# Text mining

**Ian H. Witten**
*Computer Science, University of Waikato, Hamilton, New Zealand*
*email ihw@cs.waikato.ac.nz*

## Index terms

"Bag of words" model, acronym extraction, authorship ascription, coordinate matching, data mining, document clustering, document frequency, document retrieval, document similarity metrics, entity extraction, hidden Markov models, hubs and authorities, information extraction, information retrieval, key-phrase assignment, key-phrase extraction, knowledge engineering, language identification, link analysis, machine learning, metadata, natural language processing, n-grams, rule learning, syntactic analysis, term frequency, text categorization, text mining, text summarization, token identification, training, wrapper induction.

## 1 Introduction

Text mining is a burgeoning new field that attempts to glean meaningful information from natural language text. It may be loosely characterized as the process of analyzing text to extract information that is useful for particular purposes. Compared with the kind of data stored in databases, text is unstructured, amorphous, and difficult to deal with algorithmically. Nevertheless, in modern culture, text is the most common vehicle for the formal exchange of information. The field of text mining usually deals with texts whose function is the communication of factual information or opinions, and the motivation for trying to extract information from such text automatically is compelling—even if success is only partial.

Four years ago, Hearst [Hearst, 1999] wrote that the nascent field of "text data mining" had "a name and a fair amount of hype, but as yet almost no practitioners." It seems that even the name is unclear: the phrase "text mining" appears 17 times as often as "text data mining" on the Web, according to a popular search engine (and "data mining" occurs 500 times as often). Moreover, the meaning of either phrase is by no means clear: Hearst defines data mining, information access, and corpus-based computational linguistics and discusses the relationship of these to text data mining—but does not define that term. The literature on data mining is far more extensive, and also more focused: there are numerous textbooks and critical reviews that trace its development from roots in machine learning and statistics. Text mining emerged at an unfortunate time in history. Data mining was able to ride the back of the high technology extravaganza throughout the 1990s, and became firmly established as a widely-used practical technology—though the dot com crash may have hit it harder than other areas [Franklin, 2002]. Text mining, in contrast, emerged just before the market crash—the first workshops were held at the *International Machine Learning Conference* in July 1999 and the *International Joint Conference on Artificial Intelligence* in August 1999—and missed the opportunity to gain a solid foothold during the boom years.

The phrase "text mining" is generally used to denote any system that analyzes large quantities of natural language text and detects lexical or linguistic usage patterns in an attempt to extract probably useful (although only probably correct) information [Sebastiani, 2002]. In discussing a topic that lacks a generally accepted definition in a practical Handbook such as this, I have chosen to cast the net widely and take a liberal viewpoint of what should be included, rather than attempting a clear-cut characterization that will inevitably restrict the scope of what is covered.

The remainder of this section discusses the relationship between text mining and data mining, and between text mining and natural language processing, to air important issues concerning the meaning of the term. The article's major section follows: an introduction to the great variety of tasks that involve mining plain text. We then examine the additional leverage that can be obtained when mining semi-structured text such as pages of the World-Wide Web, which opens up a range of new techniques that do not apply to plain text. Following that we indicate, by example, what automatic text mining techniques may aspire to in the future by briefly describing how human "text miners" who are information researchers rather than subject-matter experts may be able to discover new scientific hypotheses solely by analyzing the literature. Finally we review some basic techniques that underpin text mining systems, and look at software tools that are available to help with the work.

## Text mining and data mining

Just as data mining can be loosely described as looking for patterns in data, text mining is about looking for patterns in text. However, the superficial similarity between the two conceals real differences. Data mining can be more fully characterized as the extraction of implicit, previously unknown, and potentially useful information from data [Witten and Frank, 2000]. The information is implicit in the input data: it is hidden, unknown, and could hardly be extracted without recourse to automatic techniques of data mining. With text mining, however, the information to be extracted is clearly and explicitly stated in the text. It's not hidden at all—most authors go to great pains to make sure that they express themselves clearly and unambiguously—and, from a human point of view, the only sense in which it is "previously unknown" is that human resource restrictions make it infeasible for people to read the text themselves. The problem, of course, is that the information is not couched in a manner that is amenable to automatic processing. Text mining strives to bring it out of the text in a form that is suitable for consumption by computers directly, with no need for a human intermediary.

Though there is a clear difference philosophically, from the computer's point of view the problems are quite similar. Text is just as opaque as raw data when it comes to extracting information—probably more so.

Another requirement that is common to both data and text mining is that the information extracted should be "potentially useful." In one sense, this means *actionable*—capable of providing a basis for actions to be taken automatically. In the case of data mining, this notion can be expressed in a relatively domain-independent way: actionable patterns are ones that allow non-trivial predictions to be made on new data from the same source. Performance can be measured by counting successes and failures, statistical techniques can be applied to compare different data mining methods on the same problem, and so on. However, in many text mining situations it is far harder to characterize what "actionable" means in a way that is independent of the particular domain at hand. This makes it difficult to find fair and objective measures of success.

In many data mining applications, "potentially useful" is given a different interpretation: the key for success is that the information extracted must be *comprehensible* in that it helps to explain the data. This is necessary whenever the result is intended for human consumption rather than (or as well as) a basis for automatic action. This criterion is less applicable to text mining because, unlike data mining, the input itself is comprehensible. Text mining with comprehensible output is tantamount to summarizing salient features from a large body of text, which is a subfield in its own right: text summarization.

## Text mining and natural language processing

Text mining appears to embrace the whole of automatic natural language processing and, arguably, far more besides—for example, analysis of linkage structures such as citations in the academic literature and hyperlinks in the Web literature, both useful sources of information that lie outside the traditional domain of natural language processing. But, in fact, most text mining efforts consciously shun the deeper, cognitive, aspects of classic natural language processing in favor of shallower techniques more akin to those used in practical information retrieval.

The reason is best understood in the context of the historical development of the subject of natural language processing. The field's roots lie in automatic translation projects in the late 1940s and early 1950s, whose aficionados assumed that strategies based on word-for-word translation would provide decent and useful rough translations that could easily be honed into something more accurate using techniques based on elementary syntactic analysis. But the sole outcome of these high-profile, heavily-funded projects was the sobering realization that natural language, even at an illiterate child's level, is an astonishingly sophisticated medium that does not succumb to simplistic techniques. It depends crucially on what we regard as "common-sense" knowledge, which despite—or, more likely, because of—its everyday nature is exceptionally hard to encode and utilize in algorithmic form [Lenat, 1995].

As a result of these embarrassing and much-publicized failures, researchers withdrew into "toy worlds"—notably the "blocks world" of geometric objects, shapes, colors, and stacking operations—whose semantics are clear and possible to encode explicitly. But it gradually became apparent that success in toy worlds, though initially impressive, does not translate into success on realistic pieces of text. Toy-world techniques deal well with artificially-constructed sentences of what one might call the "Dick and Jane" variety after the well-known series of eponymous children's stories. But they fail dismally when confronted with real text, whether painstakingly constructed and edited (like this article) or produced under real-time constraints (like informal conversation).

Meanwhile, researchers in other areas simply had to deal with real text, with all its vagaries, idiosyncrasies, and errors. Compression schemes, for example, must work well with all documents, whatever their contents, and avoid catastrophic failure even when processing outrageously deviant files (such as binary files, or completely random input). Information retrieval systems must index documents of all types and allow them to be located effectively whatever their subject matter or linguistic correctness. Key-phrase extraction and text summarization algorithms have to do a decent job on any text file. Practical, working systems in these areas are topic-independent, and most are language-independent. They operate by treating the input as though it were data, not language.

Text mining is an outgrowth of this "real text" mindset. Accepting that it is probably not much, what can be done with unrestricted input? Can the ability to process huge amounts of text compensate for relatively simple techniques? Natural language processing, dominated in its infancy by unrealistic ambitions and swinging in childhood to the other extreme of unrealistically artificial worlds and trivial amounts of text, has matured and now embraces both viewpoints: relatively shallow processing of unrestricted text and relatively deep processing of domain-specific material.

It is interesting that data mining also evolved out of a history of difficult relations between disciplines, in this case machine learning—rooted in experimental computer science, with *ad hoc* evaluation methodologies—and statistics—well-grounded theoretically, but based on a tradition of testing explicitly-stated hypotheses rather than seeking new information. Early machine learning researchers knew or cared little of statistics; early researchers on structured statistical hypotheses remained ignorant of parallel work in machine learning. The result was that similar techniques (for example, decision-tree building and nearest-neighbor learners) arose in parallel from the two disciplines, and only later did a balanced rapprochement emerge.

# 2 Mining plain text

This section describes the major ways in which text is mined when the input is plain natural language, rather than partially-structured Web documents. In each case we provide a concrete example. We begin with problems that involve extracting information for human consumption—text summarization and document retrieval. We then examine the task of assessing document similarity, either to categorize documents into predefined classes or to cluster them in "natural" ways. We also mention techniques that have proven useful in two specific categorization problems—language identification and authorship ascription—and a third—identifying key-phrases—that can be tackled by categorization techniques but also by other means. The next subsection discusses the extraction of structured information, both individual units or "entities" and structured relations or "templates." Finally, we review work on extracting rules that characterize the relationships between entities.

## Extracting information for human consumption

We begin with situations in which information mined from text is expressed in a form that is intended for consumption by people rather than computers. The result is not "actionable" in the sense discussed above, and therefore lies on the boundary of what is normally meant by "text mining."

### *Text summarization*

A text summarizer strives to produce a condensed representation of its input, intended for human consumption [Mani, 2001]. It may condense individual documents or groups of documents. Text compression, a related area [Bell *et al.*, 1990], also condenses documents, but summarization differs in that its output is intended to be human-readable. The output of text compression algorithms is certainly not human-readable, but neither is it actionable—the only operation it supports is decompression, that is, automatic reconstruction of the original text. As a field, summarization differs from many other forms of text mining in that there are people, namely professional abstractors, who are skilled in the art of producing summaries and carry out the task as part of their professional life. Studies of these people and the way they work provide valuable insights for automatic summarization.

Useful distinctions can be made between different kinds of summaries; some are exemplified in Figure 1 (from [Mani, 2001]). An *extract* consists entirely of material copied from the input—for example, one might simply take the opening sentences of a document (Figure 1a) or pick certain key sentences scattered throughout it (Figure 1b). In contrast, an *abstract* contains material that is not present in the input, or at least expresses it in a different way—this is what human abstractors would normally produce (Figure 1c). An *indicative* abstract is intended to provide a basis for selecting documents for closer study of the full text, whereas an *informative* one covers all the salient information in the source at some level of detail [Borko and Bernier, 1975]. A further category is the *critical abstract* [Lancaster, 1991], which evaluates the subject matter of the source document, expressing the abstractor's views on the quality of the author's work (Figure 1d). Another distinction is between a *generic* summary, aimed at a broad readership, and a *topic-focused* one, tailored to the requirements of a particular group of users.

| | | |
|---|---|---|
| (a) | 25%<br>Leading text extract | Four score and seven years ago our fathers brought forth upon this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal. Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met here on a great battlefield of that war. |
| (b) | 25%<br>Another extract | Four score and seven years ago our fathers brought forth upon this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal. Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. The brave men, living and dead, who struggled here, have consecrated it far above our poor power to add or detract. |
| (c) | 15%<br>Abstract | This speech by Abraham Lincoln commemorates soldiers who laid down their lives in the Battle of Gettysburg. It reminds the troops that it is the future of freedom in America that they are fighting for. |
| (d) | 15%<br>Critical abstract | The Gettysburg address, though short, is one of the greatest American speeches. Its ending words are especially powerful—"that government of the people, by the people, for the people, shall not perish from the earth." |

Figure 1 Applying text summarization to the Gettysburg Address

While they are in a sense the archetypal form of text miners, summarizers do not satisfy the condition that their output be actionable.

### Document retrieval

Given a corpus of documents and a user's information need expressed as some sort of query, document retrieval is the task of identifying and returning the most relevant documents. Traditional libraries provide catalogues (whether physical card catalogues or computerized information systems) that allow users to identify documents based on surrogates consisting of *metadata*—salient features of the document such as author, title, subject classification, subject headings, keywords. Metadata is a kind of highly structured (and therefore actionable) document summary, and successful methodologies have been developed for manually extracting metadata and for identifying relevant documents based on it, methodologies that are widely taught in library school (e.g. [Mann, 1993]).

Automatic extraction of metadata (e.g. subjects, language, author, key-phrases; see below) is a prime application of text mining techniques. However, contemporary automatic document retrieval techniques bypass the metadata creation stage and work on the full text of the documents directly [Salton and McGill, 1983]. The basic idea is to index every individual word in the document collection. Effectively, documents are represented as a "bag of words"—that is, the set of words that they contain, along with a count of how often each one appears in the document. Despite the fact that this representation discards the sequential information given by the word order, it underlies many remarkably effective and popular document retrieval techniques. There are some practical problems: how to define a "word," what to do with numbers; these are invariably solved by simple *ad hoc* heuristics. Many practical systems discard common words or "stop words", primarily for efficiency reasons, although suitable compression techniques obviate the need for this [Witten *et al.*, 1999]. A query is expressed as a set, or perhaps a Boolean combination, of words and phrases, and the index is consulted for each word in the query to determine which documents satisfy the query. A well-developed technology of *relevance ranking* allows the salience of each term to be assessed relative to the document collection as a whole, and also relative to each document that contains it. These measures are combined to give an overall ranking of the relevance of each document to the query, and documents are presented in relevance order.

Web search engines are no doubt the most widely-used of document retrieval systems. However, search queries are typically restricted to just a few words or phrases—usually one or two. In contrast, queries made by professionals to advanced document retrieval systems are often far more complex and specific. For example, Figure 2 shows one of the "topics" or queries used for evaluation in TREC, a series of conferences in which different document retrieval systems are compared on queries written by experienced users of information systems [Harman, 1995].

| | |
|---|---|
| Topic: | Financing AMTRAK |
| Description: | A document will address the role of the Federal Government in financing the operation of the National Railroad Transportation Corporation (AMTRAK) |
| Narrative: | A relevant document must provide information on the government's responsibility to make AMTRAK an economically viable entity. It could also discuss the privatization of AMTRAK as an alternative to continuing government subsidies. Documents comparing government subsidies given to air and bus transportation with those provided to AMTRAK would also be relevant. |

Figure 2 Sample TREC query

A set of documents returned in response to a query is a kind of *topic-focused extract* from the corpus. Like a summary, it is not normally actionable.

### Information retrieval

Information retrieval might be regarded as an extension to document retrieval where the documents that are returned are processed to condense or extract the particular information sought by the user. Thus document retrieval could be followed by a text summarization stage that focuses on the query posed by the user, or an information extraction stage using techniques described below. In practice, however, standard textbooks (e.g. [Baeza-Yates and Ribiero-Neto, 1999]) use the term simply for plain document retrieval. Of course, the granularity of documents may be adjusted so that each individual subsection or paragraph comprises a unit in its own right, in an attempt to focus results on individual nuggets of information rather than lengthy documents.

## Assessing document similarity

Many text mining problems involve assessing the similarity between different documents; for example, assigning documents to pre-defined categories and grouping documents into natural clusters. These are standard problems in data mining too, and have been a popular focus for research in text mining, perhaps because the success of different techniques can be evaluated and compared using standard, objective, measures of success.

### Text categorization

Text categorization (or text classification) is the assignment of natural language documents to pre-defined categories according to their content [Sebastiani, 2002]. The set of categories is often called a "controlled vocabulary." Document categorization is a long-standing traditional technique for information retrieval in libraries, where subjects rival authors as the predominant gateway to library contents—although they are far harder to assign objectively than authorship. The Library of Congress Subject Headings (LCSH) are a comprehensive and widely used controlled vocabulary for assigning subject descriptors. They occupy five large printed volumes of 6,000 pages each—perhaps two million descriptors in all. The aim is to provide a standardized vocabulary for all categories of knowledge, descending to quite a specific level, so that books—on any subject, in any language—can be described in a way that helps librarians retrieve all books on a given subject [Witten and Bainbridge, 2003].

Automatic text categorization has many practical applications, including indexing for document retrieval, automatically extracting metadata, word sense disambiguation by detecting the topics a document covers, and organizing and maintaining large catalogues of Web resources. As in other areas of text mining, until the 1990s text categorization was dominated by *ad hoc* techniques of "knowledge engineering" that sought to elicit categorization rules from human experts and code them into a system that could apply them automatically to new documents. Since then—and particularly in the research community—the dominant approach has been to use techniques of machine learning to infer categories automatically from a training set of pre-classified documents. Indeed, text categorization is a hot topic in machine learning today.

The pre-defined categories are symbolic labels with no additional semantics. When classifying a document, no information is used except for the document's content itself. Some tasks constrain documents to a single category, whereas in others each document may have many categories. Sometimes category labeling is probabilistic rather than deterministic, or the objective is to rank the categories by their estimated relevance to a particular document. Sometimes documents are processed one by one, with a given set of classes; alternatively there may be a single class—perhaps a new one that has been added to the set—and the task is to determine which documents it contains.

Many machine learning techniques have been used for text categorization. Early efforts used rules and decision trees. Figure 3 shows a rule (from [Apte *et al.*, 1994]) for assigning a document to a particular category. The italicized words are terms that may or may not occur in the document text, and the rule specifies a certain logical combination of occurrences. This particular rule pertains to the Reuters collection of pre-classified news articles, which is widely used for document classification research (e.g. [Hayes *et al.*, 1990]). WHEAT is the name of one of the categories.

If (*wheat* & *farm*)
or (*wheat* & *commodity*)
or (*bushels* & *export*)
or (*wheat* & *tonnes*)
or (*wheat* & *winter* & ¬ *soft*)
then WHEAT

Figure 3 Rule for assigning a document to the category WHEAT

Rules like this can be produced automatically using standard techniques of machine learning [Mitchell, 1997; Witten and Frank, 2000]. The training data comprises a substantial number of sample documents for each category. Each document is used as a positive instance for the category labels that are associated with it and a negative instance for all other categories. Typical approaches extract "features" from each document, and use the feature vectors as input to a scheme that learns how to classify documents. Using words as features—perhaps a small number of well-chosen words, or perhaps all words that appear in the document except stop words—and word occurrence counts as feature values, a model is built for each category. The documents in that category are positive examples and the remaining documents negative ones. The model predicts whether or not that category is assigned to a new document based on the words in it, and their occurrence counts. Given a new document, each model is applied to determine which categories to assign. Alternatively, the learning method may produce a likelihood of the category being assigned, and if, say, five categories were sought for the new document, those with the highest likelihoods could be chosen.

If the features are words, documents are represented using the "bag of words" model described above under document retrieval. Sometimes word counts are discarded and the "bag" is treated merely as a set (Figure 3, for example, only uses the presence of words, not their counts). Bag (or set) of words models neglect word order and contextual effects. Experiments have shown that more sophisticated representations—for example, ones that detect common phrases and treat them as single units—do not yield significant improvement in categorization ability (e.g. [Lewis, 1992]; [Apte *et al.*, 1994]; [Dumais *et al.*, 1998]), although it seems likely that better ways of identifying and selecting salient phrases will eventually pay off. Each word is a "feature". Because there are so many of them, problems arise with some machine learning methods, and a selection process is often used that identifies only a few salient features. A large number of feature selection and machine learning techniques have been applied to text categorization [Sebastiani, 2002].

## *Document clustering*

Text categorization is a kind of "supervised" learning where the categories are known beforehand and determined in advance for each training document. In contrast, document clustering is "unsupervised" learning in which there is no predefined category or "class," but groups of documents that belong together are sought. For example, document clustering assists in retrieval by creating links between similar documents, which in turn allows related documents to be retrieved once one of the documents has been deemed relevant to a query [Martin, 1995].

Clustering schemes have seen relatively little application in text mining applications. While attractive in that they do not require training data to be pre-classified, the algorithms themselves are generally far more computation-intensive than supervised schemes ([Willett, 1988] surveys classical document clustering methods). Processing time is particularly significant in domains like text classification, in which instances may be described by hundreds or thousands of attributes. Trials of unsupervised schemes include [Aone *et al.,* 1996], who use the conceptual clustering scheme COBWEB [Fisher, 1987] to induce natural groupings of close-captioned text associated with video newsfeeds; [Liere and Tadepalli, 1996], who explore the effectiveness of AutoClass [Cheeseman *et al.,* 1988] in producing a classification model for a portion of the Reuters corpus;

and [Green and Edwards, 1996], who use AutoClass to cluster news items gathered from several sources into "stories," which are groupings of documents covering similar topics.

## *Language identification*

Language identification is a particular application of text categorization. A relatively simple categorization task, it provides an important piece of metadata for documents in international collections. A simple representation for document categorization is to characterize each document by a profile that consists of the "*n*-grams," or sequences of *n* consecutive letters, that appear in it. This works particularly well for language identification. Words can be considered in isolation—the effect of word sequences can safely be neglected. Documents are preprocessed by splitting them into word tokens containing letters and apostrophes (the usage of digits and punctuation is not especially language-dependent), padding each token with spaces, and generating all possible *n*-grams of length 1 to 5 for each word in the document. These *n*-grams are counted and sorted into frequency order to yield the document profile.

The most frequent 300 or so *n*-grams are highly correlated with the language. The highest ranking ones are mostly unigrams consisting of one character only, and simply reflect the distribution of letters of the alphabet in the document's language. Starting around rank 300 or so, the frequency profile begins to be more specific to the document's topic. Using a simple metric for comparing a document profile with a category profile, each document's language can be identified with high accuracy [Cavnar and Trenkle, 1994].

An alternative approach is to use words instead of *n*-grams, and compare occurrence probabilities of the common words in the language samples with the most frequent words of the test data. This method works as well as the *n*-gram scheme for sentences longer than about 15 words, but is less effective for short sentences such as titles of articles and news headlines [Grefenstette, 1995].

## *Ascribing authorship*

Author metadata is one of the primary attributes of most documents. It is usually known and need not be mined, but in some cases authorship is uncertain and must be guessed from the document text. Authorship ascription is often treated as a text categorization problem. However, there are sensitive statistical tests that can be used instead, based on the fact that each author has a characteristic vocabulary whose size can be estimated statistically from a corpus of their work.

For example, *The Complete Works of Shakespeare* (885,000 words) contains 31,500 different words, of which 14,400 appear only once, 4300 twice, and so on. If another large body of work by Shakespeare were discovered, equal in size to his known writings, one would expect to find many repetitions of these 31,500 words along with some new words that he had not used before. According to a simple statistical model, the number of new words should be about 11,400 [Efron and Thisted, 1976]. Furthermore, one can estimate the total number of words known by Shakesepeare from the same model: the result is 66,500 words. (For the derivation of these estimates, see [Efron and Thisted, 1976].) This statistical model was unexpectedly put to the test ten years after it was developed [Kolata, 1986]. A previously unknown poem, suspected to have been penned by Shakespeare, was discovered in a library in Oxford, England. Of its 430 words, statistical analysis predicted that 6.97 would be new, with a standard deviation of ±2.64. In fact, nine of them were (*admiration*, *besots*, *exiles*, *inflection*, *joying*, *scanty*, *speck*, *tormentor* and *twined*). It was predicted that there would be 4.21±2.05 words that Shakespeare had used only once; the poem contained seven—only just outside the range. 3.33±1.83 should have been used exactly twice before; in fact five were. Although this does not prove authorship, it does suggest it—particularly since comparative analyses of the vocabulary of Shakespeare's contemporaries indicate substantial mismatches.

Text categorization methods would almost certainly be far less accurate than these statistical tests, and this serves as a warning not to apply generic text mining techniques indiscriminately. However, the tests are useful only when a huge sample of pre-classified text is available—in this case, the life's work of a major author.

## *Identifying key-phrases*

In the scientific and technical literature, keywords and key-phrases are attached to documents to give a brief indication of what they are about. (Henceforth we use the term “key-phrase” to subsume keywords, that is, one-word key-phrases.) Key-phrases are a useful form of metadata because they condense documents into a few pithy phrases that can be interpreted individually and independently of each other.

Given a large set of training documents with key-phrases assigned to each, text categorization techniques can be applied to assign appropriate key-phrases to new documents. The training documents provide a predefined set of key-phrases from which all key-phrases for new documents are chosen—a controlled vocabulary. For each key-phrase the training data defines a set of documents that are associated with it. For each key-phrase, standard machine learning techniques are used to create a “classifier” from the training documents, using those associated with it as positive examples and the remainder as negative examples. Given a new document, it is processed by each key-phrase’s classifier. Some classify it positively—in other words, it belongs to the set of documents associated with that key-phrase—while others classify it negatively—it does not. Key-phrases are assigned to the new document accordingly. The process is called key-phrase *assignment* because phrases from an existing set are assigned to documents.

There is an entirely different method for inferring key-phrase metadata called key-phrase *extraction*. Here, all the phrases that occur in the document are listed and information retrieval heuristics are used to select those that seem to characterize it best. Most key-phrases are noun phrases, and syntactic techniques may be used to identify these and ensure that the set of candidates contains only noun phrases. The heuristics used for selection range from simple ones such as the position of the phrase’s first occurrence in the document to more complex ones such as the occurrence frequency of the phrase in the document versus its occurrence frequency in a corpus of other documents in the subject area. The training set is used to tune the parameters that balance these different factors.

With key-phrase assignment, the only key-phrases that can be assigned are ones that are have already been used for training documents. This has the advantage that all key-phrases are well-formed, but the disadvantage that novel topics cannot be accommodated. The training set of documents must therefore be large and comprehensive. In contrast, key-phrase *extraction* is open-ended: phrases are selected from the document text itself. There is no particular problem with novel topics, but idiosyncratic or mal-formed key-phrases may be chosen. A large training set is not needed because it is only used to set parameters for the algorithm.

Key-phrase extraction works as follows. Given a document, rudimentary lexical techniques based on punctuation and common words are used to extract a set of candidate phrases. Then, features are computed for each phrase, like how often it appears in the document (normalized by how often that phrase appears in other documents in the corpus), how often it has been used as a key-phrase in other training documents, whether it occurs in the title, abstract, or section headings, whether it occurs in the title of papers cited in the reference list, and so on. The training data is used to form a model that takes these features and predicts whether or not a candidate phrase will actually appear as a key-phrase—this information is known for the training documents. Then the model is applied to extract likely key-phrases from new documents. Such models have been built and used to assign key-phrases to technical papers; simple machine learning schemes (e.g. Naïve Bayes) seem adequate for this task.

To give an indication of the success of machine learning on this problem, Figure 4 shows the titles of three research articles and two sets of key-phrases for each one [Frank *et al*., 1999]. One set contains the key-phrases assigned by the article’s author; the other was determined automatically from its full text. Phrases in common between the two sets are italicized. In each case the author’s key-phrases and the automatically extracted key-phrases overlap, but it is not too difficult to guess which are the author’s. The giveaway is that the machine learning scheme, in addition to choosing several good key-phrases, also chooses some that authors are unlikely to use—for example, *gauge*, *smooth*, and especially *garbage*! Despite the anomalies, the automatically extracted lists give a reasonable characterization of the papers. If no author-specified key-phrases were available, they could prove useful for someone scanning quickly for relevant information.

| Protocols for secure, atomic transaction execution in electronic commerce | | Neural multigrid for gauge theories and other disordered systems | |
|---|---|---|---|
| anonymity | *atomicity* | disordered systems | disordered |
| *atomicity* | *auction* | *gauge fields* | gauge |
| *auction* | customer | *multigrid* | *gauge fields* |
| *electronic commerce* | *electronic commerce* | neural multigrid | interpolation kernels |
| privacy | intruder | neural networks | length scale |
| real-time | merchant | | *multigrid* |
| *security* | protocol | | smooth |
| *transaction* | *security* | | |
| | third party | | |
| | *transaction* | | |

| Proof nets, garbage, and computations | |
|---|---|
| *cut-elimination* | cut |
| linear logic | *cut elimination* |
| proof nets | garbage |
| sharing graphs | proof net |
| typed lambda-calculus | weakening |

Figure 4 Titles and key-phrases—author- and machine-assigned—for three papers

## Extracting structured information

An important form of text mining takes the form of a search for structured data inside documents. Ordinary documents are full of structured information: phone numbers, fax numbers, street addresses, email addresses, email signatures, abstracts, tables of contents, lists of references, tables, figures, captions, meeting announcements, Web addresses, and more. In addition, there are countless domain-specific structures, such as ISBN numbers, stock symbols, chemical structures, and mathematical equations. Many short documents describe a particular kind of object or event, and in this case elementary structures are combined into a higher-level composite that represent the document's entire content. In constrained situations, the composite structure can be represented as a "template" with slots that are filled by individual pieces of structured information. From a large set of documents describing similar objects or events it may even be possible to infer rules that represent particular patterns of slot-fillers.

Applications for schemes that identify structured information in text are legion. Indeed, in general interactive computing, users commonly complain that they cannot easily take action on the structured information found in everyday documents [Nardi *et al.*, 1998].

### *Entity extraction*

Many practical tasks involve identifying linguistic constructions that stand for objects or "entities" in the world. Often consisting of more than one word, these terms act as single vocabulary items, and many document processing tasks can be significantly improved if they are identified as such. They can aid searching, interlinking and cross-referencing between documents, the construction of browsing indexes, and can comprise machine-processable "metadata" which, for certain operations, act as a surrogate for the document contents.

Examples of such entities are

Names of people, places, organizations, products
E-mail addresses, URLs
Dates, numbers, sums of money
Abbreviations
Acronyms and their definition

Multiword terms.

Some of these items can be spotted by a dictionary-based approach, using lists of personal names and organizations, information about locations from gazetteers, abbreviation and acronym dictionaries, and so on. Here the lookup operation should recognize legitimate variants. This is harder than it sounds—for example (admittedly an extreme one), the name of the Libyan leader *Muammar Qaddafi* is represented in 47 different ways on documents that have been received by the Library of Congress [Mann, 1993]! A central area of library science is devoted to the creation and use of standard names for authors and other bibliographic entities (called “authority control”).

In most applications, novel names appear. Sometimes these are composed of parts that have been encountered before, say *John* and *Smith*, but not in that particular combination. Others are recognizable by their capitalization and punctuation pattern (e.g. *Randall B. Caldwell*). Still others—particularly certain foreign names—will be recognizable because of peculiar language statistics (e.g. *Kung-Kui Lau*). Others will not be recognizable except by capitalization, which is an unreliable guide—particularly when only one name is present. Names that begin a sentence cannot be distinguished on this basis from other words. It is not always completely clear what to “begin a sentence” means: in some typographic conventions, itemized points have initial capitals but no terminating punctuation. Of course, words that are not names are sometimes capitalized (e.g. important words in titles—and, in German, all nouns). And a small minority of names are conventionally written unpunctuated and in lower case (e.g. some English names starting with *ff*, the poet *e e cummins* the singer *k d lang*). Full personal name recognition conventions are surprisingly complex, involving baronial prefixes in different languages (e.g. *von*, *van*, *de*), suffixes (*Snr*, *Jnr*), and titles (*Mr*, *Ms*, *Rep.*, *Prof.*, *General*).

It is generally impossible to distinguish personal names from other kinds of names in the absence of context or domain knowledge. Consider places like *Berkeley*, *Lincoln*, *Washington*; companies like *du Pont*, *Ford*, even *General Motors*; product names like *Mr Whippy* and *Dr Pepper*; book titles like *David Copperfield* or *Moby Dick*. Names of organizations present special difficulties because they can contain linguistic constructs, as in *The Food and Drug Administration* (contrast *Lincoln and Washington*, which conjoins two separate names) or the *League of Nations* (contrast *General Motors of Detroit*, which qualifies one name with a different one).

Some artificial entities like e-mail addresses and URLs are easy to recognize because they are specially designed for machine processing. They can be unambiguously detected by a simple grammar, usually encoded in a regular expression, for the appropriate pattern. Of course, this is exceptional: these items are not part of “natural” language.

Other entities can be recognized by explicit grammars—indeed, one might define structured information as “data recognizable by a grammar.” Dates, numbers, and sums of money are good examples that can be captured by simple lexical grammars. However, in practice things are often not so easy as they might appear. There may be a proliferation of different patterns, and novel ones may occur. The first step in processing is usually to divide the input into lexical tokens or “words” (e.g. split at white space or punctuation). While words delimited by non-alphanumeric characters provide a natural tokenization for many examples, such a decision will turn out to be restrictive in particular cases, for it precludes patterns that adopt a non-standard tokenization—such as *30Jul98*. In general, any prior division into tokens runs the risk of obscuring information.

To illustrate the degree of variation in these items, Figure 5 shows examples of items that are recognized by IBM’s “Intelligent Miner for Text” software [Tkach, 1998]. Dates include standard textual forms for absolute and relative dates. Numbers include both absolute numbers and percentages, and can be written in numerals or spelled out as words. Sums of money can be expressed in various currencies.

| Dates | Numbers | Sums of money |
|---|---|---|
| “March twenty-seventh, nineteen ninety-seven” | “one thousand three hundred and twenty-seven” | “twenty-seven dollars” |
| “March 27, 1997” | “thirteen twenty-seven” | “DM 27” |
| “next March 27th” | “1327” | “27,000 dollars USA” |
| “tomorrow” | “twenty-seven percent” | “27,000 marks Germany” |
| “a year ago” | 27% | |

Figure 5 Sample information items

Most abbreviations can only be identified using dictionaries. Many acronyms, however, can be detected automatically—and technical, commercial and political documents make extensive use of them. Identifying acronyms, and their definitions, in documents is a good example of a text mining problem that can usefully be tackled using simple heuristics.

The dictionary definition of “acronym” is

> A word formed from the first (or first few) letters of a series of words, as *radar*, from *ra*dio *d*etecting *a*nd *r*anging.

Acronyms are often defined by following (or preceding) their first use with a textual explanation—as in this example. Heuristics can be developed to detect situations where a word is spelled out by the initial letters of an accompanying phrase. Three simplifying assumptions that vastly reduce the computational complexity of the task while sacrificing the ability to detect just a few acronyms are to consider (a) only acronyms made up of three or more letters; (b) only the first letter of each word for inclusion in the acronym, and (c) acronyms that are written either fully- or mostly-capitalized. In fact, the acronym *radar* breaks both (b) and (c); it involves the first *two* letters of the word *radio*, and, like most acronyms that have fallen into general use, it is rarely capitalized. However, the vast majority of acronyms that pervade today’s technical, business, and political literature satisfy these assumptions, and are relatively easy to detect. Once detected, acronyms can be added to a dictionary so that they are recognized elsewhere as abbreviations. Of course, many acronyms are ambiguous: the Acronym Finder Web site (at www.mtnds.com/af/) has 27 definitions for *CIA*, ranging from *Central Intelligence Agency* and *Canadian Institute of Actuaries* to *Chemiluminescence Immunoassay*. In ordinary text this ambiguity rarely poses a problem, but in large document collections context and domain knowledge will be necessary for disambiguation.

## *Information extraction*

“Information extraction” is used to refer to the task of filling templates from natural language input [Appelt, 1999], one of the principal subfields of text mining. A commonly-cited domain is that of terrorist events, where the template may include slots for the perpetrator, the victim, type of event, where and when it occurred, etc. In the late 1980s, DARPA instituted a series of “Message understanding conferences” (MUC) to focus efforts on information extraction on particular domains and to compare emerging technologies on a level basis. MUC-1 (1987) and MUC-2 (1989) focused on messages about naval operations; MUC-3 (1991) and MUC-4 (1992) studied news articles about terrorist activity; MUC-5 (1993) and MUC-6 (1995) looked at news articles about joint ventures and management changes respectively; and MUC-7 (1997) examined news articles about space vehicle and missile launches. Figure 6 shows an example of a MUC-7 query. The outcome of information extraction would be to identify relevant news articles and, for each one, fill out a template like that shown.

| | |
|---|---|
| Query | “A relevant article refers to a vehicle launch that is scheduled, in progress or has actually occurred and must minimally identify the payload, the date of the launch, whether the launch is civilian or military, the function of the mission and its status.” |
| Template | `Vehicle:`<br>`Payload:`<br>`Mission Date:`<br>`Mission Site:`<br>`Mission Type (military, civilian):`<br>`Mission Function (test, deploy, retrieve):`<br>`Mission status (succeeded, failed, in progress, scheduled):` |

Figure 6 Sample query and template from MUC-7

Unlike text summarization and document retrieval, information extraction in this sense is not a task commonly undertaken by people because the extracted information must come from each individual article taken in isolation—the use of background knowledge and domain-specific inference are specifically forbidden. It turns out to be a difficult task for people, and inter-annotator agreement is said to lie in the 60%–80% range [Appelt, 1999].

The first job of an information extraction system is entity extraction, discussed earlier. Once this has been done, it is necessary to determine the relationship between the entities extracted, which involves syntactic parsing of the text. Typical extraction problems address simple relationships among entities, such as finding the predicate structure of a small set of pre-determined propositions. These are usually simple enough to be captured by shallow parsing techniques such as small finite-state grammars—a far easier proposition than a full linguistic parse. It may be necessary to determine the attachment of prepositional phrases and other modifiers, which may be restricted by type constraints that apply in the domain under consideration. Another problem, which is not so easy to resolve, is pronoun reference ambiguity. This arises in more general form as “co-reference ambiguity”: whether one noun phrase refers to the same real-world entity as another. Again, [Appelt, 1999] describes these and other problems of information extraction.

Machine learning has been applied to the information extraction task by seeking pattern-match rules that extract fillers for slots in the template (e.g. [Soderland, *et al*, 1995]; [Huffman, 1996]; [Freitag, 2000]). As an example, we describe a scheme investigated by [Califf and Mooney, 1999], in which pairs comprising a document and a template manually extracted from it are presented to the system as training data. A bottom-up learning algorithm is employed to acquire rules that detect the slot fillers from their surrounding text. The rules are expressed in pattern-action form, and the patterns comprise constraints on words in the surrounding context and the slot-filler itself. These constraints involve the words included, their part-of speech tags, and their semantic classes.

Califf and Mooney investigated the problem of extracting information from job ads such as those posted on Internet newsgroups. Figure 7 shows a sample message and filled template of the kind that might be supplied to the program as training data.

Newsgroup posting

```
Telecommunications. SOLARIS Systems
Administrator. 38-44K. Immediate need

Leading telecommunications firm in need
of an energetic individual to fill the
following position in our offices in
Kansas City, Missouri:

  SOLARIS SYSTEMS ADMINISTRATOR
  Salary: 38-44K with full benefits
  Location: Kansas City, Missouri
```

Filled template

```
Computer_science_job
Title: SOLARIS Systems Administrator
Salary: 38-44K
State: Missouri
City: Kansas City
Platform: SOLARIS
Area: telecommunication
```

Figure 7 Sample message and filled template

This input provides support for several rules. One example is "a noun phrase of 1 or 2 words, preceded by the word *in* and followed by a comma and a noun phrase with semantic tag *State*, should be placed in the template's *City* slot." The strategy for determining rules is to form maximally specific rules based on each example and then generalize the rules produced for different examples. For instance, from the phrase *offices in Kansas City, Missouri* in the newsgroup posting in Figure 7 a maximally specific rule can be derived that assigns the phrase *Kansas City* to the *City* slot in a context where it is preceded by *offices in* and followed by "*, Missouri*", with the appropriate parts of speech and semantic tags. A second newsgroup posting that included the phrase *located in Atlanta, Georgia*, with *Atlanta* occupying the filled template's *City* slot, would produce a similar maximally specific rule. The rule generalization process takes these two specific rules, notes the commonalities, and determines the general rule for filling the *City* slot cited above.

## *Learning rules from text*

Taking information extraction a step further, the extracted information can be used in a subsequent step to learn rules—not rules about how to extract information, but rules that characterize the content of the text itself. Following on from the project described above to extract templates from job postings in internet newsgroups, a database was formed from several hundred postings, and rules were induced from the database for the fillers of the slots for *Language*, *Platform*, *Application*, and *Area* slots [Nahm and Mooney, 2000]. Figure 8 shows some of the rules that were found.

If *Language* contains both *HTML* and *DHTML*, then *Language* also contains *XML*
If *Application* contains *Dreamweaver 4* and *Area* is *Web design* then *Application* also contains *Photoshop 6*
If *Application* is *ODBC* then *Language* is *JSP*
If *Language* contains both *Perl* and *HTML* then *Platform* is Linux

Figure 8 Sample rules induced from job postings database.

In order to create the database, templates were constructed manually from a few newsgroup postings. From these, information extraction rules were learned as described above. These rules were then used to extract information automatically from the other newsgroup postings. Finally, the whole database so extracted was input to standard data mining algorithms to infer rules for filling the four chosen slots. Both prediction rules—that is, rules predicting the filler for a predetermined slot—and association rules—that is, rules predicting the value of *any* slot—were sought. Standard techniques were employed: C4.5Rules [Quinlan, 1993] and Ripper [Cohen, 1995] for prediction rules and Apriori [Agrawal and Srikant, 1994] for association rules.

Nahm and Mooney [Nahm and Mooney, 2000] concluded that information extraction based on a few manually-constructed training examples could compete with an entire manually-constructed database in terms of the quality of the rules that were inferred. However, this success probably hinged on the highly structured domain of job postings in a tightly constrained area of employment. Subsequent work on inferring rules from book descriptions on the Amazon Web site [Nahm and Mooney, 2002] produced rules that, though interesting, seem rather less useful in practice.

# 3 Mining structured text

Much of the text that we deal with today—especially on the Internet—contains explicit structural markup and thus differs from traditional plain text. Some markup is internal and indicates document structure or format; some is external and gives explicit hypertext links between documents. These information sources give additional leverage for mining Web documents. Both sources of information are generally extremely noisy: they involve arbitrary and unpredictable choices by individual page designers. However, these disadvantages are offset by the overwhelming amount of data that is available, which is relatively unbiased because it is aggregated over many different information providers. Thus "Web mining" is emerging as a new subfield, similar to text mining but taking advantage of the extra information available in Web documents, particularly hyperlinks—and even capitalizing on the existence of topic directories in the Web itself to improve results [Chakrabarti, 2003].

We briefly review three techniques for mining structured text. The first, wrapper induction, uses internal markup information to increase the effectiveness of text mining in marked-up documents. The remaining two, document clustering and determining the "authority" of Web documents, capitalize on the external markup information that is present in hypertext in the form of explicit links to other documents.

## *Wrapper induction*

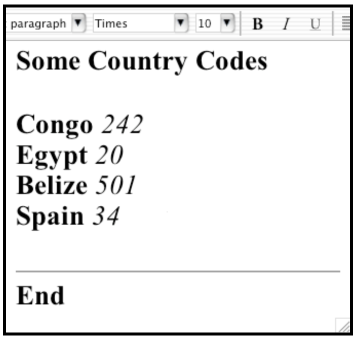
Internet resources that contain relational data—telephone directories, product catalogs, etc.—use formatting markup to clearly present the information they contain to users. However, with standard HTML, it is quite difficult to extract data from such resources in an automatic way. The XML markup language is designed to overcome these problems by encouraging page authors to mark their content in a way that reflects document structure at a detailed level; but it is not clear to what extent users will be prepared to share the structure of their documents fully in XML, and even if they do, huge numbers of legacy pages abound.

Many software systems use external online resources by hand-coding simple parsing modules, commonly called "wrappers," to analyze the page structure and extract the requisite information. This is a kind of text mining, but one that depends on the input having a fixed, predetermined structure from which information can be extracted algorithmically. Given that this assumption is satisfied, the information extraction problem is relatively trivial. But this is rarely the case. Page structures vary; errors that are insignificant to human readers throw automatic extraction procedures off completely; Web sites evolve. There is a strong case for automatic induction of wrappers to reduce these problems when small changes occur, and to make it easier to produce new sets of extraction rules when structures change completely.

Figure 8 shows an example taken from [Kushmerick *et al.*, 1997] in which a small Web page is used to present some relational information. Below is the HTML code from which the page was generated; below that is a wrapper, written in an informal pseudo-code, that extracts relevant information from the HTML. Many different wrappers could be written; in this case the algorithm is based on the formatting information present in the HTML—the fact that countries are surrounded by <B> … </B> and country codes by <I> … </I>. Used in isolation, this information fails because other parts of the page are rendered in boldface too. Consequently the wrapper in

Figure 8c uses additional information—the <P> that precedes the relational information in Figure 8b and the <HR> that follows it—to constrain the search. This wrapper is a specific example of a generalized structure that parses a page into a *head*, followed by a sequence of relational items, followed by a *tail*; where specific delimiters are used to signal the end of the head, the items themselves, and the beginning of the tail.

(a)



**Some Country Codes**

**Congo** *242*
**Egypt** *20*
**Belize** *501*
**Spain** *34*

**End**

(b)

```
<HTML><TITLE>Some Country Codes</TITLE>
<BODY><B>Some Country Codes</B><P>
<B>Congo</B> <I>242</I><BR>
<B>Egypt</B> <I>20</I><BR>
<B>Belize</B> <I>501</I><BR>
<B>Spain</B> <I>34</I><BR>
<HR><B>End</B></BODY></HTML>
```

(c)

```
ExtractCountryCodes(page P)
  Skip past first occurrence of <P> in P
  While next <B> is before next <HR> in P
    For each [s, t] ∈ {[<B>,</B>], [<I>,</I>]}
      Skip past next occurrence of s in P
      Extract attribute from P to next occurrence of t
  Return extracted tuples
```

Figure 8 Web page, underlying HTML, and wrapper extracting relational information

It is possible to infer such wrappers by induction from examples that comprise a set of pages and tuples representing the information derived from each page. This can be done by iterating over all choices of delimiters, stopping when a consistent wrapper is encountered. One advantage of automatic wrapper induction is that recognition then depends on a minimal set of cues, providing some defense against extraneous text and markers in the input. Another is that when errors are caused by stylistic variants it is a simple matter to add these to the training data and re-induce a new wrapper that takes them into account.

## *Document clustering with links*

Document clustering techniques are normally based on the documents' textual similarity. However, the hyperlink structure of Web documents, encapsulated in the "link graph" in which nodes are Web pages and links are hyperlinks between them, can be used as a different basis for clustering. Many standard graph clustering and partitioning techniques are applicable (e.g. [Hendrickson and Leland, 1995]). Link-based clustering schemes typically use factors such as these:

- The number of hyperlinks that must be followed to travel in the Web from one document to the other;
- The number of common ancestors of the two documents, weighted by their ancestry distance;

- The number of common descendents of the documents, similarly weighted.

These can be combined into an overall similarity measure between documents. In practice, a textual similarity measure is usually incorporated as well, to yield a hybrid clustering scheme that takes account of both the documents' content and their linkage structure. The overall similarity may then be determined as the weighted sum of four factors (e.g. [Weiss *et al.*, 1996]). Clearly such a measure will be sensitive to the stylistic characteristics of the documents and their linkage structure, and given the number of parameters involved there is considerable scope for tuning to maximize performance on particular data sets.

### *Determining "authority" of Web documents*

The Web's linkage structure is a valuable source of information that reflects the popularity, sometimes interpreted as "importance," "authority" or "status," of Web pages. For each page, a numeric rank is computed. The basic premise is that highly-ranked pages are ones that are cited, or pointed to, by many other pages. Consideration is also given to (a) the rank of the citing page, to reflect the fact that a citation by a highly-ranked page is a better indication of quality than one from a lesser page, and (b) the number of out-links from the citing page, to prevent a highly-ranked page from artificially magnifying its influence simply by containing a large number of pointers. This leads to a simple algebraic equation to determine the rank of each member of a set of hyperlinked pages [Brin and Page, 1998]. Complications arise from the fact that some links are "broken" in that they lead to nonexistent pages, and from the fact that the Web is not fully connected; these are easily overcome.

Such techniques are widely used by search engines (e.g. Google) to determine how to sort the hits associated with any given query. They provide a social measure of status that relates to standard techniques developed by social scientists for measuring and analyzing social networks [Wasserman and Faust, 1994].

## 4 Human text mining

All scientific researchers are expected to use the literature as a major source of information during the course of their work to provide new ideas and supplement their laboratory studies. However, some feel that this can be taken further: that new information, or at least new hypotheses, can be derived directly from the literature by researchers who are expert in information-seeking but not necessarily in the subject matter itself. Subject-matter experts can only read a small part of what is published in their fields and are often unaware of developments in related fields. Information researchers can seek useful linkages between related literatures which may be previously unknown—particularly if there is little explicit cross-reference between the literatures.

We briefly sketch an example, to indicate what automatic text mining may eventually aspire to—but is nowhere near achieving yet. By analyzing chains of causal implication within the medical literature, new hypotheses for causes of rare diseases have been discovered—some of which have received supporting experimental evidence [Swanson 1987; Swanson and Smalheiser, 1997]. While investigating causes of migraine headaches, Swanson extracted information from titles of articles in the biomedical literature, leading to clues like these:

> Stress is associated with migraines
> Stress can lead to loss of magnesium
> Calcium channel blockers prevent some migraines
> Magnesium is a natural calcium channel blocker
> Spreading cortical depression is implicated in some migraines
> High levels of magnesium inhibit spreading cortical depression
> Migraine patients have high platelet aggregability
> Magnesium can suppress platelet aggregability

These clues suggest that magnesium deficiency may play a role in some kinds of migraine headache, a hypothesis that did not exist in the literature at the time Swanson found these links.

Thus a new and plausible medical hypothesis was derived from a combination of text fragments and the information researcher's background knowledge. Of course, the hypothesis still had to be tested via non-textual means.

# 5 Techniques and tools

Text mining systems use a broad spectrum of different approaches and techniques, partly because of the great scope of text mining and consequent diversity of systems that perform it, and partly because the field is so young that dominant methodologies have not yet emerged.

## *High-level issues: Training vs. knowledge engineering*

There is an important distinction between systems that use an automatic training approach to spot patterns in data and ones that are based on a knowledge engineering approach and use rules formulated by human experts. This distinction recurs throughout the field but is particularly stark in the areas of entity extraction and information extraction. For example, systems that extract personal names can use hand-crafted rules derived from everyday experience. Simple and obvious rules involve capitalization, punctuation, single-letter initials, and titles; more complex ones take account of baronial prefixes and foreign forms. Alternatively, names could be manually marked up in a set of training documents and machine learning techniques used to infer rules that apply to test documents.

In general, the knowledge engineering approach requires a relatively high level of human expertise—a human expert who knows the domain, and the information extraction system, well enough to formulate high-quality rules. Formulating good rules is a demanding and time-consuming task for human experts, and involves many cycles of formulating, testing, and adjusting the rules so that they perform well on new data.

Markup for automatic training is clerical work that requires only the ability to recognize the entities in question when they occur. However, it is a demanding task because large volumes are needed for good performance. Some learning systems can leverage unmarked training data to improve the results obtained from a relatively small training set. For example, an experiment in document categorization used a small number of labeled documents to produce an initial model, which was then used to assign probabilistically-weighted class labels to unlabeled documents [Nigam *et al.*, 1998]. Then a new classifier was produced using all the documents as training data. The procedure was iterated until the classifier remained unchanged. Another possibility is to bootstrap learning based on two different and mutually reinforcing perspectives on the data, an idea called "co-training" [Blum and Mitchell, 1998].

## *Low-level issues: Token identification*

Dealing with natural language involves some rather mundane decisions that nevertheless strongly affect the success of the outcome. Tokenization, or splitting the input into words, is an important first step that seems easy but is fraught with small decisions: how to deal with apostrophes and hyphens, capitalization, punctuation, numbers, alphanumeric strings, whether the amount of white space is significant, whether to impose a maximum length on tokens, what to do with non-printing characters, and so on. It may be beneficial to perform some rudimentary morphological analysis on the tokens—removing suffixes [Porter, 1980] or representing them as words separate from the stem—which can be quite complex and is strongly language-dependent. Tokens may be standardized by using a dictionary to map different, but equivalent, variants of a term into a single canonical form. Some text mining applications (e.g. text summarization) split the input into sentences and even paragraphs, which again involves mundane decisions about delimiters, capitalization, and non-standard characters.

Once the input is tokenized, some level of syntactic processing is usually required. The simplest operation is to remove stop words, which are words that perform well-defined syntactic roles but from a non-linguistic point of view do not carry information. Another is to identify common phrases and map them into single features. The resulting representation of the text as a sequence of word features is commonly used in many text mining systems (e.g. for information extraction).

## *Basic techniques*

Tokenizing a document and discarding all sequential information yields the "bag of words" representation mentioned above under document retrieval. Great effort has been invested over the years in a quest for document similarity measures based on this representation. One is to count the number of terms in common between the documents: this is called *coordinate matching*. This representation, in conjunction with standard classification systems from machine learning (e.g. Naïve Bayes and Support Vector Machines; see [Witten and Frank, 2000]), underlies most text categorization systems.

It is often more effective to weight words in two ways: first by the number of documents in the entire collection in which they appear ("document frequency") on the basis that frequent words carry less information than rare ones; second by the number of times they appear in the particular documents in question ("term frequency"). These effects can be combined by multiplying the term frequency by the inverse document frequency, leading to a standard family of document similarity measures (often called "tf×idf"). These form the basis of standard text categorization and information retrieval systems.

A further step is to perform a syntactic analysis and tag each word with its part of speech. This helps to disambiguate different senses of a word and to eliminate incorrect analyses caused by rare word senses. Some part-of-speech taggers are rule based, while others are statistically based [Garside *et al.*, 1987]—this reflects the "training" vs. "knowledge engineering" referred to above. In either case, results are correct about 95% of the time—which may not be enough to resolve the ambiguity problems.

Another basic technique for dealing with sequences of words or other items is to use Hidden Markov Models (HMMs). These are probabilistic finite-state models that "parse" an input sequence by tracking its flow through the model. This is done in a probabilistic sense, so that the model's current state is represented not by a particular unique state but by a probability distribution over all states. Frequently the initial state is unknown or "hidden," and must itself be represented by a probability distribution. Each new token in the input affects this distribution in a way that depends on the structure and parameters of the model. Eventually, the overwhelming majority of the probability may be concentrated on one particular state, which serves to disambiguate the initial state and indeed the entire trajectory of state transitions corresponding to the input sequence. Trainable part-of-speech taggers are based on this idea: the states correspond to parts of speech (e.g. [Brill, 1992]).

HMMs can easily be built from training sequences in which each token is pre-tagged with its state. However, the manual effort involved in tagging training sequences is often prohibitive. There exists a "relaxation" algorithm that takes untagged training sequences and produces a corresponding HMM [Rabiner, 1989]. Such techniques have been used in text mining, for example, to extract references from plain text [McCallum *et al.*, 1999].

If the source documents are hypertext, there are various basic techniques for analyzing the linkage structure. One, evaluating page rank to determine a numeric "importance" for each page, was described above. Another is to decompose pages into "hubs" and "authorities" [Kleinberg, 1999]. These are recursively defined as follows: a good hub is a page that points to many good authorities, while a good authority is a page pointed to by many good hubs. This mutually reinforcing relationship can be evaluated using an iterative relaxation procedure. The result can be used to select documents that contain authoritative content to use as a basis for text mining, discarding all those Web pages that simply contain lists of pointers to other pages.

## *Tools*

There is a plethora of software tools to help with the basic processes of text mining. A comprehensive and useful resource at nlp.stanford.edu/lionks/statnlp.html lists taggers, parsers, language models and concordances; several different corpora (large collections, particular languages, etc.); dictionaries, lexical, and morphological resources; software modules for handling XML and SGML documents; and other relevant resources such as courses, mailing lists, people, and societies. It classifies software as freely downloadable and commercially available, with several intermediate categories.

One particular framework and development environment for text mining, called General Architecture for Text Engineering or GATE [Cunningham, 2002], aims to help users develop, evaluate and deploy systems for what the authors term “language engineering.” It provides support not just for standard text mining applications such as information extraction, but also for tasks such as building and annotating corpora, and evaluating the applications.

At the lowest level, GATE supports a variety of formats including XML, RTF, HTML, SGML, email and plain text, converting them into a single unified model that also supports annotation. There are three storage mechanisms: a relational database, a serialized Java object, and an XML-based internal format; documents can be re-exported into their original format with or without annotations. Text encoding is based on Unicode to provide support for multilingual data processing, so that systems developed with GATE can be ported to new languages with no additional overhead apart from the development of the resources needed for the specific language.

GATE includes a tokenizer and a sentence splitter. It incorporates a part-of-speech tagger and a gazetteer that includes lists of cities, organizations, days of the week, etc. It has a semantic tagger that applies hand-crafted rules written in a language in which patterns can be described and annotations created as a result. Patterns can be specified by giving a particular text string, or annotations that have previously been created by modules such as the tokenizer, gazetteer, or document format analysis. It also includes semantic modules that recognize relations between entities and detect co-reference. It contains tools for creating new language resources, and for evaluating the performance of text mining systems developed with GATE.

One application of GATE is a system for entity extraction of names that is capable of processing texts from widely different domains and genres. This has been used to perform recognition and tracking tasks of named, nominal and pronominal entities in several types of text. GATE has also been used to produce formal annotations about important events in a text commentary that accompanies football video program material.

# 6 Conclusion

Text mining is a burgeoning technology that is still, because of its newness and intrinsic difficulty, in a fluid state—akin, perhaps, to the state of machine learning in the mid-1980s. Generally accepted characterizations of what it covers do not yet exist. When the term is broadly interpreted, many different problems and techniques come under its ambit. In most cases it is difficult to provide general and meaningful evaluations because the task is highly sensitive to the particular text under consideration. Document classification, entity extraction, and filling templates that correspond to given relationships between entities, are all central text mining operations that have been extensively studied. Using structured data such as Web pages rather than plain text as the input opens up new possibilities for extracting information from individual pages and large networks of pages. Automatic text mining techniques have a long way to go before they rival the ability of people, even without any special domain knowledge, to glean information from large document collections.

# References

Agrawal, R. and Srikant, R. (1994) “Fast algorithms for mining association rules.” *Proc Int Conf on Very Large Databases VLDB-94*, Santiago, Chile, pp. 487-499.

Aone, C., Bennett, S.W., and Gorlinsky, J. (1996) “Multi-media fusion through application of machine learning and NLP.” *Proc AAAI Symposium on Machine Learning in Information Access*. Stanford, CA.

Appelt, D.E. (1999) "Introduction to information extraction technology." *Tutorial, Int Joint Conf on Artificial Intelligence IJCAI'99*. Morgan Kaufmann, San Mateo. Tutorial notes available at www.ai.sri.com/~appelt/ie-tutorial.

Apte, C., Damerau, F.J. and Weiss, S.M. (1994) "Automated learning of decision rules for text categorization." *ACM Trans Information Systems*, Vol. 12, No. 3, pp. 233-251.

Baeza-Yates, R. and Ribiero-Neto, B. (1999) *Modern information retrieval.* Addison Wesley Longman, Essex, England.

Blum, A. and Mitchell, T. (1998) "Combining labeled and unlabeled data with co-training." *Proc Conf on Computational Learning Theory COLT-98*. Madison, Wisconsin, pp. 92-100.

Borko, H. and Bernier, C.L. (1975) *Abstracting concepts and methods.* Academic Press, San Diego, California.

Brill, E. (1992) "A simple rule-based part of speech tagger." *Proc Conf on Applied Natural Language Processing ANLP-92*. Trento, Italy, pp. 152-155.

Brin, S. and Page, L. (1998) "The anatomy of a large-scale hypertextual Web search engine." *Proc World Wide Web Conference WWW-7*. In *Computer Networks and ISDN Systems*, Vol. 30, No. 1-7, pp. 107-117.

Califf, M.E. and Mooney, R.J. (1999) "Relational learning of pattern-match rules for information extraction." *Proc National Conference on Artificial Intelligence AAAI-99*. Orlando, FL, pp. 328-334.

Cavnar, W.B. and Trenkle, J.M. (1994) "N-Gram-based text categorization." *Proc Symposium on Document Analysis and Information Retrieval*. Las Vegas, NV, pp. 161-175.

Cheeseman, P., Kelly, J., Self, M., Stutz., J., Taylor, W., and Freeman, D. (1988) "AUTOCLASS: A Bayesian classification system.*" Proc Int Conf on Machine Learning ICML-88*. San Mateo, CA, pp. 54–64.

Cohen, W.W. (1995) "Fast effective rule induction." *Proc Int Conf on Machine Learning ICML-95*. Tarragona, Catalonia, Spain, pp. 115-123.

Cunningham, H. (2002) "GATE, a General Architecture for Text Engineering." *Computing and the Humanities*, Vol. 36, pp. 223-254.

Dumais, S.T., Platt, J., Heckerman, D. and Sahami, M. (1998) "Inductive learning algorithms and representations for text categorization." *Proc Int Conf on Information and Knowledge Management CIKM-98*. Bethesda, MD, pp. 148-155.

Efron, B. and Thisted, R. (1976) "Estimating the number of unseen species: how many words did Shakespeare know?" *Biometrika*, Vol. 63, No. 3, pp. 435-447.

Fisher, D. (1987) "Knowledge acquisition via incremental conceptual clustering." *Machine Learning* Vol. 2, pp. 139–172.

Frank, E., Paynter, G., Witten, I.H., Gutwin, C. and Nevill-Manning, C. (1999) "Domain-specific keyphrase extraction." *Proc Int Joint Conf on Artificial Intelligence IJCAI-99*. Stockholm, Sweden, pp. 668-673.

Franklin, D. (2002) "New software instantly connects key bits of data that once eluded teams of researchers." *Time*, December 23.

Freitag, D. (2002) "Machine learning for information extraction in informal domains." *Machine Learning*, Vol. 39, No. 2/3, pp. 169-202.

Garside, R., Leech, G. and Sampson, G. (1987) *The computational analysis of English: a corpus-based approach*. Longman, London.

Green, C.L, and Edwards, P. (1996) "Using machine learning to enhance software tools for Internet information management." *Proc AAAI Workshop on Internet based Information Systems*. Portland, OR, pp. 48–56.

Grefenstette, G. (1995) "Comparing two language identification schemes." *Proc Int Conf on Statistical Analysis of Textual Data JADT-95*. Rome, Italy.

Harman, D.K. (1995) "Overview of the third text retrieval conference." In *Proc Text Retrieval Conference TREC-3*. National Institute of Standards, Gaithersburg, pp. 1-19.

Hayes, P.J., Andersen, P.M., Nirenburg, I.B. and Schmandt, L.M. (1990) "Tcs: a shell for content-based text categorization." *Proc IEEE Conf on Artificial Intelligence Applications CAIA-90.* Santa Barbara, CA, pp. 320-326.

Hearst, M.A. (1999) "Untangling text mining." *Proc Annual Meeting of the Association for Computational Linguistics ACL99.* University of Maryland, June.

Hendrickson, B. and Leland, R.W. (1995) "A multi-level algorithm for partitioning graphs." *Proc ACM/IEEE Conf on Supercomputing.* San Diego, CA.

Huffman, S.B. (1996) "Learning information extraction patterns from examples." In *Connectionist, statistical, and symbolic approaches to learning for natural language processing*, edited by S. Wertmer, E. Riloff and G. Scheler. Springer Verlag, Berlin, pp. 246-260.

Kleinberg, J.M. (1999) "Authoritative sources in a hyperlinked environment." *J. ACM*, Vol. 46, No. 5, pp. 604-632.

Kolata, G. (1986) "Shakespeare's new poem: an ode to statistics." *Science*, No. 231, pp. 335-336; January 24.

Kushmerick, N., Weld, D.S. and Doorenbos, R. (1997) "Wrapper induction for information extraction." *Proc Int Joint Conference on Artificial Intelligence IJCAI-97.* Nayoya, Japan, pp. 729-735.

Lancaster, F.W. (1991) *Indexing and abstracting in theory and practice*. University of Illinois Graduate School of Library and Information Science, Champaign, Illinois.

Lenat, D.B. (1995) "CYC: A large-scale investment in knowledge infrastructure." *Comm ACM*, Vol. 38, No. 11, pp. 32–38.

Lewis, D.D. (1992) "An evaluation of phrasal and clustered representations on a text categorization task." *Proc Int Conf on Research and Development in Information Retrieval SIGIR-92.* pp. 37-50. Copenhagen, Denmark.

Liere, R., and Tadepalli, P. (1996) "The use of active learning in text categorization." *Proc AAAI Symposium on Machine Learning in Information Access.* Stanford, CA.

Mani, I. (2001) *Automatic summarization.* John Benjamins, Amsterdam.

Mann, T. (1993) *Library research models.* Oxford University Press, NY.

Martin, J.D. (1995) "Clustering full text documents." *Proc IJCAI Workshop on Data Engineering for Inductive Learning at IJCAI-95.* Montreal, Canada.

McCallum, A., Nigam, K., Rennie, J. and Seymore, K. (1999) "Building domain-specific search engines with machine learning techniques." *Proc AAAI Spring Symposium.* Stanford, CA.

Mitchell, T.M. (1997) *Machine learning.* McGraw Hill, NY.

Nahm, U.Y. and Mooney, R.J. (2000) "Using information extraction to aid the discovery of prediction rules from texts." *Proc Workshop on Text Mining, Int Conf on Knowledge Discovery and Data Mining KDD-2000.* Boston, USA, pp. 51-58.

Nahm, U.Y. and Mooney, R.J. (2002) "Text mining with information extraction." *Proc AAAI-2002 Spring Symposium on Mining Answers from Texts and Knowledge Bases.* Stanford, CA.

Nigam, K., McCallum, A., Thrun, S. and Mitchell, T. (1998) "Learning to classify text from labeled and unlabeled documents." *Proc National Conference on Artificial Intelligence AAAI-98.* Madison, USA, pp.792-799.

Porter, M.F. (1980) "An algorithm for suffix stripping." *Program*, Vol. 13, No. 3, pp. 130-137.

Quinlan, R. (1993) *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA.

Rabiner, L.R. (1989) "A tutorial on hidden Markov models and selected applications in speech recognition." *Proc IEEE* 77(2): 257–286.

Sebastiani, F. (2002) "Machine learning in automated text categorization." *ACM Computing Surveys*, Vol. 34, No. 1, pp. 1–47.

Salton, G. and McGill, M.J. (1983) *Introduction to modern information retrieval.* McGraw Hill, New York.

Soderland, S., Fisher, D., Aseltine, J. and Lehnert, W. (1995) "Crystal: inducing a conceptual dictionary." *Proc Int Conf on Machine Learning ICML-95*. Tarragona, Catalonia, Spain, pp. 343-351.

Swanson, D.R. (1987) "Two medical literatures that are logically but not bibliographically connected." *J American Society for Information Science*, Vol. 38, No. 4, pp. 228-233.

Swanson, D.R. and Smalheiser, N.R. (1997) "An interactive system for finding complementary literatures: a stimulus to scientific discovery." *Artificial Intelligence*, Vol. 91, pp. 183-203.

Tkach, D. (editor) (1998) "Text mining technology: turning information into knowledge." IBM White Paper, Feb 17, 1998.

Wasserman, S. and Faust, K. (1994) *Social network analysis: methods and applications*. Cambridge University Press.

Weiss, R., Velez, B., Nemprempre, C., Szilagyi, P., Duda, A. and Gifford, D.K. (1996) "HyPursuit: A hierarchical network search engine that exploits content-link hypertext clustering." *Proc ACM Conf on Hypertext*. Washington, DC; March, pp. 180-193.

Willett, P. (1988) "Recent trends in hierarchical document clustering: A critical review." *Information Processing and Management*, Vol. 24, No. 5, pp. 577–597.

Witten, I.H., Moffat, A. and Bell, T.C. (1999) *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, San Francisco, CA.

Witten, I.H. and Frank, E. (2000) *Data mining: Practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann, San Francisco, CA.

Witten, I.H. and Bainbridge, D. (2003) *How to build a digital library*. Morgan Kaufmann, San Francisco, CA.