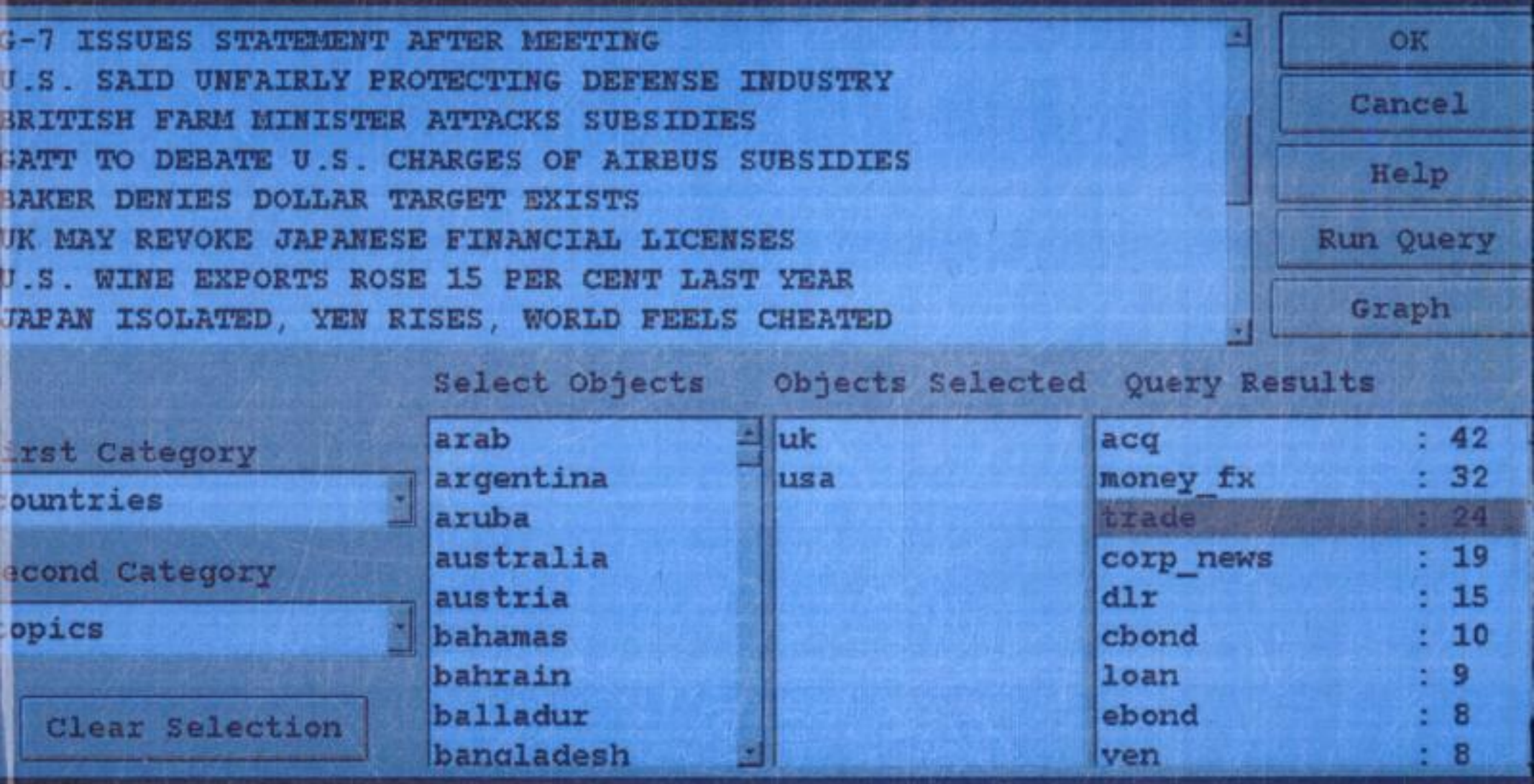
THE TEXT MINING HANDBOOK

Advanced Approaches in Analyzing Unstructured Data



RONEN FELDMAN
JAMES SANGER

CAMBRIDGE

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
32 Avenue of the Americas, New York, NY 10013-2473, USA

www.cambridge.org
Information on this title: www.cambridge.org/9780521836579

First published 2007

Printed in the United States of America

A catalog record for this publication is available from the British Library.

Library of Congress Cataloging in Publication Data

Feldman, Ronen, 1962–
The text mining handbook : advanced approaches in analyzing unstructured data /
Ronen Feldman, James Sanger.
p. cm.
Includes bibliographical references and index.
ISBN 0-521-83657-3 (hardback)
1. Data mining – Handbooks, manuals, etc. I. Sanger, James, 1965– II. Title.
QA76.9.D343F45 2006
005.74 – dc22 2005029330

ISBN-13 978-0-521-83657-9 hardback
ISBN-10 0-521-83657-3 hardback

Contents

Preface

The information age has made it easy to store large amounts of data. The proliferation of documents available on the Web, on corporate intranets, on news wires, and elsewhere is overwhelming. However, although the amount of data available to us is constantly increasing, our ability to absorb and process this information remains constant. Search engines only exacerbate the problem by making more and more documents available in a matter of a few key strokes.

Text mining is a new and exciting research area that tries to solve the information overload problem by using techniques from data mining, machine learning, natural language processing (NLP), information retrieval (IR), and knowledge management. Text mining involves the preprocessing of document collections (text categorization, information extraction, term extraction), the storage of the intermediate representations, the techniques to analyze these intermediate representations (such as distribution analysis, clustering, trend analysis, and association rules), and visualization of the results.

This book presents a general theory of text mining along with the main techniques behind it. We offer a generalized architecture for text mining and outline the algorithms and data structures typically used by text mining systems.

The book is aimed at the advanced undergraduate students, graduate students, academic researchers, and professional practitioners interested in complete coverage of the text mining field. We have included all the topics critical to people who plan to develop text mining systems or to use them. In particular, we have covered preprocessing techniques such as text categorization, text clustering, and information extraction and analysis techniques such as association rules and link analysis.

The book tries to blend together theory and practice; we have attempted to provide many real-life scenarios that show how the different techniques are used in practice. When writing the book we tried to make it as self-contained as possible and have compiled a comprehensive bibliography for each topic so that the reader can expand his or her knowledge accordingly.

1

Introduction to Text Mining

1.1 DEFINING TEXT MINING

Text mining can be broadly defined as a knowledge-intensive process in which a user interacts with a document collection over time by using a suite of analysis tools. In a manner analogous to data mining, text mining seeks to extract useful information from data sources through the identification and exploration of interesting patterns. In the case of text mining, however, the data sources are document collections, and interesting patterns are found not among formalized database records but in the unstructured textual data in the documents in these collections.

Certainly, text mining derives much of its inspiration and direction from seminal research on data mining. Therefore, it is not surprising to find that text mining and data mining systems evince many high-level architectural similarities. For instance, both types of systems rely on preprocessing routines, pattern-discovery algorithms, and presentation-layer elements such as visualization tools to enhance the browsing of answer sets. Further, text mining adopts many of the specific types of patterns in its core knowledge discovery operations that were first introduced and vetted in data mining research.

Because data mining assumes that data have already been stored in a structured format, much of its preprocessing focus falls on two critical tasks: Scrubbing and normalizing data and creating extensive numbers of table joins. In contrast, for text mining systems, preprocessing operations center on the identification and extraction of representative features for natural language documents. These preprocessing operations are responsible for transforming unstructured data stored in document collections into a more explicitly structured intermediate format, which is a concern that is not relevant for most data mining systems.

Moreover, because of the centrality of natural language text to its mission, text mining also draws on advances made in other computer science disciplines concerned with the handling of natural language. Perhaps most notably, text mining exploits techniques and methodologies from the areas of information retrieval, information extraction, and corpus-based computational linguistics.

I.1.1 The Document Collection and the Document

A key element of text mining is its focus on the *document collection*. At its simplest, a document collection can be any grouping of text-based documents. Practically speaking, however, most text mining solutions are aimed at discovering patterns across very large document collections. The number of documents in such collections can range from the many thousands to the tens of millions.

Document collections can be either *static*, in which case the initial complement of documents remains unchanged, or *dynamic*, which is a term applied to document collections characterized by their inclusion of new or updated documents over time. Extremely large document collections, as well as document collections with very high rates of document change, can pose performance optimization challenges for various components of a text mining system.

An illustration of a typical real-world document collection suitable as initial input for text mining is PubMed, the National Library of Medicine's online repository of citation-related information for biomedical research papers. PubMed has received significant attention from computer scientists interested in employing text mining techniques because this online service contains text-based document abstracts for more than 12 million research papers on topics in the life sciences. PubMed represents the most comprehensive online collection of biomedical research papers published in the English language, and it houses data relating to a considerable selection of publications in other languages as well. The publication dates for the main body of PubMed's collected papers stretch from 1966 to the present. The collection is dynamic and growing, for an estimated 40,000 new biomedical abstracts are added every month.

Even subsections of PubMed's data repository can represent substantial document collections for specific text mining applications. For instance, a relatively recent PubMed search for only those abstracts that contain the words *protein* or *gene* returned a result set of more than 2,800,000 documents, and more than 66 percent of these documents were published within the last decade. Indeed, a very narrowly defined search for abstracts mentioning *epidermal growth factor receptor* returned more than 10,000 documents.

The sheer size of document collections like that represented by PubMed makes manual attempts to correlate data across documents, map complex relationships, or identify trends at best extremely labor-intensive and at worst nearly impossible to achieve. Automatic methods for identifying and exploring interdocument data relationships dramatically enhance the speed and efficiency of research activities. Indeed, in some cases, automated exploration techniques like those found in text mining are not just a helpful adjunct but a baseline requirement for researchers to be able, in a practicable way, to recognize subtle patterns across large numbers of natural language documents.

Text mining systems, however, usually do not run their knowledge discovery algorithms on unprepared document collections. Considerable emphasis in text mining is devoted to what are commonly referred to as *preprocessing operations*. Typical text mining preprocessing operations are discussed in detail in Chapter III.

Text mining preprocessing operations include a variety of different types of techniques culled and adapted from information retrieval, information extraction, and

computational linguistics research that transform raw, unstructured, original-format content (like that which can be downloaded from PubMed) into a carefully structured, intermediate data format. Knowledge discovery operations, in turn, are operated against this specially structured intermediate representation of the original document collection.

The Document

Another basic element in text mining is the *document*. For practical purposes, a document can be very informally defined as a unit of discrete textual data within a collection that usually, but not necessarily, correlates with some real-world document such as a business report, legal memorandum, e-mail, research paper, manuscript, article, press release, or news story. Although it is not typical, a document can be defined a little less arbitrarily within the context of a particular document collection by describing a *prototypical document* based on its representation of a similar class of entities within that collection.

One should not, however, infer from this that a given document necessarily exists only within the context of one particular collection. It is important to recognize that a document can (and generally does) exist in any number or type of collections – from the very formally organized to the very ad hoc. A document can also be a member of different document collections, or different subsets of the same document collection, and can exist in these different collections at the same time. For example, a document relating to Microsoft's antitrust litigation could exist in completely different document collections oriented toward current affairs, legal affairs, antitrust-related legal affairs, and software company news.

"Weakly Structured" and "Semistructured" Documents

Despite the somewhat misleading label that it bears as *unstructured data*, a text document may be seen, from many perspectives, as a structured object. From a linguistic perspective, even a rather innocuous document demonstrates a rich amount of semantic and syntactical structure, although this structure is implicit and to some degree hidden in its textual content. In addition, typographical elements such as punctuation marks, capitalization, numerics, and special characters – particularly when coupled with layout artifacts such as white spacing, carriage returns, underlining, asterisks, tables, columns, and so on – can often serve as a kind of "soft markup" language, providing clues to help identify important document subcomponents such as paragraphs, titles, publication dates, author names, table records, headers, and footnotes. Word sequence may also be a structurally meaningful dimension to a document. At the other end of the "unstructured" spectrum, some text documents, like those generated from a WYSIWYG HTML editor, actually possess from their inception more overt types of embedded metadata in the form of formalized markup tags.

Documents that have relatively little in the way of strong typographical, layout, or markup indicators to denote structure – like most scientific research papers, business reports, legal memoranda, and news stories – are sometimes referred to as *free-format* or *weakly structured* documents. On the other hand, documents with extensive and consistent format elements in which field-type metadata can be more easily inferred – such as some e-mail, HTML Web pages, PDF files, and word-processing

files with heavy document templating or style-sheet constraints – are occasionally described as *semistructured* documents.

1.1.2 Document Features

The preprocessing operations that support text mining attempt to leverage many different elements contained in a natural language document in order to transform it from an irregular and implicitly structured representation into an explicitly structured representation. However, given the potentially large number of words, phrases, sentences, typographical elements, and layout artifacts that even a short document may have – not to mention the potentially vast number of different senses that each of these elements may have in various contexts and combinations – an essential task for most text mining systems is the identification of a simplified subset of document features that can be used to represent a particular document as a whole. We refer to such a set of features as the *representational model* of a document and say that individual documents are *represented by* the set of features that their representational models contain.

Even with attempts to develop efficient representational models, each document in a collection is usually made up of a large number – sometimes an exceedingly large number – of features. The large number of features required to represent documents in a collection affects almost every aspect of a text mining system's approach, design, and performance.

Problems relating to high *feature dimensionality* (i.e., the size and scale of possible combinations of feature values for data) are typically of much greater magnitude in text mining systems than in classic data mining systems. Structured representations of natural language documents have much larger numbers of potentially representative features – and thus higher numbers of possible combinations of feature values – than one generally finds with records in relational or hierarchical databases.

For even the most modest document collections, the number of word-level features required to represent the documents in these collections can be exceedingly large. For example, in an extremely small collection of 15,000 documents culled from Reuters news feeds, more than 25,000 nontrivial word stems could be identified.

Even when one works with more optimized feature types, tens of thousands of concept-level features may still be relevant for a single application domain. The number of attributes in a relational database that are analyzed in a data mining task is usually significantly smaller.

The high dimensionality of potentially representative features in document collections is a driving factor in the development of text mining preprocessing operations aimed at creating more streamlined representational models. This high dimensionality also indirectly contributes to other conditions that separate text mining systems from data mining systems such as greater levels of pattern overabundance and more acute requirements for postquery refinement techniques.

Another characteristic of natural language documents is what might be described as *feature sparsity*. Only a small percentage of all possible features for a document collection as a whole appears in any single document, and thus when a document is represented as a binary vector of features, nearly all values of the vector are zero.

The tuple dimension is also sparse. That is, some features often appear in only a few documents, which means that the support of many patterns is quite low.

Commonly Used Document Features: Characters, Words, Terms, and Concepts

Because text mining algorithms operate on the feature-based representations of documents and not the underlying documents themselves, there is often a trade-off between two important goals. The first goal is to achieve the correct calibration of the volume and semantic level of features to portray the meaning of a document accurately, which tends to incline text mining preprocessing operations toward selecting or extracting relatively more features to represent documents. The second goal is to identify features in a way that is most computationally efficient and practical for pattern discovery, which is a process that emphasizes the streamlining of representative feature sets; such streamlining is sometimes supported by the validation, normalization, or cross-referencing of features against controlled vocabularies or external knowledge sources such as dictionaries, thesauri, ontologies, or knowledge bases to assist in generating smaller representative sets of more semantically rich features.

Although many potential features can be employed to represent documents,[1] the following four types are most commonly used:

- ***Characters.*** The individual component-level letters, numerals, special characters and spaces are the building blocks of higher-level semantic features such as words, terms, and concepts. A character-level representation can include the full set of all characters for a document or some filtered subset. Character-based representations without positional information (i.e., bag-of-characters approaches) are often of very limited utility in text mining applications. Character-based representations that include some level of positional information (e.g., bigrams or trigrams) are somewhat more useful and common. In general, however, character-based representations can often be unwieldy for some types of text processing techniques because the feature space for a document is fairly unoptimized. On the other hand, this feature space can in many ways be viewed as the most complete of any representation of a real-world text document.
- ***Words.*** Specific words selected directly from a "native" document are at what might be described as the basic level of semantic richness. For this reason, word-level features are sometimes referred to as existing in the *native feature space* of a document. In general, a single word-level feature should equate with, or have the value of, no more than one linguistic token. Phrases, multiword expressions, or even multiword hyphenates would not constitute single word-level features. It is possible for a word-level representation of a document to include a feature for each word within that document – that is the "full text," where a document is represented by a complete and unabridged set of its word-level features. This can

[1] Beyond the three feature types discussed and defined here – namely, words, terms, and concepts – other features that have been used for representing documents include linguistic phrases, nonconsecutive phrases, keyphrases, character bigrams, character trigrams, frames, and parse trees.

be applied – perhaps with varying results – to document collections represented by other feature models.

Domains and Background Knowledge

In text mining systems, concepts belong not only to the descriptive attributes of a particular document but generally also to *domains*. With respect to text mining, a domain has come to be loosely defined as a specialized area of interest for which dedicated *ontologies, lexicons*, and *taxonomies* of information may be developed.

Domains can include very broad areas of subject matter (e.g., *biology*) or more narrowly defined specialisms (e.g., *genomics or proteomics*). Some other noteworthy domains for text mining applications include financial services (with significant sub-domains like corporate finance, securities trading, and commodities.), world affairs, international law, counterterrorism studies, patent research, and materials science. Text mining systems with some element of domain-specificity in their orientation – that is, most text mining systems designed for a practical purpose – can leverage information from formal external knowledge sources for these domains to greatly enhance elements of their preprocessing, knowledge discovery, and presentation-layer operations.

Domain knowledge, perhaps more frequently referred to in the literature as *background knowledge*, can be used in text mining preprocessing operations to enhance concept extraction and validation activities. Access to background knowledge – although not strictly necessary for the creation of concept hierarchies within the context of a single document or document collection – can play an important role in the development of more meaningful, consistent, and normalized concept hierarchies.

Text mining makes use of background knowledge to a greater extent than, and in different ways from, data mining. For advanced text mining applications that can take advantage of background knowledge, features are not just elements in a flat set, as is most often the case in structured data applications. By relating features by way of lexicons and ontologies, advanced text mining systems can create fuller representations of document collections in preprocessing operations and support enhanced query and refinement functionalities.

Indeed, background knowledge can be used to inform many different elements of a text mining system. In preprocessing operations, background knowledge is an important adjunct to classification and concept-extraction methodologies. Background knowledge can also be leveraged to enhance core mining algorithms and browsing operations. In addition, domain-oriented information serves as one of the main bases for search refinement techniques.

In addition, background knowledge may be utilized by other components of a text mining system. For instance, background knowledge may be used to construct meaningful constraints in knowledge discovery operations. Likewise, background knowledge may also be used to formulate constraints that allow users greater flexibility when browsing large result sets.

I.1.3 The Search for Patterns and Trends

Although text mining preprocessing operations play the critical role of transforming unstructured content of a raw document collection into a more tractable

concept-level data representation, the core functionality of a text mining system resides in the analysis of *concept co-occurrence* patterns across documents in a collection. Indeed, text mining systems rely on algorithmic and heuristic approaches to consider *distributions, frequent sets*, and various *associations* of concepts at an *interdocument* level in an effort to enable a user to discover the nature and relationships of concepts as reflected in the collection as a whole.

For example, in a collection of news articles, a large number of articles on politician X and "scandal" may indicate a negative image of the character of X and alert his or her handlers to the need for a new public relations campaign. Or, a growing number of articles on company Y and product Z may indicate a shift of focus in company Y's interests – a shift that should be noted by its competitors. In another example, a potential relationship might be inferred between two proteins P_1 and P_2 by the pattern of (a) several articles mentioning the protein P_1 in relation to the enzyme E_1, (b) a few articles describing functional similarities between enzymes E_1 and E_2 without referring to any protein names, and (c) several articles linking enzyme E_2 to protein P_2. In all three of these examples, the information is not provided by any single document but rather from the totality of the collection. Text mining's methods of pattern analysis seek to discover co-occurrence relationships between concepts as reflected by the totality of the corpus at hand.

Text mining methods – often based on large-scale, brute-force search directed at large, high-dimensionality feature sets – generally produce very large numbers of patterns. This results in an overabundance problem with respect to identified patterns that is usually much more severe than that encountered in data mining applications aimed at structured data sources.

A main operational task for text mining systems is to enable a user to limit pattern overabundance by providing refinement capabilities that key on various specifiable measures of "interestingness" for search results. Such refinement capabilities prevent system users from getting overwhelmed by too many uninteresting results.

The problem of pattern overabundance can exist in all knowledge discovery activities. It is simply heightened when interacting with large collections of text documents, and, therefore, text mining operations must necessarily be conceived to provide not only relevant but also manageable result sets to a user.

Text mining also builds on various data mining approaches first specified in Lent, Agrawal, and Srikant (1997) to identify trends in data. In text mining, *trend analysis* relies on date-and-time stamping of documents within a collection so that comparisons can be made between a subset of documents relating to one period and a subset of documents relating to another.

Trend analysis across document subsets attempts to answer certain types of questions. For instance, in relation to a collection of news stories, Montes-y-Gomez, Gelbukh, and Lopez-Lopez (2001b) suggests that trend analysis concerns itself with questions such as the following:

- *What is the general trend of the news topics between two periods (as represented by two different document subsets)?*
- *Are the news topics nearly the same or are they widely divergent across the two periods?*
- *Can emerging and disappearing topics be identified?*
- *Did any topics maintain the same level of occurrence during the two periods?*

(3) If a particular part of a system is compared, all other parts must be exactly the same. For instance, when comparing learning algorithms, the document representations must be the same, and when comparing the dimension reduction methods, the learning algorithms must be fixed together with their parameters.

These conditions are very difficult to meet – especially the last one. Thus, in practice, the only reliable comparisons are those done by the same researcher.

Other frequently used benchmark collections are the OHSUMED collection of titles and abstracts of papers from medical journals categorized with MESH thesaurus terms, 20 Newsgroups collection of messages posted to newsgroups with the newsgroups themselves as categories, and the TREC-AP collection of newswire stories.

IV.7.3 Comparison among Classifiers

Given the lack of a reliable way to compare classifiers across researchers, it is possible to draw only very general conclusions in reference to the question Which classifier is the best?

- According to most researchers, the top performers are SVM, AdaBoost, *k*NN, and Regression methods. Insufficient statistical evidence has been compiled to determine the best of these methods. Efficiency considerations, implementation complexity, and other application-related issues may assist in selecting from among these classifiers for specific problems.
- Rocchio and Naïve Bayes have the worst performance among the ML classifiers, but both are often used as baseline classifiers. Also, NB is very useful as a member of classifier committees.
- There are mixed results regarding the neural networks and decision tree classifiers. Some of the experiments have demonstrated rather poor performance, whereas in other experiments they performed nearly as well as SVM.

IV.8 CITATIONS AND NOTES

Section IV.1

Applications of text categorization are described in Hayes et al. (1988); Ittner, Lewis, and Ahn (1995); Larkey (1998); Lima, Laender, and Ribeiro-Neto (1998); Attardi, Gulli, and Sebastiani (1999); Drucker, Vapnik, and Wu (1999); Moens and Dumortier (2000); Yang, Ault, Pierce, and Lattimer (2000); Gentili et al. (2001); Krier and Zaccà (2002); Fall et al. (2003); and Giorgetti and Sebastiani (2003a, 2003b).

Section IV.2

For a general introduction to text categorization, refer to Sebastiani (2002) and Lewis (2000), which provides an excellent tutorial on the subject.

Section IV.3

Approaches that integrate linguistic and background knowledge into the categorization process can be found in Jacobs (1992); Rodriguez et al. (1997); Aizawa (2001); and Benkhalifa, Mouradi, and Bouyakhf (2001a, 2001b).

Section IV.5.3–IV.5.4

The following papers discuss how to use decision trees and decision lists for text categorization: Apte, Damerau, and Weiss (1994a, 1994b, 1994c); Li and Yamanishi (1999); Chen and Ho (2000); and Li and Yamanishi (2002).

Section IV.5.5

The use of regression for text categorization is discussed in Zhang and Oles (2001), Zhang et al. (2003), and Zhang and Yang (2003).

Section IV.5.8

The *k*NN algorithm is discussed and described in Yavuz and Guvenir (1998); Han, Karypis, and Kumar (2001); Soucy and Mineau (2001b); and Kwon and Lee (2003).

Section IV.5.9

The SVM algorithm is described and discussed in Vapnik (1995); Joachims (1998); Kwok (1998); Drucker, Vapnik, et al. (1999); Joachims (1999); Klinkenberg and Joachims (2000); Siolas and d'Alche-Buc (2000); Tong and Koller (2000); Joachims (2001); Brank et al. (2002); Joachims (2002); Leopold and Kindermann (2002); Diederich et al. (2003); Sun, Naing, et al. (2003); Xu et al. (2003); and Zhang and Lee (2003).

Section IV.5.10

Approaches that combine several algorithms by using committees of algorithms or by using boosting are described in Larkey and Croft (1996); Liere and Tadepalli (1997); Liere and Tadepalli (1998); Forsyth (1999); Ruiz and Srinivasan (1999a, 1999b); Schapire and Singer (2000); Sebastiani, Sperduti, and Valdambrini (2000); Al-Kofahi et al. (2001); Bao et al. (2001); Lam and Lai (2001); Taira and Haruno (2001); and Nardiello, Sebastiani, and Sperduti (2003).

Additional Algorithms

There are several *adaptive* (or *online*) algorithms that build classifiers incrementally without requiring the whole training set to be present at once. A simple *perceptron* is described in Schutze et al. (1995) and Wiener (1995). A WINNOW algorithm, which is a multiplicative variant of perceptron, is described in Dagan, Karov, and Roth (1997). Other online algorithms include WIDROW_HOFF, EXPONENTIATED GRADIENT (Lewis et al. 1996), and SLEEPING EXPERTS (Cohen and Singer 1999).

Relational and rule-based approaches to text categorization are discussed in Cohen (1992); Cohen (1995a, 1995b); and Cohen and Hirsh (1998).

Section IV.7

Comparisons between the categorization algorithms are discussed in Yang (1996) and Yang (1999).

V

Clustering

Clustering is an unsupervised process through which objects are classified into groups called clusters. In categorization problems, as described in Chapter IV, we are provided with a collection of preclassified training examples, and the task of the system is to learn the descriptions of classes in order to be able to classify a new unlabeled object. In the case of clustering, the problem is to group the given unlabeled collection into meaningful clusters without any prior information. Any labels associated with objects are obtained solely from the data.

Clustering is useful in a wide range of data analysis fields, including data mining, document retrieval, image segmentation, and pattern classification. In many such problems, little prior information is available about the data, and the decision-maker must make as few assumptions about the data as possible. It is for those cases the clustering methodology is especially appropriate.

Clustering techniques are described in this chapter in the context of textual data analysis. Section V.1 discusses the various applications of clustering in text analysis domains. Sections V.2 and V.3 address the general clustering problem and present several clustering algorithms. Finally Section V.4 demonstrates how the clustering algorithms can be adapted to text analysis.

V.1 CLUSTERING TASKS IN TEXT ANALYSIS

One application of clustering is the analysis and navigation of big text collections such as Web pages. The basic assumption, called the *cluster hypothesis*, states that relevant documents tend to be more similar to each other than to nonrelevant ones. If this assumption holds for a particular document collection, the clustering of documents based on the similarity of their content may help to improve the search effectiveness.

V.1.1 Improving Search Recall

Standard search engines and IR systems return lists of documents that match a user query. It is often the case that the same concepts are expressed by different terms in different texts. For instance, a "car" may be called "automobile," and a query for

"car" would miss the documents containing the synonym. However, the overall word contents of related texts would still be similar despite the existence of many synonyms. Clustering, which is based on this overall similarity, may help improve the recall of a query-based search in such a way that when a query matches a document its whole cluster can be returned.

This method alone, however, might significantly degrade precision because often there are many ways in which documents are similar, and the particular way to cluster them should depend on the particular query.

V.1.2 Improving Search Precision

As the number of documents in a collection grows, it becomes a difficult task to browse through the lists of matched documents given the size of the lists. Because the lists are unstructured, except for a rather weak relevance ordering, he or she must know the exact search terms in order to find a document of interest. Otherwise, the he or she may be left with tens of thousands of matched documents to scan.

Clustering may help with this by grouping the documents into a much smaller number of groups of related documents, ordering them by relevance, and returning only the documents from the most relevant group or several most relevant groups.

Experience, however, has shown that the user needs to guide the clustering process so that the clustering will be more relevant to the user's specific interest. An interactive browsing strategy called scatter/gather is the development of this idea.

V.1.3 Scatter/Gather

The scatter/gather browsing method (Cutting et al. 1992; Hearst and Pedersen 1996) uses clustering as a basic organizing operation. The purpose of the method is to enhance the efficiency of human browsing of a document collection when a specific search query cannot be formulated. The method is similar to the techniques used for browsing a printed book. An index, which is similar to a very specific query, is used for locating specific information. However, when a general overview in needed or a general question is posed, a table of contents, which presents the logical structure of the text, is consulted. It gives a sense of what sorts of questions may be answered by more intensive exploration of the text, and it may lead to the particular sections of interest.

During each iteration of a scatter/gather browsing session, a document collection is *scattered* into a set of clusters, and the short descriptions of the clusters are presented to the user. Based on the descriptions, the user selects one or more of the clusters that appear relevant. The selected clusters are then *gathered* into a new subcollection with which the process may be repeated. In a sense, the method dynamically generates a table of contents for the collection and adapts and modifies it in response to the user's selection.

V.1.4 Query-Specific Clustering

Direct approaches to making the clustering query-specific are also possible. The hierachical clustering is especially appealing because it appears to capture the essense

V.3.1 K-Means Algorithm

The K-means algorithm partitions a collection of vectors $\{x_1, x_2, \ldots x_n\}$ into a set of clusters $\{C_1, C_2, \ldots C_k\}$. The algorithm needs k cluster seeds for initialization. They can be externally supplied or picked up randomly among the vectors.

The algorithm proceeds as follows:

Initialization:

k seeds, either given or selected randomly, form the core of k clusters. Every other vector is assigned to the cluster of the closest seed.

Iteration:

The *centroids* M_i of the current clusters are computed:

$$M_i = |C_i|^{-1} \sum_{x \in c_i} x.$$

Each vector is reassigned to the cluster with the closest centroid.

Stopping condition:

At convergence – when no more changes occur.

The K-means algorithm maximizes the clustering quality function Q:

$$Q(C_1, C_2, \ldots, C_k) = \sum_{C_i} \sum_{x \in C_i} Sim(x - M_i).$$

If the distance metric (inverse of the similarity function) behaves well with respect to the centroids computation, then each iteration of the algorithm increases the value of Q. A sufficient condition is that the centroid of a set of vectors be the vector that maximizes the sum of similarities to all the vectors in the set. This condition is true for all "natural" metrics. It follows that the K-means algorithm always converges to a local maximum.

The K-means algorithm is popular because of its simplicity and efficiency. The complexity of each iteration is $O(kn)$ similarity comparisons, and the number of necessary iterations is usually quite small.

A major problem with the K-means algorithm is its sensitivity to the initial selection of seeds. If a bad set of seeds is used, the generated clusters are often very much suboptimal. Several methods are known to deal with this problem. The simplest way is to make several clustering runs with different random choices of seeds. Another possibility is to choose the initial seeds utilizing external domain-dependent information.

Several algorithmic methods of dealing with the K-means suboptimality also exist. One possibility is to allow *postprocessing* of the resulting clusters. For instance, the ISO-DATA algorithm (Jensen 1996) merges clusters if the distance between their centroids is below a certain threshold, and this algorithm splits clusters having excessively high variance. Another possibility is employed by the Buckshot algorithm described at the end of this section.

The best number of clusters, in cases where it is unknown, can be computed by running the K-means algorithm with different values of k and choosing the best one according to any clustering quality function.

V.3.2 EM-based Probabilistic Clustering Algorithm

The underlying assumption of *mixture-resolving* algorithms is that the objects to be clustered are drawn from k distributions, and the goal is to identify the parameters of each that would allow the calculation of the probability $P(C_i \mid x)$ of the given object's belonging to the cluster C_i.

The expectation maximization (EM) is a general purpose framework for estimating the parameters of distribution in the presence of hidden variables in observable data. Adapting it to the clustering problem produces the following algorithm:

Initialization:
The initial parameters of k distributions are selected either randomly or externally.
Iteration:
E-Step: Compute the $P(C_i \mid x)$ for all objects x by using the current parameters of the distributions. Relabel all objects according to the computed probabilities.
M-Step: Reestimate the parameters of the distributions to maximize the likelihood of the objects' assuming their current labeling.
Stopping condition:
At convergence – when the change in log-likelihood after each iteration becomes small.

After convergence, the final labelings of the objects can be used as the fuzzy clustering. The estimated distributions may also be used for other purposes.

V.3.3 Hierarchical Agglomerative Clustering (HAC)

The HAC algorithm begins with each object in separate cluster and proceeds to repeatedly merge pairs of clusters that are most similar according to some chosen criterion. The algorithm finishes when everything is merged into a single cluster. The history of merging provides the binary tree of the clusters hierarchy.

Initialization:
Every object is put into a separate cluster.
Iteration:
Find the pair of most similar clusters and merge them.
Stopping condition:
When everything is merged into a single cluster.

Different versions of the algorithm can be produced as determined by how the similarity between clusters is calculated. In the *single-link* method, the similarity between two clusters is the maximum of similarities between pairs of objects from the two clusters. In the *complete-link* method, the similarity is the minimum of similarities of such pairs of objects. The single-link approach may result in long and thin chainlike clusters, whereas the complete-link method results in tight and compact clusters. Although the single-link method is more versatile, experience suggests that the complete-link one produces more useful results.

Other possible similarity measures include "center of gravity" (similarity between centroids of clusters), "average link" (average similarity between pairs of objects of

clusters), and a "group average" (average similarity between all pairs of objects in a merged cluster), which is a compromise between the single- and complete-link methods.

The complexity of HAC is $O(n^2 s)$, where n is the number of objects and s the complexity of calculating similarity between clusters. For some object similarity measures it is possible to compute the group average cluster similarity in constant time, making the complexity of HAC truly quadratic. By definition, the group average similarity between clusters C_i and C_j is

$$Sim(C_i, C_j) = \frac{1}{|C_i \cup C_j|(|C_i \cup C_j| - 1)} \sum_{x,y \in C_i \cup C_j, x \neq y} Sim(x, y).$$

Assuming that the similarity between individual vector is the cosine similarity, we have

$$Sim(C_i, C_j) = \frac{(S_i + S_j) \cdot (S_i + S_j) - (|C_i| + |C_j|)}{|C_i \cup C_j|(|C_i \cup C_j| - 1)},$$

where $S_i = \sum_{x \in Ci} x$ is the sum of all vectors in the ith cluster. If all S_i's are always maintained, the cosine similarity between clusters can always be computed in a constant time.

V.3.4 Other Clustering Algorithms

Several graph-theoretic clustering algorithms exist. The best known is based on construction of the minimal spanning tree (MST) of the objects and then deleting the edges with the largest lengths to generate clusters. In fact, the hierarchical approaches are also related to graph theoretic clustering. Single-link clusters are subgraphs of the MST, which are also the connected components (Gotlieb and Kumar 1968). Complete-link clusters are the maximal complete subgraphs (Backer and Hubert 1976).

The nearest neighbor clustering (Lu and Fu 1978) assigns each object to the cluster of its nearest labeled neighbor object provided the similarity to that neighbor is sufficiently high. The process continues until all objects are labeled.

The Buckshot algorithm (Cutting et al. 1992) uses the HAC algorithm to generate a good initial partitioning for use by the K-means algorithm. For this purpose, $\sqrt{kn}$ objects are randomly selected, and the group-average HAC algorithm is run on the set. The k clusters generated by HAC are used to initialize the K-means algorithm, which is then run on the whole set of n objects. Because the complexity of HAC is quadratic, the overall complexity of Buckshot remains $O(kn)$ linear in the number of objects.

V.4 CLUSTERING OF TEXTUAL DATA

The clustering of textual data has several unique features that distinguish it from other clustering problems. This section discusses the various issues of representation, algorithms, data abstraction, and evaluation of text data clustering problems.

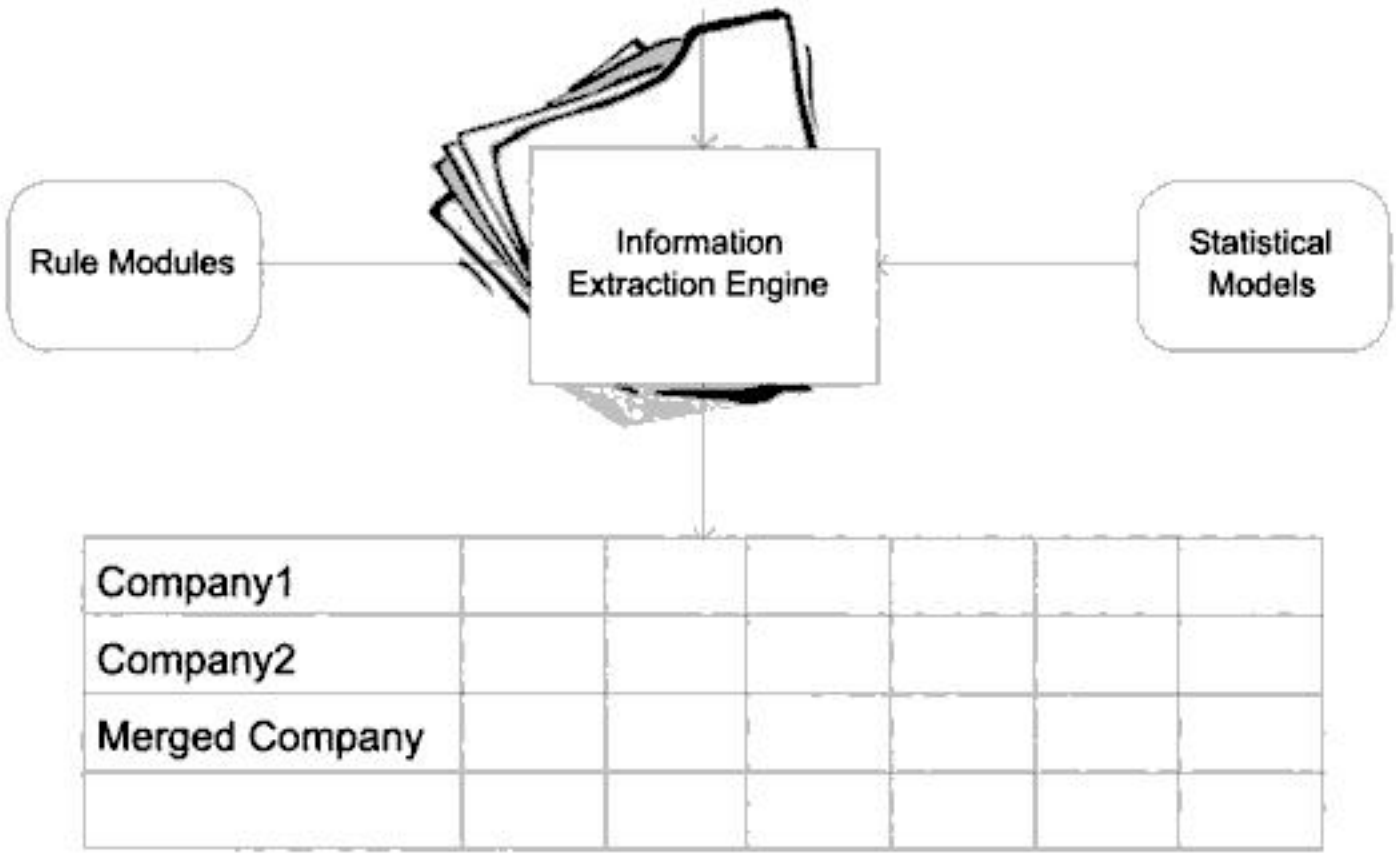


Figure VI.1. Schematic view of the information extraction process.

Consequently, IE methods allow for mining of the actual information present within the text rather than the limited set of tags associated with the documents. The IE process makes the number of different relevant *entities* and *relationships* on which the text mining is performed unbounded – typically thousands or even millions, which would be far beyond the number of tags any automated categorization system could handle. Thus, preprocessing techniques involving IE tend to create more rich and flexible representation models for documents in text mining systems.

IE can be seen as a limited form of "complete text comprehension." No attempt is made to understand the document at hand fully. Instead, one defines a priori the types of semantic information to be extracted from the document. IE represents documents as sets of entities and frames that are another way of formally describing the relationships between the entities.

The set of all possible entities and frames is usually open and very big compared with the set of categorization keywords. It cannot be created manually. Instead, the features are extracted directly from the text. The hierarchy relation between the entities and frames is usually a simple tree. The root has several children – the entity types (e.g., "*Company*," "*Person*," "*Gene*," etc.) under which the actual entities are automatically added as they are being discovered.

The frames constitute structured objects, and so they cannot be directly used as features for text mining. Instead, the frame attributes and its label are used for features. The frame itself, however, may bypass the regular text mining operations and may be fed directly to the querying and visualization components.

The simplest kind of information extraction is called *term extraction*. There are no frames, and there is only one entity type – simply "*term*."

Figure VI.1 gives a schematic view of the IE process. At the heart of the process we have the IE engine that takes a set of documents as input. The engine works by using a statistical model, a rule module, or a mix of both.

The output of the engine is a set of annotated frames extracted from the documents. The frames actually populate a table in which the fields of the frame are the rows of the table.

VI.1.1 Elements That Can Be Extracted from Text

There are four basic types of elements that can, at present, be extracted from text.

- ***Entities.*** Entities are the basic building blocks that can be found in text documents. Examples include people, companies, locations, genes, and drugs.
- ***Attributes.*** Attributes are features of the extracted entities. Some examples of attributes are the title of a person, the age of a person, and the type of an organization.
- ***Facts.*** Facts are the relations that exist between entities. Some examples are an employment relationship between a person and a company or phosphorylation between two proteins.
- ***Events.*** An event is an activity or occurrence of interest in which entities participate such as a terrorist act, a merger between two companies, a birthday and so on.

Figure VI.2 shows a full news article that demonstrates several tagged entities and relationships.

VI.2 HISTORICAL EVOLUTION OF IE: THE MESSAGE UNDERSTANDING CONFERENCES AND TIPSTER

The Defense Advanced Research Project Agency (DARPA) has been sponsoring efforts to codify and expand IE tasks, and the most comprehensive work has arisen from MUC-6 (Message Understanding Conference) and MUC-7 conferences. We now describe the various tasks introduced during the MUC conferences.

VI.2.1 Named Entity Recognition

The named entity recognition (NE, sometimes denoted also as NER) phase is the basic task-oriented phase of any IE system. During this phase the system tries to identify all mentions of proper names and quantities in the text such as the following types taken from MUC-7:

- People names, geographic locations, and organizations;
- Dates and times; and
- Monetary amounts and percentages.

The accuracy (F1) of the extraction results obtained on the NE task is usually quite high, and the best systems manage to get even up to 95-percent breakeven between precision and recall.

The NE task is weakly domain dependent – that is, changing the domain of the texts being analyzed may or may not induce degradation of the performance levels. Performance will mainly depend on the level of generalization used while developing the NE engine and on the similarity between the domains.

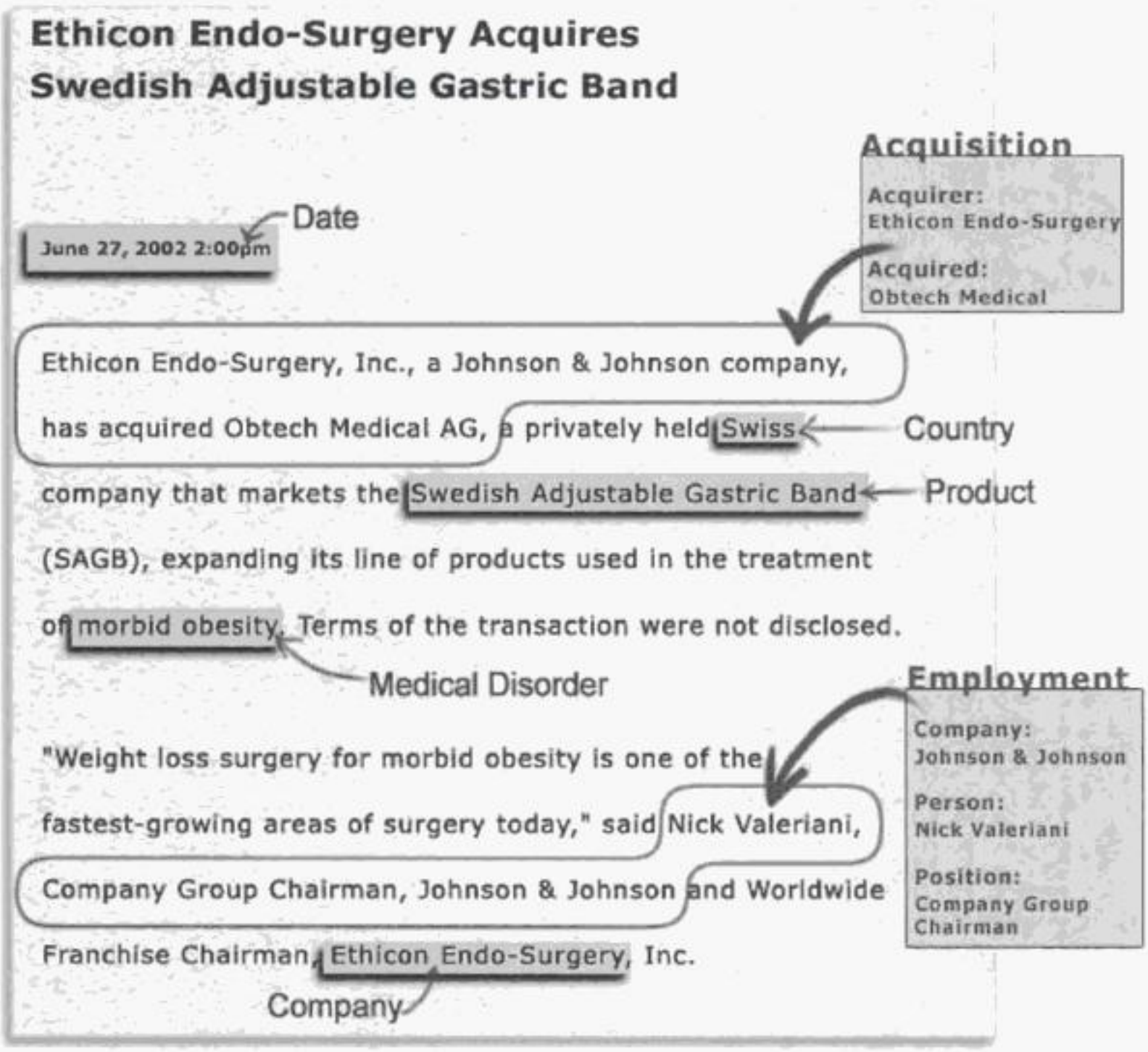


Figure VI.2. A tagged news article.

Proper names usually account for 70 percent of the named entities in the MUC corpuses, dates and times account for 25 percent, and monetary amounts and percentages account for less than 5 percent of the total named entities. Out of the named entities, about 45–50 percent are organization names, 12–32 percent are location tags, and 23–39 percent are people tags.

The MUC committee stipulated that the following types of noun phrases should not be extracted because they do not refer to any specific entity:

- Artifacts (e.g., *Wall Street Journal*, MTV, etc.),
- Common nouns used in anaphoric reference (such as the plane, the company, etc.),
- Names of groups of people and laws named after people (e.g., Republicans, "Gramm–Rudman amendment," "the Nobel Prize," etc.),
- Adjectival forms of location names (e.g., "American," "Japanese," etc.), and
- Miscellaneous uses of numbers that are not specifically currency or percentages.

VI.2.2 Template Element Task

Template element tasks (TEs) are independent or neutral with respect to scenario or domain. Each TE consists of a generic object and some attributes that describe it. This enables separating domain-independent from domain-dependent aspects of extraction.

The TE following types were included in MUC-7:

- Person
- Organization
- Location (airport, city, country, province, region, water)
- Artifact.

Here are examples of TEs. A typical paragraph of text from a press release is as follows below (taken from <http://www.itl.nist.gov/iaui/894.02/related_projects/muc/>):

> Fletcher Maddox, former Dean of the UCSD Business School, announced the formation of La Jolla Genomatics together with his two sons. La Jolla Genomatics will release its product Geninfo in June 1999. L.J.G. is headquartered in the Maddox family's hometown of La Jolla, CA.

One can extract various entities and descriptors. For instance, some of the entities and descriptors that can be automatically extracted from this paragraph by using information extraction algorithms include the following:

```
entity {
ID = 1,
NAME = "Fletcher Maddox"
DESCRIPTOR = "Former Dean of USCD Business School"
CATEGORY = person
}
entity {
ID = 2
NAME = "La Jolla Genomatics"
ALIAS = "LJG"
DESCRIPTOR = ""
CATEGORY = organization
}
entity {
ID = 3
NAME = "La Jolla"
DESCRIPTOR = "the Maddox family hometown"
CATEGORY = location
}
```

VI.2.3 Template Relationship (TR) Task

The Template relationship task (TR) expresses a domain-independent relationship between entities as compared with TEs, which just identify entities themselves. The goal of the TR task is to find the relationships that exist between the template elements extracted from the text (during the TE task). Just like the definition of an entity, entity attributes depend on the problem and the nature of the texts being analyzed; the relationships that may exist between template elements is domain dependent too. For example, persons and companies may be related by employee_of relation, companies and locations may be related by located_of relations, and companies may be interrelated by subdivision_of relations.

The following TRs were extracted from the sample text:

```
employee_of (Fletcher Maddox, UCSD Business School)
employee_of (Fletcher Maddox, La Jolla Genomatics)
product_of (Geninfo, La Jolla Genomatics)
location_of (La Jolla, La Jolla Genomatics)
location_of (CA, La Jolla Genomatics)
```

VI.2.4 Scenario Template (ST)

Scenario templates (STs) express domain and task-specific entities and relations. The main purpose of the ST tasks is to test portability to new extraction problems quickly. This task gives advantage to technologies that are not so labor intensive and hence can port the extraction engine to a new domain in a short time (couple of weeks).

Here are a few events that were extracted from the sample text:

```
company-formation-event {
PRINCIPAL = "Fletcher Maddox"
DATE = ""
CAPITAL = ""
}
product-release-event {
COMPANY = "La Jolla Genomatics"
PRODUCS = "Geninfo"
DATE = "June 1999"
COST = ""
}
```

VI.2.5 Coreference Task (CO)

The coreference task (CO) captures information on coreferring expressions (e.g., pronouns or any other mentions of a given entity), including those tagged in the NE, TE tasks. This CO focuses on the IDENTITY (IDENT) relation, which is symmetrical and transitive. It creates equivalence classes (or coreference chains) used for scoring. The task is to mark nouns, noun phrases, and pronouns.

The Texaco station, at 102 Main Street, Farmers Branch, TX, was severely damaged, but no injuries were reported. Total property damages are estimated at $350,000.

Event:	tornado
Date:	4/3/97
Time:	19:15
Location:	Farmers Branch : "northwest of Dallas" : TX : USA
Damage:	mobile homes
	Texaco station
Estimated Losses:	$350,000
Injuries:	none

VI.3.3 Case 3: Terror-Related Article, MUC-4

19 March – a bomb went off this morning near a power tower in San Salvador leaving a large part of the city without energy, but no causalities have been reported. According to unofficial sources, the bomb – allegedly detonated by urban guerrilla commandos – blew up a power tower in the northwestern part of San Salvador at 0650 (1250 GMT).

Incident Type:	Bombing
Date:	March 19th
Location:	El Salvador: San Salvador (City)
Perpetrator:	urban guerrilla commandos
Physical Target:	power tower
Human Target:	–
Effect of Physical Target:	destroyed
Effect on Human Target:	no injury or death
Instrument	bomb

VI.3.4 Technology-Related Article, TIPSTER-Style Tagging

Here is an article from the MUC-5 evaluation dealing with microelectronics.

<doc>
<REFNO> 000019641 </REFNO>
<DOCNO> 3560177 </DOCNO>
<DD> November 25, 1991 </DD>
<SO> News Release </SO>
<TXT>
Applied Materials, Inc. today announced its newest source technology, called the Durasource, for the Endura(™) 5500 PVD system. This enhanced source includes new magnet configurations, giving the industry's most advanced

sputtered aluminum step coverage in sub-micron contacts, and a new one piece target that more than doubles target life to approximately 8000 microns of deposition compared to conventional two-piece "bonded" targets. The Durasource enhancement is fully retrofittable to installed Endura 5500 PVD systems. The Durasource technology has been specially designed for 200 mm wafer applications, although it is also available for 125 mm and 1s0mm wafer sizes. For example, step coverage symmetry is maintained within 3% between the inner and outer walls of contacts across a 200 mm wafer. Film thickness uniformity averages 3% (3 sigma) over the life of the target.
</TXT>
</doc>

<TEMPLATE-3560177-1> :=
DOC NR: 3560177
DOC DATE: 251192
DOCUMENT SOURCE: "News Release"
CONTENT:
<MICROELECTRONICS_CAPABILITY-3560177-1>
DATE TEMPLATE COMPLETED: 021292
EXTRACTION TIME: 5
COMMENT: "article focuses on nonreportable target source but reportable info available"
/"TOOL_VERSION: LOCKE.3.4"
/"FILLRULES_VERSION: EME.4.0"

<MICROELECTRONICS_CAPABILITY-3560177-1> :=
PROCESS: <LAYERING-3560177-1>
MANUFACTURER: <ENTITY-3560177-1>

<ENTITY-3560177-1> :=
NAME: Applied Materials, INC
TYPE: COMPANY

<LAYERING-3560177-1> :=
TYPE: SPUTTERING
FILM: ALUMINUM
EQUIPMENT: <EQUIPMENT-3560177-1>

<EQUIPMENT-3560177-1> :=
NAME_OR_MODEL: "Endura(TM) 5500"
MANUFACTURER: <ENTITY-3560177-1>
EQUIPMENT_TYPE: PVD_SYSTEM
STATUS: IN_USE
WAFER_SIZE: (200 MM)
(125 MM)
COMMENT: "actually three wafer sizes, third is error 1s0mm"

VI.3.5 Case 5: Comprehensive Stage-by-Stage Example

- ***Original Sentence*:** Mr. Eskew was Vice President of Worldwide Sales for Sandpiper Networks, which was recently acquired by Digital Island where he created the worldwide sales strategy.
- ***After Part of Speech Tagging*:**
  <Prop>Mr. Eskew</Prop> <Verb>was</Verb> <Prop>Vice President</Prop> <Prep>of</Prep> <Prop>Worldwide Sales</Prop> <Prep>for</Prep> <Prop>Sandpiper Networks</Prop> which <Verb>was</Verb> <Adv>recently</Adv> <Verb>acquired</Verb> <Prep>by</Prep> <Prop>Digital Island</Prop> where <Pron>he</Pron> <Verb>created</Verb> <Det>the</Det> <Adj>worldwide</Adj> <Nn>sales strategy.</Nn>
- ***After Shallow Parsing*:**
  NP:{Mr. Eskew} was NP:{Vice President of Worldwide Sales} for NP:{Sandpiper Networks} which was ADV:{recently} V:{acquired} by NP:{Digital Island} where NP:{he} V:{created} NP:{the worldwide sales strategy}
- ***After Named Entity Recognition*:**
  Person:{Mr. Eskew} was Position:{Vice President of Worldwide Sales} for Company:{Sandpiper Networks} which was ADV:{recently} V:{acquired} by Company:{Digital Island} where Person:{he} V:{created} NP:{the worldwide sales strategy}
- ***After Merging (Anaphora Resolution)*:**
  Person:{Mr. Eskew} was Position:{Vice President of Worldwide Sales} for Company:{Sandpiper Networks} which was ADV:{recently} V:{acquired} by Company:{Digital Island} where Person:{Mr. Eskew} V:{created} NP:{the worldwide sales strategy}
- ***Frames Extracted*:**
  Frame Type: **Acquisition**
  Acquiring Company: Digital Island
  Acquired Company: Sandpiper Networks
  Acquisition Status: Historic

  FrameType: **PersonPositionCompany**
  Person: Mr. Eskew
  Position: Vice President of Worldwide Sales
  Company: Sandpiper Networks
  Status: Past

VI.4 ARCHITECTURE OF IE SYSTEMS

Figure VI.5 shows the generalized architecture for a basic IE system of the type that would be used for text mining preprocessing activities. The subcomponents are colored according to their necessity within the full system.

A typical general-use IE system has three to four major components. The first component is a tokenization or *zoning* module, which splits an input document into its basic building blocks. The typical building blocks are words,

Processing the Initial Lexical Content: Tokenization and Lexical Analysis

The first two phases of an IE system really both concern themselves with processing a document to identify various elements of its basic lexical content. As a first pass, a document is divided into tokens, sentences, and possibly paragraphs. Then, each word is tagged by its part of speech and lemma.

In addition, an IE system can use specialized dictionaries and gazetteers to tag words that appear in those word lists. Typical dictionaries include names of countries, cities, people's first names, public companies, company suffixes, common titles in companies, and so on. Dictionary support during initial tagging creates richer document representations. For example, using appropriate dictioniaries, the word "Robert" would be tagged as a "first name," "IBM" would be tagged as a company, and the acronym "spa" could be tagged as "company suffix."

Proper Name Identification

Commonly, the next phase is proper name identification. After an IE system performs the basic lexical analysis, it is typically designed to try to identify a variety of simple entity types such as dates, times, e-mail address, organizations, people names, locations, and so on. The entities are identified by using regular expressions that utilize the context around the proper names to identify their type. The regular expressions can use POS tags, syntactic features, and orthographic features such as capitalization.

Proper name identification is performed by scanning the words in the sentence while trying to match one of the patterns in the predefined set of regular expressions. Each proper name type has its associated set of regular expressions. All patterns are attempted for each word. If more than one pattern is matched, the IE system picks the pattern that matches the longest word sequence. If there is a tie, the IE system usually just uses the first pattern. If no pattern matches, the IE system moves to the next word and reapplies the entire set of patterns. The process continues until the end of the sentence is reached.

To illustrate how such regular expressions are constructed, we present several regular expressions for identifying people names below.

1. @Honorific CapitalizedWord CapitalizedWord
   a. @Honorific is a list of honorific titles such as {Dr., Prof., Mr., Ms., Mrs. etc.)
   b. Example: Mr. John Edwards
2. @FirstNames CapitalizedWord
   a. @FirstNames is a list of common first names collected from sites like the U.S. census and other relevant sites
   b. Example: Bill Hellman
3. CapitalizedWord CapitalizedWord [,] @PersonSuffix
   a. @PersonSuffix is a list of common suffixes such as {Jr., Sr., II, III, etc.}
   b. Example: Mark Green, Jr.
4. CapitalizedWord CapitalLetter [.] CapitalizedWord
   a. CapitalLetter followed by an optional period is a middle initial of a person and a strong indicator that this is a person name
   b. Example: Nancy M. Goldberg

will be useful as a building block for further phases of IE-related preprocessing operations.

Building Relations

The construction of relations between entities is done by using domain-specific patterns. The generality of the patterns depends on the depth of the linguistic analysis performed at the sentence level. If one just performs this analysis against individual noun or verb phrases, or both, then one will need to develop five to six times more patterns than if simply the subject, verb, and object of each sentence were identified. To extract an executive appointment event from the text fragment above, one could use the following pattern:

> Company [Temporal] @Announce Connector Person PersonDetails @Appoint Position

This pattern can be broken down in the following way:

- ***Temporal*** is a phrase indicating a specific date and/or time such as {yesterday, today, tomorrow, last week, an hour ago}
- ***@Announce*** is a set of phrases that correspond to the activity of making a public announcement like {announced, notified, etc.}
- ***Connector*** is a set of connecting words like {that, . . . }
- ***PersonDetails*** is a phrase describing some fact about a person (such as his or her age, current position, etc.); it will be usually surrounded by commas
- ***@Appoint*** is a set of phrases that correspond to the activity of appointing a person to a position like {appointed, named, nominated, etc.}

One of the main tasks during the relation extraction is coreference resolution, which is introduced in Section VI.5. We expand on coreference resolution in Section VI.5.

Inferencing

In many cases, an IE system has to resort to some kind of common sense reasoning and infer missing values to complete the identification of events. The inference rules are written as a formalism similar to Prolog clauses. Common examples include family relations, management changes, spatial relations, and so on.

Below are two examples, one related to location of a person and the other to the position a person is going to fill. The first example is a simple two-sentence text fragment.

> **Example 1:** John Edgar was reported to live with Nancy Leroy. His Address is 101 Forest Rd., Bethlehem, PA.

From this, it is possible to extract the following entities and events:

1. person(John Edgar)
2. person(Nancy Leroy)
3. livetogether(John Edgar, Nancy Leroy)
4. address(John Edgar, 101 Forest Rd., Bethlehem, PA)

Using the following rule, one can infer that Nancy Leroy lives at 101 Forest Rd., Bethlehem, PA.

address(P1,A) :- person(P1), person(P2), livetogether(P1,P2), address(P1,A).

The second example is also a two-sentence text fragment.

Example 2: RedCarpet Inc. announced that its President, JayGoldman, has resigned. The company appointed Fred Robbins to the position.

From this one can extract the following entities and events:

1. company(RedCarpet)
2. person(Jay Goldman)
3. personLeftPosition(Jay Goldman, RedCarpet, President)
4. personReplacesPerson(Fred Robbins, Jay Goldman)

Using the following rule in this second example, one can infer that Fred Robbins is the new President of RedCarpet:

newposition(P2,Pos) :- person(P1), person(P2), company(C1), personLeftPosition(P1,C1,Pos), personReplacesPerson (P2,P1).

VI.5 ANAPHORA RESOLUTION

Anaphora or *coreference resolution* is the process of matching pairs of NLP expressions that refer to the same entity in the real world. It is a process that is critical to the proper function of advanced text mining preprocessing systems.

Below is an example of an annotated text fragment that includes chains of coreffering phrases. We can see here two chains referring to a person (#1, #5), one chain referring to an incident (#2), one chain referring to groups of people (#4), two chains referring to locations (#3, #7), and one chain referring to an organization (#6).

HADERA, Israel$_3$ (AP) – A Palestinian gunman$_1$ walked into a wedding hall in northern Israel$_3$ late Thursday and opened fire, killing six people and injuring 30$_2$, police$_6$ said.... Police$_6$ earlier said the attacker$_1$ threw hand grenades but witnesses and later police$_6$ accounts said the attacker$_1$ opened fire with an M-16 and was$_1$ stopped before he$_1$ could throw a grenade. The Al Aqsa Brigades$_4$, a militia$_4$ linked to Yasser Arafat's Fatah claimed responsibility. The group$_4$ said that Abed Hassouna$_1$ from a village$_7$ near the Palestinian town of Nablus carried out the attack$_2$ to avenge the death of Raed Karmi$_5$, (the militia)$_4$'s leader$_5$ in the town of Tulkarem. Hassouna$_1$ had been a Palestinian policeman$_1$ but left$_1$ the force two years ago, residents of his$_1$ village$_7$ said.

There are two main approaches to anaphora resolution. One is a knowledge-based approach based on linguistic analysis of the sentences and is coded as a rigid algorithm. The other approach is a machine learning approach based on an annotated corpus.

VI.5.1 Pronominal Anaphora

Pronominal anaphora deals with resolving pronouns such as he, she, and they. It is the most common type of coreference. There are three types of pronouns:

- ***Reflexive pronouns***: himself, herself
- ***Personal pronouns***: he, him, you
- ***Possessive pronouns***: her, his, hers

It should be pointed out that not all pronouns in English are anaphoric. For instance, "it" can often be nonanaphoric as in the case of the previous sentence. Other examples of nonanaphoric "it" include expressions such as "It is important," "It is necessary," or "It has to be taken into account." A nonanaphoric "it" is described as *pleonastic*.

VI.5.2 Proper Names Coreference

The task here is to link together all the variations of a proper name (person, organization, location) that are observed in text. For example,

> Former President Bush$_1$ defended the U.S. military Thursday during a speech at one of the nation's largest Army posts, where one private accused of abusing Iraqi prisoners awaits a court-martial. "These are difficult times for the Army as the actions of a handful in Iraq violate the soldier's code," said George H. W. Bush$_1$.

Additional examples can be observed in the example above.

VI.5.3 Apposition

Appositives are used to provide auxiliary information for a named entity. This information is separated from the entity by a comma and either precedes it or comes directly after it as in the following example:

> said George H. W. Bush$_1$, the father of President Bush$_1$. the father of President Bush$_1$, George H. W. Bush$_1$ said. . . .

A necessary condition for an appositional phrase to corefer to a named entity is that they occur in different noun phrases. If the apposition is a modifier of the named entity within the same noun phrase, then they are not considered coreferring as in the following phrase "Former President Bush." In this case *Former President* is not coreferring to Bush.

VI.5.4 Predicate Nominative

A predicate nominative occurs after a copulative verb (is, seems, looks like, appears, etc.) and completes a reference to the subject of a clause.

An example follows:

> Bill Gates$_1$ is the Chairman of Microsoft Corporation$_1$
> Subject: Bill Gates
> Predicate Nominative: the Chairman of Microsoft Corporation

A predicate nominative is a candidate for coreference only if it is stated in a firm way. If it is stated in a speculative or negative way, then it is not a candidate for coreference.

VI.5.5 Identical Sets

In this type of coreference the anaphor and the antecedent both refer to sets that are identical or to identical types. In the following example, "The Al Aqsa Brigades," "a militia," and "The group" all refer to the same set of people.

> The Al Aqsa Brigades$_4$, a militia$_4$ linked to Yasser Arafat's Fatah claimed responsibility. The group$_4$ said that Abed Hassouna$_1$ from a village$_7$ near the Palestinian town of Nablus carried out the attack$_2$ to avenge the death of Raed Karmi$_5$, (the militia)$_4$'s leader$_5$

Identifying identical sets is usually extremely difficult because deep knowledge about the domain is needed.

If we have a lexical dictionary such as WordNet that include hyponyms and hypernyms, we may be able to identify identical sets. We can deduce, for instance, that "militia" is a kind of "group."

VI.5.6 Function–Value Coreference

A function–value coreference is characterized by phrases that have a function–value relationship. Typically, the function will be descriptive and the value will be numeric.

In the following text there are two function–value pairs:

> Evolved Digital Systems's Revenues$_1$ were \$4.1M$_1$ for the quarter, up 61% compared to the first quarter of 2003. Net Loss$_2$ declined by 34% to \$5.6M$_2$.

Function: Evolved Digital Systems's Revenues
Value: \$4.1M
Function: Net Loss
Value: \$5.6M

VI.5.7 Ordinal Anaphora

Ordinal anaphora involves a cardinal number like *first* or *second* or an adjective such as *former* or *latter* as in the following example:

> IBM and Microsoft$_1$ were the final candidates, but the agency preferred the latter company$_1$.

VI.5.8 One-Anaphora

A one-anaphora consists of an anaphoric expression realized by a noun phrase containing the word "one" as in the following:

> If you cannot attend a tutorial$_1$ in the morning, you can go for an afternoon one$_1$.

VI.5.11 CogNIAC (Baldwin 1995)

CogNIAC is a pronoun resolution engine designed around the assumption that there is a subclass of anaphora that does not require general purpose reasoning. Among the kinds of information CogNIAC does require are POS tagging, simple noun phrase recognition, and basic semantic category information like gender and number.

The system is based on a set of high-confidence rules that are successively applied over the pronoun under consideration. The rules are ordered according to their importance and relevance to anaphora resolution. The processing of a pronoun stops when one rule is satisfied. Below are listed the six rules used by the system. For each of them, the ***sentence prefix of anaphor*** is defined as the text portion of the sentence from the beginning of the sentence to the position of the anaphor.

1. ***Unique Antecedent.***
   Condition: If there is a single valid antecedent ***A*** in the relevant discourse.
   Action: ***A*** is the selected antecedent.
2. ***Reflexive Pronoun.***
   Condition: If the anaphor is a reflexive pronoun.
   Action: Pick the nearest valid antecedent in the anaphor prefix of the current sentence.
3. ***Unique in Current + Preceding.***
   Condition: If there is a single valid antecedent A in the preceding sentence and anaphor prefix of the current sentence.
   Action: ***A*** is the selected antecedent.
   Example: **Rupert Murdock**'s News Corp. confirmed his interest in buying back the ailing New York Post. But analysts said that if **he** winds up bidding for the paper, ...
4. ***Possessive Pronoun.***
   Condition: If the anaphor is a possessive pronoun and there is a single exact copy of the possessive phrase in the previous sentence.
   Action: The antecedent of the latter copy is the same as the former.
   Example: After he was dry, Joe carefully laid out the damp towel in front of **his locker**. Travis went over to **his locker**, took out a towel and started to dry off.
5. ***Unique in Current Sentence.***
   Condition: If there is a single valid antecedent ***A*** in the anaphor-prefix of the current sentence
   Action: ***A*** is the selected antecedent.
6. ***Unique Subject–Subject Pronoun.***
   Condition: If the subject of the previous sentence is a valid antecedent *A* and the anaphor is the subject of the current sentence.
   Action: ***A*** is the selected antecedent.

VI.5.11.1 Kennedy and Boguraev

This approach is based on Lappin and Leass's (1994) method but without the need for full parsing. This algorithm was used to resolve personal pronouns, reflexives, and possessives. The algorithm works by constructing coreference equivalence classes.

The boosting indicators assign a positive score to a matching candidate, reflecting a positive likelihood that it is the antecedent of the current pronoun. In contrast, the impeding indicators apply a negative score to the matching candidate, reflecting a lack of confidence that it is the antecedent of the current pronoun. The candidate with the highest combined score is selected.

Here are some of the indicators used by Mitkov:

- ***Definiteness.*** Definite noun phrases in previous sentences are more likely antecedents of pronominal anaphors than indefinite ones (definite noun phrases score 0 and indefinite ones are penalized by −1).
- ***Givenness.*** Noun phrases in previous sentences representing the "given information" are deemed good candidates for antecedents and score 1 (candidates not representing the theme score 0). The given information is usually the first noun phrase in a nonimperative sentence.
- ***Indicating Verbs.*** If a verb in the sentence has a stem that is a member of {discuss, present, illustrate, identify, summarize, examine, describe, define, show, check, develop, review, report, outline, consider, investigate, explore, assess, analyze, synthesize, study, survey, deal, cover}, then the first NP following the verb is the preferred antecedent.
- ***Lexical Reiteration.*** Lexically reiterated noun phrases are preferred as candidates for antecedent (an NP scores 2 if is repeated within the same paragraph twice or more, 1 if repeated once, and 0 if it is not repeated). The matching is done in a loose way such that synonyms and NPs sharing the same head are considered identical.
- ***Section Heading Preference.*** If a noun phrase occurs in the heading of the section containing the current sentence, then we consider it the preferred candidate.
- ***"Nonprepositional" Noun Phrases.*** A "nonprepositional" noun phrase is given a higher preference than a noun phrase that is part of a prepositional phrase (0, −1). Example: `Insert the cassette into the VCR making sure it is suitable for the length of recording`. Here VCR is penalized for being part of a prepositional phrase and is resolved to the cassette.
- ***Collocation Pattern Preference.*** This preference is given to candidates having an identical verb collocation pattern with a pronoun of the pattern "noun phrase (pronoun), verb" and "verb, noun phrase (pronoun)." *Example*: `Press the key down and turn the volume up... Press it again`. Here `key` is preferred antecedent because it shares the same verb (press) with the pronoun ("it").
- ***Immediate Reference.*** Given a pattern of the form "... You? V1 NP... con you? V2 it (*con* you? V3 it)", where *con* ∈ {and/or/before/after...}, the noun phrase immediately after V1 is a very likely candidate for the antecedent of the pronoun "it" immediately following V2 and is therefore given preference (scores 2 and 0). Example:

> To print the paper$_1$, you can stand the printer$_2$ up or lay it$_2$ flat. To turn on the printer$_2$, press the Power button$_3$ and hold it$_3$ down for a moment. Unwrap the the paper$_1$, form it$_1$ and align it$_1$ then load it$_1$ into the drawer.

- ***Referential distance.*** In complex sentences, noun phrases receive the following scores based on how close they are to the anaphor:
  - previous clause: 2
  - previous sentence: 1
  - 2 sentences: 0
  - 3 sentences further back: −1

  In simple sentences, the scores are as follows:
  - previous sentence: 1
  - 2 sentences: 0
  - 3 sentences further back: −1
- ***Domain Terminology Preference.*** NPs representing domain terms are more likely to be the antecedent (score 1 if the NP is a term and 0 if not).

VI.5.11.3 Evaluation of Knowledge-Poor Approaches

For many years, one of the main problems in contrasting the performance of the various systems and algorithms had been that there was no common ground on which such a comparison could reasonably be made. Each algorithm used a different set of documents and made different types of assumptions.

To solve this problem, Barbu (Barbu and Mitkov 2001) proposed the idea of the "evaluation workbench" – an open-ended architecture that allows the incorporation of different algorithms and their comparison on the basis of the same preprocessing tools and data. The three algorithms just described were all implemented and compared using the same workbench.

The three algorithms implemented receive as input the same representation of the input file. This representation is generated by running an XML parser over the file resulting from the preprocessing phase. Each noun phrase receives the following list of features:

- the original word form
- the lemma of the word or of the head of the noun phrase
- the starting and ending position in the text
- the part of speech
- the grammatical function (subject, object . . .)
- the index of the sentence that contains the referent
- the index of the verb related to the referent.

In addition, two definitions should be highlighted as follows:

- ***Precision*** = number of correctly resolved anaphors / number of anaphors attempted to be resolved
- ***Success Rate*** = number of correctly resolved anaphors / number of all anaphors.

The overall results as reported in Mitkov are summarized in the following table:

	K&B	Cogniac	Mitkov
Precision	52.84%	42.65%	48.81%
Success	61.6%	49.72%	56.9%

VI.5.11.4 Machine Learning Approaches

One of the learning approaches (Soon et al. 2001) is based on building a classifier based on the training examples in the annotated corpus. This classifier will be able to take any pair of NLP elements and return true if they refer to the same real-world entity and false otherwise. The NLP elements can be nouns, noun phrases, or pronouns and will be called *markables*.

The markables are derived from the document by using the regular NLP preprocessing steps as outlined in the previous section (tokenization, zoning, part of speech tagging, noun phrase extraction and entity extraction). In addition to deriving the markables, the preprocessing steps make it possible to create a set of features for each of the markables. These features are used by the classifier to determine if any two markables have a coreference relation.

Some Definitions

- ***Indefinite Noun Phrase.*** An indefinite noun phrase is a phrase that is used to introduce a specific object or set of objects believed to be new to the addressee (e.g., a new automobile, some sheep, and five accountants).
- ***Definite Noun Phrase.*** This is a noun phrase that starts with the article "the."
- ***Demonstrative Noun Phrase.*** This is a noun phrase that starts with "this," "that," "those," or "these."

Features of Each Pair of Markables

- Sentence Distance: 0 if the markables are in the same sentence.
- Pronoun: 1 if one of the markables is a pronoun; 0 otherwise.
- Exact Match: 1 if the two markables are identical; 0 otherwise.
- Definite Noun Phrase: 1 if one of the markables if a definite noun phrase; 0 otherwise.
- Demonstrative Noun Phrase: 1 if one of the markables is a demonstrative noun phrase.
- Number Agreement: 1 if the both markables are singular or plural; 0 otherwise.
- Semantic Agreement: 1 if the markables belong to the same semantic class (based on the entity extraction component).
- Gender Agreement: 1 if the two markables have the same gender (male, female), 0 if not, and 2 if it is unknown.
- Proper Name: 1 if both markables are proper names; 0 otherwise.
- Alias: 1 if one markable is an alias of the other entity (like GE and General Motors).

Generating Training Examples

- ***Positive Examples.*** Assume that in a given document we have found four markables that refer to the same real-world entity, {M1,M2,M3,M4}. For each adjacent pair of markables we will generate a positive example. In this case, we will have three positive examples – namely {M1,M2}, {M2,M3} and {M3,M4}.
- ***Negative Examples.*** Assume that markables a,b,c appear between M1 and M2; then, we generate three negative examples {a,M2}, {b,M2}, {c,M2}.

The Algorithm

```
Identify all markables
For each anaphor A
    Let M1 to Mn be all markables from the
beginning of the document till A
    For i=n;i=1;i - -
        if PairClassifier(A,Mi)=true then
            A, Mi is an anaphoric pair
            exit
        end if
    end for
end for
```

Evaluation

Training on 30 documents yielded a classifier that was able to achieve precision of 68 percent and recall of 52 percent (F1 = 58.9%).

Ng and Cardie (Ng and Cardie 2002) have suggested two types of extensions to the Soon et al. corpus-based approach. First, they applied three extralinguistic modifications to the machine learning framework, which together provided substantial and statistically significant gains in coreference resolution precision. Second, they expanded the Soon et al. feature set from 12 features to an arguably deeper set of 53.

Ng and Cardie have also proposed additional lexical, semantic, and knowledge-based features – most notably, 26 additional grammatical features that include a variety of linguistic constraints and preferences. The main modifications that were suggested by Ng and Cardie are as follows:

- ***Best-first Clustering.*** Rather than a right-to-left search from each anaphoric NP for the first coreferent NP, a right-to-left search for a *highly likely antecedent* was performed. As a result, the coreference clustering algorithm was modified to select as the antecedent of NP the NP with the highest coreference likelihood value from among preceding NPs with coreference class values above 0.5.
- ***Training Set Creation.*** Rather than generate a positive training example for each anaphoric NP and its **closest** antecedent, a positive training example was generated for its **most confident** antecedent. For a nonpronominal NP, the closest **nonpronominal** preceding antecedent was selected as the most confident antecedent. For pronouns, the closest preceding antecedent was selected as the most confident antecedent.
- ***String Match Feature.*** Soon's string match feature (SOON STR) tests whether the two NPs under consideration are the same string after removing determiners from each. Rather than using the same string match for all types of anaphors, finer granularity is used. Exact string match is likely to be a better coreference predictor for proper names than it is for pronouns, for example. Specifically, the SOON STR feature is replaced by three features – PRO STR, PN STR, and WORDS STR – that restrict the application of string matching to pronouns, proper names, and nonpronominal NPs, respectively.

end a field. The learning algorithm approximates each X function by taking a set of pairs of the form $\{i, X\}(i)$ as training data. Each field is extracted by a wrapper $W = <F, A, H>$ where

F is a set of begin boundary detectors
A is a set of end boundary detectors
$H(k)$ is the probability that the field has length k

A boundary detector is just a sequence of tokens with wild cards (some kind of a regular expression).

$$W(i, j) = \begin{cases} 1 & \text{if} \quad F(i)A(j)H(j - i + 1) > \sigma \\ 0 & \text{otherwise} \end{cases}$$

$$F(i) = \sum_k C_{F_k} F_k(i), \quad A(i) = \sum_k C_{A_k} A_k(i).$$

$W(i, j)$ is a nave Bayesian approximation of the probability that we have a field between token i and j with uniform priors. Clearly, as σ is set to be higher we get better precision and lower recall, and if we set σ to be 0 we get the highest recall but compromise precision.

The BWI algorithm learns two detectors by using a greedy algorithm that extends the prefix and suffix patterns while there is an improvement in the accuracy. The sets $F(i)$ and $A(i)$ are generated from the detectors by using the AdaBoost algorithm. The detector pattern can include specific words and regular expressions that work on a set of wildcards such as <num>, <Cap>, <LowerCase>, <Punctuation> and <Alpha>.

When the BWI algorithm was evaluated on the acquisition relations from the Reuters news collection, it achieved the following results compared with HMM:

Slot	BWI	HMM
Acquiring Company	34.1%	30.9%
Dollar Amount	50.9%	55.5%

VI.6.3 The (LP)^2 Algorithm

The (LP)^2 algorithm learns from an annotated corpus and induces two sets of rules: tagging rules generated by a bottom-up generalization process and correction rules that correct mistakes and omissions done by the tagging rules.

A tagging rule is a pattern that contains conditions on words preceding the place where a tag is to be inserted and conditions on the words that follow the tag. Conditions can be either words, lemmas, lexical categories (such as digit, noun, verb, etc), case (lower or upper), and semantic categories (such as time-id, cities, etc).

VI.7 STRUCTURAL IE

VI.7.1 Introduction to Structural IE

Most text mining systems simplify the structure of the documents they process by ignoring much of the structural or visual characteristics of the text (e.g., font type, size, location, etc.) and process the text either as a linear sequence or as a bag of words. This allows the algorithms to focus on the semantic aspects of the document. However, valuable information is lost in these approaches, which ignore information contained in the visual elements of a document.

Consider, for example, an article in a journal. The title is readily identifiable based on its special font and location but less so based on its semantic content alone, which may be similar to the section headings. This holds true in the same way for the author names, section headings, running title,and so on. Thus, much important information is provided by the visual layout of the document – information that is ignored by most text mining and other document analysis systems.

One can, however, leverage preprocessing techniques that do not focus on the semantic content of the text but instead on the visual layout alone in an effort to extract the information contained in layout elements. These type of techniques entail an IE task in which one is provided a document and seeks to discover specific fields of the document (e.g., the title, author names, publication date, figure captions, bibliographical citings, etc.). Such techniques have been termed *structural* or *visual information extraction*.

Of course, it goes without saying that, within the overall context of text mining preprocessing, a structural or visual IE approach is not aimed at replacing the semantic one. Instead, the structural IE approach can be used to complement other more conventional text mining preprocessing processes.

This section describes a recently developed general algorithm that allows the IE task to be performed based on the visual layout of the document. The algorithm employs a machine learning approach whereby the system is first provided with a set of training documents in which the desired fields are manually tagged. On the basis of these training examples, the system automatically learns how to find the corresponding fields in future documents.

VI.7.2 Overall Problem Definition

A document D is a set of primitive elements $D = \{e_1, \ldots, e_n\}$. A primitive element can be a character, a line, or any other visual object as determined by the document format. A primitive element can have any number of visual attributes such as font size and type, physical location, and so on. The *bounding box* attribute, which provides the size and location of the bounding box of the element, is assumed to be available for all primitive elements. We define an *object* in the document to be any set of primitive elements.

The *visual information extraction (VIE)* task is as follows. We are provided with a set of *target fields* $F = \{f_1, \ldots, f_k\}$ to be extracted and a set of *training documents* $T = \{T_1, \ldots, T_m\}$ wherein all occurrences of the target fields are annotated. Specifically, for each target field f and training document T, we are provided with the object

$f(T)$ of T that is of type f ($f(T) = 0$ if f does not appear in T). The goal, when presented with an unannotated query document Q, is to annotate the occurrences of target fields that exist in Q (not all target fields need be present in each document).

Practically, the VIE task can be decomposed into two subtasks. First, for each document (both training and query) one must group the primitive elements into meaningful objects (e.g., lines, paragraphs, etc.) and establish the hierarchical structure among these objects. Then, in the second stage, the structure of the query document is compared with the structures of the training documents to find the objects corresponding to the target fields.

It has also proven possible to enhance the results by introducing the notion of *templates*, which are groups of training documents with a similar layout (e.g., articles from the same journal). Using templates, one can identify the essential features of the page layout, ignoring particularities of any specific document. Templates are discussed in detail in the sections that follow.

A brief examination is also made of a real-world system that was implemented for a typical VIE task involving a set of documents containing financial analyst reports. The documents were in PDF format. Target fields included the title, authors, publication dates, and others.

VI.7.3 The Visual Elements Perceptual Grouping Subtask

Recall that a document is a set of primitive elements such as characters, figures, and so on. The *objects* of a document are sets of primitive elements. Target fields, in general, are objects.

Thus, the first step in the visual IE task is to group the primitive elements of the documents into higher level objects. The grouping should provide the conceptually meaningful objects of the document such as paragraphs, headings, and footnotes.

For humans, the grouping process is easy and is generally performed unconsciously based on the visual structure of the document. As with other types of IE perceptual grouping requirements, the goal is to mimic the human perceptual grouping process.

VI.7.4 Problem Formulation for the Perceptual Grouping Subtask

One can model the structure of the objects of a document as a tree, of which the leaves are primitive elements and the internal nodes are (composite) objects. This structure is called the object tree or *O-Tree* of the document.

The O-Tree structure creates a hierarchal structure among objects in which higher level objects consist of groups of lower level objects. This hierarchal structure reflects the conceptual structure of documents in which objects such as columns are groups of paragraphs, which, in turn, are groups of lines, and so on. The exact levels and objects represented in the O-Tree are application and format dependent.

For an HTML document, for example, the O-Tree may include objects representing tables, menus, the text body, and other elements, whereas for PDF documents the O-Tree may include objects representing paragraphs, columns, lines, and so on. Accordingly, for each file format and application we define the *object hierarchy*, H,

which determines the set of possible *object types*, and a hierarchy among these objects. Any object hierarchy must contain an object of type document, which is at the root of the hierarchy. When constructing an O-Tree for a document, each object is labeled by one of the *object types* defined in the object hierarchy, and the tree structure must correspond to the hierarchical structure defined in the hierarchy.

Formally, an object hierarchy H is a rooted DAG that satisfies the following:

- The leaf nodes are labeled by primitive element types.
- Internal nodes are labeled by objects types.
- The root node is labeled by the document object type.
- For object types x and y, type y is a child of x if an object of type x can (directly) contain an object type y.

For a document $D = \{e_1, \ldots, e_n\}$ and an object hierarchy H, an *O-Tree of D* according to H is a tree O with the following characteristics:

- The leaves of O consist of all primitive elements of D.
- Internal nodes of O are objects of D.
- If X and X' are nodes of O (objects or primitive elements) and $X \subset X'$, then X' is an ancestor (or parent) of X.
- Each node X is labeled by a label from H denoted *label*(X).
- If X' is a parent of X in T, then *label*(X') is a parent of *label*(X) in H.
- *label*$(root)$ = Document.

VI.7.5 Algorithm for Constructing a Document O-Tree

Given a document, one constructs an O-Tree for the document. In doing so, the aim is to construct objects best reflecting the true grouping of the elements into "meaningful" objects (e.g., paragraphs, columns, etc.). When constructing an object we see to it that the following requirements are met:

- The elements of the objects are within the same physical area of the page. Specifically, each object must be *connected* – that is, an object X cannot be decomposed into two separate objects $X1$ and $X2$ such that any line connecting $X1$ and $X2$ necessarily crosses an element in a different object.
- The elements of the object have similar characteristics (e.g., similar font type, similar font size, etc.). Specifically, one must assume a *fitness function fit*(.,.) such that for any two objects X and Y, where *label*(Y) is child of *label*(X), *fit*(Y,X) provides a measure of how fit Y is as an additional member to X (e.g., if X is a paragraph and Y a line, then how similar is Y the other lines in X). One adds Y to X only if *fit*$(Y;X)$ is above some threshold value, $^\circ$. The exact nature of the function *fit*$(¢; ¢)$ and the threshold value are format and domain dependent.

Given these two criteria, the O-Tree can be constructed in a greedy fashion, from the bottom up, layer by layer. In doing so, one should always prefer to enlarge existing objects of the layer, starting with the largest object. If no existing object can be enlarged, and there are still "free" objects of the previous layer, a new object is created. The procedure terminates when the root object, labeled Document, is

completed. A description of the algorithm is provided in the following pseudocode algorithm:

Input: D - Document
Output: O-Tree for D

1. For each type $t \in H$ do
   let *level(t)* be the length of the longest path from t to a leaf
2. Let h = *level*(Document)
3. *Objects*(0) $\leftarrow D$
4. For $i = 1$ to h do
5. *Objects(i)* $\leftarrow 0$
6. *free* $\leftarrow$ *Objects(i* $-$ 1)
7. *While free* $\neq 0$ do
8. For each X $\in$ *Objects(i)* in order of descending size do
9. For each $Y \in$ *free* in order of increasing distance from X do
10. If Y is a neighbor of X and *fit(Y, X)* $\geq \gamma$ then
11. $X \leftarrow X \cup Y$
12. *make Y a child of X*
13. Remove Y from *free*
14. Break (go to line 7)
15. For each $t \in H$ such that *level(t)* = i do
16. if *Objects(i)* does not include an empty object of type t
17. Add an empty set of type t to *Objects(i)*
18. end while
19. Remove empty objects from *Objects*(i)
20. end for
21. return resulting O-Tree

VI.7.6 Structural Mapping

Given a Visual Information Extraction task, one first constructs an O-Tree for each of the training documents as well as for the query document, as described in the previous section. Once all the documents have been structured as O-Trees, it is necessary to find the objects of Q (the query document) that correspond to the target fields. This is done by comparing the O-Tree of Q, and the objects therein, to those of the training documents.

This comparison is performed in two stages. First, the training document that is visually most similar to the query document is found. Then, one maps between the objects of the two documents to discover the targets fields in the query document.

VI.7.6.1 Basic Algorithm

- ***Document Similarity.*** Consider a query document Q and training documents $T = \{T_1, \ldots, T_n\}$. We seek to find the training document T_{opt} that is *visually*

Text mining is a new and exciting area of computer science research that tries to solve the crisis of information overload by combining techniques from data mining, machine learning, natural language processing, information retrieval, and knowledge management. Similarly, link detection – a rapidly evolving approach to the analysis of text that shares and builds on many of the key elements of text mining – also provides new tools for people to better leverage their burgeoning textual data resources. Link detection relies on a process of building up networks of interconnected objects through various relationships in order to discover patterns and trends. The main tasks of link detection are to extract, discover, and link together sparse evidence from vast amounts of data sources, to represent and evaluate the significance of the related evidence, and to learn patterns to guide the extraction, discovery, and linkage of entities.

The Text Mining Handbook presents a comprehensive discussion of the state of the art in text mining and link detection. In addition to providing an in-depth examination of core text mining and link detection algorithms and operations, the work examines advanced preprocessing techniques, knowledge representation considerations, and visualization approaches. Finally, the book explores current real-world, mission-critical applications of text mining and link detection in such varied fields as corporate finance business intelligence, genomics research, and counterterrorism activities.

Dr. Ronen Feldman is a Senior Lecturer in the Mathematics and Computer Science Department of Bar-Ilan University and Director of the Data and Text Mining Laboratory. Dr. Feldman is cofounder, Chief Scientist, and President of ClearForest, Ltd., a leader in developing next-generation text mining applications for corporate and government clients. He also recently served as an Adjunct Professor at New York University's Stern School of Business. A pioneer in the areas of machine learning, data mining, and unstructured data management, he has authored or coauthored more than 70 published articles and conference papers in these areas.

James Sanger is a venture capitalist, applied technologist, and recognized industry expert in the areas of commercial data solutions, Internet applications, and IT security products. He is a partner at ABS Ventures, an independent venture firm founded in 1982 and originally associated with technology banking leader Alex. Brown and Sons. Immediately before joining ABS Ventures, Mr. Sanger was a Managing Director in the New York offices of DB Capital Venture Partners, the global venture capital arm of Deutsche Bank. Mr. Sanger has been a board member of several thought-leading technology companies, including Inxight Software, Gomez Inc., and ClearForest, Inc; he has also served as an official observer to the boards of AlphaBlox (acquired by IBM in 2004), Intralinks, and Imagine Software and as a member of the Technical Advisory Board of Qualys, Inc.

CAMBRIDGE
UNIVERSITY PRESS
www.cambridge.org

ISBN 0-521-83657-3

9 780521 836579 >