

From data to knowledge mining

ANA CRISTINA BICHARRA GARCIA, INHAUMA FERRAZ, AND ADRIANA S. VIVACQUA

Laboratório de Documentação Ativa e Design Inteligente, Universidade Federal Fluminense, Fluminense, Brazil

(RECEIVED November 14, 2007; ACCEPTED November 27, 2008)

Abstract

Most past approaches to data mining have been based on association rules. However, the simple application of association rules usually only changes the user's problem from dealing with millions of data points to dealing with thousands of rules. Although this may somewhat reduce the scale of the problem, it is not a completely satisfactory solution. This paper presents a new data mining technique, called knowledge cohesion (KC), which takes into account a domain ontology and the user's interest in exploring certain data sets to extract knowledge, in the form of semantic nets, from large data sets. The KC method has been successfully applied to mine causal relations from oil platform accident reports. In a comparison with association rule techniques for the same domain, KC has shown a significant improvement in the extraction of relevant knowledge, using processing complexity and knowledge manageability as the evaluation criteria.

Keywords: Data Mining; Knowledge Cohesion; Ontology; Sense Making; Text Mining

1. INTRODUCTION

We live in a world of fast generation and distribution of enormous amounts of information. One way of dealing with that is simply to randomly pick a manageable amount of information and react to it. However, this strategy is not well suited to the business world, given that knowledge is crucial to survival in a competitive environment. This creates a need for automated information extraction techniques.

Any engineering field creates artifacts such as products or processes. During an artifact's life cycle a great deal of data is produced, starting with an initial set of specifications and continuing through maintenance reports. Many of these documents are textual. Data management technology has evolved and has provided the means for organizing and storing information. Having information available is a necessary, but not sufficient, condition for learning. People must be able to retrieve and digest information to craft new knowledge from the stored information. Unfortunately, making sense of engineering project data, even for a specific domain area or company, overwhelms human capabilities. Decision-making processes improve whenever successful and unsuccessful cases are understood, revealing flawed decision patterns that should be avoided (Maher & Garza, 1995; Soibelman & Kim, 2000).

Despite the difficulties, the importance of learning from the past has led researchers to carefully look for analytic tech-

niques to handle large amounts of engineering data. In this scenario, the use of data mining (DM) techniques has become an attractive alternative to classic mathematical models when applied to specific tasks such as problem diagnosis and failure prediction (Varde et al., 2007). DM is the process of discovering relevant knowledge from large data repositories (Fayyad et al., 1996). Although this assertion is broadly accepted, making this process work is still a challenge. There are many DM algorithms that differ in the type of information they extract, ranging from association rules to classifications or patterns. One of the most popular DM strategies is the association rules technique (Agrawal & Srikant, 1994), which takes a data set of items, and generates IF–THEN rules based on relations of co-occurrence between elements. Unfortunately, association rule techniques still fail to produce a manageable amount of relevant information, usually producing too many results, which include much redundancy.

A large amount of research has been done on DM, most of it applied to classification tasks, such as identifying patterns of good and bad maintenance strategies, knowledge transfer from maintenance to design (Jadhav et al., 2007), identifying calibration models in building structural components (Saitta et al., 2005), and eliciting the customer's project requirements (Lo et al., 2007).

Refrat et al. (2004) investigated the benefits of using DM techniques to assist facility managers in the identification of potential building maintenance problems given data from building air handling units and thermostatic mixing valves. Maintenance data were organized in a database. Five thousand

Reprint requests to: Ana Cristina Bicharra Garcia, Rua Passo da Pátria, 156, Bloco E, sl. 326, São Domingos Niterói, RJ, CEP 24210-240, Brazil.
E-mail: bicharra@ic.uff.br

work orders for a specific building, collected during a period of approximately 2 years, were used. Classic statistical methods and different DM techniques such as association rules, clustering algorithms, and decision trees were applied. They concluded that DM techniques support the identification of critical cost issues, as well as the creation of predictive plans.

Jadhav et al. (2007) used classification DM techniques to identify the causes of high severity events that lead to major financial losses in aeroengine artifacts. They focused on providing answers to a predefined set of maintenance questions that could improve posterior design. Information was organized in a data warehouse, which makes the mining process easier because such data contains no noise.

Saitta et al. (2005) investigated an innovative use of DM for system identification, also known as model calibration, in structural engineering. Classic approaches (Modak et al., 2002; Koh et al., 2003) have been based on calculations of structural stiffness coefficients to predict structural behavior. They took a different approach, using DM techniques to reveal properties of good models that could explain observations of structural behavior and create models from these observations.

Chen et al. (2003) claimed that product design would benefit from prior customer feedback and marketing analysis. They investigated the use of neural networks to elicit user requirements for the design of wooden golf clubs. Their approach requires an input matrix of previously recorded graded imposed constructs that reflect the perception of the product, with an emphasis on requirements originating from multicultural differences among users.

1.1. The problem

The main challenge for knowledge engineers when using the association rules technique is tuning syntactic metrics, such as support and confidence. These parameters prune the result set, reducing it to a manageable size (Silberschatz & Tuzhilin, 1995). However, there might be relevant data patterns with low statistical significance that will remain hidden. Relaxing the pruning task to allow these patterns to appear will often lead to an unmanageable result set, and a strict cut might cause relevant knowledge to be discarded from the result set. This is the well-established trade-off analysis of balancing precision and recall. We believe that a semantic-oriented pruning technique that increases the number of relevant extracted knowledge items while decreasing irrelevant outcomes is in order.

1.2. The knowledge cohesion (KC) hypothesis

We use the association rules DM technique (AC) as a research baseline, because of its large acceptance by the artificial intelligence (AI) community. As in any AC algorithm, frequent data ensembles become seeds for finding relevant patterns.

However, instead of finding IF–THEN rules that exceed a certain confidence threshold, we take a semantic approach, generating instantiated semantic nets as the output of the mining process. We claim that the inclusion of available domain knowledge and emphasis on the relations leads to a more manageable result set and more significant mined patterns. Our premises are the following:

1. In general, before analyzing data from a domain, knowledge engineers investigated the field by looking at documents or interviewing experts in order to build up a minimum amount of domain understanding that helps interpret results. For example, soda is a refrigerant in a supermarket domain.
2. The knowledge engineer may want to steer the search toward specific combinations, for instance, an *itemset*, which contains the elements *people* and *major failure* (not necessarily a frequent pair of information items in the source data).

Based on these two premises, we created a new DM technique called KC. The KC technique is dependent on a domain ontology: the better the ontology, the more efficient the technique. KC has two components: from the domain ontology, we calculate the semantic distance (SD) between two elements, and from the user's preferences and own view of the domain, we obtain the relevance assessment (RA) of the item pairs.

SD and RA are the two domain-dependent components that guide the KC method during *itemset* selection phase. In addition, instead of creating IF–THEN rules, the KC method produces instantiated semantic nets as its outcome, which facilitates the understanding of the results.

1.3. Paper outline

This paper continues by providing background information on domain ontologies, followed by a detailed description of the KC approach. A scenario using a syntactic database illustrates the working of the KC method, and a comparative analysis provides support for our claim that KC brings a considerable advantage to DM techniques. Data results from applying KC to oil field accident reports' data set and a performance comparison to association rule techniques are presented in the discussion section.

2. BACKGROUND KNOWLEDGE

In this section we present the background necessary to understand our proposed DM technique. Because we consider the existence of a formal domain description, we present the concept of domain ontology and introduce the AC DM technique.

2.1. Domain ontology

Ontology is a broad concept with different meanings. Philosophers use the term to refer to the general study of the nature of being. Since the late 1980s, computer scientists have applied the concept of ontology to knowledge acquisition fields. Thus, in his seminal work in this area, Gruber (1993) refers to ontology as a specification of a conceptualization. This definition narrows the representation power of a domain ontology because it neglects the fact that an ontology represents a group's view of a domain, fits a set of tasks, and may change with time and use, as in any knowledge representation (Shirky, 2007). In our work we adopt a more specific definition to emphasize these three issues. We define ontology as a description of a domain negotiated by a community that will use it for specific purposes such as computational reuse or human shared understanding. The description, which may change over time, includes a list of terms and the relations between them. Each term contains a name, a list of pertinence rules, exceptions, and comments. A relation contains a name, the number of terms included in the relationship (the relation's *arity*), and the prescribed behavior resulting from applying the relation to the terms.

Ontology has long been a means for sharing domain data, information, and knowledge among system development group members. For example, Lenat et al. (1990) tried to build a single ontology for everything, but at that time, they could not envision the challenge this task presents, as it involves keeping up to date knowledge that, by definition, evolves with time.

Although ontology was at first primarily associated with knowledge-based systems (KBS) development, currently, it has become embedded in many other systems. Its importance soared as it was increasingly used in Web information retrieval systems (Etzkorn, 2006; Li & Ramani, 2007). However, ontology is not a panacea, for it can only represent a

group view of a domain to be used for predefined tasks. To determine the applicability of an ontology we need evaluation metrics (Sirim et al., 2003), such as cohesion, which measures the modular relatedness between concepts of a domain ontology (Yao, 2005).

In addition to building systems, ontologies have been used for evaluating software quality, based on the degree of coupling among ontologies used in the software. Orme et al. (2006) claim that the more coupled the ontologies, the harder it will be to maintain the software bug-free.

Generally, an ontology has a graphical representation as illustrated in Figure 1, which describes a marketplace. An interpretation of the figure would be,

A market basket contains groceries and clothing. Clothing may be formal or sports clothing, and tennis shoes are a type of sports shoes, which are a type of sports clothing. Beverages and food are groceries, and appetizer and salad are types of food. Olives are a type of appetizer. Olives are a type of appetizer.

Although we acknowledge that an ontology may get out of date because of changes in the world it represents, we believe that for engineering domains it satisfactorily represents the meaning of things in a domain for at least a 3-year time frame. Consequently, a domain ontology may function as additional knowledge in the DM process, providing a good basis to guide the search process. However, as in all heuristic search methods, the hints provided by the knowledge within a domain ontology accelerate the process, but might also prune relevant knowledge, depending on its representativeness.

Given that an ontology can be represented as a graph in a Euclidian world, there is always a measurable distance between two directly or indirectly connected terms. Intuitively, we can infer whether a pair of terms is closer than another pair

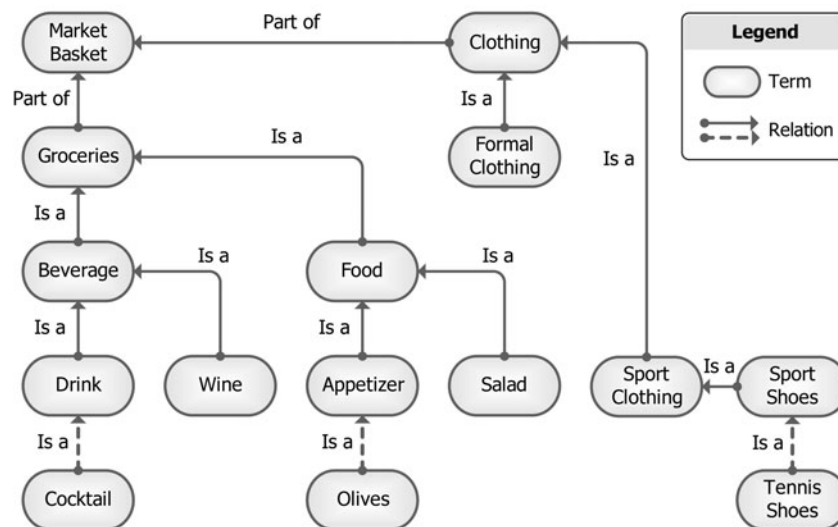


Fig. 1. A sample ontology for the marketplace.

of terms. Because the terms of the graph are connected by semantic links, we assume it is possible to at least partially quantify a semantic link, even when it might not be a precise quantification.

Our assumption is based on observations made over 10 years of building ontologies for domains related to petroleum engineering, such as oil process plant design, oil pumping equipment failure detection, and oil production failure diagnosis. Our process of building ontologies consists of gathering concepts, eliciting their meanings, negotiating a proper terminology, negotiating the relation among concepts, breaking or unifying concepts, and clustering or splitting groups of concepts. Concept relations are carefully designed; there are no random assignments. We only assign an *is-a* relationship to two concepts when our whole group agrees that the concepts share most of the properties expected by the meaning of the *is-a* relationship.

For example, in the domain description illustrated in Figure 1, we can intuitively consider that the terms “cocktail” and “olive” are closer than the terms “cocktail” and “shoes.” The challenge is thus to create an objective metric to estimate the SD between the two terms. Rather than defining a specific measurement, we want to be able to compare groups of concepts and discriminate closer pairs from more distant ones.

Building a domain ontology requires a great deal of effort to ensure that all the concepts needed for the envisioned task are properly described. “Properly” is a qualitative and subjective metric that transfers the responsibility for verifying the ontology quality to the group of knowledge engineers who developed it. Kramer and Kaidl (2004) proposed two quantitative metrics, cohesion and coupling, to evaluate the quality of a domain ontology based on a set of production rules that reflects the knowledge encoded in a KBS. The set of properties, or slots, describing a domain concept constitute a frame. A description is cohesive whenever the degree of membership between frames is a function of the number of common properties. Kramer and Kaidl claim that KBS rules can be used to measure cohesion of a frame and, indirectly, the quality of a domain ontology. According to them, the cohesion of a frame is a function of the number of its slots that are encoded in the same the KBS rules. Their proposal is to use knowledge rules to infer domain ontology quality. We adopt a similar perspective, but a different approach. We propose to use a domain ontology to evaluate the quality of knowledge rules elicited by DM techniques.

2.2. AC DM technique

A database of transactions D is a set of n transactions $\{t_1, \dots, t_n\}$. The t_i values are typical vectors of items of the form (t_{i1}, \dots, t_{im}) , whose components are discrete values, called features or attributes. Thus, an item t_{ij} denotes the values of the j th feature T_j of the transaction t_i . In the following description, the i subscript will be left out when implied by the context.

A set of items is referred to as an *itemset* (or pattern). An *itemset* that contains k items is a k -*itemset*. The set $\{X, Y\}$ represents a 2-*itemset*. The frequency of occurrence of an *itemset* is the number of records that contains the *itemset*. This frequency is also known as the support of an *itemset*. An *itemset* that has a predefined minimum support (*itemset* selection threshold) becomes a relevant candidate pattern. The number of records required for an *itemset* to satisfy the support threshold varies according to the knowledge engineer’s objectives. The objective might be to study frequent patterns (e.g., in more than 60% of the data collection) or the objective might be to study high impact patterns that, even though they may happen only few times, are responsible for major losses or gains in a company’s investment. High support values prune recall, but lessen precision. Knowledge engineers select support values considering the tradeoff between retrieving a human-manageable number of patterns to be analyzed and allowing relevant patterns emerge.

Although support is a metric to measure the frequency of an *itemset* in the database, confidence is a metric to measure the degree of confidence that a subset will be present in an *itemset* when some other subset has been found. Support and confidence metrics form the basis of the AC DM technique.

An association rule R is an expression $X \rightarrow Y$, where X (rule left-hand side or rule antecedent) and Y (rule right-hand side or rule consequent) are subsets of items.

Extracting an association rule $X \rightarrow Y$ means that the *itemset* containing only the subsets X and Y is frequent enough to be a candidate rule. In addition, it means that there is a high degree of confidence in inferring that whenever X is found in a transaction there is also a high probability that Y will appear. Rule confidence and support are represented by Eqs. (1) and (2), respectively, where n_{XY} is the number of transactions in D containing X and Y , n_X is the number of transactions in D containing only X , and n is the number of transactions in D .

$$\text{support}(R) = \frac{n_{XY}}{n}, \quad (1)$$

$$\text{confidence}(R) = \frac{n_{XY}}{n_X}. \quad (2)$$

Generation of AC is a common approach for mining data sets because it is easy to understand what a rule is. Unfortunately, this approach usually only changes the problem from dealing with huge amounts of data to dealing with an unmanageably large number of rules. Results that include more than 100,000 rules are not uncommon.

Filtering the outcomes is the main approach to address this problem (Lenca et al., 2007). It represents a postprocessing step to be executed on top of the mining method’s results. Metrics such as degree of novelty (Lavrac et al., 1999) and information entropy (Lavrac et al. 1999; Dhar et al., 2000) qualify the outcomes and help filter what should be the final result.

We propose a different DM approach. Like AC, it starts with *itemssets* as the working elements to create outcomes. However, instead of generating IF–THEN rules based on conditional probability theory, our method generates semantic networks layered over a domain ontology. Search is guided by the knowledge in the ontology. New patterns are anchored in old known knowledge through a display that overlays the rules and the ontology, and therefore users understand the results more easily.

3. THE KC DM TECHNIQUE

Our approach is based on KC. The KC algorithm is an extension of the AC technique. It is an empirical method that relies on the existence of a domain ontology to forget two selection metrics, SD and RA, to substitute for the confidence metric.

3.1. The rationale for the KC method

The KC method stems from our intuition that existing knowledge should be taken into account when searching for new knowledge in large data sets, because either new knowledge will make more sense when anchored on old knowledge, or what is already known can be removed from the search results. To find a way to transform this intuition into an objective metric, we started observing how knowledge engineers (KEs) build domain ontologies. The selected concepts are domain specific, but the relationships between any two concepts are general (e.g., *Is-a*, *Part-Of*, *Is-an-attribute-of*, *Causes*, *Time-Follows*, and *Space-Follows*). The relationship between two concepts is carefully chosen and reflects the way KEs envision the domain. Whenever two concepts are related by an *Is-a* relationship, it is expected there is a set–subset relationship. It is also expected the two concepts share most of the properties that characterize both concepts. For example, when one says that *dog Is-a mammal*, it is expected that *dog* inherits all identification properties of mammals such as the way it reproduces and breathes. This close identity relationship suggests any mined knowledge containing to concepts connected by an *Is-a* relationship has a low probability of aggregating new knowledge to the analysis.

Following the same reasoning, we verified the semantics of the *Part-of* relationship. What does it mean to say that engine 222 is *Part-of* the Toyota Corolla 2008? The composition relation generally means that the whole needs its parts to be itself, that is, there would be no Toyota Corolla 2008 without engine 222. However, engine 222 may be a part of many other cars. Consequently, a mined pattern that brings up two concepts that are bonded by a *Part-of* relationship brings up more information than it would if the concepts were only related by an *Is-a* relationship, but there is still a low degree of novelty because the existence of the Toyota would imply the specific engine.

In contrast, there are other relationships, such as *Is-an-attribute-of*, that create less expectation of related concepts

appearing. For example, being cheap may be an attribute of popular cars, but the concept “being cheap” may relate to a great variety of other things as well. Consequently, a mined pattern containing two concepts related by the *Is-an-attribute-of* relationship has a better chance of being more innovative. We did the same type of qualitative analysis on the *Causes* and *Follows* relationships.

We translate this intuition about the indicators of relationship novelty into an SD calculation. During DM, KEs might also have intuitions about specific correlated concepts they might want to check or avoid. To allow the KEs to direct the search, we have also created RA evaluation criteria.

3.2. SD

In the domain of information systems, SD is a measurement of word similarity. This measure has been widely used for document retrieval on the Web. Calculation of SD begins by considering how the distance between two adjacent concepts reflects their similarity (Roddick Hornsby & de Vries, 2003). The world is represented as a directed graph of terms in which each node must have a label reflecting its meaning. Implicitly, a domain is represented as a finite state machine in which there is a cost to navigate from one concept to another. SD reflects how related two terms are in a specific domain, and how much information one term aggregates to another when they appear together.

Instead of assigning a numeric value to links between specific terms, we take a broader approach. We consider an ontology of the domain, represented as a semantic net in which nodes and links have meaning. We assign to each semantic link a numeric value that reflects the correlation between two adjacent nodes. Note that the numeric value is neither a measure of similarity between two adjacent concepts nor a reflection of the relation between two specific concepts. Rather, the value reflects the relationship weight when connecting two terms. Because the values assigned to the correlation between nodes are never precise, we decided to lose accuracy to gain flexibility to incorporate more relationships and make the method more applicable in actual data sets.

In addition, as terms are connected through semantic links, the distance between two terms may vary depending on the relationship(s) connecting them. Each semantic link connecting terms has an associated relationship weight. The numeric value reflects the amount of information a semantic link brings to the understanding of a domain. The closer the information nodes, the smaller the numeric values that map the semantic link.

The semantic relation is directional; that is, when one decides to use the relation *Is-a* between two terms, there is a clear understanding about which term is the subset and which is the superset. This is an important observation because when evaluating the SD between two terms, the distance may be different depending on the direction of the path. SD has the following properties:

- *Asymmetry*: $SD(A, B) \neq SD(B, A)$. The SD between terms A and B is generally different from B to A . For example, take the relation *Is-a* (cat, mammal). No additional information is added to *mammal* when one already knows that the element in question is a cat. However, knowing that the element is a cat when all that is known is that it is a mammal adds a restriction on what subset of mammal is being referred to.
- *Transitivity*: $SD(A, B) \wedge SD(B, C) \rightarrow SD(A, C)$. This behavior is inherited from the fact that the ontology is represented by a graph.
- *Reflexivity*: $SD(A, A)$. This behavior denotes a relation between instances of the same concept. For instance, consider a term called *Cause-of-Accident*. There may be an immediate cause, but during investigation we discover a sequence of events from which one causes the other, such as “the pump burned” because “the oil flow was too low” because “the main oil vessel was empty” because “the automatic pump control did not work properly” because “equipment maintenance was not performed as scheduled.” The events are semantically close examples of things that should have not happened, but they might have occurred in different times, which may explain an SD between any the two adjacent instances.

We have created a provisional weight assignment for the most common semantic links, illustrated in Table 1. The assigned values are a subjective measurement that reflects the expected behavior of the relationship.

- *Is-a* relation denotes the set–subset relationship. The relation connects elements that are closely related. It includes the inheritance behavior, through which a subset inherits the attributes and behaviors of its correlated set. SD between subset and set is extremely small, mostly insignificant on the order of 0.5. We have considered the *Is-a* relation as the baseline for which SD equals zero. However, following the same relation in the opposite direction provides additional information, namely, that they are not the same. It reflects a restriction on the overall set. Although the terms are closely related we assigned a small extra point, as illustrated in Table 1, to signal the semantic nuance.

Table 1. *Semantic distances*

Relation	$SD(A, B)$	$SD(B, A)$	$SD(A, A)$
Is-a	0	0.5	0
Part-of	1.5	1.0	0
Is-attribute	2.5	2.0	0
Causes	2.5	2.5	0.5
Precedes (time)	2.5	2.5	0
Precedes (space)	2.5	2.5	0
Others	3.0	3.0	0

- *Part-of* relation denotes composition. If A and B are parts of C , then C is only present if A and B are present. The elements are not from the same semantic, but are fairly close. We consider that when the whole is present, the parts are present as well. As a result, the additional information, provided by the parts, is small. However, the whole and its parts are of a different nature; consequently, the SD between them must be greater than the one associated with the *Is-a* relationship. For example, knowing that there is a car reminds us that there must be a wheel. In contrast, knowing that there is a wheel does not necessarily lead to the image of a car.
- *Is-attribute-of* relation denotes properties of an object, such as color or price of a car. Properties describe objects or concepts of a domain. For example, it is expected that a car object will have properties that describe it, such as its color, price, and brand. However, color is a concept related to objects other than cars.
- *Cause–Effect* relation denotes the causality relationship between two terms. In general, the objects are from different semantic classes such as sickness and symptoms.
- *Precedence* relationship: precedence, both temporal and spatial, relates terms from the same domain but happening in a different time or place. They are semantically close but must reflect the temporal or spatial distance.
- *Other* relations between objects are considered the upper value. As we adjust the model, other relations may demonstrate enough importance to receive special treatment. For now, relations different from the ones mentioned here are treated as *others*.

After transforming the semantic relationships into weights on graph relationships, finding the correlation between two terms implies applying a function that combines all weights in the path from a starting node to a target node. To simplify, we consider this function as a simple sum of weights. This means we are considering the Euclidean distance between two nodes in the graph.

3.3. RA

Another important concept in our work is the user’s current interests. The user may query the database to verify the frequency of specific events or even the existence of a specific correlation between events. Any on-line analytical processing (OLAP) technique can satisfy this need.

We have used explicit targets with a different perspective. Unlike an OLAP search in a database, we proceed by finding (or removing) patterns that contain significant evidence of the correlations selected. This RA impacts the mining process itself because it guides the search.

User-provided target items increase the importance of patterns that contain target elements during the mining process, thus increasing the chances of emergence of infrequent patterns that may be interesting to the user.

Table 2. Example data set

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8
X	Y	A	B	Z			K
A	B	C					
X	Y	C	D	Z	K		
B	C						
X	Y	C	F	Z			K
X	Y	F		Z			K
X	Y	G		Z			K
X	Y			Z			K
A	B						
A	C	D					
Y	K						

For example, suppose we have the data set shown in Table 2. A user might query this data set to obtain the relevance of the correlation between X and Y. The answer reflects the frequency with which X and Y appear together. The OLAP approach takes care of this need. In contrast, users might be misled by the answer. As illustrated in Table 2, X and Y appear in the context of X–Y–Z–K. In an OLAP intervention, the context in which the pattern X–Y appears is lost; that is, the OLAP answer does not uncover the fact that X–Y pattern may be present only within a macro pattern X–Y–Z–K.

Using RA, the pattern X–Y–Z–K will be returned. RA is a qualitative measurement assigned to pairs of concepts. We consider three levels of RA:

1. Low RA (1) means that the user is not interested in patterns that contain this pair of concepts.
2. Average RA (2) is the default measure for any pair of concepts.
3. High RA (4) means that the user is interested in patterns containing this pair of concepts.

We have mapped this qualitative measurement to the numeric values presented inside the parentheses.

3.4. KC calculation algorithm

As explained above, KC is based on the knowledge of the domain, represented by domain ontology, and the user’s estimations of the relevance assessment of co-occurring concepts. The first provides the degree of soundness and surprise that a given pattern may generate, whereas the second reflects the refinements in the domain representation caused by the different perspectives different analysts may have about the world.

KC takes the domain knowledge as input to calculate the SD between terms described in the ontology. Initially, relation weights are as shown in Table 1. The SD between any two terms equals the Euclidian distance considering the minimum path connecting them. As domain understanding grows, the weights of the relation might change.

In our first implementation, a Dijkstra method is used to calculate the minimum path between any two terms. An $n \times n$ matrix is created to store distances between terms (where n is the number of terms). Each cell of this matrix contains the SD between the term in the line and the term in the column, that is, $cell_{ij}$ contains the SD between the term described in matrix line $_i$ and the term described in matrix column $_j$.

The second piece of input information guiding the KC mining process is the degree of relevance assessment. Any pair of terms is considered of medium relevance assessment unless a user says otherwise. For instance, assume biologists dealing with events that describe feeding behaviors in animals. It might be the case that the correlation between date and place is known and obvious for them. Consequently, it seems reasonable that they will want to minimize their attention to patterns that include date–time relationship. This lack of interest, when disclosed, helps filter irrelevant patterns. On the other hand, it becomes possible for low-frequency term correlations to appear, because of users’ assignment of high relevance assessment to them.

KC starts by generating *itemset* candidates from the data set. As in any Apriori-based algorithm (Agrawal & Srikant, 1994), it considers only *itemsets* with enough support, that is, *itemsets* that appear with a certain frequency in the data set. However, frequency is only used to provide an initial cut. Frequent *itemsets* compose an unordered list of candidate solutions. Each candidate *itemset* is further analyzed using the KC metric. Whenever an *itemset* KC is greater than the minimum KC, the candidate *itemset* is considered a sound piece of knowledge to be displayed using the ontology as a mask that provides extra information, such as the relation between terms in the *itemset*. We define the KC of an *itemset* as shown in Eq. (3):

$$\begin{aligned}
 & itemset = term_1, term_2, term_3, \dots, term_n. \\
 & length(itemset) = n \\
 & [C] = C_{length(itemset)}^2 \\
 & KC_{(itemset)} = \frac{\log(length(itemset))}{[C]} \\
 & \quad \times \sum_i^{[C]} \frac{RA(ItemPair_i)}{SD(ItemPair_i)} \tag{3}
 \end{aligned}$$

where

- *Itemset* is the set of items that together might form a pattern.
- $Length(itemset)$ is the number of items in an *itemset*.
- $[C]$ is the number of combinations containing two different items composing the *itemset*. Each possible combination is called *ItemPair*.
- $ItemPair_i$ is one of the possible combination of two elements forming the *itemset*.
- $SD(ItemPair_i)$ is the SD between the two elements in $ItemPair_i$.

- $RA(\text{ItemPair}_i)$ is the attractiveness degree assigned by users to the ItemPair_i .
- $KC(\text{itemset})$ is the KC degree of the itemset .

The number of elements is included in the formula to reflect our preference for larger itemsets ; that is, we prefer more specific to more general knowledge inferences. Instead of directly using the number of elements, we use $\log(\text{number of elements})$ to create a scale from 0 to 1 in KC. KC is an empirical formula that reflects known knowledge from the world. The results have been promising, but we are still looking for a more robust formulation.

For example,

$$\begin{aligned} \text{itemset} &= \{A, B, C\} \\ \text{length}(\text{itemset}) &= 3 \\ [C] &= C_3^2 = 3, \quad \text{that is, } 3 \text{ pairs}\{AB, AC, BC\} \\ KC(ABC) &= \frac{\log 3}{3} \left\{ \frac{RA(AB)}{SD(AB)} + \frac{RA(BC)}{SD(BC)} + \frac{RA(AC)}{SD(AC)} \right\} \end{aligned}$$

Thus, the KC algorithm is composed of three steps:

1. Generate candidate patterns using the itemset support metric.
2. For each candidate itemset ,
 - a. calculate $KC(\text{itemset})$.
 - b. If $KC(\text{itemset}) > \min KC$,
 - i. itemset is the solution.
3. For each itemset solution,
 - a. create a solution pattern.
 - i. Place the itemset solution into the semantic net created by the domain ontology

This method allows low frequency patterns to emerge because of their KC values, which in turn enables important outliers to stand out in a crowded data set.

To evaluate the quality of our metrics, we used the following set of conditions prescribed for a good metric (Kitchenham et al., 1995):

- *Property validity* asks whether it is possible to observe the property in a deterministic way. We are observing pairs of items in a given itemset , so the answer is yes. We are accounting for the distance between pairs of items considering their relative position within the domain ontology. All items are members of the ontology.
- *Instrumental validity* asks whether there is a method that describes the way the measurement should be made. Ours is an objective method for accounting for the measurement.
- *Protocol validity* asks if the set of steps described to measure the attribute prevents double counting. In our method, there is no possible double counting.

4. APPLICATION DOMAIN: KNOWLEDGE EXTRACTION FROM OIL FIELD ACCIDENT REPORTS

In the Brazilian petroleum industry most drilling and production are offshore. The offshore oil fields are being exploited by 64 oil platforms, operated by nearly 40,000 workers. The large number of workers and the nature of oil platform operation make this an activity of high economic, environmental, and human risk. A simplified ontology mapping this domain is shown in Figure 2.

The description associated with this ontology reads as follows:

An anomaly occurs at a place, in a specific moment, involving people, during an activity, which is part of a process. The anomaly may involve equipment and has a specific cause, which may involve an external agent. This anomaly has impacts that might be human-related, economic, environmental, or any combination of these three, and may have potential impacts.

There are strict safety regulations for platform operation. One of the regulations requires the existence of a method to register unexpected events (usually classified as accidents, incidents, or deviations), which collects not only information about the people involved and consequences for the processing unit, but also the actions taken to solve the problem and prevent recurrence. The organizational process for fulfilling this requirement is to create textual documents containing this information, called anomaly handling reports. These reports are generated every time an unexpected event happens (e.g., fire, equipment failure, injuries, nonobservation of procedures). The reports are electronic files created and stored in a public location, available to all company employees. The reports are organized in nested folders, where the folder's name indicates the subject: an NY subfolder of the nonconformity folder contains reports on nonconformities events that occurred in the NY unit.

An analysis of this data is important to higher level management, as it enables detection of recurring problems and any existing actions that might mitigate them. Reports are generated through an automated system that guides the reporting process. To perform statistical analysis, identify the real cause of accidents, and determine the correlations between platform measurements and accidents, the company usually hires experts who carefully read and interpret each report and try to consolidate the information within them. If the number of reports is up to 100, a human expert can easily take care of the job. However, because the number of reports is already over 10,000 and increasing at a rate of dozens a week, extracting meaning from them requires an automated approach. In this context, we created the DMRisco system, which applies both the KC and Apriori methods to deal with the problem of extracting information from these data sets.

DMRisco is a system to mine textual data. It relies on a domain ontology and a manual tagging process that maps the

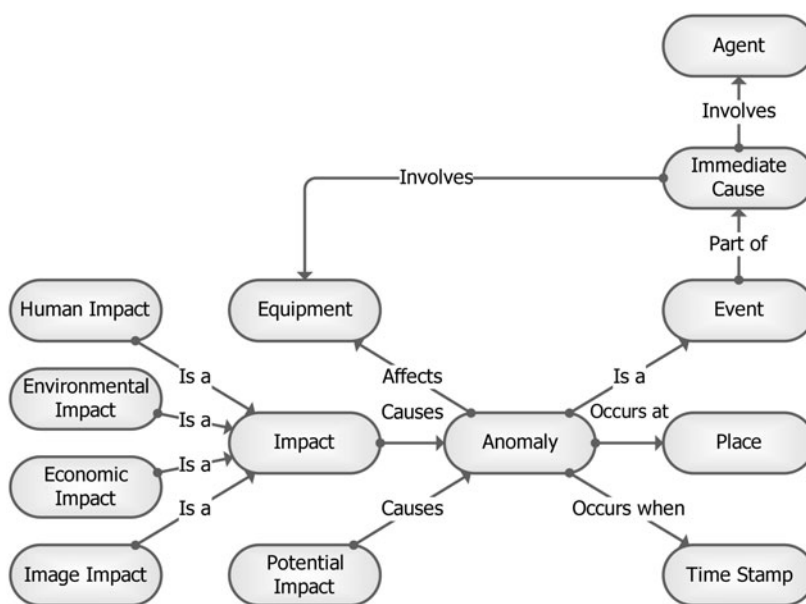


Fig. 2. The oil platform anomaly ontology (simplified).

anomaly handling report onto the domain ontology concepts. Therefore, each report is recorded as an *itemset*, and each item of this *itemset* is an instance of a concept in the domain ontology. DM is then performed on the *itemsets*.

An accident report is a document with multiple authors. The person involved in the accident must report it. Then a team of experts investigate the case and write down their analysis of the accident, possible causes, and suggestions for corrective actions. Upper level managers read the accident description and analysis, and decide what to do, who will be in charge of doing it, and who will be in charge of checking the effectiveness of the actions. This information is also written in the report. Finally, the person in charge of checking the effectiveness of the actions closes the report by writing down the results of the actions. Although each report may seem to be about a singular event, when accidents keep happening in very similar ways, DM might reveal patterns.

For the pilot study, we asked a group of 10 users to tag 1500 accident reports from a 2-year period. The time allotted for this task was 1 week, and at the end of the week, they had finished 932 reports. Although the task was straightforward, the users had trouble reconstituting parts of the reports, and were unable to completely reconstitute 12 reports. In contrast, users found the tagging task a good opportunity to gather important information that was missing from the reports.

The tagging task was assigned to trained people who worked in pairs. Each pair was composed of an expert in oil production and an expert in safety procedure. Our project benefited from an organizational decision to change the way in which accidents are reported by having specifically trained people in charge of the task.

Tagging required a domain ontology. The accident analysis ontology was collaboratively constructed by a group of ex-

perts who understood not only the domain but also the expected outcome of the analysis. They participated in a negotiation process over concepts terminology that lasted over 3 months to build a consolidated ontology for data extraction. The experts were midlevel managers, who will be using the system and know what types of interpretations they want to get out of it. To enable verification of the KC algorithm, the reports were imported to the system and tagged with the corresponding ontological elements before processing.

It is important to note that there are three types of experts involved in our project:

1. the group of experts that generate the ontology,
2. the group of people that tag accident reports, and
3. the group of experts that mine the tagged data, applying KC to the tagged data set of accident reports.

The first group of experts is composed of senior consultants who have been in the company enough time to understand the domain and the different perspectives that exist in the community. The second group is composed of people who work in the operational line. They have a great deal of experience analyzing accidents and they received training on tagging. The last group of experts is composed of decision makers or their direct assistants. They are the ones who will decide which actions should be taken after investigating accident patterns. They are the final users of KC and DMRisco.

5. EXPLAINING KC'S BEHAVIOR USING A SYNTHETIC DATA SET

As explained in Section 4, KC method works on a matrix in which lines represent different transactions and columns

represent the attributes describing the transaction. Each cell contains a value of the attribute for a given transaction. Generally, domain data are stored in databases or data sets that must be translated into matrix representations for KC to work.

In our application domain, the database is generated through the tagging of accident reports. Because tagging accident reports requires people to interpret and think about the facts before actually registering the story, we believe that the number of mistakes will be low, and that existing mistakes will have insignificant impact on the final result, because the database contains a large number of cases. Of course, DM results will only be as good as the data itself.

To illustrate our method, we present an example with few attributes. Consider a database that registers accidents, as illustrated in Table 3. Each transaction (line) represents an accident story. Each cell in the matrix means the value of the attribute indicated by the column name for the transaction (accident report) indicated by the line. Each line has a unique identifier. Instead of using attribute values, we simplified this example, by creating columns for each possible value. This means cells marked with “1” indicate the presence of that attribute (column name) in the specified record line, “0” means the negation of the attribute value and “empty” means there is no information on whether or not the value exists. To further simplify, we have eliminated columns that have no records for them. For instance, we included Wednesday afternoon, but we did not include Wednesday morning or any other weekday because there was no record of them in the data set.

Following our example, record 1 is {Plant Shutdown, __, __, John Mayer, Control Room, __}. Tables 4 and 5 show, respectively, the relevance degree and SD measures between attributes of the database showed in Table 3.

Both tables are computational representations of domain knowledge that should be encoded by domain experts before they can be used in our method. The measurements are represented by asymmetric matrices, because SD between attributes is not a symmetric measure. In our discussion, we apply the convention in which A to F represent plant shutdown,

Table 3. Sample database

A Plant Shutdown (Impact)	B Pump (Equip.)	C Polluted Water Disposal (Anomaly)	D John Mayer (Agent)	E Control Room (Place)	F Wed. Afternoon (Time Stamp)
1			1	1	
	1			1	1
1	1	1	1	1	1
	1			1	1
1	1	1	1	1	1
	1			1	1
1	1	1	1	1	1
	1			1	1

Table 4. Relevance assessment of elements of the ontology

	A	B	C	D	E	F
A	—	2	4	4	2	2
B		—	4	1	2	1
C			—	4	4	4
D				—	2	1
E					—	2
F						—

pump, polluted water disposal, John Mayer, control room, and Wednesday afternoon, respectively. Considering this representation, all possible *itemsets* that combine all items shown in Table 3, and their respective support values, are shown in Table 6. Highlighted *itemsets* have support greater than the minimum support, which was set at 0.5.

The rules generated using Apriori, with minimum support equal to 0.5, are shown in Table 7, with their respective confidence rule measure.

The *itemsets* generated using the KC algorithm, also with minimum support equal to 0.5, are shown in Table 8, with their respective KC metric value. If we consider the minimum KC value 0.5, only *itemsets* AE and ADE are considered valid by the KC algorithm. Because ADE contains AE, KC outputs the graph of ADE, which is equivalent to Plant Shutdown, John Mayer, Control Room.

When the distance and relevance measurements are adjusted correctly, the number of KC-based results, represented by semantic nets, is lower than the number of Apriori-based results, represented by association rule, as seen in the previous example. If we consider the minimum KC value to be equal to 0.5, only *itemsets* ACE, ABC, and ACF, are returned by the KC algorithm.

KC outputs a graph combining all three possible answers, containing the attributes plant shutdown, pump, polluted water disposal, and control room as a semantic net, as shown in Figure 3.

Considering the number of frequent *itemsets* that returned, 30 *itemsets* were returned by the Apriori (Agrawal & Srikant, 1994) association rule DM technique (shown in Table 8) versus only three *itemsets* by KC. Furthermore, *itemsets* ACE, ABC, and ACF are much more semantically relevant for anomaly detection than *itemset* BEF or DEF. When looking at their contents,

Table 5. Semantic distances according to the domain ontology

	A	B	C	D	E	F
A	—	2	5	4	2	2
B		—	1	2	2	2
C			—	3	1	1
D				—	4	4
E					—	2
F						—

Table 6. Itemset support measures for at most three elements

No. Items	Itemsets				
2	AB	AC	AD	AE	AF
	0, 1	0, 4	0, 2	0, 1	0, 1
	BC	BD	BE	BF	CD
	0, 4	0, 1	0, 1	0, 1	0, 3
	CE	CF	DE	DF	EF
3	0, 4	0, 4	0, 1	0, 0	0, 1
	ABC	ABD	ABE	ABF	ACD
	0, 5	0, 2	0, 2	0, 2	0, 4
	ACE	ACF	ADE	ADF	AEF
	0, 5	0, 5	0, 2	0, 2	0, 2
	BCD	BCE	BCF	BDE	BDF
	0, 4	0, 4	0, 4	0, 1	0, 1
	BEF	CDE	CDF	CEF	DEF
	0, 2	0, 4	0, 4	0, 4	0, 1

- ACE ≡ plant shutdown (impact), polluted water disposal (anomaly), control room (place);
- ABC ≡ plant shutdown (impact), pump (equipment), polluted water disposal (anomaly);
- ACF ≡ Plant shutdown (impact), polluted water disposal (anomaly), Wednesday afternoon (time stamp);
- BEF ≡ pump (equipment), control room (place), Wednesday afternoon (time stamp); and
- DEF ≡ John Mayer (agent), control room (localization), Wednesday afternoon (time stamp).

Because our initial results with this example data set are extremely encouraging, we are extending our research with experiments using the complete accident report data collection as well as XML file repositories.

6. COMPARISON BETWEEN APRIORI AND KC

In this section we present a comparison between KC and Apriori, based on real data extracted from the domain of oil processing.

To reiterate, our original hypothesis was that our semantically driven KC method would yield better results than a regular as-

Table 7. Rules generated by Apriori’s algorithm with minimum support of 0.5 and their respective confidence measurements

A → D	D → A	A → E	E → A	B → E
1.00	0.63	1.00	0.50	1.00
E → B	D → E	E → D	D → F	F → D
0.60	1.00	0.80	0.75	0.75
E → F	F → E	A → DE	D → AE	E → AD
0.80	1.00	1.00	0.63	0.50
AD → E	AE → D	DE → A	B → EF	E → BF
1.00	1.00	0.63	0.83	0.50
F → EB	BE → F	BF → E	EF → B	D → EF
0.63	0.83	1.00	0.63	0.75
E → DF	F → DE	DE → F	EF → D	DF → E
0.60	0.75	0.75	0.75	1.00

Table 8. Itemsets generated by KC and their respective KC measure with minimum support of 0.5

ACE	ABC	ACE
0, 52	0, 52	0, 52

sociation rule DM technique. We selected the Apriori technique as our benchmark, as it is a well-known and frequently used DM method. The null hypothesis thus reads *KC results will be no worse than those yielded by the Apriori algorithm*. For our first analysis, we compare KC and Apriori along two metrics:

1. *Algorithm complexity* is the number of iterations necessary to run the algorithm, defined in Eq. (4):

$$\text{AlgoComplexity} = O(\text{number of iterations}) \quad (4)$$

2. *Knowledge manageability* measures how easy it is for the user to search the result set, as defined in Eq. (5). For this measurement, we assume that it will be easier for a user to analyze a smaller result set, ranging from several dozen to a few hundred, and we assume that the smaller the set, the better the result.

$$\text{KnowledgeManageability} \in [0..1], \quad (5)$$

$$\underbrace{0}_{\text{VeryHardSearch}} \xrightarrow{\text{OutcomeUsability}} \underbrace{1}_{\text{VeryEasySearch}}$$

The lower the algorithm complexity, the better its performance will be. The last metric is subjective, as it is dependent on the evaluator’s experience with the software and the data. We believe that a small output set containing up to 10 patterns is highly manageable. It becomes less manageable as the output set gets bigger, but evaluator training and familiarity with the techniques may lessen the impact of output size. Although

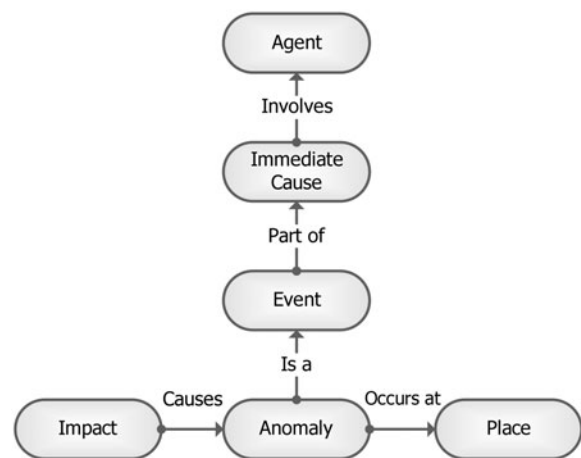


Fig. 3. Knowledge cohesion output.

subjective, this metric is important, because DM is the process of helping real people extract relevant knowledge from a data set.

6.1. Algorithm complexity

A complexity measurement provides an indication of the processing time needed by an algorithm to run, because it accounts for the number of iterations to find a solution. The KC and Apriori approaches are identical with respect to the speed of frequent *itemset* calculation.

The AC are obtained from the most frequent *itemsets* that present a high certainty factor (confidence). The KC method uses the frequent *itemsets* themselves in the semantic networks without any further processing. Thus, its complexity is $O(p)$, where p is the number of frequent *itemsets*, whereas Apriori AC are obtained by combining items in $O(n!)$, where n is the *itemset* cardinality. After this step, each rule is compared with the database to check the confidence of the rule in $O(m)$ time where m is the database cardinality. Thus, the Apriori process has complexity $O(n! * m)$.

To be fair, we need to take into account the SD calculation used in the KC method. The best possible complexity for Dijkstra's algorithm, using a Fibonacci heap, is $O(|E| + |V| * \log(|V|))$, where $|E|$ is the numbers of edges and $|V|$ is the number of vertices of the ontology graph (Barbehenn, 1998). This calculation can be done once and maintained in memory, as Apriori TID does (Agrawal & Srikant, 1994). Thus, KC needs to calculate *itemsets* as Apriori does, but it does not need to calculate all possible combinations as required in Apriori's algorithms. Consequently, we can argue that KC's algorithm time complexity will be lower than Apriori-like algorithms.

6.2. Knowledge manageability

Knowledge manageability is related to the number of results yielded by the method. To verify our algorithm, we ran it against a standard Apriori algorithm on a data set containing 1720 textual anomaly handling reports, containing 62 fields (among categories, numeric, and time stamps), which may or may not contain data. Actual engineers in oil production plants generated the reports.

Both Apriori and KC were applied to the same set of data. We configured the application to generate result sets with at least five items, with support 20% (i.e., at least 20% of the data had to conform to that *itemset*), confidence 80% for the Apriori algorithm and KC equal to 0.8. Apriori generated 26,334 rules versus 4000 generated by KC.

Figures 4 and 5 show screenshots of DMRisco: Figure 4 shows a KC's result set screenshot, and Figure 5 shows an Apriori's result set screenshot. The screenshots clearly show the difference in the number of results generated by each algorithm.

Given that KC outperforms Apriori on both algorithm complexity (number of iterations) and manageability, we con-

sider that we have strong positive indication that KC is indeed better than Apriori.

6.3. Pilot study: Generating understanding with a domain ontology

A first version of the system was shown to four users, with whom we conducted informal, unstructured interviews to receive feedback. Apriori generates several *itemsets* that are essentially the same, as it does not take into account the relationships between items. Thus, a number of items was shown which was essentially the same set, repeated several times (e.g., $[A, B, C]$, $[B, A, C]$, $[A, C, B]$, regardless of whether $A \rightarrow B$, but not $B \rightarrow A$). This happens because Apriori algorithm generated rules are composed of elements that occur simultaneously, regardless of the interrelation between them. The users did not understand why different *itemsets* were generated with the same elements and had difficulty handling the large number of result sets generated.

We considered using KC as a postprocessing step, to reduce the result set and restrict interpretation. However, we first added a visualization panel (see Fig. 6), which uses the ontology as a template for the display of the generated *itemsets*. The ontology brings additional semantics to the data, helping users interpret what they see. Apriori provides relationships between item occurrence, and the ontology provides the relationships between the concepts that the items relate to, adding domain knowledge and facilitating interpretation. With the added panel, users could more easily understand the result sets, as they were contextualized by the underlying ontology.

The smaller number of result sets generated by KC made the information more manageable to users involved, and the contextualized display made it simpler to understand the meaning of the extracted rules.

In addition, DMRisco also allowed the user to specify certain *itemsets* to be mined, even when using the Apriori algorithm. Whenever the user did so, results containing the selected *itemsets* were preferred to others, gaining additional weight.

7. FINAL REMARKS

In this paper we have presented a novel DM technique, KC, which takes advantage of existing domain ontology representing known domain knowledge to guide search for new knowledge in large data sets. We compared our approach to a classic DM algorithm and showed its potential benefits. The premise is that there is always something known in a domain that might help prune the data set, eliciting novel and relevant knowledge. This premise also highlights the weakness of the method: it will only be as good as the underlying ontology. This weakness has two corollaries. First, because KC crucially relies on previously known knowledge to guide the search, it will be useless for domains

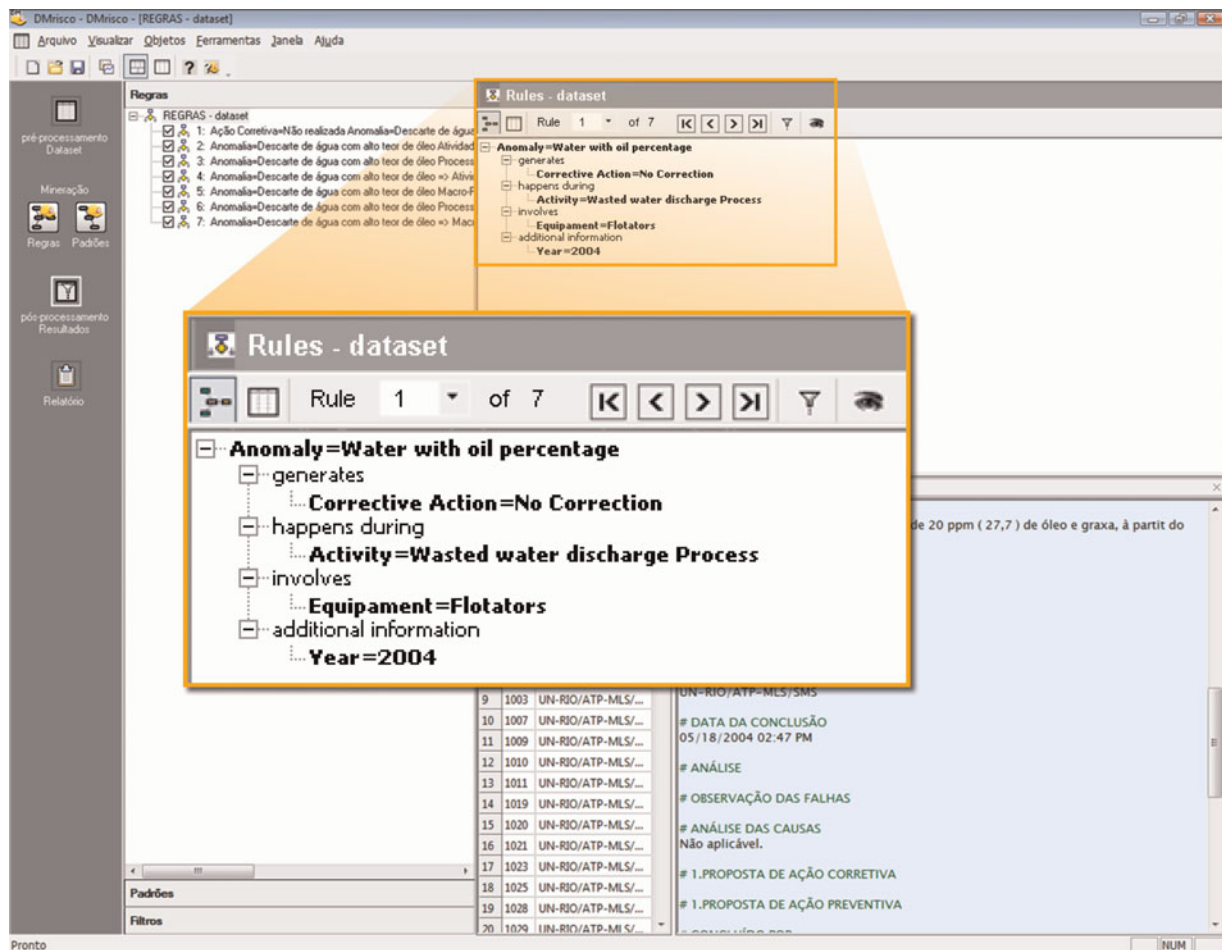


Fig. 4. DMRisco screenshot showing knowledge cohesion results. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

without such knowledge. Second, when ontologies are incorrect, KC will provide misleading results. The result itself will not necessarily be incorrect, but it will not be as relevant as desired. In addition, because results are displayed layered over the ontology, understanding might be jeopardized.

In Apriori-based algorithms, AC are extracted from frequent *itemsets* and accepted if their confidence is greater than the minimum confidence threshold. For a k -frequent *itemset* it is possible to extract AC. Accepted AC may be redundant, irrelevant, or obvious, and the number of really useful rules is much smaller than the number of generated rules. In the KC method, frequent *itemsets* are accepted if their KC measure is greater than the minimum KC threshold. In our tests all frequent *itemsets* with high KC measures are useful.

The relevance metrics for classical information retrieval are precision and recall. Precision is obtained from the ratio between the relevant information retrieved and the retrieved information. Recall is the ratio between the relevant information retrieved and the relevant information in the database. Certainly these measures are human dependent because rele-

vance is a subjective metric assigned by experts to the outcomes.

Given the difficulty of analyzing the relevance of the information retrieved/stored, which would require domain experts to verify results, precision, recall, and singularity were not investigated at this stage. The next step in our research will be to conduct formal analyses using human coders to verify quality and relevance of the results. In addition, we plan on testing KC against other DM techniques. Regardless, the results obtained so far already show that this is a feasible method, which produces results better than standard Apriori methods and generates better understanding of result sets.

Our experiences with ontology-based DM provide strong indication of a considerable improvement on precision retrieving knowledge from large data repositories that reports facts on a well-defined domain, such as the oil field accident report application described in this paper. We believe that this is a promising approach, which deserves further investigation. Although the development of an ontology may add a cost to the project, it quickly pays off in the quality of results and the domain understanding generated.

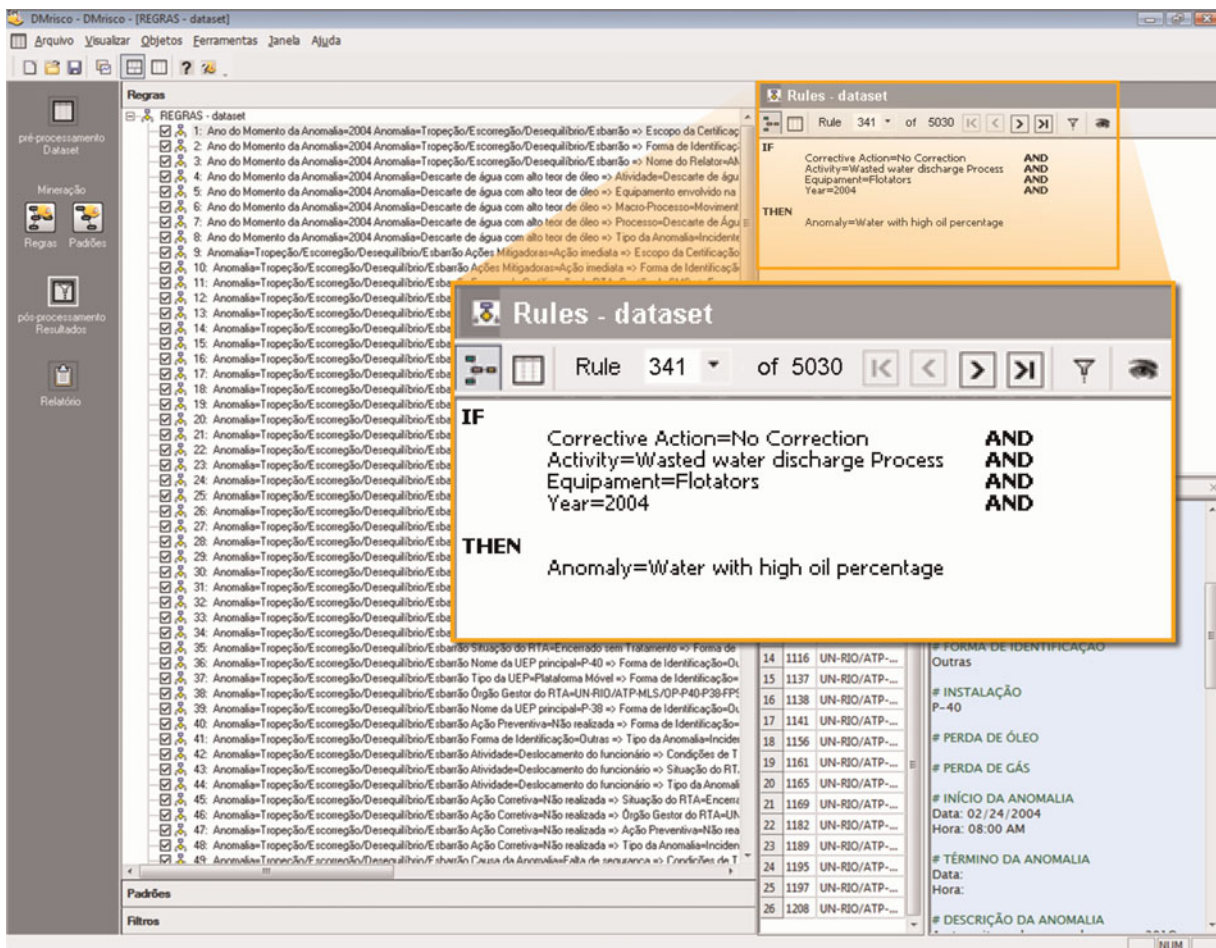


Fig. 5. DMRisco screenshot showing Apriori results. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

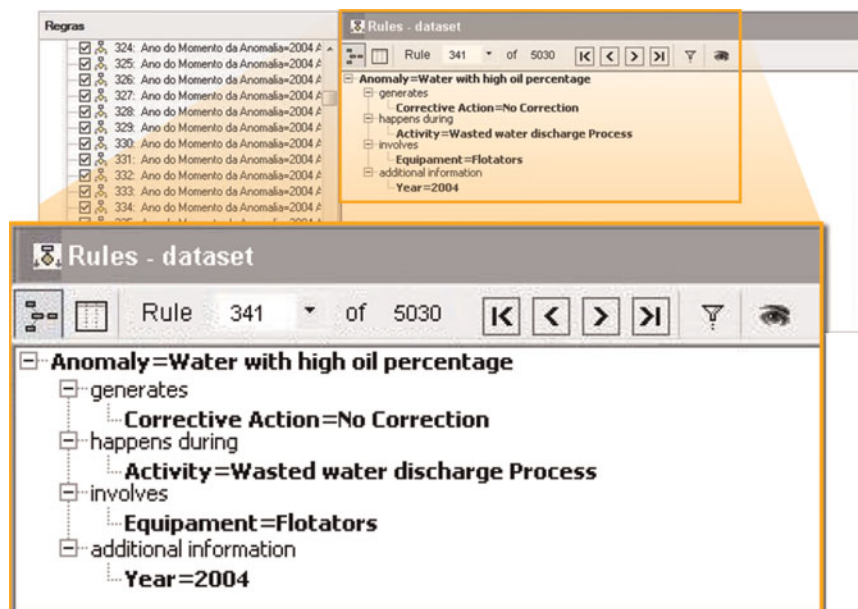


Fig. 6. A rule shown according to an ontological framework. It is easier for a user to understand the meaning of an *itemset* when the relations between the items are explicitly shown. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

ACKNOWLEDGMENTS

We are deeply indebted to Prof. Claude Reichard from Stanford University whose help on the paper review was fundamental to make the material clear and unambiguous. We also express our gratitude to the anonymous reviewers who gave us the possibility to improve the ideas written in this paper.

REFERENCES

- Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proc. 20th Int. Conf. Very Large Data Bases*, pp. 487–499.
- Barbehenn, M. (1998). A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. *IEEE Transactions on Computers* 47(2), 263.
- Chen, C., Khoo, L.P., & Yan, W. (2003). A strategy for acquiring customer requirement patterns using ladder technique and ART2 neural network. *Advanced Engineering Informatics* 16(3), 229–240.
- Dhar, V., Chou, D., & Provost, F. (2000). Discovering interesting patterns for investment decision making with GLOWER—a genetic learner overlaid with entropy reduction. *Data Mining and Knowledge Discovery Journal* 4(4), 251–280.
- Etzkorn, L.H. (2006). Semantic metrics, conceptual metrics, and ontology metrics: an analysis of software quality using IR-based systems, potential applications and collaborations. *Proc. Int. Conf. Software Maintenance*, Philadelphia, PA.
- Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (Eds.). (1996). From data mining to knowledge discovery: an overview. In *Advances in Knowledge Discovery and Data Mining*, pp. 1–34. Menlo Park, CA: AAAI Press/MIT Press.
- Gruber, T.R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition* 5(2), 199–220.
- Jadhav, P., Wong, S.C., Wills, G.B., Crowder, R.M., & Shadbolt, N.R. (2007). Data mining to support engineering design decision. *Proc. Workshop Semantic Web and Web 2.0 in Architectural, Product and Engineering Design Within the 6th Int. Semantic Web Conf. (ISWC)*, Busan, Korea.
- Kitchenham, B., Pfleeger, S.L., & Fenton, N.E. (1995). Towards a framework for software measurement validation. *IEEE Transaction on Software Engineering* (21)12, 929–943.
- Koh, C.G., Chen, Y.F., & Liaw, C.Y. (2003). A hybrid computational strategy for identification of structural parameters. *Computers and Structures* 81, 107–117.
- Kramer, S., & Kaidl, H. (2004). Coupling cohesion metrics for knowledge-based systems using frames and rules. *ACM Transaction on Software Engineering and Methodology* 3(13), 332–358.
- Lavrac, N., Flach, P., & Zupan, B. (1999). Rule evaluation measures: a unifying view. *Proc. 9th Int. Workshop on Inductive Logic Programming (ILP '99)*, pp. 174–185.
- Lenat, D., Guha, R.V., Pittman, K., Pratt, D., & Shepherd, M. (1990). Cyc: towards programs with common sense. *Communications of the ACM* 33(8), 30–49.
- Lenca, P., Vaillant, B., Meyer, P., & Lallich, S. (2007). Association rule interestingness measures: experimental and theoretical studies. In *Quality Measures in Data Mining* (Guillet, F., & Hamilton, H.J., Eds.), pp. 251–276. New York: Springer.
- Li, Z., & Ramani, K. (2007). Ontology-based design information extraction and retrieval. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 21(2), 137–154.
- Lo, D., Khoo, S., & Liu, C. (2007). Efficient mining of iterative patterns for software specification discovery. *Proc. 13th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD '07)*, San Jose, CA.
- Maher, M.L., & Garza, A.G.S. (1997). Case-based reasoning in design. *IEEE Expert: Intelligent Systems and Their Applications* 12(2), 34–41.
- Modak, S.V., Kundra, T.K., & Nakra, B.C. (2002). Comparative study of model updating methods using simulated experimental data. *Computers and Structures* 80(5), 437–447.
- Orme, A.M., Yao, H., & Etzkorn, L.H. (2006). Coupling metrics for ontology-based systems. *IEEE Software* 23(2), 102–108.
- Refrat, R.M., Gero, J., & Peng, W. (2004). Using data mining for improving building life cycle. *Proc. 8th Pacific-Asia Conf. Knowledge Discovery and Data Mining*, Sydney, Australia.
- Roddick, J.F., Hornsby, K., & de Vries, D. (2003). A unifying semantic distance model for determining the similarity of attribute values. *Proc. 26th Australasian Computer Science Conf.*, pp. 111–118.
- Saitta, S., Raphael, B., & Smith, I.F.C. (2005). Data mining techniques for improving the reliability of system identification. *Advanced Engineering Informatics* 19(4), 289–298.
- Shirky, C. (2007). *Ontology is overrated: categories, links, and tags*. Accessed at http://shirky.com/writings/ontology_overrated.html
- Silberschatz, A., & Tuzhilin, A. (1995). On subjective measures of interestingness in knowledge discovery. *Proc. 1st ACM SIGKDD Conf. Knowledge Discovery and Data Mining*, pp. 275–281.
- Sirim, E., Hendler, J., & Parsia, B. (2003). Semi-automatic composition of web services using semantic description. *Proc. Web Services: Modeling Architectures and Infrastructures Workshop in Conjunction With ICEIS 2003*.
- Smith, I.F.C., Saitta, S., Ravindran, S., & Kripakaran, P. (2006). Challenges of data interpretation. *Proc. 18th SAMCO Workshop*, pp. 37–57.
- Soibelman, L., & Kim, H. (2000). Generating construction knowledge with knowledge discovery in databases. *Proc. 8th Int. Conf. Computing in Civil and Building Engineering (VIII-ICCCBE)*, Stanford, CA.
- Varde, A., Ma, S., Maniruzzaman, M., Brown, D.C., Rundensteiner, E., & Sisson, R., Jr. (2007). Comparing mathematical and heuristic approaches for scientific data analysis. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 22(1), 53–69.
- Yao, H. (2005). Cohesion metrics for ontology design and application. *Journal of Computer Science* 1(1), 107–113.

Ana Cristina Bicharra Garcia is an Associate Professor in computer science at Universidade Federal Fluminense. She obtained her PhD in civil engineering from Stanford University in 1992. Since 1996, she has been the Director of the Active Design Documents Laboratory (ADDLabs), an AI research laboratory with special emphasis on developing technologies to solve problems related to the petroleum engineering domain. Dr. Garcia has written journal articles and conference papers (*IEEE*, *ACM*, and *AI EDAM*) on knowledge acquisition, design rationale, recommendation systems, and data mining areas.

Inhauma Neves Ferraz is an Associate Professor in the Department of Computer Science, Universidade Federal Fluminense. His current interests involve AI, neural networks, and data mining. In the last 10 years he has been developing AI applications for the petroleum engineering domain at ADDLabs.

Adriana S. Vivacqua is a Postdoctoral Researcher at ADDLabs at Universidade Federal Fluminense. She obtained her PhD in computer science from the Universidade Federal Rio de Janeiro in 2007. Dr. Vivacqua's research focuses on user modeling and on the application of AI techniques for the cooperation of support systems and human-computer interaction.