II

Core Text Mining Operations

Core mining operations in text mining systems center on the algorithms that underlie the creation of queries for discovering patterns in document collections. This chapter describes most of the more common – and a few useful but less common – forms of these algorithms. Pattern-discovery algorithms are discussed primarily from a high-level definitional perspective. In addition, we examine the incorporation of background knowledge into text mining query operations. Finally, we briefly treat the topic of text mining query languages.

II.1 CORE TEXT MINING OPERATIONS

Core text mining operations consist of various mechanisms for discovering patterns of concept occurrence within a given document collection or subset of a document collection. The three most common types of patterns encountered in text mining are *distributions* (and *proportions*), *frequent and near frequent sets*, and *associations*.

Typically, when they offer the capability of discovering more than one type of pattern, text mining systems afford users the ability to toggle between displays of the different types of patterns for a given concept or set of concepts. This allows the richest possible exploratory access to the underlying document collection data through a browser.

II.1.1 Distributions

This section defines and discusses some of text mining's most commonly used distributions. We illustrate this in the context of a hypothetical text mining system that has a document collection *W* composed of documents containing news wire stories about world affairs that have all been preprocessed with concept labels.

Whether as an initial step, to create a baseline, or to create more meaningful subdivisions of a single document collection for comparison purposes, text mining systems generally need to refer to some subcollection of a complete document collection. This activity is commonly referred to as *concept selection*. Given some collection of documents D, a text mining system will have a requirement to refer to some subcollection of D that is labeled by one or more given concepts.

Definition II.1. Concept Selection: If D is a collection of documents and K is a set of concepts, D/K is the subset of documents in D labeled with all of the concepts in K. When it is clear from the context, given a single concept k, rather than writing $D/\{k\}$ we use the notation D/k.

For example, the collection W contains a subset of the World Affairs collection – namely those documents that are labeled with the concepts *iran, nicaragua*, and *reagan;* $W/bush$ contains the subset of documents that are labeled (at least) with *reagan*; and $W/G8$ contains those documents that are labeled with any terminal node under *G8* (i.e., labeled with any *G8* country). *G8* is treated as a concept here when is being performed concept selection (rather than being viewed as the set of concepts under it, in which case it would have required *all* of its descendants to be present).

Text mining systems often need to identify or examine the proportion of a set of documents labeled with a particular concept. This analytic is commonly referred to as *concept proportion*.

Definition II.2. Concept Proportion: If D is a collection of documents and K is a set of concepts, $f(D, K)$ is the fraction of documents in D labeled with all of the concepts in K, that is, $f(D, K) = \frac{|D/k|}{|D|}$. Given one concept k, rather than writing $f(D, \{k\})$, we use the notation $f(D, k)$. When D is clear from context, we drop it and write $f(k)$.

Thus, for example, $f(W, \{iran, nicaragua, reagan\}$ is the fraction of documents in the World Affairs collection labeled with *iran, nicaragua*, and reagan; $f(reagan)$ is the proportion of the collection labeled with the concept *reagan*; and $f(G8)$ is the proportion labeled with any $(G8)$ country.

By employing definitions of selection and proportion, text mining systems can already begin identifying some useful quantities for analyzing a set of documents. For example, a text mining system might want to identify the proportion of those documents labeled with K_2 that are also labeled by K_1, which could be designated by expression $f(D/K_2, K_1)$.

This type of proportion occurs regularly enough that it has received an explicit name and notation: *conditional concept proportion*.

Definition II.3. Conditional Concept Proportion: If D is a collection of documents and K_1 and K_2 are sets of concepts, $f(D, K_1 \mid K_2)$ is the proportion of all those documents in D labeled with K_2 that are also labeled with K_1, that is, $f(D, K_1 \mid K_2) = f(D/K_2, K_1)$. When D is clear from context, we will write this as $f(K_1 \mid K_2)$.

Applying this definition, we find that $f(reagan \mid iran)$ would represent the proportion of all documents labeled by the concept *iran* that are also labeled by the concept *reagan*.

Commonly, a text mining system needs to analyze the distribution of concepts that are descendents of a particular node in a concept hierarchy. For example, a text mining system might need to allow the analysis of the distribution of concepts denoting *finance topics* – that is, descendents of the *finance topics* node in an example concept hierarchy. To accomplish this, a text mining system could use the expression

$P_K(x)$ to refer to such distributions – it will assign to any concept x in K a value between 0 and 1 – where the values are not required to add up to 1.

This type of proportion can be referred to as a *concept distribution*. In the following sections we present several specific examples of such $P_K(x)$ distributions.

One particularly important concept distribution for knowledge discovery operations is the *concept proportion distribution*, which gives the proportion of documents in some collection that are labeled with each of a number of selected concepts:

Definition II.4. Concept Proportion Distribution: If D is a collection of documents and K is a set of concepts, $F_K(D, x)$ is the proportion of documents in D labeled with x for any x in K. When D is clear from context, we will write this as $F_K(x)$.

Note the distinction between $P_K(x)$ and $F_K(x)$. $P_K(x)$ refers generically to any function that is a concept distribution. $F_K(x)$ is a specific concept distribution defined by a particular concept-labeled set of documents.

Thus, for example $F_{\text{topics}}(R, x)$ would represent the proportions of documents in W labeled with keywords under the *topics* node in the concept hierarchy. In this expression, *topics* is used as shorthand for referring to a set of concepts – namely, all those that occur under the *topics* node – instead of explicitly enumerating them all.

Also, note that $F_{\{k\}}(D, k) = f(D, k)$ – that is, F_K subsumes the earlier defined f when it is applied to a single concept. Unlike f, however, F_K is restricted to refer only to the proportion of occurrences of *individual* concepts (those occurring in the set K).[1] Thus f and F are not comparable.

Mathematically, F is not a true frequency distribution, for each document may be labeled by multiple items in the set K. Thus, for example, a given document may be labeled by two (or more) *G8* countries because occurrences of concepts are not disjoint events. Therefore, the sum of values in F_{G8} may be greater than one.

In the worst case, if all concepts in K label all documents, the sum of the values in a distribution F can be as large as $|K|$. Furthermore, because some documents may contain none of the concepts in a given K, the sum of frequencies in F might also be smaller than one – in the worst case, zero. Nonetheless, the term "distribution" is used for F, for many of the connotations this term suggests still hold true.

Just as was the case for concept proportions, text mining systems can also leverage conditional keyword-proportion distributions, which are probably one of the most used concept distributions in text mining systems.

Definition II.5. Conditional Concept Proportion Distribution: If D is a collection of documents and K and K' are sets of concepts, $F_K(D, x \mid K')$ is the proportion of those documents in D labeled with all the concepts in K' that are also labeled with concept x (with x in K), that is, $F_K(D, x \mid K') = F_K(D/K \mid K', x)$. We often write this as $F_K(x \mid K')$ when D is clear from context.

Thus, for example, $F_{\text{topics}}(x \mid \textit{Argentina})$ would assign any concept x under *topics* in the hierarchy with the proportion of documents labeled by x within the set of all documents labeled by the concept *Argentina*, and $F_{\text{topics}}(x \mid \{\textit{UK}, \textit{USA})$ is the similar distribution for those documents labeled with both the *UK* and *USA* concepts.

[1] It is also quite simple to define a similar notion for *sets* of concepts, for example, by computing the proportions for each subset of a set K (Feldman, Dagan, and Hirsh, 1998).

One of the baseline distributions text mining systems use to compare distributions is the average distribution over a set of sibling nodes in the hierarchy. For example, when looking at the proportions of *loan* within South American countries such as $f(W, loan \mid Argentina)$, $f(W, loan \mid Brazil)$, and $f(W, loan \mid Columbia)$), an end user may be interested in the average of all proportions of this form for all the South American countries – that is, the average of all proportions of the form $f(W, loan \mid k)$, where k ranges over all South American countries.

Definition II.6. Average Concept Proportion: Given a collection of documents D, a concept k, and an internal node in the hierarchy n, an *average concept proportion*, denoted by $a(D, k \mid n)$, is the average value of $f(D, k \mid k')$, where k' ranges over all immediate children of n – that is, $a(D, k \mid n) = \mathrm{Avg}_{\{k' \text{ is a child of } n\}}\{f(D, k \mid k')\}$. When D is clear from context, this will be written $a(k|n)$.

For example, $a(loan \mid South\_America)$ is the average concept proportion of $f(loan \mid k')$ as k' varies over each child of the node *South_America* in the concept hierarchy; that is, it is the average conditional keyword proportion for *loan* within South American countries.

This quantity does *not* average the values weighted by the number of documents labeled by each child of n. Instead, it equally represents each descendant of n and should be viewed as a summary of what a typical concept proportion is for a child of n.

An end user may be interested in the distribution of averages for each economic topic within South American countries. This is just another keyword distribution referred to as an *average concept distribution*.

Definition II.7. Average Concept Distribution: Given a collection of documents D and two internal nodes in the hierarchy n and n', an *average concept distribution*, denoted by $A_n(Dx \mid n')$, is the distribution that, for any x that is a child of n, averages x's proportions over all children of n' – that is, $A_n(D, x \mid n') = \mathrm{Avg}_{\{k' \text{ is a child of } n'\}}\{F_n(D, x \mid k')\}$. When clear from context, this will be written $A_n(x \mid n')$.

For example $A_{\text{topics}}(x|South\_America)$, which can be read as "the average distribution of topics within South American countries," gives the average proportion within all South American countries for any topic x.

A very basic operation for text mining systems using concept-distributions is the display of conditional concept-proportion distributions. For example, a user may be interested in seeing the proportion of documents labeled with each child of *topics* for all those documents labeled by the concept *Argentina*, that is, the proportion of *Argentina* documents that are labeled with each topic keyword.

This distribution would be designated by $F_{\text{topics}}(W, x \mid Argentina)$, and a correlating graph could be generated, for instance, as a bar chart, which might display the fact that 12 articles among all articles of *Argentina* are annotated with *sorghum*, 20 with *corn*, 32 with *grain*, and so on, providing a summary of the areas of economical activity of Argentina as reflected in the text collection. Conditional concept-proportion distributions can also be conditioned on *sets* of concepts.

In some sense, this type of operation can be viewed as a more refined form of traditional concept-based retrieval. For example, rather than simply requesting all documents labeled by *Argentina* or by both *UK* and *USA*, the user can see the documents at a higher level by requesting documents labeled by *Argentina* for example, and first seeing what proportions are labeled by concepts from some secondary set of concepts of interest with the user being able to access the documents through this more fine-grained grouping of *Argentina*-labeled documents.

Comparing with Average Distributions

Consider a conditional proportion of the form $F_k(D, x \mid k) f$, the distribution over K of all documents labeled with some concept k (not necessarily in K). It is natural to expect that this distribution would be similar to other distributions of this form over conditioning events k' that are siblings of k. When they differ substantially it is a sign that the documents labeled with the conditioning concept k may be of interest.

To facilitate this kind of comparison of concept-labeled documents with the average of those labeled with the concept and its siblings, a user can specify two internal nodes of the hierarchy and compare individual distributions of concepts under one of the nodes conditioned on the concept set under the other node – that is, compute $D(F_n(x \mid k) || A_n(x \mid n'))$ for each k that is a child of n'.

In addition to their value in finding possible interesting concept labelings, comparisons of this type also provide a hierarchical browsing mechanism for concept co-occurrence distributions. For example, an analyst interested in studying the topic distribution in articles dealing with *G8* countries may first browse the average class distribution for *G8*. This might reveal the major topics that are generally common for *G8* countries. Then, an additional search could be used to reveal the major characteristics specific for each country.

Comparing Specific Distributions

The preceding mechanism for comparing distributions with an average distribution is also useful for comparing conditional distributions of two specific nodes in the hierarchy. For example, one could measure the distance from the average topic distribution of *Arab_League* countries to the average topic distribution of *G8* countries. An answer set could be returned from a query into a table with countries sorted in decreasing order of their contribution to the distance (second column) – namely $d(A_{\text{topics}}(K \mid Arab\_League) \mid\mid A_{\text{topics}}(k \mid G8))$.

Additional columns could show, respectively, the percentage of the topic in the average topic distribution of the *Arab_League* countries ($A_{\text{topics}}(x \mid G8)$) and in the average topic distribution of the *G8* countries ($A_{\text{topics}}(x \mid G8)$). One could also show the total number of articles in which the topic appears with any *Arab_League* country and any *G8* country. This would reveal the topics with which *Arab_League* countries are associated much more than *G8* countries such as grain, wheat, and crude oil. Finally, one could show the comparison in the opposite direction, revealing the topics with which *G8* countries are highly associated relative to the *Arab_League*.

II.1.2 Frequent and Near Frequent Sets

Frequent Concept Sets

In addition to proportions and distributions, another basic type of pattern that can be derived from a document collection is a *frequent concept set*. This is defined as a set of concepts represented in the document collection with co-occurrences at or above a minimal support level (given as a threshold parameter *s*; i.e., all the concepts of the frequent concept set appear together in at least *s* documents). Although originally defined as an intermediate step in finding *association rules* (see Section II.1.3), frequent concept sets contain a great deal of information of use in text mining.

The search for frequent sets has been well treated in data mining literature, stemming from research centered on investigating *market basket*–type associations first published by Agrawal et al. in 1993. Essentially, a document can be viewed as a market basket of named entities. Discovery methods for frequent concept sets in text mining build on the *Apriori* algorithm of Agrawal et al. (1993) used in data mining for market basket association problems. With respect to frequent sets in natural language application, *support* is the number (or percent) of documents containing the given rule – that is, the co-occurrence frequency. *Confidence* is the percentage of the time that the rule is true.

```
$L_1$ = {large 1 – itemsets}
for ($k = 2$; $L_{k-1} \neq \emptyset$; $k$++) do begin
        $C_k$ = apriori-gen ($L_{k-1}$) // new candidates
        forall transactions $t \in D$ do begin
            $C_1$ = subset ($C_k$, $t$) // candidates contained in $t$
            forall candidates $c \in C_t$ do
                c.count ++;
            end
            $L_k = \{c \in C_k \mid c.count \geq minsupport\}$
end
Answer = $\bigcup_k L_k$;
```

Algorithm II.1: The Apriori Algorithm (Agrawal and Srikant 1994)[2]

A frequent set in text mining can be seen directly as a query given by the conjunction of concepts of the frequent set. Frequent sets can be partially ordered by their generality and hold the simple but useful pruning property that each subset of a frequent set is a frequent set. The discovery of frequent sets can be useful both as a type of search for patterns in its own right and as a preparatory step in the discovery of *associations*.

Discovering Frequent Concept Sets

As mentioned in the previous section, frequent sets are generated in relation to some support level. Because support (i.e., the frequency of co-occurrence) has been by convention often expressed as the variable σ, frequent sets are sometimes also referred to as *σ-covers*, or *σ-cover sets*. A simple algorithm for generating frequent sets relies on incremental building of the group of frequent sets from singleton *σ-covers*, to which additional elements that continue to satisfy the support constraint

[2] In data mining, the expression *item* is commonly used in a way that is roughly analogous to the expression *feature* in text mining. Therefore, the expression *item set* can be seen here, at least, as analogous to the expression *concept set*.

are progressively added. Algorithm II.2 is a typical algorithm for discovering frequent concept sets.

```
$L_1 = \{\{A\} \mid A \in R$ and $[A] \geq \sigma\}$
$i = 1$
While $L_i \neq \emptyset$ do
        $L_{i+1} = \{S1 \cup S2 \mid S1, S2 \in L_i, \ \mid S1 \cup S2 \mid \ = i + 1,$
            all subsets of $S1 \cup S2$ are in $L_i\}$
        $i = i + 1$
end do
return $(\{X \mid X \in \bigcup_i L_i$ and $|[X]| \geq \sigma\})$
```

Algorithm II.2: Algorithm for Frequent Set Generation

Near Frequent Concept Sets

Near frequent concept sets establish an undirected relation between two frequent sets of concepts. This relation can be quantified by measuring the degree of overlapping, for example, on the basis of the number of documents that include all the concepts of the two concept sets. This measure can be regarded as a distance function between the concept sets. Several distance functions can be introduced (e.g., based on the cosine of document vectors, Tanimoto distance, etc.).

Directed relations between concept sets can also be identified. These are considered types of *associations* (see Section II.1.3).

II.1.3 Associations

A formal description of association rules was first presented in the same research on “market basket” problems that led to the identification of *frequent sets* in data mining. Subsequently, associations have been widely discussed in literature on knowledge discovery targeted at both structured and unstructured data.

In text mining, *associations* specifically refer to the directed relations between concepts or sets of concepts. An *association rule* is generally an expression of the form $A \Rightarrow B$, where A and B are sets of features. An association rule $A \Rightarrow B$ indicates that transactions that involve A tend also to involve B.

For example, from the original market-basket problem, an association rule might be *25 percent of the transactions that contain pretzels also contain soda; 8 percent of all transactions contain both items*. In this example, 25 percent refers to the *confidence* level of the association rule, and 8 percent refers to the rule’s level of *support*.

With respect to concept sets, association rule $A \Rightarrow B$, relating two frequent concept sets A and B, can be quantified by these two basic measures of support and confidence. *Confidence* is the percentage of documents that include all the concepts in B within the subset of those documents that include all the concepts in A. *Support* is the percentage (or number) of documents that include all the concepts in A and B.

More precisely, we can describe association rules as follows:

- Let $r = \{t_1, \ldots, t_n\}$ be a collection of documents, each labeled with some subset of concepts from the m-concept set $R = \{I_1, I_2, \ldots, I_m\}$.

- Given a concept A and document t, we write $t(A) = 1$ if A is one of the concepts labeling t, and $t(A) = 0$ otherwise.
- If W is a subset of the concepts in R, $t(W) = 1$ represents the case that $t(A) = 1$ for every concept $A \in W$.
- Given a set X of concepts from R, define $(X) = \{i \mid t_i(X) = 1\}$; (X) is the set of all documents t_i that are labeled (at least) with all the concepts in X.
- Given some number σ (the support threshold), X is called a σ-covering if $|(\mathrm{X})| \geq \sigma$.

$W \Rightarrow B$ is an association rule over r if $W \subseteq R$ and $B \subseteq R \backslash W$. We refer to W as the *left-hand side* (LHS) of the association and B as the *right-hand side* (RHS).

Finally, we say that r satisfies $W \Rightarrow B$ with respect to $0 < \gamma \leq 1$ (the confidence threshold) and σ (the support threshold) if $W \cup B$ is a σ-covering (i.e., $|(W \cup B)| \geq \sigma$ and $|(W \cup B)|/|(W)| \geq \gamma$). Intuitively, this means that, of all documents labeled with the concepts in W, at least a proportion γ of them are also labeled with the concepts in B; further, this rule is based on at least σ documents labeled with all the concepts in both W and B.

For example, a document collection has documents labeled with concepts in the following tuples: {x, y, z, w}, {x, w}, {x, y, p}, {x, y, t}. If $\gamma = 0.8$ and $\sigma = 0.5$, and {x}, {y}, {w}, {x, w}, and {x, y} are coverings, then {y} ⇒ {x} and {w} ⇒ {x} are the only associations.

Discovering Association Rules

The discovery of association rules is the problem of finding all the association rules with a confidence and support greater than the user-identified values *minconf* (i.e., γ, or the minimum confidence level) and *minsup* (i.e., σ, or the minimum support level) thresholds.

The basic approach to discovering associations is a generally straightforward two-step process as follows:

- Find all frequent concept sets X (i.e., all combinations of concepts with a support greater than *minsup*);
- Test whether $X \backslash B \Rightarrow B$ holds with the required confidence.

The first step – namely the generation of frequent concept sets (see Algorithm II.2) – has usually been found to be by far the most computationally expensive operation. A typical simple algorithm for the second step – generating associations (after the generation of maximal frequent concept sets has been completed) – can be found below in Algorithm II.3.

```
foreach X maximal frequent set do
        generate all the rules $X \setminus \{b\} \Rightarrow \{b\}$, where $b \in X$, such that
              $\frac{|[X\setminus\{b\}]|}{|[X]|} \geq \sigma$
endfch
```

Algorithm II.3: Simple Algorithm for Generating Associations (Rajman and Besancon 1998)

Thus, essentially, if {w, x} and {w, x, y, z} are frequent concept sets, then the association rule {w, x} ⇒ {y, z} can be computed by the following ratio:

$$c = \frac{support\ (\{\text{w, x, y, z}\})}{support\ (\{\text{w, x}\})}.$$

Again, however, in this case the association rule will only hold if c $\geq \sigma$.

Given these steps, if there are m concepts in a document collection, then, in a single pass, all possible 2^m subsets for that document collection can be checked. Of course, in extremely large, concept-rich document collections, this can still be a nontrivial computational task. Moreover, because of the implications of generating an overabundance of associations, additional procedures – such as structural or statistical pruning, redundancy elimination, and so on – are sometimes used to supplement the main association rule extraction procedure in order to limit the number of generated associations.

Maximal Associations

Association rules are very useful in helping to generally describe associations relevant between concepts. *Maximal associations* represent a more specialized type of relationship between concepts in which associations are identified in terms of their relevance to one concept and their lack of relevance to another. These associations help create solutions in the particular problem space that exists within text document collections, where closely related items frequently appear together. Conventional association rules fail to provide a good means for allowing the specific discovery of associations pertaining to concepts that most often do not appear alone (but rather together with closely related concepts) because associations relevant only to these concepts tend to have low confidence. Maximal association rules provide a mechanism for discovering these types of specialized relations.

For example, in a document collection, the concept "Saddam Hussein" may most often appear in association with "Iraq" and "Microsoft" most often with "Windows." Because of the existence of these most common relationships, associations especially relevant to the first concept in the association, but not the other, will tend to have low confidence. For instance, an association between "Iraq" and the "Arab League" would have low confidence because of the many instances in which "Iraq" appears with "Saddam Hussein" (and not "Arab League"). Likewise, an association between "Microsoft" and "Redmond" would potentially be left unidentified because of the many more instances in which "Microsoft" appears with "Windows." Maximal associations identify associations relevant to one concept but not the other – that is, associations relating to "Iraq" or "Microsoft" alone.

Maximal Association Rules: Defining M-Support and M-Confidence

Fundamentally, a maximal association rule $X \overset{max}{\Rightarrow} Y$ states that, whenever X is the only concept of its type in a transaction (i.e., when X appears *alone*), then Y also appears with some confidence. To understand the notion of a maximal association rule it is important define the meaning of *alone* in this context. We can do so with respect to categories of G:

Definition II.8. Alone with Respect to Maximal Associations: For a transaction t, a category g, and a concept-set $X \subseteq g_i$, one would say that X is alone in t if $t \cap g_i = X$.

That is, X is alone in t if X is the largest subset of g_i that is in t. In such a case, one would say that X is maximal in t and that t *M-supports* X. For a document collection D, the *M-support* of X in D, denoted as $s_{\underset{D}{max}}(X)$, is the number of transactions $t \in D$ that *M-support X*.

A maximal association rule, or *M-association*, is a rule of the form $X \overset{max}{\Rightarrow} Y$, where X and Y are subsets of distinct categories that could be identified as $g\ (X)$ and $g\ (Y)$, respectively. The *M-support* for the maximal association $X \overset{max}{\Rightarrow} Y$, which can be denoted as $s_{\underset{D}{max}}(X \overset{max}{\Rightarrow} Y)$, can be defined as

$$s_{\underset{D}{max}}(X \overset{max}{\Rightarrow} Y) = |\{t : t \text{ M-supports } X \text{ and } t \text{ supports } Y\}|.$$

That is, $(X \overset{max}{\Rightarrow} Y)$ is equal to the number of transactions in D that M-support X and also support Y in the conventional sense, which suggests that, whenever a transaction M-supports X, then Y also appears in the transaction with some probability.

In measuring this probability, we are generally interested only in those transactions in which some element of $g(Y)$ (i.e., the category of Y) appears in the transaction. Thus, we define confidence in the following manner. If $D(X, g(Y))$ is the subset of the document collection D consisting of all the transactions that M-support X and contain at least one element of $g(Y)$, then the M-confidence of the rule $X \overset{max}{\Rightarrow} Y$, denoted by $c_{\underset{D}{max}}(X \overset{max}{\Rightarrow} Y)$, is

$$c_{\underset{D}{max}}(X \overset{max}{\Rightarrow} Y) = \frac{s_{\underset{D}{max}}(X \overset{max}{\Rightarrow} Y)}{|D(X, g(Y))|}.$$

A text mining system can search for associations in which the M-support is higher than some user-specified *minimum M-support*, which has been denoted by the designation s, and the M-confidence is higher than some user-specified minimum M-confidence, which has been denoted by c. A set X that has M-support of at least s is said to be *M-frequent.*

M-Factor

Any maximal association rule is also a conventional association with perhaps different levels of support and confidence. The *M-factor* of the rule $X \overset{max}{\Rightarrow} Y$ is the ratio between the M-confidence of the maximal association $X \overset{max}{\Rightarrow} Y$ and the confidence of the *corresponding* conventional association $X \Rightarrow Y$. Specifically, if D is a subset of the transaction that contains at least one concept of $g(Y)$, then, the M-factor of the association $X \overset{max}{\Rightarrow} Y$ is

$$\text{M-factor } (X \overset{max}{\Rightarrow} Y) = c_{\underset{D}{max}}(X \overset{max}{\Rightarrow} Y) = \frac{c_{\underset{D}{max}}(X \overset{max}{\Rightarrow} Y)}{c_{D'}(X \Rightarrow Y)}.$$

Here, the denominator is the confidence for the rule $X \Rightarrow Y$ with respect to D'. This is because, given that the M-confidence is defined with respect to D', the comparison to conventional associations must also be with respect to the set.

From a practical perspective, one generally seeks M-associations with a higher M-factor. Such M-associations tend to represent more interesting rules.

II.1.4 Isolating Interesting Patterns

The notion of *interestingness* with respect to knowledge discovery in textual data has been viewed from various subjective and contextual perspectives. The most common method of defining interestingness in relation to patterns of distributions, frequent sets, and associations has been to enable a user to input expectations into a system and then to find some way of measuring or ranking patterns with respect to how far they differ from the user's expectations.

Text mining systems can quantify the potential degree of "interest" in some piece of information by comparing it to a given "expected" model. This model then serves as a baseline for the investigated distribution.

For example, a user may want to compare the data regarding Microsoft with an averaged model constructed for a group of computer software vendors. Alternatively, a user may want to compare the data relating to Microsoft in the last year with a model constructed from the data regarding Microsoft in previous years.

Interestingness with Respect to Distributions and Proportions

Because text mining systems rely on concept proportions and distributions to describe the data, one therefore requires measures for quantifying the distance between an investigated distribution and another distribution that serves as a baseline model (Feldman, Dagan, and Hirsh 1998). So long as the distributions are discrete, one can simply use sum-of-squares to measure the distance between two models:

$$D(p' \,||\, p) = \sum_{x} (p'(x) - p(x))^2,$$

where the target distribution is designated by p and the approximating distribution by p' and the x in the summation is taken over all objects in the domain. This measure is always nonnegative and is 0 if and only if $p' = p$.

Given this measure, one can use it as a heuristic device. With respect to distribution-based patterns, this could be used as a heuristic for judging concept-distribution similarities. This measure is referred to as *concept distribution distance*.

Definition II.9. Concept Distribution Distance: Given two concept distributions $P'_K(x)$ and $P_K(x)$, the distance $D(P'_K \,||\, P_K)$ between them is defined by $D(P'_K(x) \,||\, P_K(x)) = \sum_{x \in K} (P'_K(x) - P_K(x))^2$.

Text mining systems are also sometimes interested in the value of the difference between two distributions at a particular point. This measure is called *concept proportion distance.*

Definition II.10. Concept Proportion Distance: Given two concept distributions $P'_K(x)$ and $P_K(x)$, and a concept k in K, the distance $d(P'_K(k) \,||\, P_K(k))$ between them is defined by $D(P'_K(k) \,||\, P_K(k)) = P'_K(k) - P_K(k)$.

Thus, another way to state $D(P'_K \,||\, P_K)$ would be

$$\sum_{x \in K} [d(P_K(x) \,||\, P_K(x))]^2.$$

As an example, the distance between the distribution of *topics* within *Argentina* and the distribution of *topics* within *Brazil* would be written as $D(F_{\text{topics}}(x \mid Argentina) \,||\, F_{\text{topics}}(x \mid Brazil))$, and the distance between the distribution of *topics* within *Argentina* and the average distribution of *topics* within *South_America* would be written as $D(F_{\text{topics}}(x \mid Argentina) \,||\, A_{\text{topics}}(x \mid South\_America))$.

II.1.5 Analyzing Document Collections over Time

Early text mining systems tended to view a document collection as a single, monolithic entity – a unitary corpus consisting of one coherent and largely *static* set of textual documents. Many text mining applications, however, benefit from viewing the document collection not as a monolithic corpus but in terms of subsets or divisions defined by the date and time stamps of documents in the collection. This type of view can be used to allow a user to analyze similarities and differences between concept relationships across the various subdivisions of the corpus in a way that better accounts for the change of concept relationships over time.

Trend analysis, in text mining, is the term generally used to describe the analysis of concept distribution behavior across multiple document subsets over time. Other time-based analytics include the discovery of *ephemeral associations*, which focuses on the influence or interaction of the most frequent or "peak" concepts in a period on other concepts, and *deviation*, which concentrates on irregularities such as documents that have concepts differing from more typical documents in a document collection (or subcollection) over time. In addition, text mining systems can enable users to explore the evolution of concept relationships through *temporal context graphs* and context-oriented *trend graphs*.

Although trend analysis and related time-based analytics attempt to better account for the evolving nature of concept relationships in a document collection, text mining systems have also developed practical approaches to the real-world challenges inherent in supporting truly dynamic document collections that add, modify, or delete documents over time. Such algorithms have been termed *incremental algorithms* because they tend to be aimed at more efficient incremental update of the search information that has already been mined from a document collection to account for new data introduced by documents added to this collection over time.

Both trend analysis and incremental algorithms add a certain dynamism to text mining systems, allowing these systems to interact with more dynamic document collections. This can be critical for developing useful text mining applications targeted at handling time series–type financial reports, topical news feeds, text-based market data, time-sensitive voter or consumer sentiment commentary, and so on.

Trend Analysis

The origin of the problem of discovering trends in textual data can be traced to research on methods for detecting and presenting trends in word phrases. These methods center on a two-phase process in which, in the first phase, phrases are created as frequent *sequences* of words using the sequential patterns mining algorithm first mooted for mining structured databases and, in the second phase, a user can query the system to obtain all phrases whose trend matches a specified pattern (i.e., "recent upward trend").

More recent methods for performing trend analysis in text mining have been predicated on the notion that the various types of concept distributions are functions of document collections. It is therefore possible to compare two distributions that are otherwise identical except that they are for different subcollections. One notable example of this is having two collections from the same source (such as from a news feed) but from different points in time.

For instance, one can compare the distribution of *topics* within *Argentina*-labeled documents, as formed by documents published in the first quarter of 1987, with the same distribution formed by documents from the second quarter of 1987. This comparison will highlight those topics whose proportion changed between the two time points, directing the attention of the user to specific trends or events in these topics with respect to Argentina. If R_1 is used to designate a portion of a Reuters newswire data collection from the first quarter of 1987, and R_2 designates the portion from the second quarter of 1987, this would correspond to comparing $F_{\text{topics}}(R_1, x \mid \textit{Argentina})$ and $F_{\text{topics}}(R_2, x \mid \textit{Argentina})$.

This knowledge discovery operation can be supplemented by listing trends that were identified across different quarters in the time period represented by the Reuters collection by computing $R(F_{\text{countries}}(R_1, x \mid \textit{countries}) \,||\, F_{\text{countries}}(R_2, x \mid \textit{countries}))$, where R_1 and R_2 correspond to different subcollections from different quarters.[3] A text mining system could also calculate the percentage and absolute frequency for $F_{\text{countries}}(x \mid \textit{countries})$ for each such pair of collections.

Ephemeral Associations

An *ephemeral association* has been defined by Montes-y-Gomez et al. (2001b) as a direct or inverse relation between the probability distributions of given topics (concepts) over a fixed time span. This type of association differs notionally from the more typical association form $A \Rightarrow B$ because it not only indicates the co-occurrence of two topics or sets of topics but primarily indicates how these topics or sets of topics are related within the fixed time span.

Examples of ephemeral associations can be found in news feeds in which one very frequently occurring or "peak" topic during a period seems to influence either the emergence or disappearance of other topics. For instance, news stories (documents) about a close election that involve allegations of election machine fraud may correlate with the emergence of stories about election machine technology or vote fraud stories from the past. This type of ephemeral association is referred to as a *direct ephemeral association*.

On the other hand, news stories relating to the victory of a particular tennis player in a major tournament may correlate with a noticeable and timely decrease in stories mentioning other tennis players who were formerly widely publicized. Such

[3] It would also be quite fair to ask for a distribution $F_K(x \mid K)$, which analyzes the co-occurrences of different keywords under the same node of the hierarchy. Thus, for example, $F_{\text{countries}}(x \mid \textit{countries})$ would analyze the co-occurrences of country labels on the various documents.

momentary negative influence between one topic and another is referred to as an *inverse ephemeral association*.

One statistical method suggested by Montes-y-Gomez et al. (2001b) to detect ephemeral associations has been the correlation measure r. This method has been expressed as

$$r = \frac{S_{01}}{\sqrt{S_{00} S_{01}}},$$

$$S_{kl} = \sum_{i=1}^{n} (p_k^i, p_l^i) - \frac{1}{n}\left(\sum_{i=1}^{n} p_k^i\right)\left(\sum_{i=1}^{n} p_l^i\right),$$

$$k, l = 0, 1.$$

Within this method, p_0^i is the probability of the peak topic, and p_1^i is the probability of the other topic in the period i. The correlation coefficient r attempts to measure how well two variables – here, topics or concepts – are related to one another. It describes values between −1 and 1; the value −1 means there is a perfect inverse relationship between two topics, whereas the value 1 denotes a perfect direct relationship between two topics. The value 0 indicates the absence of a relation.

Deviation Detection

Users of text mining systems are sometimes interested in deviations – that is, the identification of anomalous instances that do not fit a defined "standard case" in large amounts of data. The normative case is a representation of the average element in a data collection. For instance, in news feed documents and the topics (concepts) that they contain, a particular topic can be considered a deviation if its probability distribution greatly diverges from distributions of other topics in the same sample set.

Research into deviation detection for text mining is still in its early, formative stages, and we will not discuss it in detail here. However, work has been done by Montes-y-Gomez, Gelbukh, and Lopez-Lopez (Montes-y-Gomez et al. 2001b) and others to examine the difficult task of detecting deviations among documents in large collections of news stories, which might be seen as an application of knowledge discovery for distribution-type patterns. In such applications, time can also be used as an element in defining the norm.

In addition, one can compare norms for various time-based subsets of a document collection to find individual news documents whose topics substantially deviate from the topics mentioned by other news sources. Sometimes such deviating individual documents are referred to as *deviation sources*.

From Context Relationships to Trend Graphs

Another approach to to exploring the evolution of concept relationships is to examine *temporal context relationships*. Temporal context relationships are most typically represented by two analytical tools: the *temporal context graph* and the *trend graph*.

Before describing these time-based, context-oriented analytical tools, we expend a little effort explicating the more general notions of *context* in document collections.

Indeed, both temporal context graphs and trend graphs build on the notion of the *context relationship* and its typical visual representation in the form of the *context graph*.

Context Phrases and Context Relationships

Generally, a *context relationship* in a document collection is the relationship within a set of concepts found in the document collection in relation to a separately specified concept (sometimes referred to as the *context* or the *context concept*). A context relationship search might entail identifying all relationships within a set of company names within the context of the concept "bankruptcy." A *context phrase* is the name given to a subset of documents in a document collection that is either labeled with all, or at least one, of the concepts in a specified set of concepts.

Formal definitions for both *context phrases* and the *context relationship* are as follows:

Definition II.11. Context Phrase: If D is a collection of documents and C is a set of concepts, $D/A(C)$ is the subset of documents in D labeled with all the concepts in C, *and $D/O(C)$ is the subset of documents in D labeled with at least one of the concepts in C. Both $A(C)$ and $O(C)$ are referred to as context phrases.*

Definition II.12. Context Relationship: If D is a collection of documents, c_1 and c_2 are individual concepts, and P is a context phrase, $R(D, c_1, c_2 \mid P)$ is the number of documents in D/P which include both c_1 and c_2. Formally, $R(D, c_1, c_2 \mid P) = |(D/A(\{c_1, c_2\}))|P|$.

The Context Graph

Context relationships are often represented by a *context graph*, which is a graphic representation of the relationship between a set of concepts (e.g., *countries*) as reflected in a corpus respect to a given context (e.g., *crude_oil*).

A context graph consists of a set of *vertices* (also sometimes referred to as *nodes*) and *edges*. The vertices (or nodes) of the graph represent concepts. Weighted "edges" denote the affinity between the concepts.

Each vertex in the context graph signifies a single concept, and two concepts are connected by an edge if their similarity, with respect to a predefined similarity function, is larger than a given threshold (similarity functions in graphing are discussed in greater detail in Chapter X). A context graph is defined with respect to a given context, which determines the context in which the similarity of concepts is of interest (see Figure II.1).

A context graph also has a formal definition:

Definition II.13. Context Graph: If D is a collection of documents, C is a set of concepts, and P is a context phrase, the *concept graph* of D, C, P is a weighted graph $G = (C, E)$, with nodes in C and a set of edges $E = (\{c_1, c_2\} \mid R(D, c_1, c_2 \mid P) > 0)$. For each edge, $\{c_1, c_2\} \in E$, one defines the weight of the edge, $w\{c_1, c_2\} = R(D, c_1, c_2 \mid P)$.

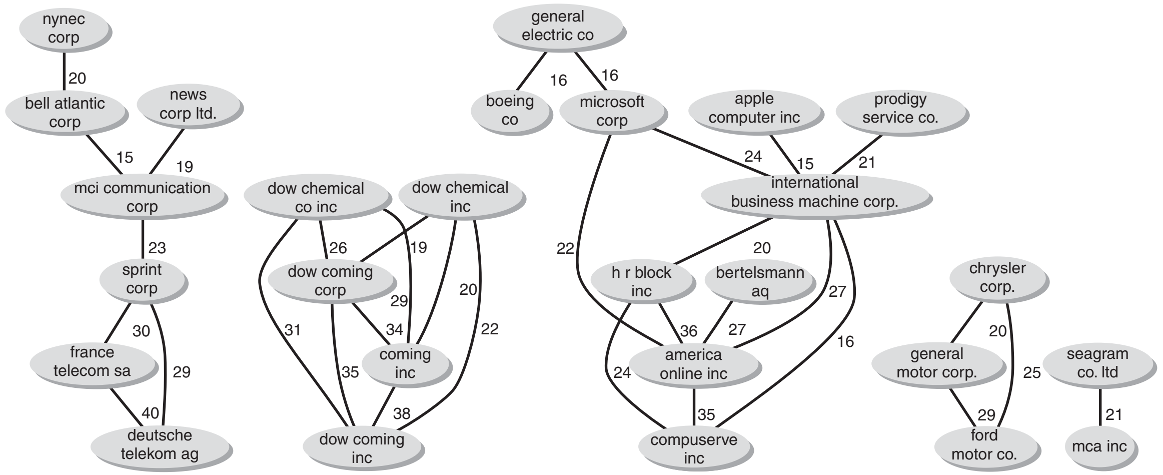


Figure II.1. Context graph for companies in the context of "joint venture." (From Feldman, Fresko, Hirsh, et al. 1998.)

It is often useful to be able to examine not just concept relationships within a given concept context but also to analyze the similarities and differences in context relationships across different temporal segments of the corpus. A *temporal context relationship* refers specifically to the relationship between a set of concepts, as reflected across these segments (identified by individual document date and time stamps) with respect to specified contexts over time. For investigation across segments, a selected subset of documents must be created that constitute a given temporal "segment" of the document collection as a whole.

Definition II.14. Temporal Selection ("Time Interval"): If D is a collection of documents and I is a time range, date range, or both, D_I is the subset of documents in D whose time stamp, date stamp, or both, is within I. The resulting selection is sometimes referred to as the *time interval*.

The formal definition for temporal context relationship builds on both this definition and that supplied earlier for a generic concept relationship (Definition II.12).

Definition II.15. Temporal Context Relationship: If D is a collection of documents, c_1 and c_2 are individual concepts, P is a context phrase, and I is the time interval, then $R_I(D, c_1, c_2 \mid P)$ is the number of documents in D_I in which c_1 and c_2 co-occur in the context of P – that is, $R_I(D, c_1, c_2 \mid P)$ is the number of D_I/P that include both c_1 and c_2.

A *temporal context graph*, then, can be defined as follows:

Definition II.16. Temporal Context Graph: If D is a collection of documents, C is a set of concepts, P is a context phrase, and I is the time range, the *temporal concept graph* of D, C, P, I is a weighted graph $G = (C, E_I)$ with set nodes in C and a set of edges E_I, where $E_I = (\{c_1, c_2\} \mid R(D, c_1, c_2 \mid P) > 0)$. For each edge, $\{c_1, c_2\} \in E$, one defines the weight of the edge by $w_I\{c_1, c_2\} = R_I(D, c_1, c_2 \mid P)$.

The Trend Graph

A *trend graph* is a very specialized representation that builds on the temporal context graph as informed by the general approaches found in trend analysis. A trend graph can be obtained by partitioning the entire timespan covered by a time- or date-stamped document collection, or both, into a series of consecutive time intervals. These intervals can then be used to generate a corresponding sequence of temporal context graphs.

This sequence of temporal context graphs can be leveraged to create combined or cumulative trend graphs that display the evolution of concept relationships in a given context by means of visual cues such as the character and relative weight of edges in the graph. For instance, several classes of edges may be used to indicate various conditions:

- ***New Edges*:** edges that did not exist in the previous graph.
- ***Increased Edges*:** edges that have a relatively higher weight in relation to the previous interval.

- ***Decreased Edges*:** edges that have a relatively decreased weight than the previous interval.
- ***Stable Edges*:** edges that have about the same weight as the corresponding edge in the previous interval.

Handling Dynamically Updated Data

There are many situations in which the document collection for a text mining system might require frequent – perhaps even constant – updating. This regularly occurs in environments in which the maintenance of data *currency* is at a premium such as when a user wants iteratively run searches on topical news, time-sensitive financial information, and so on. In such situations, there is a need for documents to be added dynamically to the document collection and a concurrent need for a user of the text mining system always – that is to say, at every instance of a new document's being added to the collection – to know the full and *current* set of patterns for the searches that he or she has run.

An obvious solution is simply to rerun the search algorithm the user is employing from scratch whenever there is a new data update. Unfortunately, this approach is computationally inefficient and resource intensive (e.g., I/O, memory capacity, disk capacity), resulting in unnecessary performance drawbacks. Additionally, users of text mining systems with large document collections or frequent updates would have to endure more significant interruptions in their knowledge mining activities than if a quicker updating mechanism employing methods of modifying search results on an increment-by-increment basis were implemented.

The more useful and sophisticated approach is to leverage knowledge from previous search runs as a foundation to which new information can be added incrementally. Several algorithms have been described for handling the incremental update situations in data mining, and these algorithms also have applicability in text mining. These include the *FUP*, *FUP*$_2$, and *Delta* algorithms, which all attempt to minimize the recomputation required for incremental updating of Apriori-style, frequent set, and association rule search results. Another algorithm, based on the notion of *border sets* in data mining, however, also offers a very efficient and robust mechanism for treating the incremental case when dealing with discovered frequent sets and associations from natural language documents.

The Borders Incremental Text Mining Algorithm

The Borders algorithm can be used to update search pattern results incrementally. It affords computational efficiency by reducing the number of scans for relations, reducing the number of candidates, and then performing no scan if there is no frequent set. This algorithm is also robust because it supports insertions and deletions as well as absolute and percentage-based thresholds.

The Borders algorithm is based on the notions of *border sets* and *negative borders*. In a sense, a border set can be seen as a notion related to that of a frequent set and may be defined as follows:

Definition II.17. Border Set: X is a border set if it is not a frequent set, but any proper subset $Y \subset X$ is a frequent set (see also Figure II.2).

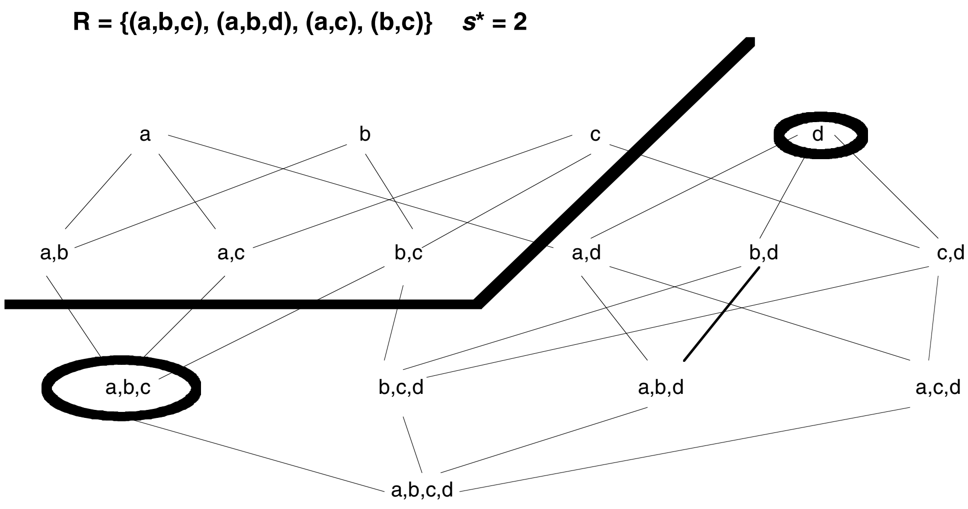


Figure II.2. Illustration of border sets.

The full benefit of the Borders algorithm can be appreciated when one attempts to accommodate incremental data updates of association rules. The Apriori algorithm for generating associations entails two main steps, beginning with the discovery of frequent sets through multiple scans of relations. This "first-step" search for frequent sets is very often the most computationally expensive part of association discovery. For each of the relation scans, a set of *candidates* is assembled and, during each scan, the support of each candidate is computed. The Borders algorithm functions initially to reduce the number of relation scans. Generally this serves to reduce the number of candidates. In addition, the algorithm does not perform a scan if no frequent set is identified.

Some important notational elements for discussing of the Borders algorithm are described below.

- Concept set $A = \{A_1, \ldots, A_m\}$
- Relations over A:
  - R_{old}: old relation
  - R_{inc}: increment
  - R_{new}: new combined relation
- $s(X/R)$: support of concept set X in the relation R
- s^*: minimum support threshold (*minsup*).

The Borders algorithm also makes use of two fundamental properties.

- ***Property 1***: if X is a new frequent set in R_{new}, then there is a subset $Y \subseteq X$ such that Y is a *promoted border*.
- ***Property 2***: if X is a new k-sized frequent set in R_{new}, then for each subset $Y \subseteq X$ of size $k - 1$, Y is one of the following: (a) a promoted border, (b) a frequent set, or (c) an old frequent set with additional support in R_{inc}.

The Borders algorithm itself can be divided into two stages.

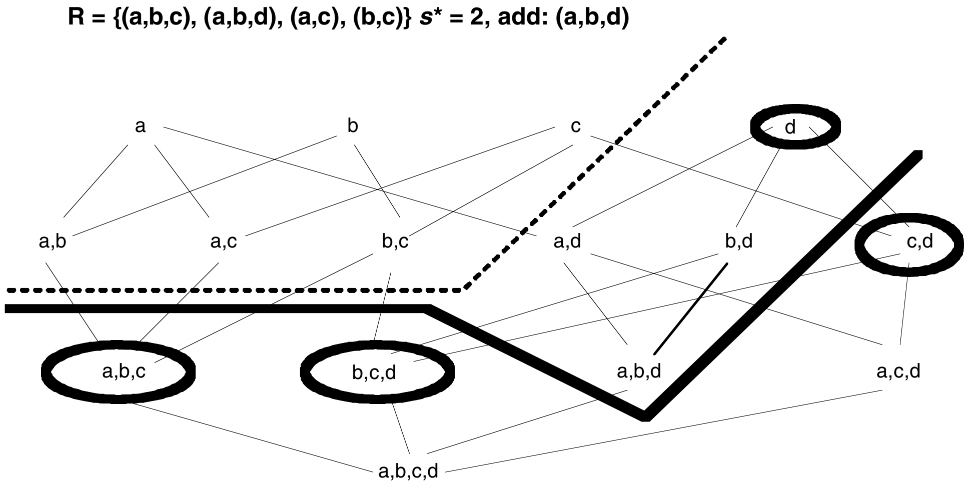


Figure II.3. Illustration of promoted borders and new borders.

- ***Stage 1: Finding Promoted Borders and Generating Candidates*.**
  - Maintain the support for all borders and frequent sets.
  - When new data arrive for each border B of R_{old},
    - Compute $s(B, R_{\text{inc}})$
    - $s(B, R_{\text{new}}) = s(B, R_{\text{old}}) + s(B, R_{\text{inc}})$
  - If $s(B, R_{\text{new}}) \geq s^*$, then B is a *promoted border.*
    - If a *promoted border* does exist,
      - Run an Apriori-like algorithm, and
      - Generate *candidates* using the Property 1 and Property 2.
- ***Stage 2: Processing Candidates.***
  - $L_0 = PB(1), i = 1$
  - Although ($L_1 \neq \emptyset$ or $i \leq$ the largest promoted border)
    - Candidates $(I + 1) = \{X \mid |X| = i + 1$
      - $\exists \quad Y \subset X, |Y| = 1, Y \in PB(i) \cup L_i$
      - $\forall \quad Z \subset X, |Z| = 1, Z \in PB(i) \cup F(i) \cup L_i\}$
    - Scan relation and compute $s(X, R_{\text{new}})$ for each candidate X
    - $L_{i+1} = \{X \text{ candidate: } s(X, R_{\text{new}}) \geq s^*\}$.

See Figure II.3 for an illustration of promoted borders. With the Borders algorithm, full relations are never scanned if there is no new frequent set. Moreover, because of its parsimony in scanning for relations, the algorithm is likely to yield a small candidate set.

Percentage thresholds can be incorporated into incremental update schemes for text mining systems in conjunction with the Borders algorithm. For instance, we can define a threshold as σ percent of the size of the relations, and thus $S^* = \sigma|R|$. The key point for this type of operation is to redetermine the type of each set according to the new threshold before running the algorithm.

Deletions with absolute thresholds for incremental data can be accommodated relatively straightforwardly:

$$s(X, R_{\text{new}}) = s(X, R_{\text{old}}) - s(X, R_{\text{inc}}).$$

For percentage-type thresholds, the approach to handling deletions is perhaps a bit less intuitive but not too complex. In these cases, one can simply look at a deletion as a decrease in the absolute threshold and approach the deletion with the following equation:

$$s^*_{\text{new}} = \sigma(|R_{\text{old}}| + |R_{\text{inc}}|) = s^*_{\text{old}} - s^*_{\text{inc}}.$$

General changes to the threshold value should also be generally supported. Increasing the threshold is relatively easy, for only *borders* and *frequent sets* need be considered. On the other hand, an approach to decreasing the threshold might be to view border X with $s(B, R_{\text{new}}) \geq s^*_{\text{new}}$ as a promoted border before running the Borders algorithm.

II.1.6 Citations and Notes

Section II.1.–II.1.1

The primary source leveraged for information throughout Section II.1 is Feldman, Dagan, et al. (1998). Although focused more on visualization, Hearst (1995) also provides some interesting general background for the topic. Definitions II.1. through II.7. derive from descriptions of distribution and proportion types identified in Feldman, Dagan, et al. (1998).

Section II.1.2

Agrawal, Imielinski, and Swami (1993) and Agrawal and Srikant (1994) introduce the generation of frequent sets as part of the Apriori algorithm. Beyond Agrawal et al.'s seminal research on investigating *market basket*–type associations (Agrawal et al. 1993), other important works shaping the present-day understanding of frequent concept sets include Agrawal and Srikant (1994) and Silverstein, Brin, and Motwani (1999). In addition, Clifton and Cooley (1999) provides a useful treatment of market basket problems and describes how a document may be viewed as a market basket of named entities. Feldman, Aumann, Amir, et al. (1997); Rajman and Besancon (1997b); and Rajman and Besancon (1998) discuss the application of elements of the Apriori algorithm to textual data.

Algorithm 1 in Section II.1.2. was taken from Agrawal and Srikant (1994).

Rajman and Besancon (1997b) provides the background for Section II.1.2.'s discussion of the discovery of frequent concept sets. Although Algorithm 2 in Section II.1.2 is a generalized and simple one for frequent set generation based on the notions set forth in Agrawal et al. (1993) and Agrawal and Srikant (1994), Rajman and Besancon (1997b) provides a slightly different but also useful algorithm for accomplishing the same task.

Section II.1.3

In addition to presenting the framework for generating frequent sets, the treatment of the Apriori algorithm by Agrawal et al. (1993) also provided the basis for generating associations from large (structured) data sources. Subsequently, associations have been widely discussed in literature relating to knowledge discovery targeted at both structured and unstructured data (Agrawal and Srikant 1994; Srikant and Agrawal 1995; Feldman, Dagan, and Kloesgen 1996a; Feldman and Hirsh 1997; Feldman and Hirsh 1997; Rajman and Besancon 1998; Nahm and Mooney 2001; Blake and Pratt 2001; Montes-y-Gomez et al. 2001b; and others).

The definitions for association rules found in Section II.1.3. derive primarily from Agrawal et al. (1993), Montes-y-Gomez et al. (2001b), Rajman and Besancon (1998), and Feldman and Hirsh (1997). Definitions of minconf and minsup thresholds have been taken from Montes-y-Gomez et al. (2001b) and Agrawal et al. (1993). Rajman and Besancon (1998) and Feldman and Hirsh (1997) both point out that the discovery of frequent sets is the most computationally intensive stage of association generation.

The algorithm example for the discovery of associations found in Section II.3.3's Algorithm 3 comes from Rajman and Besancon (1998); this algorithm was directly inspired by Agrawal et al. (1993). The ensuing discussion of this algorithm's implications was influenced by Rajman and Besancon (1998), Feldman, Dagan, and Kloesgen (1996a), and Feldman and Hirsh (1997).

Maximal associations are most recently and comprehensively treated in Amir et al. (2003), and much of the background for the discussion of maximal associations in Section II.1.3 derives from this source. Feldman, Aumann, Amir, et al. (1997) is also an important source of information on the topic. The definition of a maximal association rule in Section II.1.3, along with Definition II.8 and its ensuing discussion, comes from Amir, Aumann, et al. (2003); this source is also the basis for Section II.1.3's discussion of the *M-factor* of a maximal association rule.

Section II.1.4

Silberschatz and Tuzhilin (1996) provides perhaps one of the most important discussions of interestingness with respect to knowledge discovery operations; this source has influenced much of Section II.1.5. Blake and Pratt (2001) also makes some general points on this topic.

Feldman and Dagan (1995) offers an early but still useful discussion of some of the considerations in approaching the isolation of interesting patterns in textual data, and Feldman, Dagan, and Hirsh (1998) provides a useful treatment of how to approach the subject of interestingness with specific respect to distributions and proportions. Definitions II.9 and II.10 derive from Feldman, Dagan, and Hirsh (1998).

Section II.1.5

Trend analysis in text mining is treated by Lent et al. (1997); Feldman and Dagan (1995); Feldman, Dagan, and Hirsh (1998); and Montes-y-Gomez et al. (2001b). Montes-y-Gomez et al. (2001b) offers an innovative introduction to the notions of *ephemeral associations* and *deviation detection*; this is the primary recent source for information relating to these two topics in Section II.1.5.

The analysis of sequences and trends with respect to knowledge discovery in structured data has been treated in several papers (Mannila, Toivonen, and Verkamo 1995;

Srikant and Agrawal 1996; Keogh and Smyth 1997; Bettini, Wang, and Joiodia 1996; Mannila et al. 1995; and Mannila, Toivonen, and Verkamo 1997). Algorithms based on the identification of *episodes* (Mannila et al. 1995) and *sequential patterns* (Srikant and Agrawal 1996) in large data repositories have been described as mechanisms for better mining of implicit trends in data over time. Related work on the discovery of time series analysis has also been discussed (Agrawal and Srikant 1995; Keogh and Smyth 1997).

Lent et al. (1997) and Feldman, Aumann, Zilberstein, et al. (1997) emphasize that trend analysis focused on text mining relates to collections of documents that can be viewed as subcollections defined, in part, by time. These two works are among the most important entry points for the literature of trend analysis in text mining. Montes-y-Gomez et al. (2001b) also makes very interesting contributions to the discussion of the topic.

Definitions related to ephemeral associations come from Montes-y-Gomez et al. (2001b); the terms *ephemeral association* and *deviation detection* are used in this chapter within the general definitional context of this source. Use of the correlation measure r in the detection of ephemeral associations also comes from this source, building on original work found in Freund and Walpole (1990). Finally, the examples used to illustrate direct and inverse ephemeral associations are based on the discussions contained in Montes-y-Gomez et al. (2001b).

The discussion of deviation detection in Section II.1.5 has been shaped by several sources, including Montes-y-Gomez et al. (2001b); Knorr, Ng, and Tucatov (2000); Arning, Agrawal, Raghavan (1996); Feldman and Dagan (1995); and Feldman, Aumann, Zilberstein, et al. (1997). Much of the terminology in this section derives from Montes-y-Gomez et al. (2001b). The term *deviation sources* was coined in Montes-y-Gomez et al. (2001b).

Much of Section II.1.5's discussion of context and trend graphs derives directly from Feldman, Aumann, Zilberstein, et al. (1997) as do Definitions II.11, II.12, II.13, II.14, II.15, and II.16. The trend graph described in Section II.3.5 has also, in a general way, been influenced by Lent et al. (1997).

Feldman, Amir, et al. (1996) was an early work focusing on measures that would support a text mining system's ability to handle dynamically updated data. The *FUP* incremental updating approach comes from Cheung et al. (1996), the *FUP*$_2$ is formalized in Cheung, Lee, and Kao (1997), and the *Delta* algorithms were identified in Feldman, Amir, et al. (1996).

The notion of border sets was introduced, with respect to data mining, in Mannila and Toivonen (1996). Much of the discussion of border sets in this section is an application of the border set ideas of Mannila and Toivonen (1996) to collections of text documents. The Apriori algorithm for generating associations was identified in Agrawal et al. (1993) and Agrawal and Srikant (1994).

II.2 USING BACKGROUND KNOWLEDGE FOR TEXT MINING

II.2.1 Domains and Background Knowledge

As has already been described in Section II.1, concepts derived from the representations of documents in text mining systems belong not only to the descriptive attributes of particular documents but generally also to *domains*. A domain can be loosely defined as a specialized area of interest for which formal *ontologies, lexicons*, and *taxonomies* of information may be created. Domains can exist for very broad areas of interest (e.g., *economics* or *biology*) or for more narrow niches (e.g., *macroeconomics, microeconomics, mergers, acquisitions, fixed income, equities, genomics, proteomics, zoology, virology, immunology*, etc.).

Much of what has been written about the use of *domain knowledge* (also referred to as *background knowledge*) in classic data mining concerns its use as a mechanism for constraining knowledge discovery search operations. From these works, it is possible to generalize three primary forms of usable background knowledge from external sources for data mining applications: (a) constraints, (b) attribute relationship rules, and (c) "hierarchical trees" or "category domain knowledge." More recent literature, however, suggests that other types and implementations of background knowledge may also be useful in data mining operations.

Text mining systems, particularly those with some pronounced elements of domain specificity in their orientation, can leverage information from formal external knowledge sources for these domains to greatly enhance a wide variety of elements in their system architecture. Such elements include those devoted to preprocessing, knowledge discovery, and presentation-layer operations. Even text mining systems without pronounced elements of domain specificity in their design or usage, however, can potentially benefit by the inclusion of information from knowledge sources relating to broad but still generally useful domains such as *the English language* or *world almanac–type facts*.

Indeed, background knowledge can be used in text mining preprocessing operations to enhance concept extraction and validation activities. Furthermore, access to background knowledge can play a vital role in the development of meaningful, consistent, and normalized concept hierarchies.

Background knowledge, in addition, may be utilized by other components of a text mining system. For instance, one of the most clear and important uses of background knowledge in a text mining system is the construction of meaningful constraints for knowledge discovery operations. Likewise, background knowledge may also be used to formulate constraints that allow users greater flexibility when browsing large result sets or in the formatting of data for presentation.

II.2.2 Domain Ontologies

Text mining systems exploit background knowledge that is encoded in the form of *domain ontologies*. A domain ontology, sometimes also referred to less precisely as a *background knowledge source* or *knowledge base*, might be informally defined as the set of all the classes of interest and all the relations between these classes for a given domain. Perhaps another way of describing this is to say that a domain ontology houses all the facts and relationships for the domain it supports. Some see a grouping of facts and relationships as a *vocabulary* constructed in such a way as to be both understandable by humans and readable by machines.

A more formal – albeit very generic – definition for a domain ontology can be attempted with the following notation proposed by Hotho et al. (2003) derived generally from research into formal concept analysis:

Definition II.18. Domain Ontology with Domain Hierarchy: A *domain ontology* is a tuple $O := (C, \leq c)$ consisting of a set C whose elements are called *concepts* and a partial order $\leq c$ on C, which is labeled a *concept hierarchy* or *taxonomy*.

One example of a real-world ontology for a broad area of interest can be found in *WordNet*, an online, public domain ontology originally created at Princeton University that has been designed to model the domain of the English language. Version 1.7 of WordNet contains approximately 110,000 unique concepts (referred to as *synsets* by WordNet's designers); the ontology also has a sophisticated concept hierarchy that supports relation-type information.

WordNet can be used as a "terminological knowledge base" of concepts, concept types, and concept relations to provide broadly useful background knowledge relating to the domain of the English language. A WordNet *synset* represents a single unique instance of a concept meaning related to other *synsets* by some type of specified relation.

Interestingly, WordNet also supports a lexicon of about 150,000 lexical entries (in WordNet's terminology "words") that might more generally be viewed as a list of lexical identifiers or "names" for the concepts stored in the WordNet ontology. Users of WordNet can query both its ontology and its lexicon.

Another ontology implementation that models a narrower subject area domain is the Gene Ontology$^{\text{TM}}$ or GO knowledge base administered by the Gene Ontology Consortium. The GO knowledge base serves as a controlled vocabulary that describes gene products in terms of their associated biological processes, cellular components, and molecular functions. In this controlled vocabulary, great care is taken both to construct and define concepts and to specify the relationships between them. Then, the controlled vocabulary can be used to annotate gene products.

GO actually comprises several different structured knowledge bases of information related to various species, coordinates, synonyms and so on. Each of these ontologies constitutes structured vocabularies in the form of directed acyclic graphs (DAGs) that represent a network in which each concept ("term" in the GO terminology) may be the "child" node of one or more than one "parent" node. An example of this from the GO molecular function vocabulary is the function concept *transmembrane receptor protein-tyrosine kinase* and its relationship to other function concepts; it is a subclass both of the parent concept *transmembrane receptor* and of the parent concept *protein tyrosine kinase*. Figure II.4 provides a high-level view of the Gene Ontology structure.

Several researchers have reported that the GO knowledge base has been used for background knowledge and other purposes. Moreover, the Gene Ontology Consortium has developed various specialized browsers and mapping tools to help developers of external systems leverage the background knowledge extractable from the GO knowledge base.

II.2.3 Domain Lexicons

Text mining systems also leverage background knowledge contained in *domain lexicons*. The names of domain concepts – and the names of their relations – make up a domain ontology's lexicon. The following definitions come from Hotho et al. (2003).

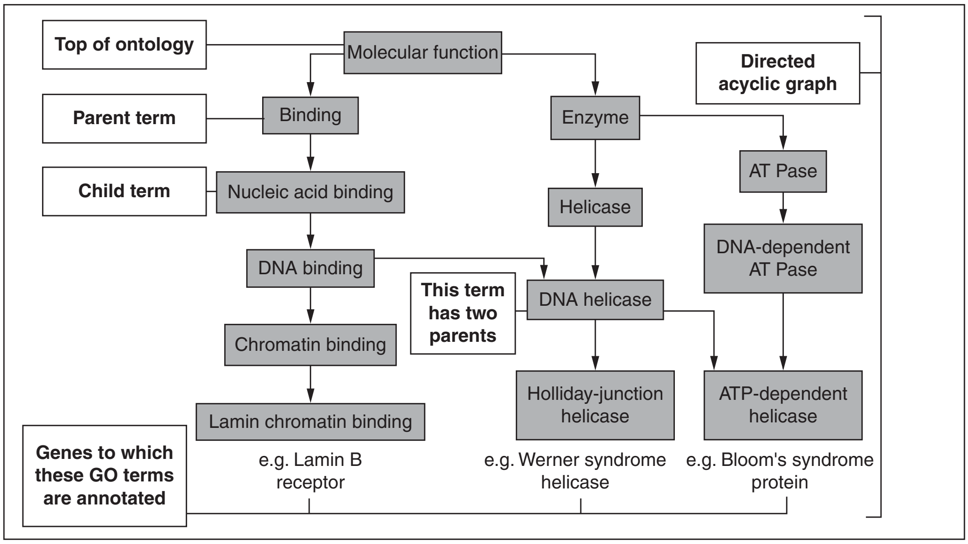


Figure II.4. Schematic of the Gene Ontology structure. (From GO Consortium 2001.)

Definition II.19. Domain Lexicon: A lexicon for an ontology O is a tuple $Lex:=(S_C, Ref_C)$ consisting of a set S_C, whose elements are called *names of concepts*, and a relation $Ref_C \subseteq S_C \times c$ called *lexical reference for concepts* for which $(c, c) \in Ref_C$ holds for all $c \in C \cap S_C$.

Based on Ref_C, we define, for $s \in S_C$, $Ref_C\,(s):=\{c \in C \mid (s, c) \in Ref_C\}$ and, for $c \in C$, $Ref^{-1}{}_C\,(c):=\{s \in S_C \mid (s, c) \in Ref_C\}$.

For the typical situation – such as the WordNet example – of an ontology with a lexicon, one could also use a simple notation:

Definition II.20. Domain Ontology with Lexicon: An *ontology with lexicon* is a pair (O, Lex), where O is an ontology and Lex is a lexicon for O.

A lexicon such as that available with WordNet can serve as the entry point to background knowledge. Using a lexicon, a text mining system could normalize the concept identifiers available for annotation of documents in its corpus during preprocessing in a way that supports, by means of the lexicon's related ontology, both the resolution of synonyms and the extraction of rich semantic relationship information about concepts.

II.2.4 Introducing Background Knowledge into Text Mining Systems

Background knowledge can be introduced into text mining systems in various ways and at various points in a text mining system's architecture. Although there are may be any number of arguments about how background knowledge can enrich the value of knowledge discovery operations on document collections, there are three main *practical* reasons why background information is so universally important in text mining systems.

First, background knowledge can be used in a text mining system to limit pattern abundance. Background knowledge can be crafted into constraints that allow for more efficient and meaningful queries; such constraints can be used for a variety of other purposes as well. Second, background knowledge is an extremely efficient mechanism for resolving questions of concept synonymy and polysemy at the level of search. Access to an ontology that stores both lexical references and relations allows for various types of resolution options. Third, background knowledge can be leveraged in preprocessing operations to create both a consistent lexical reference space and consistent hierarchies for concepts that will then be useful throughout other subsequent query, presentation, and refinement operations.

Perhaps the simplest method to integrate background knowledge into a text mining system is by using it in the construction of meaningful query constraints. For instance, with respect to association discovery, concepts in a text mining system can be preprocessed into either some hierarchical form or clusters representing some limited number of categories or classes of concepts. These categories can then be compared against some relevant external knowledge source to extract interesting attributes for these categories and relations between categories.

A tangible example of this kind of category- or class-oriented background knowledge constraint is a high-level category like *company*, which might, after reference to some commercial ontology of company information, be found to have commonly occurring attributes such as *ProductType, Officers*, *CEO*, *CFO*, *BoardMembers, CountryLocation, Sector, Size*, or *NumberOfEmployees*. The category *company* could also have a set of relations to other categories such as *IsAPartnerOf, IsACustomerOf, IsASupplierTo, IsACompetitorTo,* or *IsASubsidiaryOf*. These category attributes and relations could then be used as constraints available to a user on a pick list when forming a specific association-discovery query relating either to the class *company* or to a concept that is a particular member of that class.

The resulting query expression (with constraint parameter) would allow the user to specify the LHS and RHS of his or her query more carefully and meaningfully. The inclusion of these types of constraints not only increases user interactivity with a text mining system because the user will be more involved in specifying interesting query parameters but can also limit the amount of unwanted patterns resulting from underspecified or inappropriately specified initial queries.

Further, background information constraints can be used in an entirely different way – namely, in the formatting of presentation-level displays of query results. For instance, even if a user did not specify particular constraints as parameters to his or her query expression, a text mining system could still "add value" to the display of the result set by, for instance, highlighting certain associations for which particular preset constraint conditions have been met. An example of this might be that, in returning a result set to a query for all *companies* associated with *crude oil*, the system could highlight those companies identified as suppliers of crude oil in blue whereas those companies that are buyers of crude oil could be highlighted in red. Such color coding might aid in users' exploration of data in the result set because these data provide more information to the user than simply presenting a bland listing of associations differentiated only by confidence level.

Another common use of background knowledge is in the creation of consistent hierarchical representations of concepts in the document collection. During preprocessing – or even during a query – groups of concepts can be compared against some normalized hierarchical form generated from an ontology. The resulting concept hierarchy has the benefit of being both informed by the domain knowledge about relationships collected in the ontology and more consistently integrated with the external source in the event that other types of system operations require reference to information contained in the ontology.

II.2.5 Real-World Example: FACT

FACT (Finding Associations in Collections of Text) was a text mining system developed by Feldman and others during the late 1990s. It represented a focused effort at enhancing association discovery by means of several constraint types supplied by a background knowledge source. In this, it created a very straightforward example of how background knowledge could be leveraged to clear practical effect in knowledge discovery operations on document collections.

General Approach and Functionality

The FACT system might essentially be seen as an advanced tool focused specifically on the discovery of associations in collections of keyword (concept)-labeled text documents. Centering on the association discovery query, the FACT system provided a robust query language through which a user could specify queries over the implicit collection of possible query results supported by the documents in the collection.

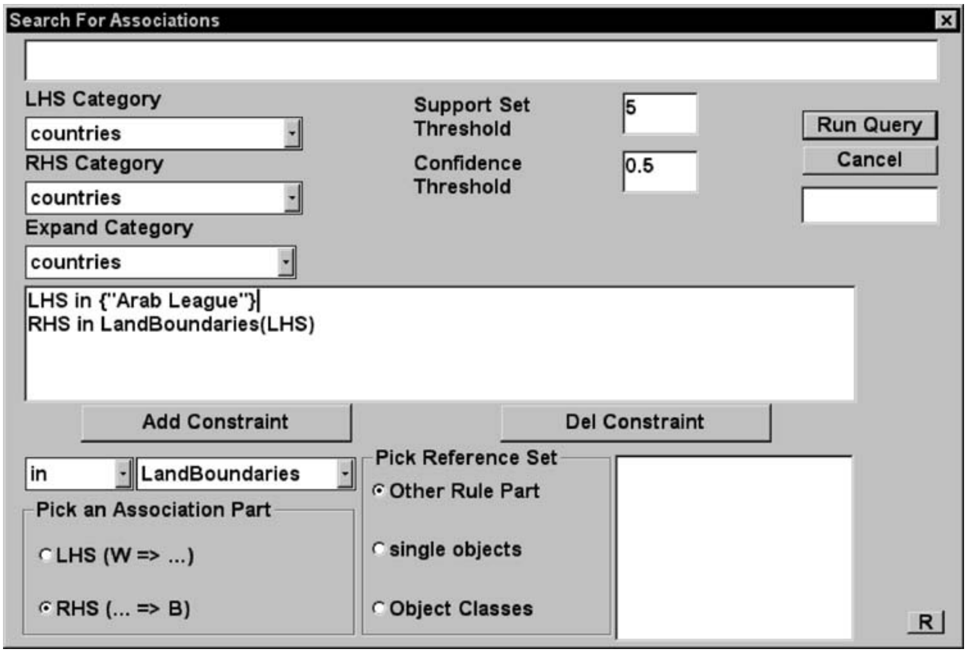
Rather than requiring the specification of an explicit query expression in this language, FACT presented the user with a simple-to-use graphical interface in which a user's various discovery tasks could be specified, and the underlying query language provided a well-defined semantics for the discovery actions performed by the user through the interface (see Figure II.5).

Perhaps most importantly, FACT was able to exploit some basic forms of background knowledge. Running against a document collection of newswire articles, FACT used a simple textual knowledge source (the CIA World Factbook) to exploit knowledge relating to countries. FACT was able to leverage several attributes relating to a country (size, population, export commodities, organizational memberships, etc.) as well as information about relationships between countries (e.g., whether countries were neighbors or trading partners, had a common language, had a common border, etc.).

Using this background knowledge to construct meaningful constraints, FACT allowed a user, when making a query, to include constraints over the set of desired results. Finally, FACT also exploited these constraints in how it structured its search for possible results. This background knowledge thus enabled FACT to, for example, discover associations between a *G7* country, for instance, that appeared as a concept label of a document and some other *nonbordering G7* countries that also appeared as concept labels of the document.

System Architecture

FACT's system architecture was straightforward. In a sense, all system components centered around the execution of a query (see Figure II.6). The system's query



LHS Category: countries
RHS Category: countries
Expand Category: countries
Support Set Threshold: 5
Confidence Threshold: 0.5

LHS in {"Arab League"}
RHS in LandBoundaries(LHS)

in LandBoundaries

Pick an Association Part: LHS (W => ...) [✓ RHS (... => B)]

Pick Reference Set: [✓ Other Rule Part] single objects Object Classes

Figure II.5. FACT's query specification interface. (From Feldman and Hirsh 1997. Reprinted with permission of John Wiley and Sons.)

execution core operations took three inputs – the annotated document collection, distilled background knowledge, and a user's knowledge-discovery query – to create output that was passed to a presentation-layer tool that formatted the result set for display and user browsing.

The system provided an easy-to-use interface for a user to compose and execute an association discovery query, supplemented by constraints for particular types of keywords that had been derived from an external knowledge source. The system then ran the fully constructed query against a document collection whose documents were represented by keyword annotations that had been pregenerated by a series of text categorization algorithms.

Result sets could be returned in ways that also took advantage of the background knowledge–informed constraints. A user could explore a result set for a query and then refine it using a different combination of constraints.

Implementation

The document collection for the FACT system was created from the Reuters-22173 text categorization test collection, a collection of documents that appeared on the Reuters newswire in 1987. This collection obviated the need to build any system elements to preprocess the document data by using categorization algorithms.

The Reuters-22173 documents were preassembled and preindexed with categories by personnel from Reuters Ltd. and Carnegie Group, Inc., and some final formatting was manually applied. The Reuters personnel tagged each document

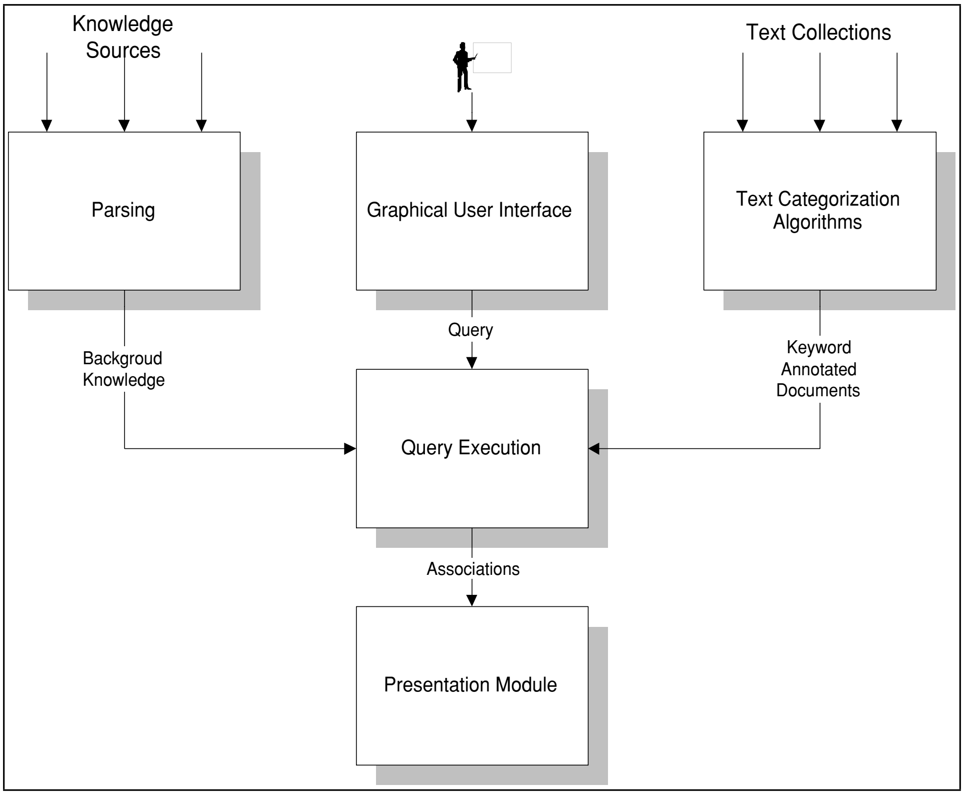
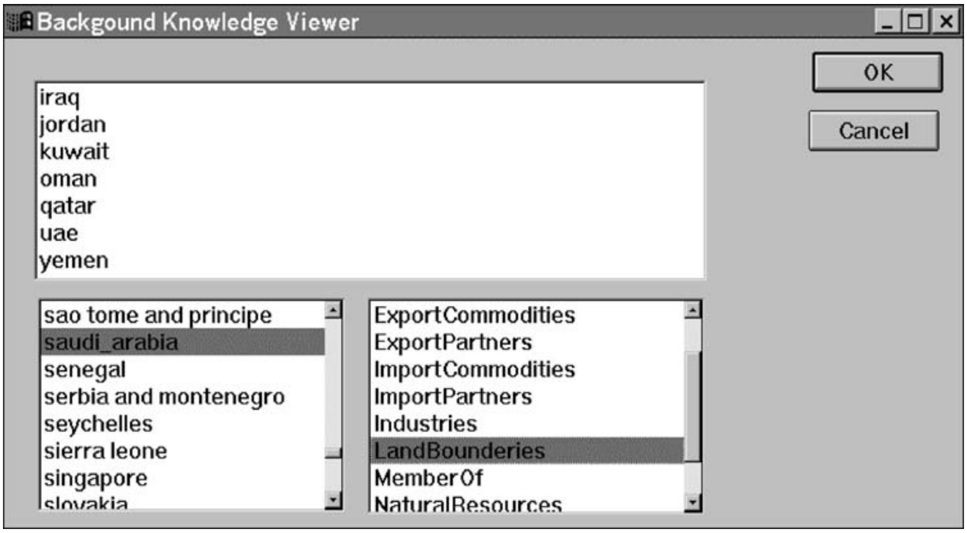


Figure II.6. System architecture of FACT. (From Feldman and Hirsh 1997. Reprinted with permission of John Wiley and Sons.)

with a subset of 135 keywords that fell into five overarching categories: *countries*, *topics*, *people*, *organizations*, and *stock exchanges*.

The 1995 CIA World Factbook that served as the FACT system's ostensible ontology amd background knowledge source was a structured document containing information about each of the countries of the world and was divided into six sections: Geography, People, Government, Economy, Communications, and Defense Forces. For experimentation with the Reuters-22173 data, the following background information was extracted for each country *C*:

- **MemberOf:** all organizations of which *C* is a member (e.g., G7, Arab League, EC),
- **LandBoundaries:** the countries that have a land border with *C*,
- **NaturalResources:** the natural resources of *C* (e.g., crude, coal, copper, gold),
- **ExportCommodities:** the main commodities exported by *C* (e.g., meat, wool, wheat),
- **ExportPartners:** the principal countries to which *C* exports its ExportCommodities,
- **ImportCommodities:** the main commodities imported by *C* (e.g., meat, wool, wheat),



iraq
jordan
kuwait
oman
qatar
uae
yemen

sao tome and principe
saudi_arabia
senegal
serbia and montenegro
seychelles
sierra leone
singapore
slovakia

ExportCommodities
ExportPartners
ImportCommodities
ImportPartners
Industries
LandBounderies
MemberOf
NaturalResources

Figure II.7. FACT's background knowledge viewer showing the countries having land boundaries with Saudi Arabia. (From Feldman and Hirsh 1997. Reprinted with permission of John Wiley and Sons.)

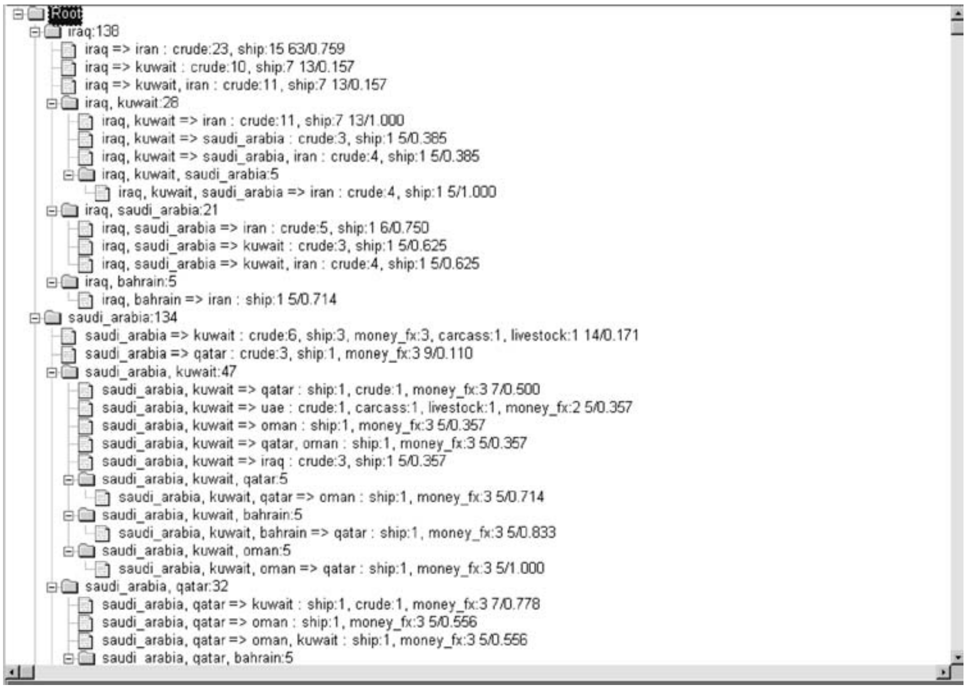
- **ImportPartners:** the principal countries from which *C* imports its Import Commodities,
- **Industries:** the main industries of *C* (e.g., iron, steel, machines, textiles, chemicals), and
- **Agriculture:** the main agricultural products of *C* (e.g., grains, fruit, potatoes, cattle).

The first boldfaced element before the colon defines a unary predicate, and the remainder of each entry constitutes a binary predicate over the set of keywords that can label the documents in the Reuters-22173 collection. Users could browse this background knowledge in FACT by means of a utility (see Figure II.7).

For its main association-discovery algorithm, FACT implemented a version of the two-phase Apriori algorithm. After generating σ-covers, however, FACT modified the traditional association-discovery phase to handle the various types of constraints that had been generated from the CIA World Factbook.

Upon completion of a query, FACT executed its query code and passed a result set back to a specialized presentation tool, the FACT system's association browser. This browser performed several functions. First, it filtered out redundant results. Second, it organized results hierarchically – identifying commonalties among the various discovered associations and sorting them in decreasing order of confidence.

Further, the tool housed this hierarchical, sorted representation of the result set in a screen presentation that enabled a user to browse the titles of documents supporting each of the individual associations in the result set simply by pointing and clicking on that association (see Figure II.8).



```
Root
  iraq:138
    iraq => iran : crude:23, ship:15 63/0.759
    iraq => kuwait : crude:10, ship:7 13/0.157
    iraq => kuwait, iran : crude:11, ship:7 13/0.157
    iraq, kuwait:28
      iraq, kuwait => iran : crude:11, ship:7 13/1.000
      iraq, kuwait => saudi_arabia : crude:3, ship:1 5/0.385
      iraq, kuwait => saudi_arabia, iran : crude:4, ship:1 5/0.385
      iraq, kuwait, saudi_arabia:5
        iraq, kuwait, saudi_arabia => iran : crude:4, ship:1 5/1.000
    iraq, saudi_arabia:21
      iraq, saudi_arabia => iran : crude:5, ship:1 6/0.750
      iraq, saudi_arabia => kuwait : crude:3, ship:1 5/0.625
      iraq, saudi_arabia => kuwait, iran : crude:4, ship:1 5/0.625
    iraq, bahrain:5
      iraq, bahrain => iran : ship:1 5/0.714
  saudi_arabia:134
    saudi_arabia => kuwait : crude:6, ship:3, money_fx:3, carcass:1, livestock:1 14/0.171
    saudi_arabia => qatar : crude:3, ship:1, money_fx:3 9/0.110
    saudi_arabia, kuwait:47
      saudi_arabia, kuwait => qatar : ship:1, crude:1, money_fx:3 7/0.500
      saudi_arabia, kuwait => uae : crude:1, carcass:1, livestock:1, money_fx:2 5/0.357
      saudi_arabia, kuwait => oman : ship:1, money_fx:3 5/0.357
      saudi_arabia, kuwait => qatar, oman : ship:1, money_fx:3 5/0.357
      saudi_arabia, kuwait => iraq : crude:3, ship:1 5/0.357
      saudi_arabia, kuwait, qatar:5
        saudi_arabia, kuwait, qatar => oman : ship:1, money_fx:3 5/0.714
      saudi_arabia, kuwait, bahrain:5
        saudi_arabia, kuwait, bahrain => qatar : ship:1, money_fx:3 5/0.833
      saudi_arabia, kuwait, oman:5
        saudi_arabia, kuwait, oman => qatar : ship:1, money_fx:3 5/1.000
    saudi_arabia, qatar:32
      saudi_arabia, qatar => kuwait : ship:1, crude:1, money_fx:3 7/0.778
      saudi_arabia, qatar => oman : ship:1, money_fx:3 5/0.556
      saudi_arabia, qatar => oman, kuwait : ship:1, money_fx:3 5/0.556
      saudi_arabia, qatar, bahrain:5
```

Figure II.8. FACT's association browser presentation module showing a result set for associations of Arab League countries with countries sharing a border. (From Feldman and Hirsh 1997. Reprinted with permission of John Wiley and Sons.)

Experimental Performance Results

FACT appeared to perform well on queries of the form "find all associations between a set of countries including *Iran* and any person" and "find all associations between a set of topics including *Gold* and any country" as well as more complex queries that included constraints. One interesting – albeit still informal and crude – experiment performed on the system was to see if there was any performance difference (based on a comparison of CPU time) between query templates with and without constraints. In most cases, the queries involving constraints extracted from background knowledge appeared to be noticeably more efficient in terms of CPU time consumption.

Some practical difficulties were encountered when trying to convert the CIA World Factbook into unary and binary predicates when the vocabulary in the Factbook differed from the universe of keywords labeling the Reuters documents (Feldman and Hirsh 1997). This is a problem that can creep into almost any text mining system that attempts to integrate background knowledge. FACT's designers put in place a point solution to resolve this problem by including additional background knowledge from a standard reference dictionary to help at least provide a basic definition of synonyms.

Obviously, today, advanced text mining systems involving background knowledge can integrate with more sophisticated dictionary-type ontologies like WordNet to resolve problems with synonymy. Further, today's designers of text mining systems can also consider various strategies for including background knowledge in preprocessing routines to help create more consistency in the concept tags that annotate document collections before the execution of any knowledge discovery algorithms.

II.2.6 Citations and Notes

Section II.2.1

For general discussion of the use of background knowledge to construct constraints in classic data mining, see Anand, Bell, and Hughes (1995) and Yoon et al. (1999). Kopanis, Avouris, and Daskalaki (2002) discusses other uses for background knowledge in data mining systems. Feldman and Hirsh (1996a) provides an early discussion of various uses of background knowledge within a text mining system.

Section II.2.2

The informal definition for a domain ontology in Section II.2.2 comes from Craven and Kumlien (1999). The definition for a domain vocabulary was derived from Gruber (1993). Definition II.18 has been taken from Hotho et al. (2003); this source provides much of the background and definitional information for the topics discussed throughout Sections II.2.2 through II.2.4.

A large body of literature exists on the subject of WordNet, but the basic overview is contained in Martin (1995); the identification of WordNet as a “terminological knowledge base” also comes from this source. Descriptions of WordNet’s lexicon, concept hierarchy, and ontological structure rely on information published in Rodriguez, Gomez-Hidalgo, and Diaz-Agudo (1997) and Hotho et al. (2003).

The Gene Ontology knowledge base is described in GO Consortium (2000). The schematic of the GO knowledge base displayed in Figure II.4 comes from GO Consortium (2001); the example in Section II.2.2 involving the function concept *transmembrane receptor protein-tyrosine kinase* was also taken from this source. Hill et al. (2002) and Hirschman et al. (2002) have both reported use of the GO knowledge base for background knowledge purposes in knowledge discovery systems.

Section II.2.3

Definitions II.19 and II.20 as well as the WordNet examples used in discussing these definitions come from Hotho et al. (2003).

Sections II.2.4.–II.2.5

The FACT system is described in Feldman and Hirsh (1996a), Feldman and Hirsh (1996b), and Feldman and Hirsh (1997), and it influenced a substantial amount of later discussion of text mining systems (Landau, Feldman, Aumann, et al. 1998; Blake and Pratt 2001; Montes-y-Gomez et al. 2001b; Nahm and Mooney 2001; and others). Most of the descriptions of the FACT system found in Section II.2.5 derive from Feldman and Hirsh (1997).

II.3 TEXT MINING QUERY LANGUAGES

Query languages for the type of generalized text mining system described in this chapter must serve several straightforward purposes. First, these languages must allow for the specification and execution of one of the text mining system's search algorithms. Second, they generally need to allow for multiple constraints to be appended to a search argument; such constraints need to be specifiable by a user. Third, the query languages typically also need to perform some types of auxiliary filtering and redundancy to minimize pattern overabundance in result sets.

Most text mining systems offer access to their query language either through a more abstracted and "friendly" interface that acts to assist the user by means of pick lists, pull-down menus, and scroll bars containing preset search types and constraints or through more direct "command-line" access to the query language that exposes query language expressions in their full syntax. Some text mining systems offer both.

It is important in any implementation of a query language interface for designers of text mining systems to consider carefully the usage situations for the interfaces they provide. For instance, having a user-friendly, graphically oriented tool may greatly enhance a system's ease of use, but if this tool severely limits the types of queries that may be performed it may not meet a strict cost–benefit analysis.

Similarly, direct access to a text mining system's query language to support the construction of ad hoc queries can be very advantageous for some users trying to experiment with queries involving complex combinations of constraints. If, however, such a direct query interface does not allow for robust storage, reuse, renaming, and editing of ad hoc queries as query templates, such "low level" access to the query language can become very inefficient and frustrating for users.

II.3.1 Real World Example: KDTL

The text mining query language KDTL (knowledge discovery in text language) was first introduced in 1996 as the query language engine supporting the FACT system and was subsequently more fully described as a central element of Feldman, Kloesgen, et al.'s later Document Explorer system.

KDTL's primary function is to provide a mechanism for performing queries that isolate interesting patterns. A Backus Naur Form (BNF) description of KDTL is shown in Figure II.9.

KDTL supports all three main patter-discovery query types (i.e., distributions, frequent sets, and associations) as well as less common graphing outputs (i.e., keyword graph, directed keyword graph). Also notice that each query contains one algorithmic statement and several constraint statements.

The constraint part of the query is structured in such a way that the user needs first to select a single relevant component – that is, the left-hand side (LHS) of the association, right-hand side (RHS), frequent set, or a path in a keyword graph. Then, all subsequent constraint statements are applied to this component.

When specifying set relations, the user can optionally specify background predicates to be applied to the given expressions. KDTL intentionally contains some redundancy in the constraints statements to facilitate easier specification of queries.

II.3.2 KDTL Query Examples

Here are some typical examples of KDTL queries executed on the Reuters-22173 document collection used by FACT and described in Section II.5.

```
Algorithmic statements:

gen_rule() : generate all matching association rules
gen_frequent_set(): generate all matching frequent sets
gen_kg() : generate a keyword graph
gen_dkg() : generate a directed keyword graph
gen_dist() : generate a distribution

Constraint statements:

set_filter(<Set>) - the set MUST meet the following
constraints
set_not_filter(<Set>) - the set MUST NOT meet the following
constraints

<Set> ::=  frequent_set | left | right | path

contain([<background predicate>], <Expression>) -
      the designated set must contain <expression> (or
      <background predicate>(<expression>))
subset([<background predicate>],<Expression>) -
     the designated set is a subset of <expression> (or
      <background predicate>(<expression>))
disjoint([<background predicate>],<Expression>) -
     the designated set and <expression> are disjoint (or
      <background predicate>(<expression>))
equal([<background predicate>],<Expression>) -
     the designated set is equal to <expression> (or
      <background predicate>(<expression>))

all_has(<Expression>) -
     all members of the designated set are descendents of
      <expression> in the taxonomy
one_has(<Expression>) -
     at least one of the members of the designated set is a
      descendent of <expression>

property_count(<Expression>,<low>,<high>) -
     # of members that are descendents of <expression> is in
      the specified range
size(<Expression>,<low>,<high>) -
     size of the designated set is in the specified range

Set_conf(real)
Set_supp(integer)

<Expression> ::= Keyword | Category |
     <Expression>,<Expression> |
                         <Expression> ; <Expression>
<high> ::= integer
<low>  ::= integer
```

Figure II.9. BNF description of KDTL. (From Feldman, Kloesgen, and Zilberstein 1997a. Reprinted with permission of Springer Science and Business Media.)

In order to query only those associations that correlate between a set of countries including Iran and a person, the KDTL query expression would take the following form:

```
set_filter(left); all_has({"countries"}); contain({"iran"});
set_filter(right); all_has({"people"}); property_count("people",1,1);
set_supp(4); set_conf(0.5); gen_rule();
```

Run against the Reuters collection, the system would find four associations as a result of this particular query, all of which would have Reagan in the RHS.

(6.54%) Iran, Nicaragua, USA ⇒ Reagan
(6.50%) Iran, Nicaragua ⇒ Reagan
(18.19%) Iran, USA ⇒ Reagan
(19.10%) Iran ⇒ Reagan

The interesting associations are those that include Iran and Nicaragua on the LHS. Upon querying the document collection, one can see that, when Iran and Nicaragua are in the document, then, if there is any person in the document, Reagan will be in that document too. In other words, the association Iran, Nicaragua, <person> ⇒ Reagan has 100-percent confidence and is supported by six documents. The <person> constraint means that there must be at least one person name in the document.

As another example, if one wanted to infer which people were highly correlated with West Germany (Reuters collection was from a period before the reunion of Germany), a query that looked for correlation between groups of one to three people and West Germany would be formulated.

```
set_filter("left"); size(1,3); all_has({"people"});
set_filter("right"); equal({"west_germany"});
set_supp(10); set_conf(0.5); gen_rule();
```

The system found five such associations; in all them the people on the LHS were senior officials of the West German government. Kohl was the Chancellor, Poehl was the president of the Central Bank, Bangemann was the Economic Minister, and Stoltenberg was the Finance Minister. If one wanted to infer from a document collection who the high officials of a given country are, a similar query would probably yield a reasonably accurate answer.

This type of example can also be used to show how background knowledge can be leveraged to eliminate trivial associations. For instance, if a user is very familiar with German politics and not interested in getting these particular associations, he or she might like to see associations between people who are not German citizens and Germany. Adding the constraints *set_filter_not("left"); equal(nationality, "west_germany");* will eliminate all the associations shown below.

(8.100%) Poehl, Stoltenberg ⇒ West Germany
(6.100%) Bangemann ⇒ West Germany
(11.100%) Kohl ⇒ West Germany
(21.80%) Poehl ⇒ West Germany
(44.75%) Stoltenberg ⇒ West Germany

II.3.3 KDTL Query Interface Implementations

In Figures II.10 and II.11, one can see two elements of a sample GUI for defining KDTL queries. In the KDTL Query Editor (see Figure II.10), a user builds a query expression with one constraint at a time.

The tabbed dialog boxes in Figure II.11 demonstrate how the user defines a single constraint. Several different types of set constraints are supported, including background and numerical size constraints.

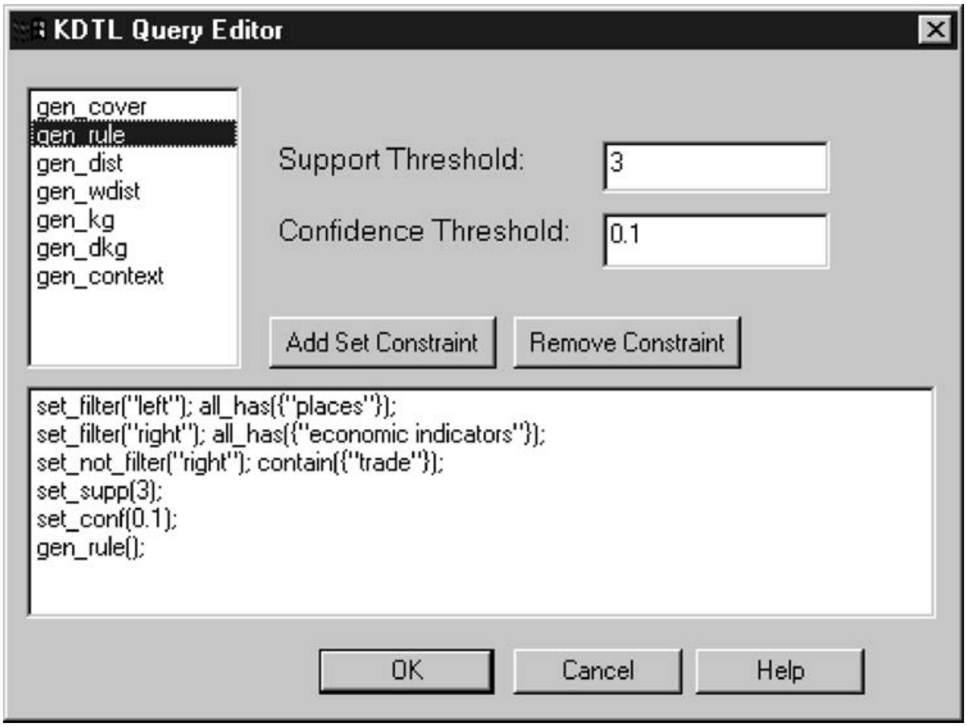
The results of a typical query – of the kind defined in Figures II.10 and II.11 – can be seen in Figure II.12.

In this query, the object was to find all associations that connect a set of countries and a set of economical indicator topics if trade is not in the set. Only one association satisfies all these constraints. If the last constraint had been lifted – and one allowed "trade" to be in the RHS of the association – the system would have returned 18 associations.

II.3.4 Citations and Notes

Sections II.3–II.3.2

The descriptions of KDTL in Section II.3, as well as the example of the language and the various screen shots of query interfaces, primarily come from Feldman, Kloesgen, and Zilberstein (1997a). See also Feldman and Hirsh (1997).



Support Threshold: 3

Confidence Threshold: 0.1

```
set_filter("left"); all_has({"places"});
set_filter("right"); all_has({"economic indicators"});
set_not_filter("right"); contain({"trade"});
set_supp(3);
set_conf(0.1);
gen_rule();
```

Figure II.10. Defining a KDTL query. (From Feldman, Kloesgen, and Zilberstein 1997a. Reprinted with permission of Springer Science and Business Media.)

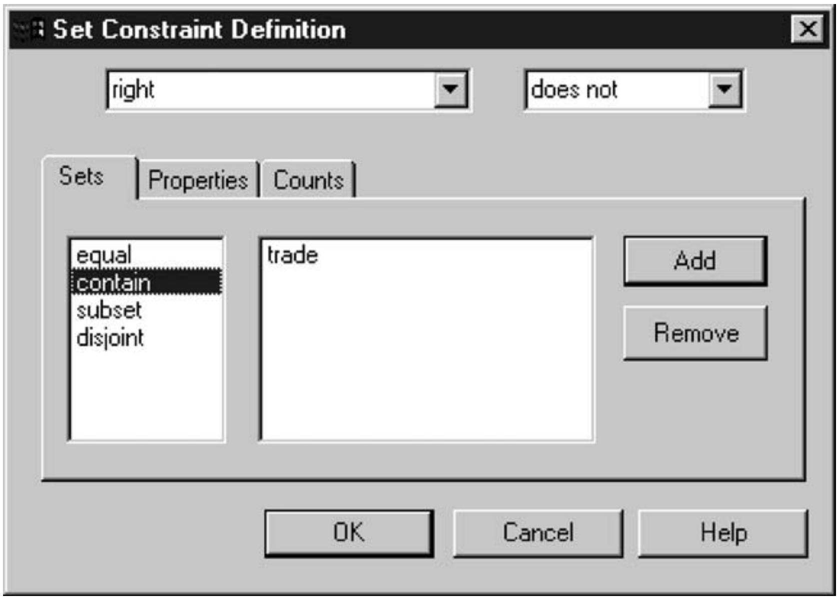
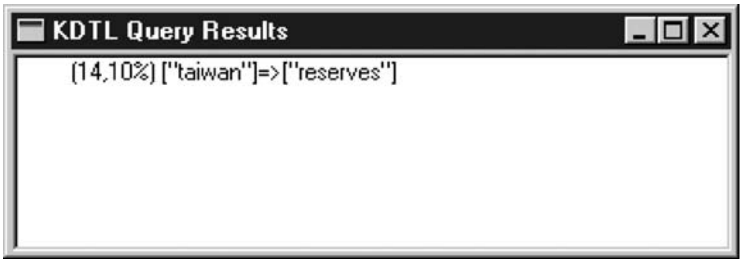


Figure II.11. Defining a KDTL set constraint. (From Feldman, Kloesgen, and Zilberstein 1997a. Reprinted with permission of Springer Science and Business Media.)



```
(14,10%) ["taiwan"]=>["reserves"]
```

Figure II.12. Interface showing KDTL query results. (From Feldman, Kloesgen, and Zilberstein 1997a. Reprinted with permission of Springer Science and Business Media.)