1

Introduction to Text Mining

I.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.

I.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.

lead to some word-level representations of document collections having tens or even hundreds of thousands of unique words in its feature space. However, most word-level document representations exhibit at least some minimal optimization and therefore consist of subsets of representative features filtered for items such as stop words, symbolic characters, and meaningless numerics.

- ***Terms.*** *Terms* are single words and multiword phrases selected directly from the corpus of a native document by means of *term-extraction* methodologies. Term-level features, in the sense of this definition, can *only* be made up of specific words and expressions found within the native document for which they are meant to be generally representative. Hence, a term-based representation of a document is necessarily composed of a subset of the terms in that document. For example, if a document contained the sentence

  > President Abraham Lincoln experienced a career that took him from log cabin to White House,

  a list of terms to represent the document could include single word forms such as "Lincoln," "took," "career," and "cabin" as well as multiword forms like "President Abraham Lincoln," "log cabin," and "White House."

  Several of term-extraction methodologies can convert the raw text of a native document into a series of *normalized terms* – that is, sequences of one or more tokenized and lemmatized word forms associated with part-of-speech tags. Sometimes an external lexicon is also used to provide a controlled vocabulary for term normalization. Term-extraction methodologies employ various approaches for generating and filtering an abbreviated list of most meaningful *candidate terms* from among a set of normalized terms for the representation of a document. This culling process results in a smaller but relatively more semantically rich document representation than that found in word-level document representations.
- ***Concepts.***[2] *Concepts* are features generated for a document by means of manual, statistical, rule-based, or hybrid *categorization* methodologies. Concept-level features can be manually generated for documents but are now more commonly extracted from documents using complex preprocessing routines that identify single words, multiword expressions, whole clauses, or even larger syntactical units that are then related to specific *concept identifiers*. For instance, a document collection that includes reviews of sports cars may not actually include the specific word "automotive" or the specific phrase "test drives," but the concepts "automotive" and "test drives" might nevertheless be found among the set of concepts used to to identify and represent the collection.

  Many categorization methodologies involve a degree of cross-referencing against an external knowledge source; for some statistical methods, this source might simply be an annotated collection of training documents. For manual and rule-based categorization methods, the cross-referencing and validation of prospective concept-level features typically involve interaction with a "gold standard" such as a preexisting domain ontology, lexicon, or formal concept

[2] Although some computer scientists make distinctions between keywords and concepts (e.g., Blake and Pratt 2001), this book recognizes the two as relatively interchangeable labels for the same feature type and will generally refer to either under the label *concept*.

hierarchy – or even just the mind of a human domain expert. Unlike word- and term-level features, concept-level features can consist of words not specifically found in the native document.

Of the four types of features described here, terms and concepts reflect the features with the most condensed and expressive levels of semantic value, and there are many advantages to their use in representing documents for text mining purposes. With regard to the overall size of their feature sets, term- and concept-based representations exhibit roughly the same efficiency but are generally much more efficient than character- or word-based document models. Term-level representations can sometimes be more easily and automatically generated from the original source text (through various term-extraction techniques) than concept-level representations, which as a practical matter have often entailed some level of human interaction.

Concept-level representations, however, are much better than any other feature-set representation at handling synonymy and polysemy and are clearly best at relating a given feature to its various hyponyms and hypernyms. Concept-based representations can be processed to support very sophisticated concept hierarchies, and arguably provide the best representations for leveraging the domain knowledge afforded by ontologies and knowledge bases.

Still, concept-level representations do have a few potential drawbacks. Possible disadvantages of using concept-level features to represent documents include (a) the relative complexity of applying the heuristics, during preprocessing operations, required to extract and validate concept-type features and (b) the domain-dependence of many concepts.[3]

Concept-level document representations generated by categorization are often stored in vector formats. For instance, both CDM-based methodologies and Los Alamos II–type concept extraction approaches result in individual documents being stored as vectors.

Hybrid approaches to the generation of feature-based document representations can exist. By way of example, a particular text mining system's preprocessing operations could first extract terms using term extraction techniques and then match or normalize these terms, or do both, by winnowing them against a list of meaningful entities and topics (i.e., concepts) extracted through categorization. Such hybrid approaches, however, need careful planning, testing, and optimization to avoid having dramatic – and extremely resource-intensive – growth in the feature dimensionality of individual document representations without proportionately increased levels of system effectiveness.

For the most part, this book concentrates on text mining solutions that rely on documents represented by concept-level features, referring to other feature types where necessary to highlight idiosyncratic characteristics or techniques. Nevertheless, many of the approaches described in this chapter for identifying and browsing patterns within document collections based on concept-level representations can also

[3] It should at least be mentioned that there are some more distinct disadvantages to using manually generated concept-level representations. For instance, manually generated concepts are fixed, labor-intensive to assign, and so on. See Blake and Pratt (2001).

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 subdomains 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?*

In these illustrative questions, individual "news topics" can be seen as specific concepts in the document collection. Different types of trend analytics attempt to compare the frequencies of such concepts (i.e., number of occurrences) in the documents that make up the two periods' respective document subcollections. Additional types of analysis, also derived from data mining, that can be used to support trend analysis are *ephemeral association discovery* and *deviation detection*. Some specific methods of trend analysis are described in Section II.1.5.

I.1.4 The Importance of the Presentation Layer

Perhaps the key presentation layer functionality supported by text mining systems is *browsing*. Most contemporary text mining systems support browsing that is both dynamic and *content-based*, for the browsing is guided by the actual textual content of a particular document collection and not by anticipated or rigorously prespecified structures. Commonly, user browsing is facilitated by the graphical presentation of concept patterns in the form of a hierarchy to aid interactivity by organizing concepts for investigation.

Browsing is also *navigational*. Text mining systems confront a user with extremely large sets of concepts obtained from potentially vast collections of text documents. Consequently, text mining systems must enable a user to move across these concepts in such a way as to always be able to choose either a "big picture" view of the collection in toto or to drill down on specific – and perhaps very sparsely identified – concept relationships.

Visualization tools are often employed by text mining systems to facilitate navigation and exploration of concept patterns. These use various graphical approaches to express complex data relationships. In the past, visualization tools for text mining sometimes generated static maps or graphs that were essentially rigid snapshots of patterns or carefully generated reports displayed on the screen or printed by an attached printer. State-of-the-art text mining systems, however, now increasingly rely on highly interactive graphic representations of search results that permit a user to drag, pull, click, or otherwise directly interact with the graphical representation of concept patterns. Visualization approaches, like that seen in Figure I.1, are discussed more fully in Chapter X.

Several additional types of functionality are commonly supported within the front ends of text mining systems. Because, in many respects, the presentation layer of a text mining system really serves as the front end for the execution of the system's core knowledge discovery algorithms, considerable attention has been focused on providing users with friendlier and more powerful methods for executing these algorithms. Such methods can become powerful and complex enough to necessitate developing dedicated *query languages* to support the efficient parameterization and execution of specific types of pattern discovery queries. The use of the presentation layer for query execution and simple browsing is discussed in Chapter IX.

At the same time, consistent with an overall emphasis on user empowerment, the designers of many text mining systems have moved away from limiting a user to running only a certain number of fixed, preprogrammed search queries. Instead, these text mining systems are designed to expose much of their search functionality to the user by opening up direct access to their query languages by means of *query language interfaces* or *command-line query interpreters*.

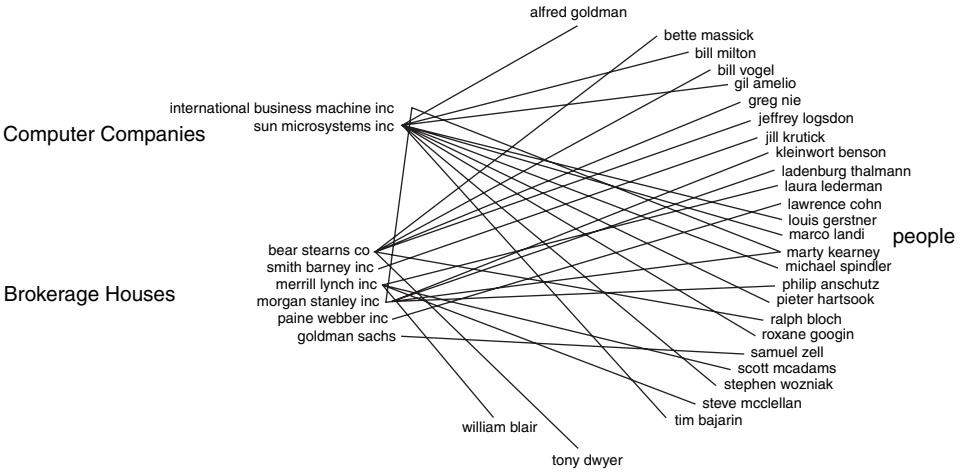

Figure I.1. Example of a visualization tool – mapping concepts (keywords) within the context of categories by means of a "category graph." (From Feldman, Kloesgen, Ben-Yehuda, et al. 1997.)

Furthermore, text mining front ends may offer a user the ability to *cluster* concepts through a suite of *clustering tools* (discussed in Chapter V) in ways that make the most cognitive sense for a particular application or task. Text mining systems can also allow a user to create customized *profiles* for concepts or concept relationships to produce a richer knowledge environment for interactive exploration.

Finally, some text mining systems offer users the ability to manipulate, create, or concatenate *refinement constraints* to assist in producing more manageable and useful result sets for browsing. Like other aspects relating to the creation, shaping, and parameterization of queries, the use of such refinement constraints can be made much more user-friendly by incorporating graphical elements such as pull-downs, radio boxes, or context- or query-sensitive pick lists.

I.1.5 Citations and Notes

Sections I.1–I.1.1

Useful introductions to text mining include Feldman and Dagan (1995), Dixon (1997), Rajman and Besancon (1997b), Feldman (1998), Rajman and Besancon (1998), Hearst (1999a), Tan (1999), and Porter (2002).

Feldman (1998) points out some of the distinctions between classic data mining preprocessing operations, such as table joins, and those of text mining systems. Feldman and Hirsh (1996) discusses text mining's indebtedness to information retrieval. Feldman, Fresko, Hirsh et al. (1998) and Nahm and Mooney (2000), among other works, indicate text mining's dependence on information extraction methodologies – especially in terms of preprocessing operations. Hearst (1999) notes text mining's relatedness to some elements of corpus-based computational linguistics.

PubMed, developed by the National Center for Biotechnology Information (NCBI) at the National Library of Medicine (NLM), a division of the U.S. National Institutes of Health (NIH), is the overall name given to the NLM's database access system, which provides access to resources such as the MEDLINE and

OLDMEDLINE databases. Full information on PubMed can be found at <www.ncbi.nih.gov/entrez/ query.fcgi>.

Hirschman et al. (2002) and Blake and Pratt (2001) both highlight PubMed's attractiveness as a data source for text mining systems. The estimate that 40,000 new biomedical abstracts are being added to PubMed every month comes from Pustejovsky et al. (2002).

Rajman and Besancon (1998) introduced the notion of a prototypical document with respect to text mining document collections.

Freitag (1998b) makes the point that a text document can be viewed as a structured object and discusses many of the semantic and syntactical structures that lend structure to a document. Freitag (1998b) and Zaragoza, Massih-Reza, and Gallinari (1999) both indicate that word sequence may also be a structurally meaningful dimension in documents.

Section I.1.2

Blake and Pratt (2001) presents a discussion of document features in a light useful to understanding text mining considerations. The definition of feature dimensionality that we rely on in Chapter II is shaped by the notion as it is described in Pedersen and Bruce (1997). Statistics for the number of word-level features in a collection of 15,000 documents come from Feldman (1998). Yang and Pedersen (1997) points out that tens of thousands of concept-level features may be relevant for a single application domain.

Blake and Pratt (2001) and Yang and Pedersen (1997) are generally valuable for understanding some distinctions between different types of document features. The phrase *native feature space* was borrowed from Yang and Pedersen (1997). Term-extraction methodologies in text mining are fully treated in Feldman, Fresko, Hirsh et al. (1998). Feldman et al. (2002), Hull (1996), and Brill (1995) are classic works on information extraction useful for understanding lemmatized forms, normalized terms, and so on.

Although some computer scientists make distinctions between keywords and concepts (e.g., Blake and Pratt 2001), this book recognizes the two as relatively interchangeable labels for the same feature type and will generally refer to either under the label *concept*.

It should at least be mentioned that there are some more distinct disadvantages to using manually generated concept-level representations. Manually generated concepts, for example, are fixed and labor-intensive to assign (Blake and Pratt 2001). CDM-based methodologies are discussed in Goldberg (1996).

Feldman and Hirsh (1996a) presents one of the first formal discussions regarding the use of background knowledge in text mining. Other relevant works include Kosmynin and Davidson (1996); Zelikovitz and Hirsh (2000); and Hotho, Staab, and Stumme (2003).

Section I.1.3

Feldman, Kloesgen, Ben-Yehuda, et al. (1997) provides an early treatment of knowledge discovery based on co-occurrence relationships between concepts in documents within a document collection. Lent, Agrawal, and Srikant (1997) is the seminal early work for identifying trends in large amounts of textual data. The high-level

questions important to trend analysis identified in Section I.1.3 are based on similar questions presented in Montes-y-Gomez et al. (2001b). The terms *ephemeral association discovery* and *deviation detection* are used here in the manner introduced in Montes-y-Gomez et al. (2001b).

Section I.1.4

Treatments of browsing germane to text mining and related applications include Chang and Rice (1993); Dagan, Feldman, and Hirsh (1996); Feldman, Kloesgen, Ben-Yehuda, et al. (1997); Smith (2002); and Dzbor, Domingue, and Motta (2004). Browsing is discussed in Chapter IX, while a detailed discussion of more elaborate visualization approaches for supporting user interactivity in text mining applications can be found in Chapter X.

I.2 GENERAL ARCHITECTURE OF TEXT MINING SYSTEMS

At an abstract level, a text mining system takes in input (raw documents) and generates various types of output (e.g., patterns, maps of connections, trends). Figure I.2 illustrates this basic paradigm. A human-centered view of knowledge discovery, however, yields a slightly more complex input–output paradigm for text mining (see Figure I.3). This paradigm is one in which a user is part of what might be seen as a prolonged interactive loop of querying, browsing, and refining, resulting in answer sets that, in turn, guide the user toward new iterative series of querying, browsing, and refining actions.

I.2.1 Functional Architecture

On a functional level, text mining systems follow the general model provided by some classic data mining applications and are thus roughly divisible into four main areas: (a) preprocessing tasks, (b) core mining operations, (c) presentation layer components and browsing functionality, and (d) refinement techniques.

- ***Preprocessing Tasks*** include all those routines, processes, and methods required to prepare data for a text mining system's core knowledge discovery operations. These tasks typically center on data source preprocessing and categorization

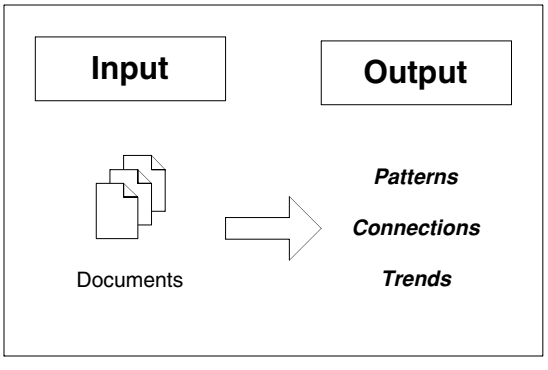

Figure I.2. Simple input–output model for text mining.

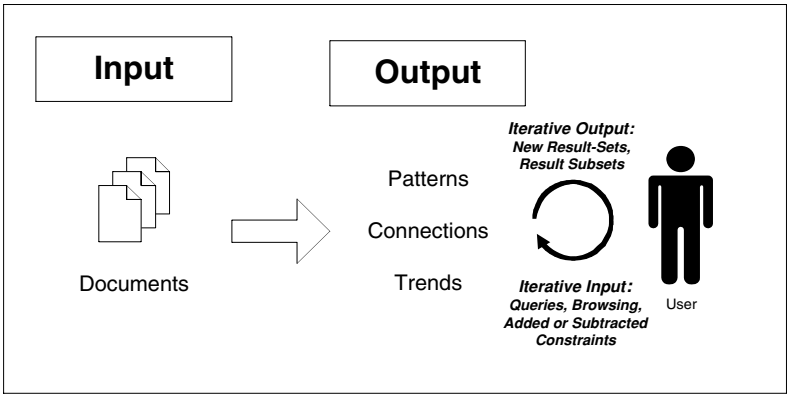

Figure I.3. Iterative loop for user input and output.

activities. Preprocessing tasks generally convert the information from each original data source into a canonical format before applying various types of feature extraction methods against these documents to create a new collection of documents fully represented by concepts. Where possible, preprocessing tasks may also either extract or apply rules for creating document date stamps, or do both. Occasionally, preprocessing tasks may even include specially designed methods used in the initial fetching of appropriate "raw" data from disparate original data sources.

- ***Core Mining Operations*** are the heart of a text mining system and include pattern discovery, trend analysis, and incremental knowledge discovery algorithms. Among the commonly used patterns for knowledge discovery in textual data are distributions (and proportions), frequent and near frequent concept sets, and associations. Core mining operations can also concern themselves with comparisons between – and the identification of levels of "interestingness" in – some of these patterns. Advanced or domain-oriented text mining systems, or both, can also augment the quality of their various operations by leveraging background knowledge sources. These core mining operations in a text mining system have also been referred to, collectively, as *knowledge distillation* processes.
- ***Presentation Layer Components*** include GUI and pattern browsing functionality as well as access to the query language. Visualization tools and user-facing query editors and optimizers also fall under this architectural category. Presentation-layer components may include character-based or graphical tools for creating or modifying concept clusters as well as for creating annotated profiles for specific concepts or patterns.
- ***Refinement Techniques***, at their simplest, include methods that filter redundant information and cluster closely related data but may grow, in a given text mining system, to represent a full, comprehensive suite of suppression, ordering, pruning, generalization, and clustering approaches aimed at discovery optimization. These techniques have also been described as *postprocessing*.

Preprocessing tasks and core mining operations are the two most critical areas for any text mining system and typically describe serial processes within a generalized view of text mining system architecture, as shown in Figure I.4.

Figure I.4. High-level text mining functional architecture.

At a slightly more granular level of detail, one will often find that the processed document collection is, itself, frequently intermediated with respect to core mining operations by some form of flat, compressed or hierarchical representation, or both, of its data to better support various core mining operations such as hierarchical tree browsing. This is illustrated in Figure I.5. The schematic in Figure I.5 also factors in the typical positioning of refinement functionality. Further, it adds somewhat more

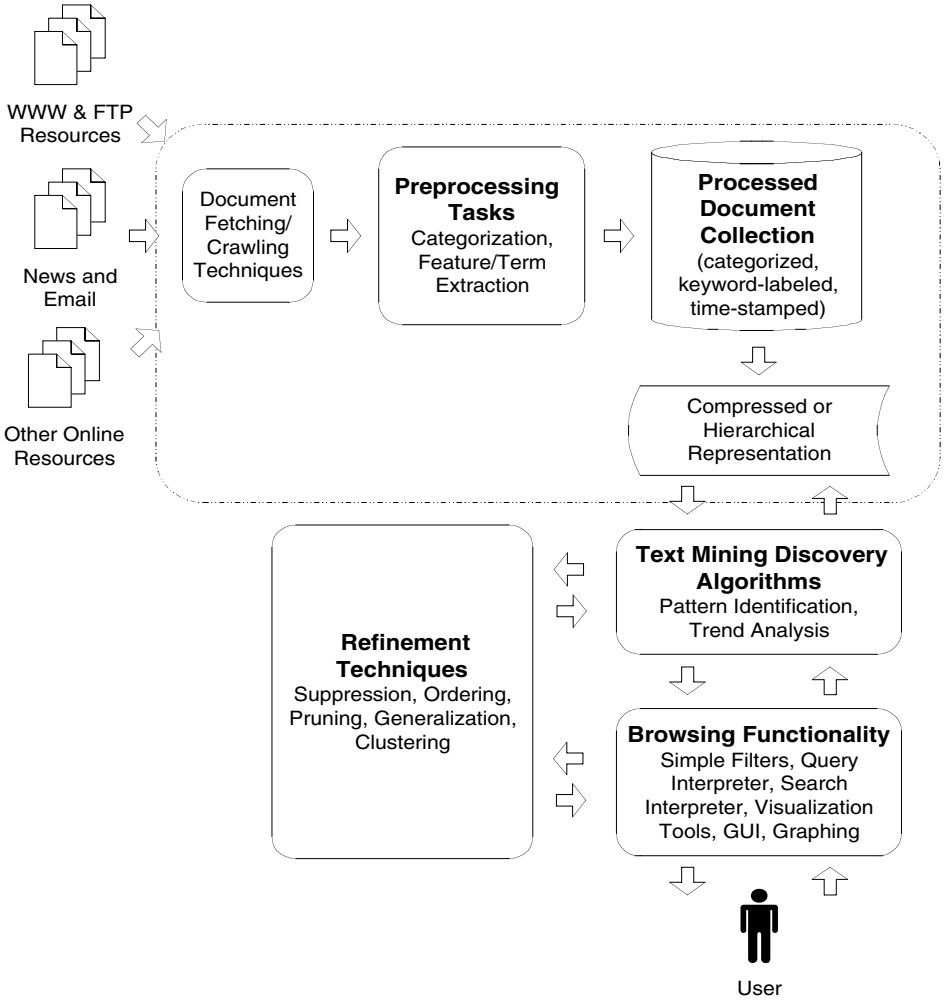

Figure I.5. System architecture for generic text mining system.

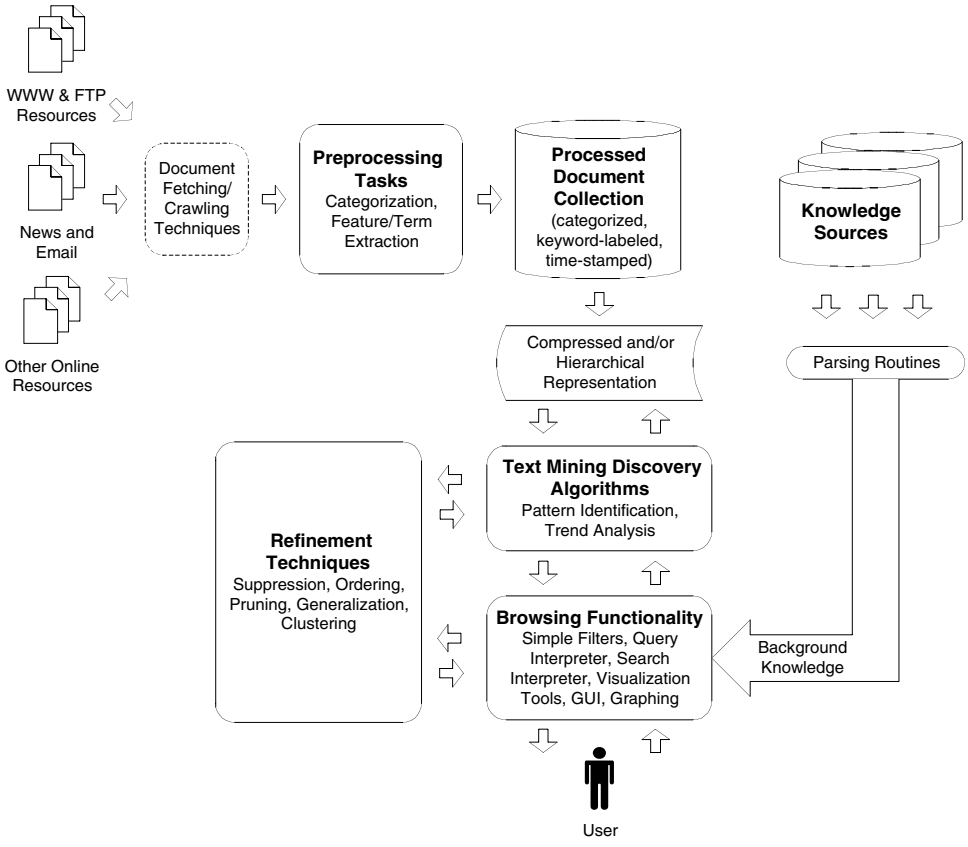

Figure I.6. System architecture for an advanced or domain-oriented text mining system.

detail with respect to relative functioning of core data mining algorithms. Many text mining systems – and certainly those operating on highly domain-specific data sources, such as medicine, financial services, high tech, genomics, proteomics, and chemical compounds – can benefit significantly from access to special background or domain-specific data sources. See Figure I.6.

Background knowledge is often used for providing constraints to, or auxiliary information about, concepts found in the text mining collection's document collection. The background knowledge for a text mining system can be created in various ways. One common way is to run parsing routines against external knowledge sources, such as formal ontologies, after which unary or binary predicates for the concept-labeled documents in the text mining system's document collection are identified. These unary and binary predicates, which describe properties of the entities represented by each concept deriving from the expert or "gold standard" information sources, are in turn put to use by a text mining system's query engine. In addition, such constraints can be used in a text mining system's front end to allow a user to either (a) create initial queries based around these constraints or (b) refine queries over time by adding, substracting, or concatenating constraints.

Commonly, background knowledge is preserved within a text mining system's architecture in a persistent store accessible by various elements of the system. This type of persistent store is sometimes loosely referred to as a system's

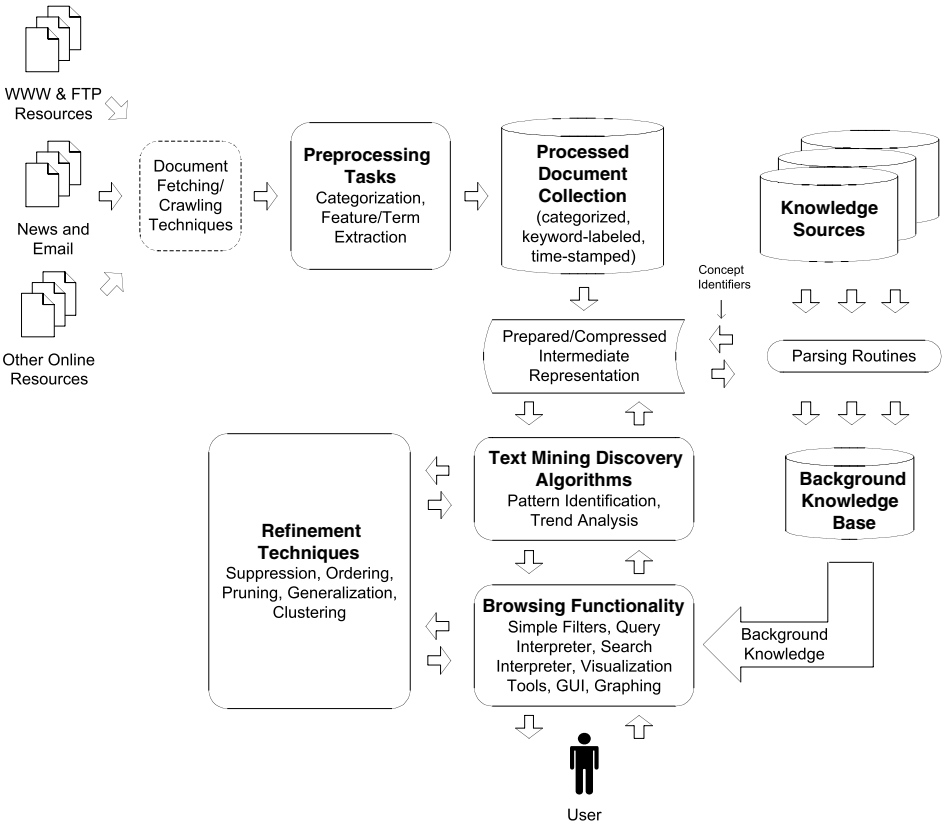

Figure I.7. System architecture for an advanced text mining system with background knowledge base.

knowledge base. The typical position of a knowledge base within the system architecture of a text mining system can be seen in Figure I.7.

These generalized architectures are meant to be more descriptive than prescriptive in that they represent some of the most common frameworks found in the present generation of text mining systems. Good sense, however, should be the guide for prospective system architects of text mining applications, and thus significant variation on the general themes that have been identified is possible. System architects and developers could include more of the filters typically found in a text mining system's browser or even within subroutines contained among the system's store of refinement techniques as "preset" options within search algorithms included in its main discovery algorithms. Likewise, it is conceivable that a particular text mining system's refinement techniques or main discovery algorithms might later find a very fruitful use for background knowledge.

I.2.2 Citations and Notes

Section I.2

The view of human-centered knowledge discovery introduced in Brachman and Anand (1996) and to some degree echoed in Grinstein (1996) influences much

of the discussion of text mining systems in Chapter II and indeed throughout this book.

Section I.2.1

The architectural elements of the systems elaborated on here reflect a composite of operations developed in several widely described real-world text mining applications, most especially the KDT (Feldman and Dagan 1995), FACT (Feldman and Hirsh 1996a; Feldman and Hirsh 1996b; Feldman and Hirsh 1997), and Document Explorer (Feldman, Kloesgen, Ben Yehuda, et al. 1997; Feldman, Kloesgen, and Zilberstein 1997a) systems. Besides these text mining applications, other systems at least referentially contributing in some way to this composite include the TEXTRISE system (Nahm and Mooney 2000), the SYNDICATE system (Hahn and Schnattinger 1997), the Explora System (Kloesgen 1995b), and the LINDI project (Hearst 1999).

In particular, Feldman, Kloesgen, and Zilberstein (1997a) includes a pertinent discussion of the architecture of the Document Explorer System. Tan (1999) also proposes a generalized architecture for text mining systems, using the term "knowledge distillation processes" in roughly the same way as this section refers to "core mining operations." The term "postprocessing" – as a general label for what this book refers to as refinement techniques – comes from Hotho et al. (2002).