In Proceedings of the Second International Conference on Knowledge Discovery from Databases, 1996

Mining Associations in Text in the Presence of Background Knowledge

Ronen Feldman

Mathematics and Computer Science Department
Bar-Ilan University
Ramat-Gan, ISRAEL 52900
feldman@bimacs.cs.biu.ac.il

Haym Hirsh

Department of Computer Science
Rutgers University
Piscataway, NJ USA 08855
hirsh@cs.rutgers.edu

Abstract

This paper describes the FACT system for knowledge discovery from text. It discovers associations – patterns of co-occurrence – amongst keywords labeling the items in a collection of textual documents. In addition, FACT is able to use background knowledge about the keywords labeling the documents in its discovery process. FACT takes a query-centered view of knowledge discovery, in which a discovery request is viewed as a query over the implicit set of possible results supported by a collection of documents, and where background knowledge is used to specify constraints on the desired results of this query process. Execution of a knowledge-discovery query is structured so that these background-knowledge constraints can be exploited in the search for possible results. Finally, rather than requiring a user to specify an explicit query expression in the knowledge-discovery query language, FACT presents the user with a simple-to-use graphical interface to the query language, with the language providing a well-defined semantics for the discovery actions performed by a user through the interface.

Introduction

Suppose someone comes along and gives you a large collection of textual documents – newswire stories, internal business memos, netnews articles, email messages, or even WWW pages – and asks you to find something interesting in the collection. We have previously labeled this problem "Knowledge Discovery from Text" (KDT) [Feldman and Dagan, 1995; Dagan et al., 1996].

This paper describes FACT (Finding Associations in Collections of Text), a tool for discovering associations in collections of textual documents given background knowledge about the topics of documents in the collection. Central to this work is a query-centered view of the discovery process [Imielinski, 1995]. Given a collection of data, there is a corresponding implicit collection of possible results supported by the data. FACT provides a query language for the discovery process in which a user can specify queries over this implicit collection of possible results supported by the data. However, rather than requiring the specification of an explicit query expression in this language, FACT presents the user with a simple-to-use graphical interface in which a user's various discovery tasks are specified, with the underlying query language providing a well-defined semantics for the discovery actions performed by the user through the interface. As part of this interface a user can specify constraints over the set of desired results in terms of background knowledge about the topics of the documents, with FACT exploiting such constraints in how it structures its search for possible results.

We begin the paper with an overview of the FACT system. We then describe the general problem of finding associations, our association-discovery query language, and our algorithms for executing queries in this language. Finally, we discuss the use of FACT on a collection of Reuters newswire stories using background knowledge automatically extracted from the CIA World Factbook.

The FACT System Architecture

FACT takes as input three sources of information. The First is a collection of textual data on which the discovery process takes place. Since our approach begins with the assumption borrowed from the Information Retrieval literature that each document is labeled with a set of keywords representing the topics of the document, the input text collections must either already be labeled with such keywords (as is the case for the Reuters data discussed in Section 6), or must be fed through a text categorization system that annotates documents with such keywords.

In addition to that, FACT also takes as input background knowledge for its discovery process. To be usable by FACT such knowledge must define unary and binary predicates over the keywords labeling the documents, representing properties of the entities represented by each keyword and relationships between them. Thus for the Reuters newswire data, for example, FACT is told for each country-keyword the organizations of which that country is a mem-

ber, thereby defining a set of unary predicates over the country-keywords (one per organization). FACT is also given information about which countries neighbor one another, defining a set of binary predicates over the country-keywords. Since such information is rarely available in the precise form needed by the FACT system, it will usually be necessary to develop tools that understand the format of an information source and can translate into the necessary format for FACT. For example, the background knowledge used in our Reuters newswire experiments comes from the CIA World Factbook, a structured textual document with information about the various countries of the world. To make it possible for FACT to use this knowledge we had to develop a tool that parses the Factbook's well-structured text and converts it into the format used by FACT.

Finally, FACT is provided with the user's specification of a knowledge-discovery task, which is acquired from the user via a simple graphical user interface. The interface knows about the various keywords that can label a document, as well as the various unary and binary predicates defined by the background knowledge that can be applied to these keywords, and allows the user to specify a query using this keyword and predicate vocabulary via a collection of menus.

The results of this discovery process are then passed on to a tool, that can effectively present the results and allow the user to browse them. This component of FACT filters out redundant results [Feldman et al., 1996], sorts results in decreasing order of confidence, and enables the user to access and browse those documents that support each of the individual results that it presents to the user.

The Query Language

To execute an association-discovery task a user specifies a query in FACT's association query language – the association-discovery process should only return results that satisfy the query . Each association-discovery query has three parts. The first part specifies what types of keywords are desired in the left-hand and right-hand sides of any found associations, as well as what support and confidence the association should have. Thus, for example, a user can express an interest in associations that relate a set of countries labeling a document to a person also labeling the document, as long as the association has sufficient support and confidence in the collection.

The second (possibly empty) part of a query specifies constraints – in terms of the predicates defined by the background knowledge – that the user wants any found association to satisfy. There are two types of background knowledge that can be used in queries. The first are unary predicates over keywords. In specifying a query, each unary predicate is viewed as a class of keywords, specifying the set of keywords for which it is true. Thus, for example, the unary predicate EC that is true if a keyword is the name of a country that is a member of the European Community is viewed as a class whose members are those keywords that are European-Community countries. A user can request that a unary predicate be true of some keyword in an association by specifying that the keyword be a member of the class defined by the unary predicate.

The second type of background knowledge that can be used in queries are binary predicates, which define relationships between keywords. Thus, for example, the background knowledge might define the binary predicate Nationality, which is true whenever the first argument is the name of some person whose nationality is the country appearing as the second argument of the predicate. For the query language each binary predicate is viewed as a function: given the value of the first argument, it returns the set of values for the second argument that would make the predicate true. The predicate Nationality, for example, would be viewed as a function that takes a person's name as input and returns the country that is that person's nationality; the predicate ExportCommodity becomes the function Export-Commodities that takes a country keyword as input and outputs the keywords representing that country's export commodities. Further, whenever a function is applied to a set of keywords, the function returns all second arguments that make the predicate true for any element in the input set of keywords. A user can request that a binary predicate be true of some keywords in an association by specifying that one keyword be amongst the values returned by the function when it is applied to some other keyword in the association.

Finally, the third (also possibly empty) part of a query specifies constraints on the size of the various components of the association. Thus, for example, a user can request associations that have only one keyword on their right-hand side, or that mention at most 5 country keywords.

Figure 1 gives a BNF grammar of our association-discovery query language, , where nonterminals are written in angle brackets, “(0,1]” represents the set of reals between 0 (noninclusive) and 1 (inclusive), and <CategoryType> is defined as appropriate for a given domain, dividing the keywords labeling the documents into subclasses (such as “country”, “person”, etc., in the Reuters newswire data). Any expansion for “<Arg>” to “<Var>” must a variable that was previously defined in “<Pattern>” for that query (i.e., this is a context-sensitive portion of the language that cannot be represented in BNF).

<Query> ::= *Find* (<support>/<confidence>) Pattern
Where: <BackgroundConstraint>*
<KeywordConstraint>*

<support> ::= <integer>
<confidence> ::= (0,1]
<Pattern> ::= <VarList> ⇒ <VarList>
<VarList> ::= <VarExp> | <VarExp>, <VarList>
<VarExp> ::= <Var> : <TypeExp>
<TypeExp> ::= <CategoryType> | <CategoryType>+

<BackgroundConstraint> ::= <Arg> <Operator> <Arg>
<Operator> ::= ∈ | ∉ | ⊆ | ⊄ | = | ≠
<Arg> ::= <Var> | <Keyword> | <Class> | <BgExpression> | LHS | RHS | All
<BgExpression> ::= <BackgroundFunction>(<Arglist>)
<Arglist> ::= <Arg> | <Arg>,<Arglist>

<KeywordConstraint> ::= <#Exp> <CompOperator> <#Exp>
<#Exp> ::= <numeric constant> | #(<category>) | #(LHS) | #(RHS) | #(All)

<CompOperator> ::= > | ≥ | < | ≤ | = | ≠

Figure 1 - BNF grammar of the Query language of FACT

For example, the query "*Find*: (10/0.2) c:country+ ⇒ p:person, *Where*: Nationality(p) ∉ c, #(LHS) ≤ 3" (taken from the Reuters newswire domain) requests associations where, at least 20% of the time, whenever some set of at most three countries labels a document it is also labeled with some person whose nationality is not one of those countries, and this occurs at least 10 times in the collection.

Query Execution

Our algorithms for executing association-discovery queries in the presence of background knowledge are based on those described by Agrawal and Srikant [1994] and Mannila et al. [1994]. Once a collection of all σ-covers (X is called a σ-cover if $|[X]| \geq \sigma$) has been found, the traditional association-discovery process attempts to find all subsets B for each σ-cover X for which X\B ⇒ B holds with the desired confidence γ. To limit the search to those associations satisfying the constraints we use the σ-cover algorithm in a slightly different fashion, so as to use the constraints to reduce the search space and make the association generation process more efficient.

To do this, we divide the constraints on possible associations into two classes. The first class contains those "simple" constraints that refer to only one side of the association, such as LHS ⊆ Arab League, or Iran ∈ RHS, as well as those constraints that require some property to hold on the whole association, such as #(All) < 5. The second class contains those "complex" constraints that require some relationship to hold between elements of the two side of the association, such as RHS ⊆ LandBoundaries(LHS). We use both classes of constraints to reduce the space of possible σ-covers that must be considered.

Figure 2 gives an outline of this algorithm for finding associations in the presence of such constraints. It takes as input the collection of documents, *Ds*; *K(D)* is used to refer to the collection of keywords labeling document *D*. The algorithm finds all possible LHS candidates, only searching through those that satisfy the simple constraints on the LHS. For each such result the algorithm considers which other keywords could appear as the RHS of the association, constrained according to whatever additional constraints are present. At the end of the process it determines which associations satisfy the support and confidence from those that were created satisfying the given constraints.

Use the σ-cover algorithm to create *Ls*, the set of all left-hand sides that could satisfy the association-discovery query, constrained to only consider those keywords satisfying the simple constraints on the LHS.
For all $D \in Ds$
  For all $X \in Ls$ do
    if $X \subseteq K(D)$ then
      B = The keywords in $K(D) \backslash X$ that satisfy the constraints on RHS (either simple constraints or composite constraints) and that appear with the required support.
      Update co-occurrence counters for X and all subsets of B
    end if
  end do
end do
Form associations based on the accumulated co-occurrence counters. Remove those associations that do not satisfy the required support and confidence.

Figure 2 - Query evaluation algorithm

Applying FACT to Newswire Data

To investigate the use of FACT to find associations in text we used it on the Reuters-22173 newswire data often used in research in Information Retrieval. Our goal is not just the discovery of associations in text, but doing so in the presence of background knowledge of the domain. To investigate the role of background knowledge in association discovery for the Reuters-22173 collection we used background knowledge extracted from the 1995 CIA World FactBook, a structured textual document containing information about each of the countries in the world. The information about each country is divided to 6 sections: Geog-

raphy, People, Government, Economy, Communications, and Defense Forces.

As a crude measure of the efficiency of our algorithms, we ran a series of queries using FACT and compared the cpu time (on a 486/50) and the number of associations found for each query. Each query was created by instantiating one of two query templates. The first template (T1) includes a background-knowledge constraint that requires the right-hand side of any found association to be in the Land-Boundaries of the left-hand side: "*Find:* (5/0.1) c1:country+ ⇒ c2:country+ *Where:* c1 ⊆ **CountryGroup**, c2 ⊆ LandBoundaries(c1)". To generate a query **CountryGroup** is replaced in this template by some country organizations defined in the background knowledge. The second query is generated in the same way from an identical template (T2), only without the LandBoundaries constraint: "*Find:* (5/0.1) c1:country+ ⇒ c2:country+ *Where:* c1 ⊆ **CountryGroup**."
Figure 3 shows a graph giving the cpu time it took FACT for evaluating each of the queries for those country organizations whose queries generated from template T2 produced at least 25 associations (the execution time for those giving fewer associations was negligible, and was excluded to avoid a cluttered graph with many nearly-zero values). Figure 4 gives a graph that shows for each country organization the number of associations produced for that query. (Organizations listed on the X axis are ordered identically for both graphs, according to the number of results generated for its instantiation of template T2.) These results show that rather than slowing down the association-discovery process, the specification of background-knowledge constraints actually provides information that is exploited by our discovery algorithms, speeding up the association-discovery process.

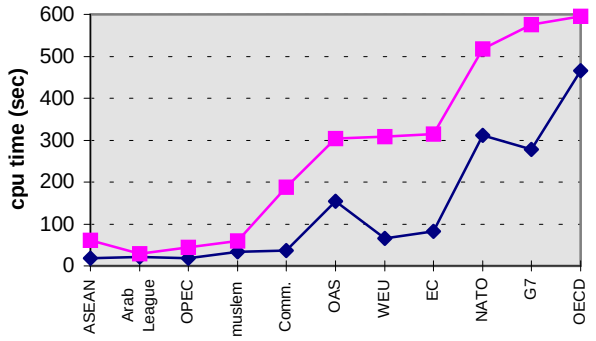

Figure 3 Cpu time found for two sets of queries

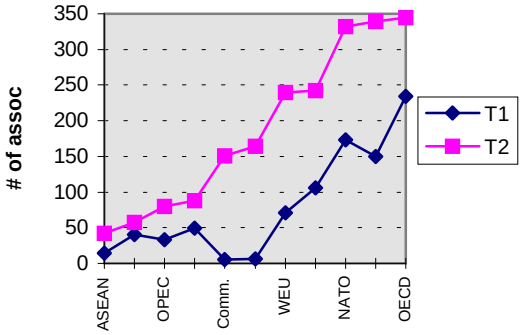

Figure 4 - # of associations found for two sets of queries

Summary

This paper has described the FACT system for knowledge discovery in collections of textual documents, which finds associations amongst the keywords labeling the documents given background knowledge about the keywords and relationships between them. Rather than forcing the user to specify an explicit query expression in some arcane knowledge-discovery query language, FACT presents the user with an easy-to-use graphical interface in which discovery tasks can be specified, with the language providing a well-defined semantics for the discovery actions performed by a user via the interface.

References

[Agrawal and Srikant, 1994] Agrawal A. and Srikant R. Fast algorithms for mining association rules. In *Proceedings of the VLDB Conference*, Santiago, Chile.

[Dagan et al., 1996] Dagan I., Feldman R., and Hirsh H. , Keyword-Based Browsing and Analysis of Large Document Sets. To appear, In *Proceedings of SDAIR96*, Las Vegas, Nevada, April 1996.

[Feldman et al., 1996] Feldman R., Dagan I., and Kloesgen W. Efficient Algorithms for Mining and Manipulating Associations in Texts. To appear, In *Proceedings of EMCSR96*, Vienna, Austria, April 1996.

[Feldman and Dagan, 1995] Feldman R. and Dagan I. KDT - knowledge discovery in texts. In *Proceedings of the First International Conference on Knowledge Discovery(KDD-95)*, August 1995.

[Imielinski, 1995] T. Imielinski. Invited talk. *The First International Conference on Knowledge Discovery(KDD-95).*

[Mannila et al., 1994] Mannila H., Toivonen H., and Verkamo A. Efficient Algorithms for Discovering association rules. In *Proceedings of KDD-94*, pages 181-192.