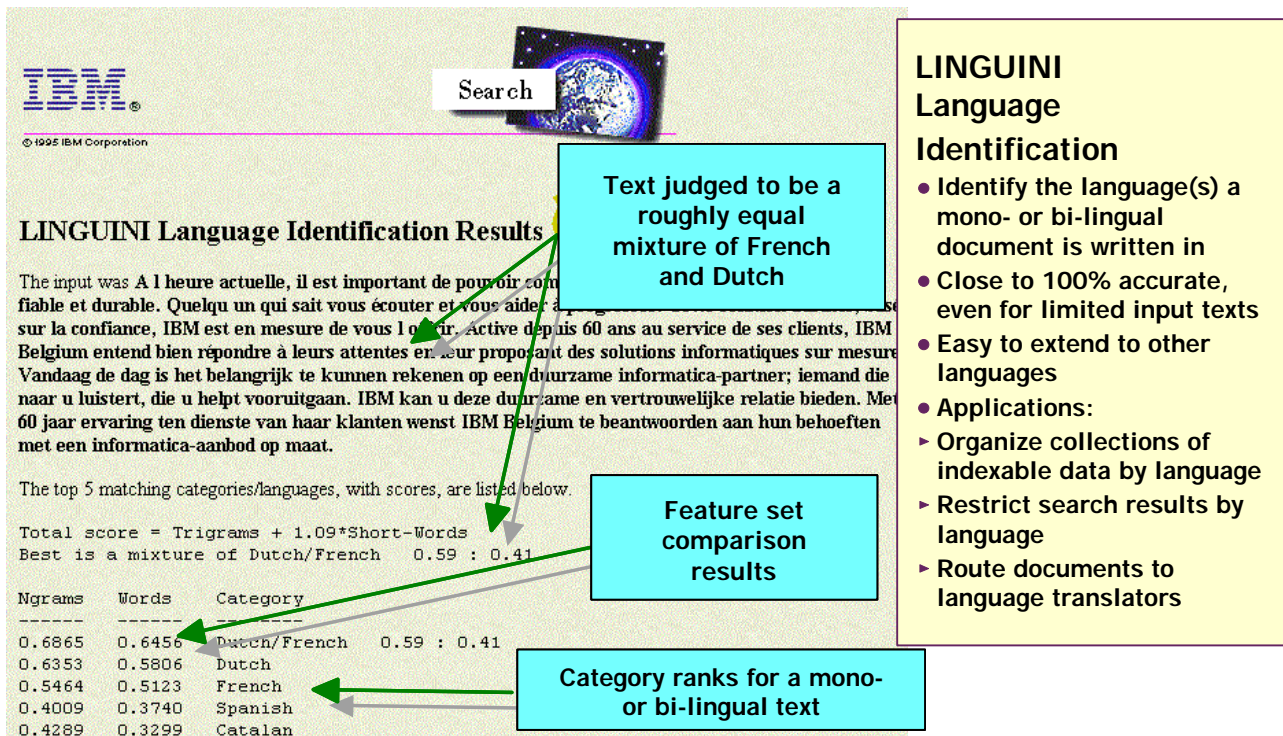


Figure 2: Linguini for identifying the language(s) of a document.

Feature Extraction

The feature extraction component of the Intelligent Miner for Text recognizes significant vocabulary items in text. Examples of the vocabulary it finds are shown in figure 3. The process is fully automatic -- the vocabulary is not predefined. Nevertheless, as the figure shows, the names and other multiword terms that are found are of high quality and in fact correspond closely to the characteristic vocabulary used in the domain of the documents being analyzed. In fact what is found is to a large degree the vocabulary in which concepts occurring in the collection are expressed. This makes Feature Extraction a powerful Text Mining technique. An example of an application built with the feature extractor is shown in figure 4.

Among the features automatically recognized are

- ▼ Names, of people, organizations and places
- ▼ Multiword terms
- ▼ Abbreviations
- ▼ Other vocabulary, such as dates and currency amounts

When analyzing single documents, the feature extractor can operate in two possible modes. In the first, it analyzes that document alone. In the preferred mode, it locates vocabulary in the document which occurs in a dictionary which it has previously built automatically from a collection of similar documents. When using a collection of documents, the feature extractor is able to aggregate the evidence from many documents to find the optimal

...	Credit Lyonnais
capital note	Credit Suisse
CCC	Credit Suisse First Boston
certificate of deposit	Dana
certificates of deposits	debt security
Chronar	debtor country
CMOs	Detroit Edison
commercial bank	Digital Equipment
commercial paper	dtrs of debt
Commercial Union Assurance	end-March
Commodity Futures Trading Commission	Enserch
Consul Restaurant	equity warrant
convertible bond	Eurodollar
credit facility	European Investment Bank
credit line	...

Figure 3: Some of the vocabulary found by feature extraction in a collection of financial news stories. The canonical forms are shown.

vocabulary. For example, it can often detect the fact that several different items are really variants of the same feature, in which case it picks one (usually the longest one) as the canonical form. In addition, it can then assign a statistical significance measure to each vocabulary item.

The significance measure, called “Information Quotient” (IQ), is a number which is assigned to every vocabulary item found in the collection. The calculation of IQ uses a combination of statistical measures which together measure the significance of a word, phrase or name within the documents in the collection. The vocabulary items with the highest IQ are almost always names or multiword terms because they tend to convey a more focused meaning than do single words alone.

Name extraction

Names are valuable clues to the subject of a text. Using fast and robust heuristics, the name extraction module locates occurrences of names in text and determines what type of entity the name refers to -- person, place, organization, or “other”, such as a publication, award, war, etc. The module processes either a single document or a collection of documents. For a single document, it provides the locations of all names that occur in the text. For a collection of documents, it produces a dictionary, or database, of names occurring in the collection.

All the names that refer to the same entity, for example "President Clinton", "Mr. Clinton" and "Bill Clinton", are recognized as referring to the same person. Each such group of variant names is assigned a "canonical name", (e.g., "Bill Clinton") to distinguish it from other groups referring to other entities ("Clinton, New Jersey"). The canonical name is the most explicit, least ambiguous name constructed from the different variants found in the document. Associating a particular occurrence of a variant with a canonical name reduces the ambiguity of variants. For example, in one document, "IRA" is associated with the Irish Republican Army, while in another it may be associated with an Individual Retirement Account.

The name extraction module does not require a preexisting database of names. Rather, it discovers names in the text, based on linguistically motivated heuristics[2] that exploit typography and other regularities of language. It operates at high speed because it does not need to perform full syntactic parsing of the text. Discovering names in text is challenging because of the ambiguities inherent in natural language. For example, a conjunction like "and" may join two separate names (e.g., "Spain and France") or be itself embedded in a single name (e.g., "The Food and Drug Administration"). The heuristics employed by the name extraction module correctly handle structural

[2] *Disambiguation of Proper Names in Text*, N. Wacholder and Y. Ravin, presented at the Applied Natural Language Conference (April 3, 1997)

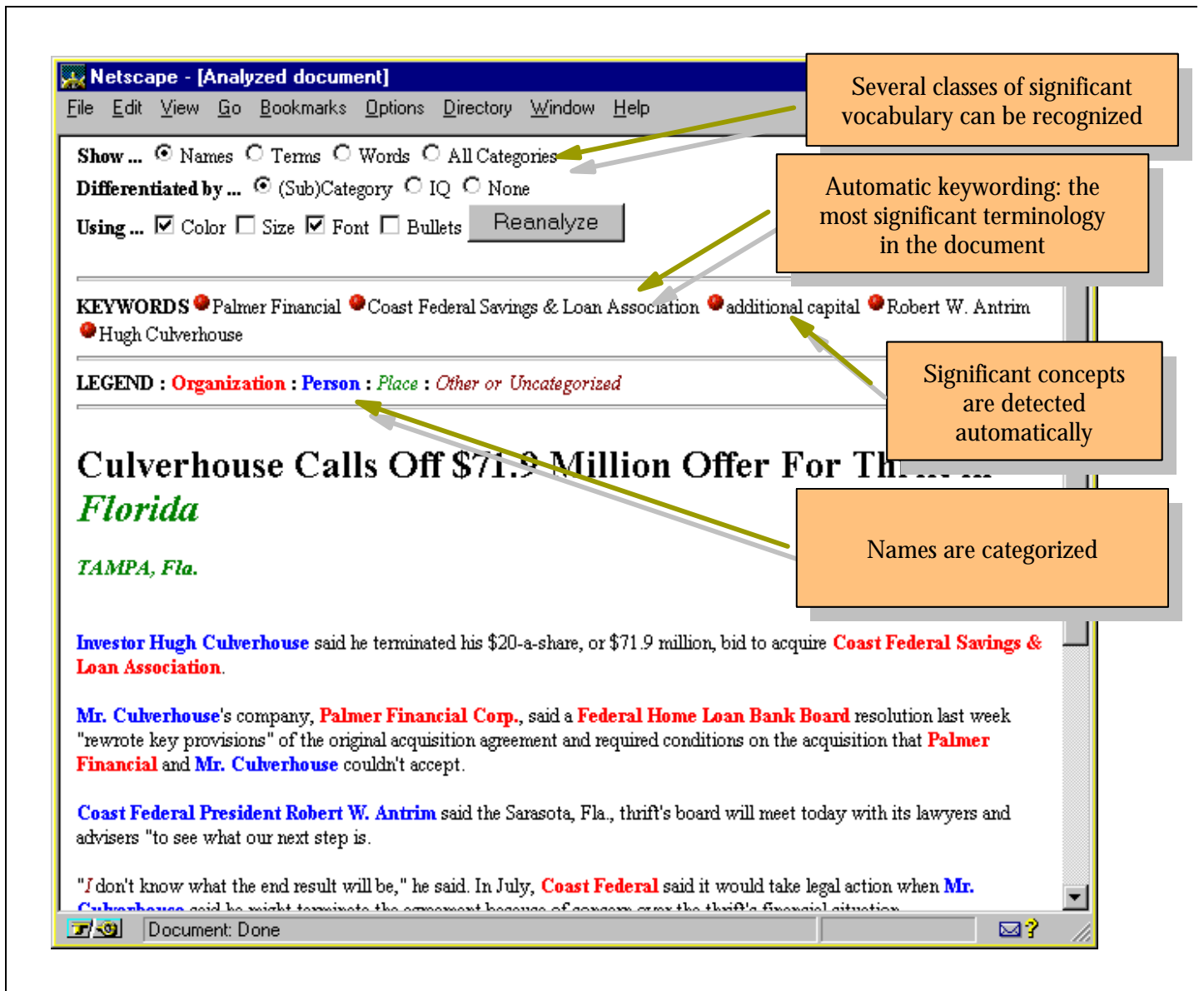


Figure 4: Example of an application, built on the feature extractor APIs, which highlights vocabulary items found in HTML documents.

ambiguity in the majority of the cases. In tests it has been found to correctly recognize 90-95% of the proper names present in edited text.

Term Extraction

The second major type of lexical clue to the subject of a document are the domain terms the document contains. The term extraction module uses a set of simple heuristics to identify multi-word technical terms in a document. Those heuristics, which are based on a dictionary containing part-of-speech information for English words, involve doing simple pattern matching in order to find expressions having the noun phrase structures characteristic of technical terms. This process is much faster than alternative approaches.

The term extraction module discovers terms automatically in text -- it is not limited to finding terms in supplied in a separate. The term extractor ensures the quality of the set of terms it proposes by requiring that they be

repeated within a document. Repetition in a single document is a signal that a term names a concept that the document is "about", thus helping to ensure that the expression is indeed a domain term. Furthermore, when Text Mining is analyzing a large collection of documents, the occurrence statistics for terms also helps to distinguish useful domain terms.

As for names, Text Mining's term extraction recognizes variant forms for the terms it identifies. For example, all of "alternate minimum tax", "alternative minimum tax", and "alternate minimum taxes" would be recognized as variants of the same canonical form.

Abbreviations

The formation of abbreviations and acronyms is a fruitful source of variants for names and terms in text. Text Mining's abbreviations recognizer identifies these short form variants and matches them with their full forms. When the full form (or something close to it) has also been recognized by name extraction or term extraction, then the short form is added to the set of already-identified variants for the existing canonical form. Otherwise, the full form becomes the canonical form of a new vocabulary item with the short form as its variant.

The recognizer is capable of handling a variety of common abbreviatory conventions. For example, both "EEPROM" and "electrically erasable PROM" are recognized as short forms for "electrically erasable programmable read-only memory". TextMining also knows about conventions involving word-internal case ("MSDOS" matches "MicroSoft DOS") and prefixation ("GB" matches "gigabyte").

Other extractors

In order for the principal extractors to work effectively, Text Mining also includes other extractors which help analyze portions of the document text. Among these, the recognizers for numbers, dates, and currency amounts extract information which is potentially useful for certain applications.

The **number** recognizer identifies any of the following text expressions as a number expression and produces a canonical representation for it consisting of an appropriate integer or fixed-point value, as required:

"one thousand three hundred and twenty-seven"
"thirteen twenty-seven"
"1327"
"twenty-seven percent"
"27%"

The **date** recognizer identifies expressions describing either absolute or relative dates and produces a canonical representation for them. The representation for relative dates (e.g., "next March 27th") assume that some external process provides a "reference date" with respect to which a date calculator can interpret the expression. Some example expressions, with their canonical representation, are:

"March twenty-seventh, nineteen ninety-seven"	1997/03/27
"March 27, 1997"	1997/03/27
"next March 27th"	ref+0000/03/27
"tomorrow"	ref+0000/00/01
"a year ago"	ref-0001/00/00

The **money** recognizer identifies expressions describing currency amounts and produces a canonical representation for them. Examples are:

"twenty-seven dollars"	"27.000 dollars USA"
"DM 27"	"27.000 marks Germany"

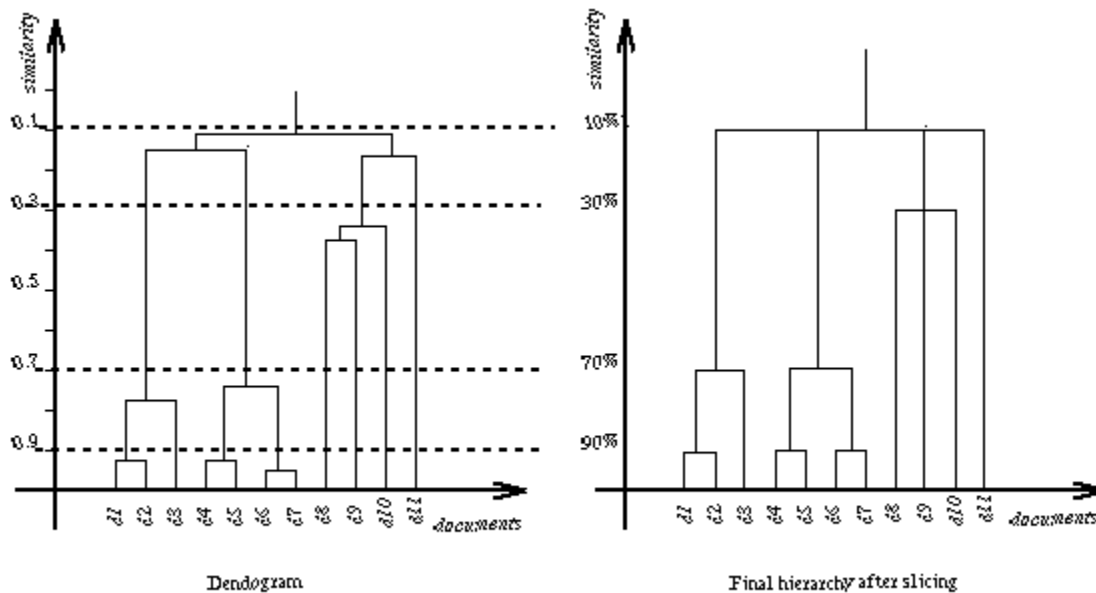


Figure 5: Dendrograms showing how the clustering tool groups documents by similarity. The left diagram shows how user-defined slicing levels relate to binary clustering. The right-hand diagram shows the final clusters.

Clustering

Clustering is a fully automatic process which divides a collection of documents into groups. The documents in each group are similar to each other in some way. When the content of documents is used as the basis of the clustering (as in the IM for Text), the different groups correspond to different topics or themes that are discussed in the collection. Thus, clustering is a way to find out what the collection contains. To help to identify the topic of a group, the clustering tool identifies a list of terms or words which are common in the documents in the group.

Clustering can also be done with respect to combinations of the properties of documents, such as their length, cost, date, etc. The clustering tools in the Intelligent Miner for Data would be applicable to that kind of problem.

An example of clustering in Text Mining would be to analyze e-mail from customers to find out if there are some common themes that have been overlooked. The effect of clustering is to segment a document collection into subsets (the clusters) within which the documents are similar in that they tend to have some common features. Clustering can be used to

- ▼ Provide an overview of the contents of a large document collection
- ▼ Identify hidden similarities
- ▼ Ease the process of browsing to find similar or related information.

Clustering can also be applied to a subset of documents. In particular, the TextMiner search engine can use clustering to reveal structure in the results list of a search, as described later.

Clusters are discovered in data by finding groups of objects which are more similar to each other than to the members of any other group. Therefore, the goal of cluster analysis is usually to determine a set of clusters such that inter-cluster similarity is minimized and intra-cluster similarity is maximized. Since in general there is no unique or best solution to this problem, the Intelligent Miner for Text provides a tool based on a robust algorithm which can be configured to match the intended application.

Hierarchical clustering

Hierarchical clustering algorithms seem to work especially well for textual data. The algorithm used by the clustering tool starts with a set of singleton clusters each containing a single document. It then identifies the two clusters in this set that are most similar and merges them together into a single cluster. This process is repeated until only a single cluster, the root, is left. The (binary) tree constructed during this process, called a *dendrogram*, contains the complete clustering information including all inter- and intra-cluster similarities.

The left diagram in figure 5 shows a graphical representation of a dendrogram. The vertical axis measures the similarity. The horizontal lines in the tree are drawn at the similarity level that formed the corresponding cluster, i.e., the intra-cluster similarity for this cluster. The inter-cluster similarity between two arbitrary clusters is the similarity level of the first common cluster. Pure dendrograms are often inconvenient to visualize and interpret as they are binary. Therefore we apply a configurable *slicing* technique that identifies the clusters within the tree which have a comparable degree of intra-cluster similarity. With this information new clusters are formed by merging previous clusters.

Slicing can be adjusted to individual needs. The clustering tool allows the user to set the upper and lower thresholds of intra-cluster similarity, and to define the number of slicing-steps at which slicing is performed. For example, specifying an upper threshold of 90%, a lower threshold of 10%, and the number of slice steps as 8, the clustering tool will check for potential clusters at similarity levels of 90%, 80%, 70%, 60%, 40%, 30%, 20%, and finally 10%. The right diagram in figure 5 shows the result of slicing for the dendrogram on the left with the above parameter setting. In this case the two clusters {d8,d9} and {d10} have been merged to a new cluster {d8,d9,d10}.

A convenient way to combine a coarse overview and detailed information of the clustering structure is to repeat the slicing at different levels of intra-cluster similarity thresholds. The combined output reflects the most important decisions of the hierarchical clustering algorithm at some snapshots, taken at different levels.

Computation of document similarity using lexical affinities

It is clear that the notion of similarity between documents and clusters is crucial. A very simple similarity measure would be the degree of overlap for single words in the documents. However, due to the ambiguity of single words, this measure of similarity is rather imprecise, and the resulting clustering would not be very useful in general.

One alternative to using single words would be to perform a semantic analysis of the documents, in order to find the concepts mentioned in the text and use them for clustering. This kind of analysis is very expensive and furthermore depends on a lot of domain-dependent knowledge that has to be constructed manually or obtained from other sources. Instead of taking this approach, the clustering tool uses *lexical affinities* instead of single words.

A lexical affinity is a correlated group of words which appear frequently within a short distance of one another. Lexical affinities include phrases like "online library" or "computer hardware" as well as other less readable word groupings. They are generated dynamically, thus they are specific for each collection. A set of semantically rich terms can be obtained without a need to hand-code a specialized lexicon or a thesaurus. The clustering tool uses a list of the lexical affinities in each document as the basis for its similarity calculation. A cluster can be labeled with the lexical affinities it contains, which allows a user to quickly assess the characteristics of the cluster.

Figure 6 shows part of a user interface for hierarchical clustering. Here we display the tree structure after clustering and slicing using the library metaphor. Clusters are represented as bookshelves, and the individual documents as books. The user can navigate through the tree and track down interesting parts and documents in the tree. Other parts of the user interface allow selection of the document set for clustering, and the definition of the various parameters.

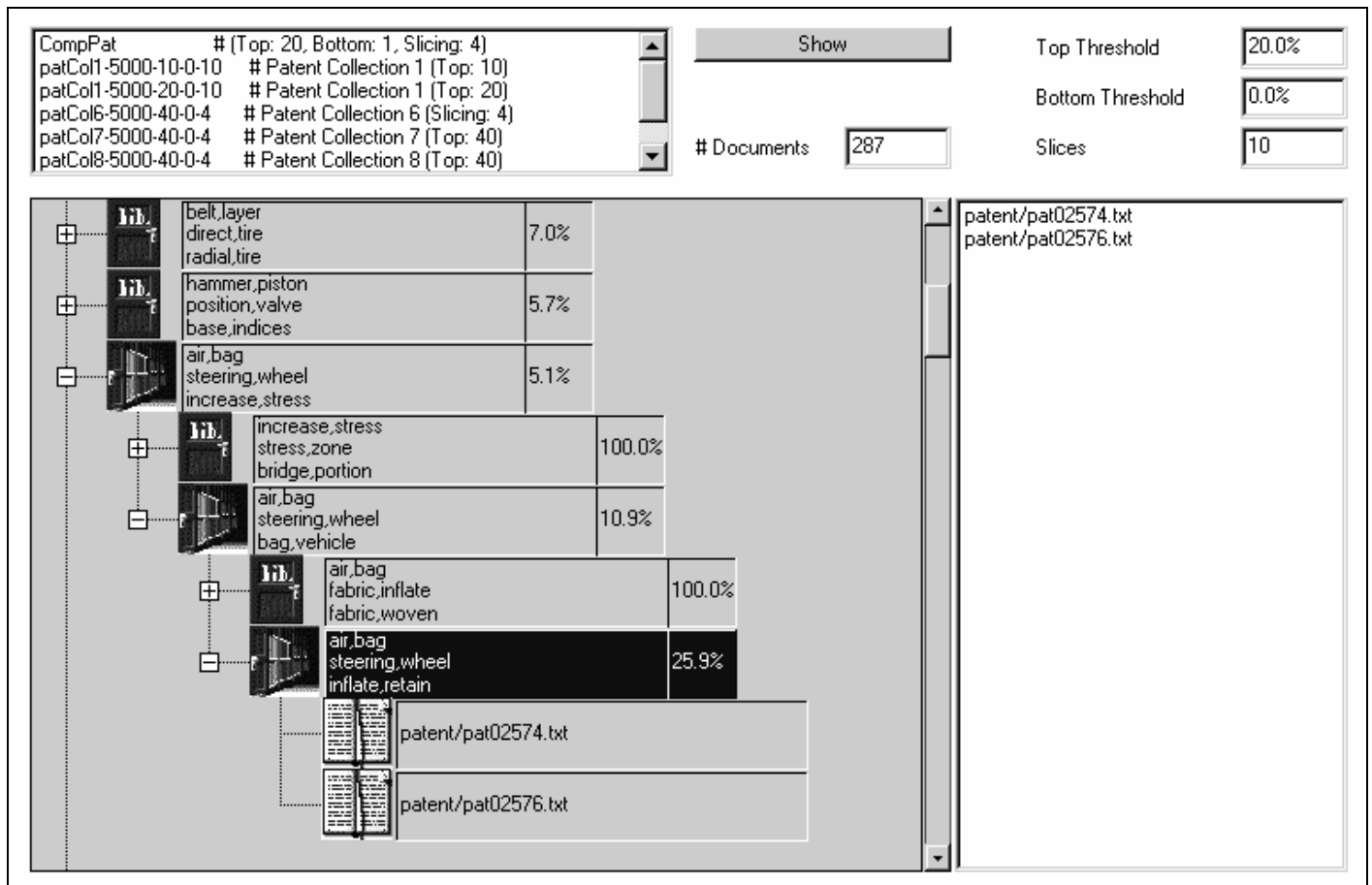


Figure 6: Example of a browser for the hierarchical clustering tool

Categorization

Categorization tools assign documents to preexisting categories, sometimes called “topics” or “themes”. The categories are chosen to match the intended use of the collection. By assigning documents to categories, the Intelligent Miner for Text can help to organize them. While categorization cannot take the place of the kind of cataloging that a librarian can do, it provides a much less expensive alternative. Applications of categorization include:

Organizing intranet documents

For example, documents on an intranet might be divided into categories like “Personnel policy”, “Lotus Notes information” or “Commuter information” as shown in figure 1. It has been estimated that it costs at least \$25 to have a librarian catalog an item. It is clearly impractical to catalog the million or so documents on a large intranet in this way. By using automatic categorization, documents can be assigned to an organization scheme which makes it easier to find them by browsing, or by restricting the scope of a text search.

Assigning documents to folders

Categorization can also help to file documents in a smarter way. For example, it could help a person who has to assign e-mail to one of a set of folders, by suggesting which folders should be considered.

While the actual categorization of documents is fast and inexpensive using the automatic tools in the Intelligent Miner for Text, the definition of the categorization scheme requires some care. However, the effort to do this for the Intelligent Miner for Text is much less than in some other systems. In the Intelligent Miner for Text, categories are defined by sample documents. All the analysis of the categories, especially the choice of key words and phrases to characterize each category, is done automatically (in older systems this had to be done manually). This “training” phase produces a special index which is subsequently used to categorize new documents. The Intelligent Miner for Text categorizers return a list of category names and confidence levels for each document being categorized. Documents can be assigned to more than one category. If the confidence level is low, then typically the document would be put aside so that a human categorizer can make the final decision. This might be necessary if the document is on a subject which doesn’t match any of the predefined categories. If this happens often it may be a sign that a new category needs to be defined. All that needs to be done is to accumulate a set of examples for the new category, then rerun the training process which takes at most a few hours.

If the set of defined categories does match the subject matter of the incoming documents, then tests have shown that the categorizers in the Intelligent Miner for Text agree with human categorizers at least as well as human categorizers agree with one another.

The categorizers in the toolkit work by extracting characteristic features from the documents being categorized, then comparing these features to a set of features for each category which was extracted from the example documents during the training phase. This approach ensures a compact index and rapid processing. Two different categorizers are delivered with the Intelligent Miner for Text, which differ in the features they use:

Linguistic features

This categorizer uses special linguistic features extracted from English text by the feature extractor components of the Intelligent Miner for Text as the basis for categorization. Their low ambiguity and domain specificity make these excellent features for this purpose.

N-grams

The n-gram categorizer uses letter groupings and short words as the features for categorization. It can extract these from documents in any language. In addition to categorizing documents by their subject matter, it can be used to determine their language (the Language Identification tool is a special application of this categorizer) or their code page.

Text Search Functions

The search engine in the text mining toolkit is a fully-featured product which incorporates several of the text analysis functions to provide search facilities, in addition to the ability to index and search in many languages, use supporting thesauri, and process queries with a combination of natural-language and Boolean parts. In addition it offers facilities like result list clustering and relevance feedback which help a user to find the most relevant documents.

Text Search Engine

TextMiner, the fully-featured search engine delivered with Intelligent Miner for Text, allows for the construction of high-quality information retrieval systems in a wide range of possible applications for large collections of documents in 16 different languages and multiple file formats. TextMiner does in-depth document analysis during indexing and allows for sophisticated query enhancement and result preparation to supply high-quality information retrieval.

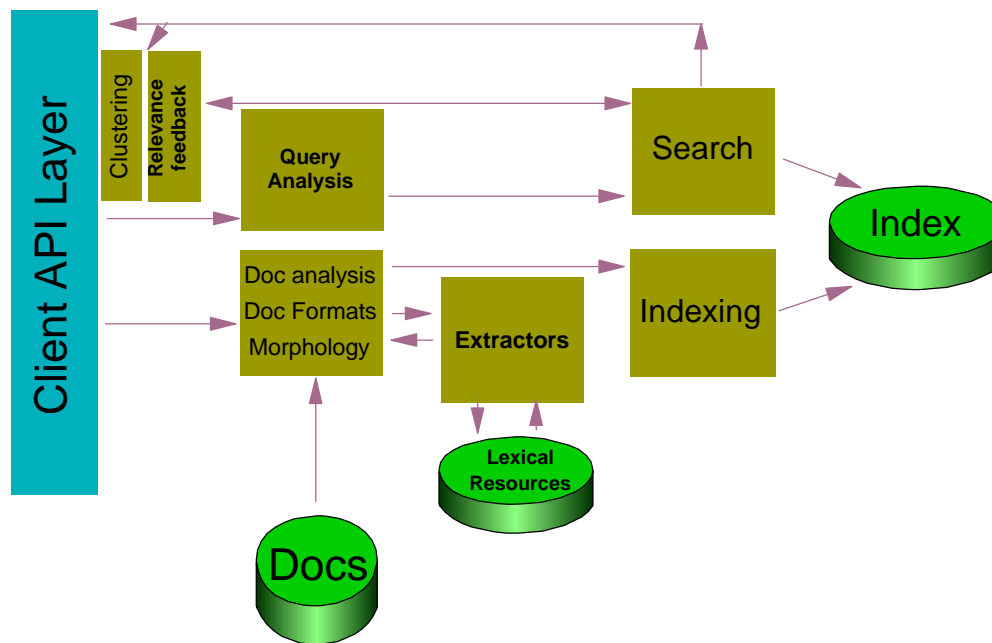


Figure 7: The architecture of TextMiner

Search Engine Features

The basic full text search engine of TextMiner offers a number of advanced features making it one of the most advanced products of its kind on the market today. In the following, we describe the document analysis and query enhancement techniques integrated into TextMiner. The architecture of TextMiner is shown schematically in figure 7. A sample graphical user interface built using the application programming interfaces of TextMiner is shown in figure 8.

TextMiner is a client/server application that allows for a great number of concurrent clients performing searches and other tasks. A important feature is the online update capability, that is, TextMiner is able to perform indexing tasks without having to suspend searches.

Multiple search paradigms

TextMiner implements a number of different search paradigms inside the same search engine. The heart of the search engine is an index structure that supports **Boolean**, **free text**, and **hybrid** queries. **Phonetic** searches are possible on the same index structure. A special purpose index supports **fuzzy** searches and the double-byte character set-based languages Japanese, Chinese and Korean..

Boolean queries allow for conjunction, disjunction, and exclusion of search terms. Individual search terms may be single words or phrases. Front-, middle-, and end-masking using wild cards for single characters or strings of characters can be used. It is also possible to specify proximity conditions, requiring that terms must occur in the same sentence or paragraph. Boolean search allows for very exact specification of a user's information need. Because of its complexity and the flexibility pure Boolean search is usually a typical "expert search", for example performed by a professional librarian.

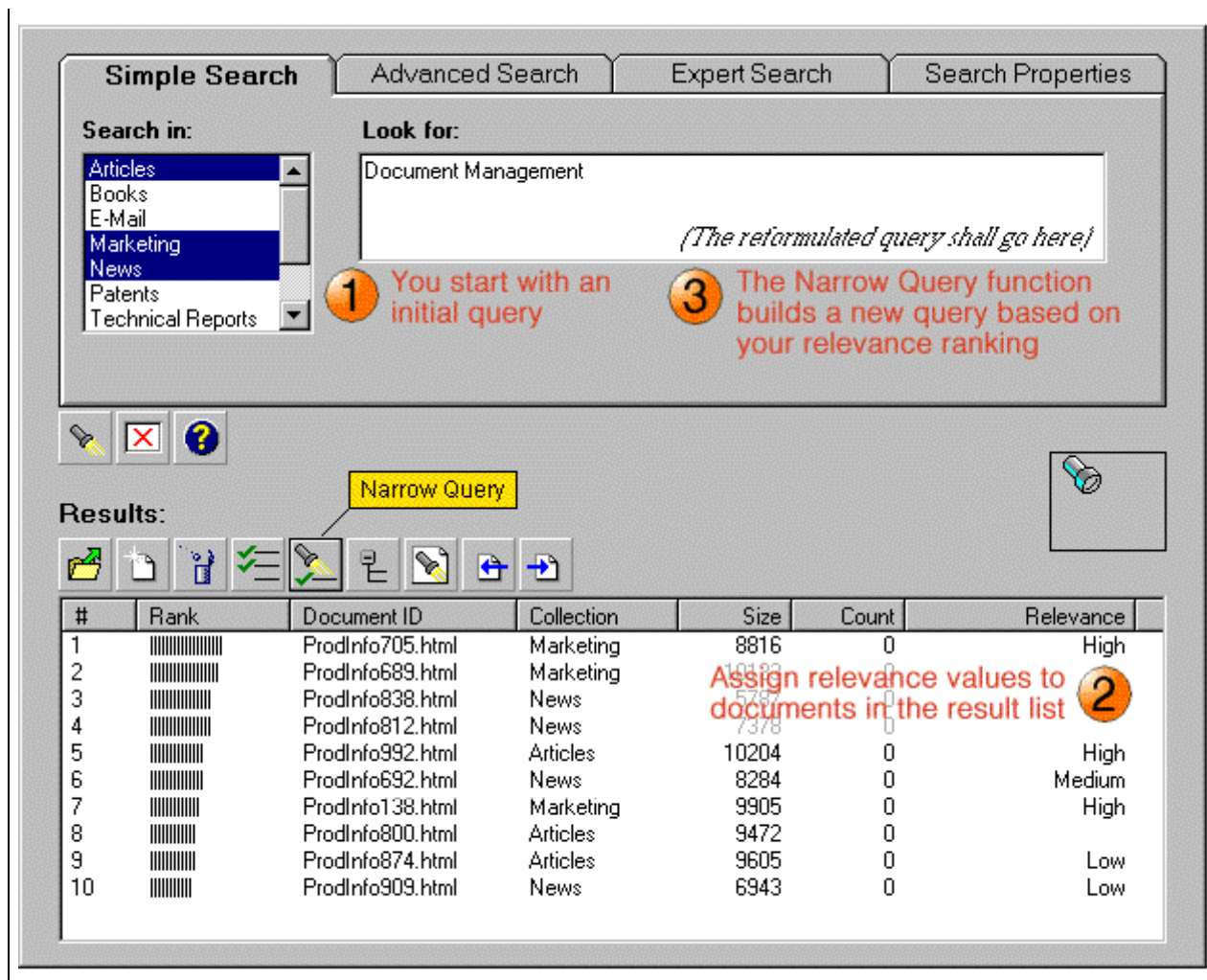


Figure 8: Example of a graphical user interface built on the application programming interfaces of TextMiner. The steps to improve a query using relevance feedback are shown.

Free text queries are based on the probabilistic retrieval model. Probabilistic retrieval covers a broad range of applications in information retrieval and has shown to have very good performance in several large-scale comparative evaluations. The basic idea here is to estimate the conditional probability $p(D|Q)$ that a document D is relevant given a query Q . The estimate is based on term frequencies and maybe other information derived from the document collection at indexing time and comparing this information to the terms in the query. The result of a query is a ranked list of documents ordered by decreasing probability. In general, pure free text queries are easy to use even by novice users of a retrieval system. The typical “simple search” in, for example, a Web search panel could be based on pure free text queries.

Hybrid queries combine free text and Boolean queries in a unique, patented way. The goal is to overcome the problems of pure free text queries. Basically, a hybrid query is a free text query that restricts the result set to the documents that also match the Boolean part of the query. This allows for negative specifications in free text queries that are not supported by pure free text system. For example, it would be possible to search for documents about “market share of Japanese cars” (free text) but not “Toyota” (Boolean). Hybrid queries could be ideally used for “advanced search” functionality to allow an experienced user to perform more effective information retrieval tasks.

For any of these query types, additional expansions using synonyms, thesaurus information, or previously extracted features as described below may be applied to the search terms.

Through the construction of an additional **n-gram index**, TextMiner also supports **fuzzy** searches. Indexing and search is based on n-grams, that is, limited-length character sequences. Searches may use exact or fuzzy matching. Fuzzy matching allows for a limited deviation of the search term from the index term in the document. For example, person names may be found without knowing the exact spelling in many cases. The input text is not processed linguistically for this index. This makes this kind of search language-independent. This search technology works extremely well for documents in Japanese, simplified and traditional Chinese, and Hangeul.

Phonetic searches are supported by performing query expansion and running the query against a standard index. The goal is to introduce wild cards and additional terms to change the query in a way such that all occurrences of similar-sounding words are also retrieved. This is particularly useful whenever the exact spelling of a term to be searched is not known.

Reducing terms to their base form

TextMiner performs morphological analysis for all words before entering them into a linguistic index. For example, the word mice in a text would be indexed as mouse. Later, a search for any form of the same word would find all of the documents containing any arbitrary form of that word. Dictionaries and morphological analysis algorithms are available for multiple languages.

Stop word filtering

Stop words are words such as prepositions and pronouns that occur frequently in texts and are not suitable as search terms. Therefore, these words are excluded from indexing and will also be eliminated from any query. For each language a stop word list is provided that can be customized by the user to exclude other non-information-bearing terms in a particular application.

Linguistic processing for queries

Linguistic processing of search terms is controlled by the user through qualifiers in the query language. Query processing aims at making search terms broader so that the recall rate of searches is increased, that is, more of the relevant documents in a collection are retrieved. This can be achieved by expanding a query to include synonyms or related terms from a user-supplied thesaurus.

Synonyms are semantically similar words. They are obtained from a dictionary which is specific for each supported language. For example, the English word “word” will be expanded to include synonyms like comment, remark, statement, utterance, term, expression, order, news, etc. Synonyms are domain-independent.

A thesaurus is a domain-dependent resource in which words are linked to each other by a set of semantic relations. These relations may be hierarchical (such as the “narrower term” relation), associative (such as a “related term” relation), or as a special case it may be a synonym relation. Thesauri can be made available to the system by the user, who provides a text file in a special format and compiles it. Thesaurus expansion of a query term can be done along a single relation, or with all relations defined in the particular thesaurus. The depth of the expansion, that is, the maximum number of transitions along links, can also be specified as a query parameter.

Text Mining Enhancements

The linguistic processing of documents and queries is further enhanced for English by applying a number of text mining techniques.

When building a **feature index**, TextMiner applies the feature extractors described above to the documents while indexing. In addition to the regular linguistic processing, TextMiner discovers and extracts names of persons, places, or organization, domain-specific multi-word terms, and abbreviations. The extracted information can be used to expand queries later on and therefore significantly enhance the recall of the retrieval system.

Result list clustering groups the results of a query into sets of related documents. This eases the user's comprehension of search results. The result list clustering uses the same technology as the clustering tool provided in the tools suite. For performance reasons it has been limited to a maximum of 200 documents.

Relevance feedback allows the user to mark documents on the result list of a query as relevant or irrelevant. The information provided is used to reformulate the query. The result of that query will then be more focused on documents related to the user's information interest, that is, the precision of the retrieval system will be improved. A user interface which incorporates this function is shown in figure 8.

Web Search

NetQuestion is the search engine designed to build global WWW search services or centralized intranet search services. It is built on the same technology that TextMiner uses, but it is optimized to handle the large amounts of information that are typically stored on Web sites. Document analysis and query processing are more shallow compared to TextMiner, providing for faster indexing and response to large volumes of queries.

Text Search Applications

TextMiner is the integrated full text search facility in IBM DB2 Universal Database V5 and in IBM Digital Library V2. NetQuestion is the search engine for IBM Lotus Go, IBM's corporate home page *www.ibm.com* and IBM's 1500-server intranet, Network Computing Framework.

Scenarios

The following scenarios describe applications in which the components of the Intelligent Miner for Text are used together.

Show me more like this

In text search, "show me more like this", also known as "query by example", is a useful technique. It works as follows. Suppose someone has issued a vague query and is browsing the documents in the results list. They may find a document which discusses exactly the kind of things they are interested in. "Show me more like this" is a function which can return similar documents.

Typically, the "show me more like this" function is implemented by turning the example document into a query, which is then submitted to the search engine. The tools in the Intelligent Miner for Text can be used to implement this function efficiently. The first step is to use the feature extractor module to build a vocabulary dictionary for the collection being searched. Then each document is pre-analyzed and the most significant (highest IQ) vocabulary items it contains are stored as metadata in a database. These items are then used as the query sent to the TextMiner search engine when the "show me more like this" function is invoked.

Searching with categories

In text search, one of main problems is to ensure that as many as possible of the hits near the top of a results list are relevant to the query. One way to do this is to have the user select some categories of information that they want to restrict their query to. For example, a person might issue the query "eggplant" and get documents back about growing eggplants and about recipes for cooking eggplants. By specifying that the category of interest is "cooking" they can see a better focused result list.

To implement this capability, documents need to be categorized when they are indexed for search, using one of the categorization tools. One way to subsequently limit the search is to embed the category label in the document before it is indexed by TextMiner. for example, a paragraph containing the tokens "\$category\$" and "\$cooking\$" could be added. Then at search time the powerful Boolean and adjacency capabilities of

TextMiner would allow the condition to be added to the query that returned documents should contain “\$\$category\$\$” in the same paragraph as “\$\$cooking\$\$”. With this scheme the performance impact of adding the category condition is minimal, because TextMiner processes the Boolean condition efficiently, and it is easy to search several categories at once. Other implementation techniques could also be used.

Processing customer e-mail

In this scenario we are looking a few years into the future when companies will conduct much of their correspondence with their customers via e-mail. To facilitate handling, each e-mail message should be automatically categorized. For example, complaints should be recognized, and the messages should be classified according to the products they deal with. Furthermore, messages that mention large amounts of money may be handled more urgently. Another kind of classification may be applied to the complaints, to suggest the kind of form letter that should be generated in reply. Lastly, the messages should be periodically clustered and the common themes evaluated, to discover if customer concerns have common themes which should be drawn to the attention of product managers.

A journalists' workstation

A journalist collects materials for use in present and future projects into folders. She uses an application which exploits the proper name extraction capabilities in the Intelligent Miner for Text to help organize the material. The application:

- ▼ Annotates folders and files with all the names of people mentioned in the text they contain
- ▼ Uses TextMiner's name search feature e.g. to distinguish the city "Houston" (a place) from occurrences of the personal name “Whitney Houston”
- ▼ Clusters articles using the names mentioned in them
- ▼ Assembles a folder of all paragraphs that contain an instance of a certain proper name
- ▼ Pushes a newswire article to the journalist's workstation as soon as an instance of some person name has been detected in it.

Many other scenarios can be implemented with the flexible and powerful tools in the Intelligent Miner for Text.

The Web Tools: NetQuestion, the Web Crawler and the Web Crawler Toolkit

The Web tools provide technologies to build intelligent Internet/intranet Web sites. They allow companies to leverage the use of Internet and intranets to gain access to relevant information. Information analysts can use these tools to crawl and collect information available on the Internet or intranet, and examine and analyze information collected for business decision making, such as competitive or marketplace information. The tools support push and pull mode of information access.

Net Question

NetQuestion is a powerful, full-text search engine that can be used to build a global WWW search service or a centralized intranet search service. It is designed and optimized to handle the large amounts of information that are typically stored on Web sites. Therefore, the document analysis and query processing are more shallow, compared to TextMiner, to provide for faster indexing and response to large volumes of queries. NetQuestion, however, features the same online update mechanism for indexing as used by TextMiner and other important components, such as client/server handling and queue mechanisms. .

The documents to be indexed by NetQuestion can be in either plain text or text with HTML markup. Sample CGI scripts and HTML forms to develop a search interface are provided with the system. Administration can be performed through command line functions. .

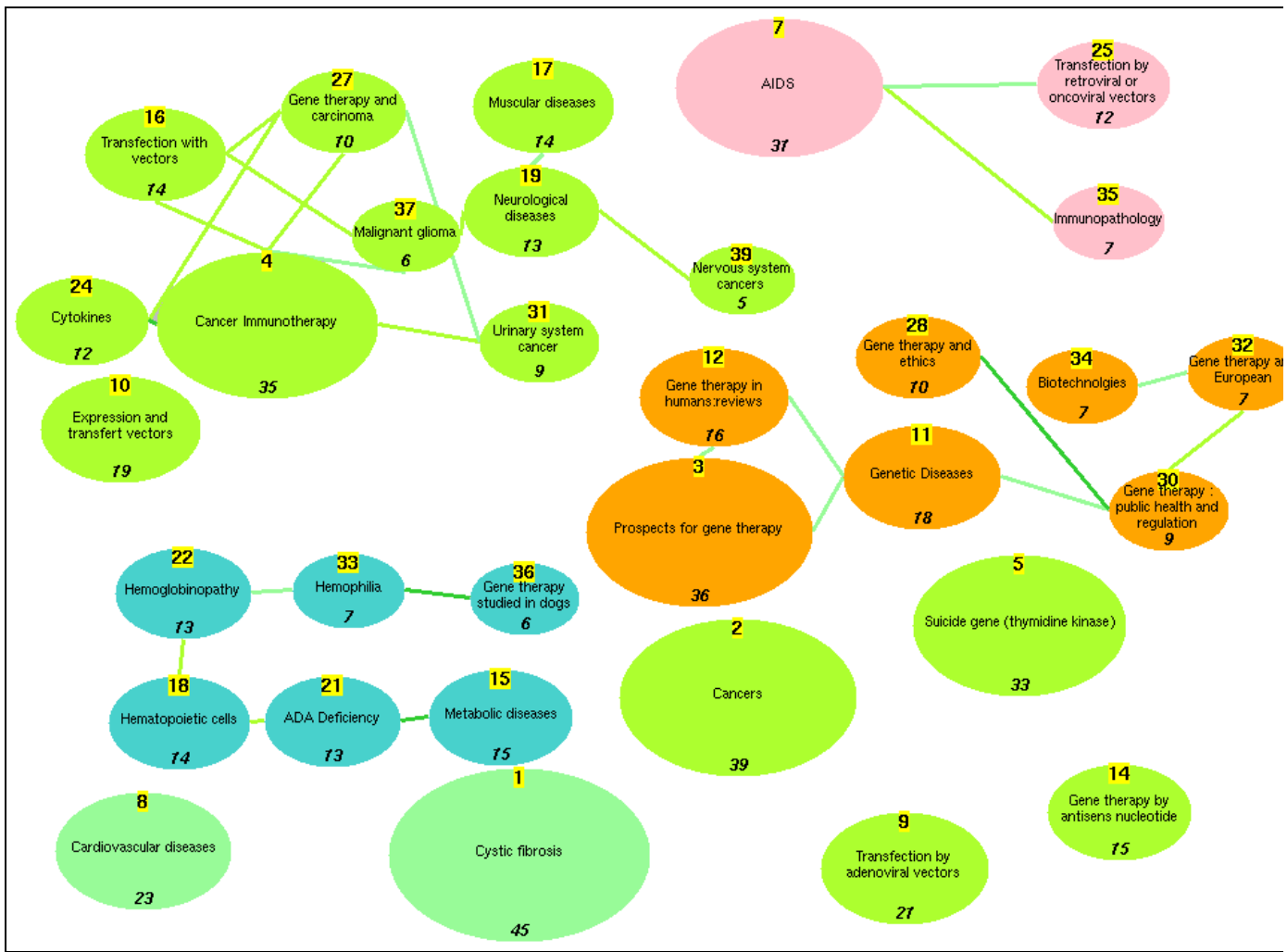


Figure 9: Clusters discovered in a database of scientific reports on gene therapy by the ECAM tool.

Since NetQuestion does not use dictionaries, it can be used for all single-byte character set languages. Boolean queries allowing for phrase and proximity searches as well as for front-, middle-, and end-masking using wildcards. Precise term searches are optimized for Web applications in both Internet and intranet environments.

NetQuestion features high speed in indexing and retrieval where one precise index is built. An optimized and reduced index spans about 35% to 40% of the document size. An ultra compact index of about 10% of the document size can be built for customers not needing full context information (no need for proximity and free-text searches.)

Net Question also provides sophisticated lexical affinities-based ranking for free-text and hybrid queries, advanced relevance ranking, and detection of misspellings in documents, expanding the search request accordingly. NetQuestion can, in some circumstances, even pick up misspellings of words that are not in a dictionary, such as brand names or new technology terms. For instance, during indexing NetQuestion notices that one of the occurrences of "Toyota" is misspelled as "Toyotta." If someone later tries to search for "Toyota", NetQuestion automatically adds "Toyotta" to the query .In addition, linguistically fuzzy searches are available for English documents

The Web Crawler.

The Web Crawler is a robot that starts at one or more Web sites and follows selected HTML links. It retrieves objects of any content type and language, such as HTML, text, images, audio, or video and stores them to the local file system for further processing. For example, an indexer can use HTML and other text documents to build an index of documents. Types and number of levels of HTML links can be selected through a customization step. The Web Crawler can monitor Web-page activities and changes to optimize retrievals.

The Web Crawler Toolkit

In addition to the Web Crawler which is a ready-to-run implementation, the IBM Intelligent Miner for Text includes a **Web Crawler Toolkit** allowing the users to develop Web crawlers according to their needs. The toolkit uses on DB2 to store metadata information, therefore a restricted use version of DB2 Universal Database is included with the product.

The Web crawlers can run on a single machine and can also spawn off a user-specified number of crawler copies that run in parallel or run on multiple machines. These copies can be configured to independently crawl disjoint subsets of very large Webs. The individual crawl results consisting of data objects and their metadata can be shared for subsequent processing across machines either through shared file systems only, or through shared file systems and the parallel query capability of DB2.

A portfolio of technology

In addition to the components in the IBM Intelligent Miner for Text product, IBM laboratories and development teams have other powerful technologies in the pipeline. Some of these are already being deployed to help solve customer's problems; others are still under development. Together these form an evolving portfolio from which solutions can be built to suit many business needs.

Among these components and technologies are

- ▼ Other clustering and visualization applications
- ▼ Prompted Query Refinement
- ▼ Lexical Navigation
- ▼ Advanced feature extraction
- ▼ Feature Extraction for non-English languages
- ▼ Text classification technologies

Clustering and visualization applications

There are a variety of different algorithms for clustering available today. For example, IBM Intelligent Miner for Data does relational data analysis to perform flat clustering, in which groups of similar data items are formed and linked to each other using the technique of Condorcet-clustering. A focused team within IBM uses similar algorithms and information and text analysis framework to build special-purpose Business Intelligence applications. Figure 9 shows an example, which is one of a number described at <http://direct.boulder.ibm.com/bi/demos/ecamdemo.htm>. This application combines the ability to discover the main themes in the collection being analyzed, and the relationships between them as shown in the figure, with a visualization tool which presents the information in a way which is easy to understand.

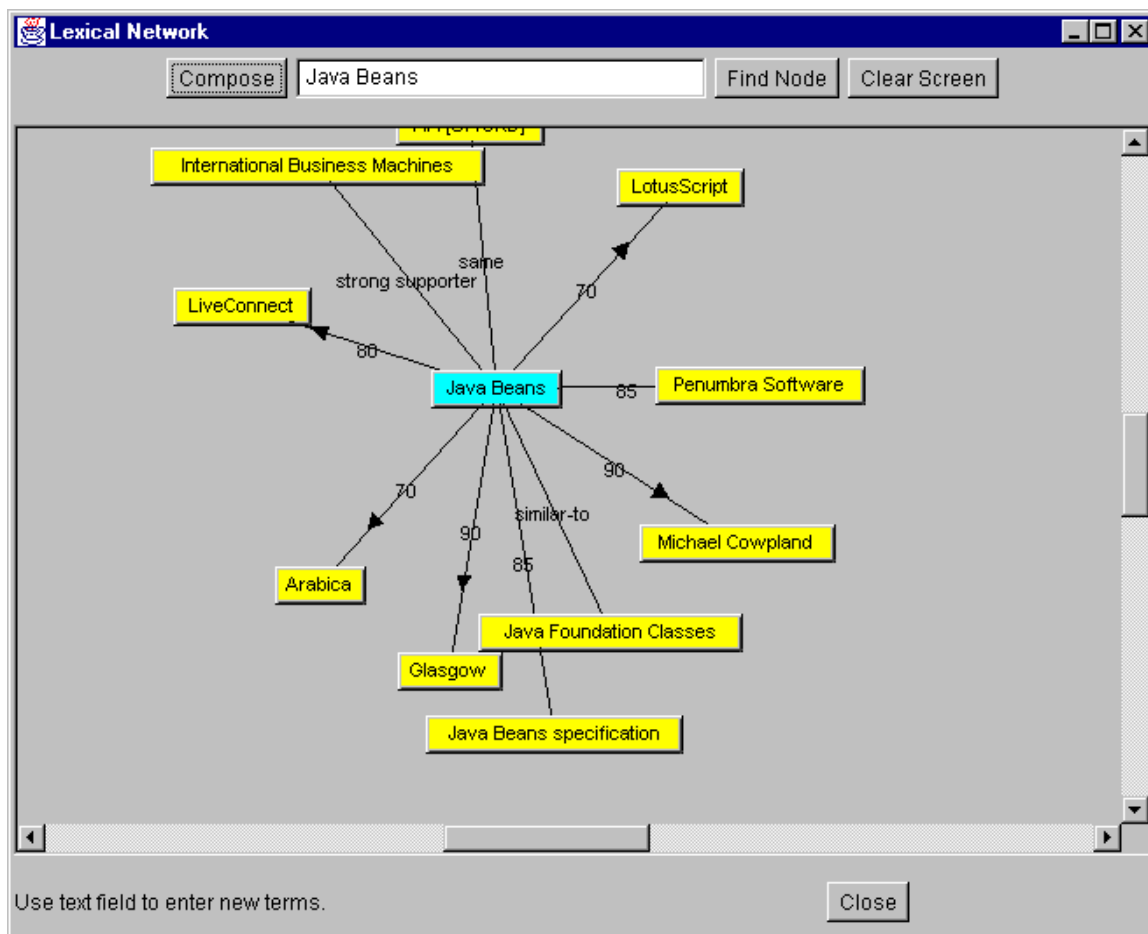


Figure 10: Visualizing the relationships used in Lexical Navigation

Prompted Query Refinement

Text search is a powerful tool for finding relevant information in a mass of documents, if a user’s query adequately describes their information need. But people find it difficult to build queries: the average query submitted to Internet search services is only 2 words long. This is not enough to focus a search in most cases, so typically the user has to read through many documents from a long results list in order to find what he or she is looking for. If the work is done up front, the user has to think of terms and perhaps Boolean conditions to add to their query -- something that is hard to do without training, and takes time and careful thought.

Prompted Query Refinement (PQR) is an application which uses Text Mining technology to help people to build a search query by simply clicking. PQR proposes terms which are relevant to an original short query, but are carefully selected to be excellent search terms if the user chooses to add them to the query. Furthermore, the terms suggested by PQR are typically phrases and names, which carry a lot of meaning. Consequently, it is easier for a user to assess their relevance. PQR is fully described in a recent conference paper.[3]

Some of the technology underlying PQR, for example the feature extraction which recognizes significant vocabulary in documents, is already available in the Intelligent Miner for Text. PQR uses the search engine of

[3] *Lexical Navigation: visually prompted query expansion and refinement*, J.W. Cooper and R.J. Byrd, in ACM Digital Libraries '97 (Association for Computing Machinery, New York NY, 1997), p.237

Intelligent Miner for Text for indexing and searching. The TextMiner engine also includes another approach to query refinement, relevance feedback, which has already been described.

Lexical Navigation

The ability to discern relationships between concepts in a mass of complex data is a key function in Information Mining. Powerful new tools which automatically discover relationships in text data have been developed in IBM Research for text mining applications. Lexical Navigation is an application which uses the network of relationships between the concepts discovered in a collection of documents as an aid to understand and explore the information present in the collection. Figure 10 shows an example of a display from a Lexical Navigation application.

The figure shows the ways in which the concept "Java Beans" is most strongly related to other vocabulary items present in a collection of consultants' reports. In some cases, the relationship is given a numerical strength, based on the tendency of these concepts to co-occur (and hence be linked in the authors' minds). In other cases a label is automatically assigned to the relationship. Note that this feature allows one to use this display to learn that IBM is a strong supporter of Java Beans, without having to read the documents which are the evidence for this relationship. Thus Lexical Navigation shows the user a selected, abstracted, layer of information which is automatically distilled from the collection. The presentation is incremental: any node of interest can be selected and further expanded, to show a network of relationships.

Advanced Feature Extraction

In addition to the technologies in the feature extraction component of the Intelligent Miner for text, IBM laboratories are developing new ways to recognize interesting features in text. For example, one approach is to build an extractor that can learn what to look for based on examples. In another case, an extractor which can find the products of companies is being tested. In yet another example an extractor that can find deadlines in e-mail messages has been developed. Each of these extractors uses a different approach based on scientific expertise in IBM's network of Research laboratories.

Feature Extraction for non-English languages

At different locations within IBM we are actively working on extending the language coverage for our text mining technology. For example, flat clustering works with documents in English, French and German.

Names recognition, which is a very important piece in feature extraction, is being extended to languages other than English. The names extractors are a set of tools for the extraction of proper names from German, Spanish, and Italian text. They extract proper names for persons, including their titles, and organization names of various types. In most cases, the extraction accuracy will be over 90 % and the recall over 80 % of all multi-word proper name occurrences. Here are some examples of names extracted for the various languages.

Person names including many titles, e.g. :

Spanish	"el ministro, Javier Solana", "Manuel Alejandro Castillo"
Italian	"signor Pier Verderio", "SANDRO CURZI"
German	"Oberbürgermeister Wolfgang Assmann", "Johann Sebastian Bach"

Names of public and private organizations, e.g. :

Spanish	"Academia Real Española", "Consejo Superior de Deportes"
Italian	"Commissione Affari Costituzionali", "Stati Uniti d'America"
German	"Adam Opel AG", "Arbeitsgemeinschaft Frankfurter Bürger"

Common place names are treated if occurring as part of the above types, e.g.:

Spanish	"Oficina Española de Turismo a Francfort"
Italian	"Forza Italia"
German	"die Frankfurter Familie Brentano"

In addition to the name string, the inames extractors output the structure of the name. For example, the name of a German company "Adam Opel AG" is recognized to contain a person name "Adam Opel". This is useful for inames' sorting of proper names, as "Adam Opel AG" will be sorted with "Opel AG", but not with "Adam Riese".

The inames extractors also group together the variants of proper names, and the assignment of a canonical form, which is the name providing most information. If, for example, a set of newspaper articles refers to "Bundeskanzler Helmut Kohl", "Bundeskanzler Kohl", and "Helmut Kohl", these forms will be grouped and the first citation will become the canonical form. On the other hand, the citation "Bundeskanzler Dr. Kohl" would be treated separately, as it contains additional information on Helmut Kohl's titles.

Text Classification Technologies

Another example of the depth of technology available for Text Mining is that several approaches to categorizing documents are being developed. Different technical approaches result in categorizers with different strengths and therefore different capabilities. The categorizers shipped with the Intelligent Miner for Text are what are known as centroid neighbor classifiers, which build a compact representation of the "average" document in a category and use that as the basis for deciding which categories a new document belongs in. This approach has the advantages of being able to run fast in a relatively small amount of memory. There are several other categorization techniques, all of which work well on test data. However, for a given application it is advantageous to be able to try different approaches because at the moment there are no accepted rules which can be used to decide which kind of categorizer to use in different situations. The categorizers in the IBM portfolio include examples of the following technical approaches:

Rule induction

A rule induction categorizer automatically learns rules from example documents, and applies them to determine which categories new documents belong in. One advantage of such a categorizer is that the rules can be read and modified, if necessary, by people. Another advantage is that rules can be written based on human expertise, to supplement the automatic learning.

Nearest-neighbor

A nearest-neighbor categorizer compares a new document to documents that have already been categorized, then categories of the most similar documents as the ones for the new document. This is particularly suitable when the category is not well described by an "average" document. This might be the case, for example, if it contains two distinct topics, as might happen over time if the category scheme is modified.