Algorithms for Storytelling

Deept Kumar, Naren Ramakrishnan, *Member*, *IEEE Computer Society*,
Richard F. Helm, and Malcolm Potts

Abstract—We formulate a new data mining problem called *storytelling* as a generalization of redescription mining. In traditional redescription mining, we are given a set of objects and a collection of subsets defined over these objects. The goal is to view the set system as a vocabulary and identify two expressions in this vocabulary that induce the same set of objects. Storytelling, on the other hand, aims to explicitly relate object sets that are disjoint (and, hence, maximally dissimilar) by finding a chain of (approximate) redescriptions between the sets. This problem finds applications in bioinformatics, for instance, where the biologist is trying to relate a set of genes expressed in one experiment to another set, implicated in a different pathway. We outline an efficient storytelling implementation that embeds the CARTwheels redescription mining algorithm in an A^* search procedure, using the former to supply next move operators on search branches to the latter. This approach is practical and effective for mining large data sets and, at the same time, exploits the structure of partitions imposed by the given vocabulary. Three application case studies are presented: a study of word overlaps in large English dictionaries, exploring connections between gene sets in a bioinformatics data set, and relating publications in the PubMed index of abstracts.

Index Terms—Data mining, mining methods and algorithms, retrieval models, graph and tree search strategies.

1 Introduction

Redescription mining is a recently introduced data mining problem [1], [2], [3] that seeks to find subsets of data that afford multiple definitions. The input to redescription mining is a set of objects and a collection of subsets defined over these objects. The goal is to view the set system as a vocabulary of descriptors and identify clusters of objects that can be defined in at least two ways using this vocabulary.

For instance, consider the set system in Fig. 1, where the six objects are books and the descriptors denote books about traveling in London (Y), books containing information about places where popes are interred (G), popular books about the history of codes and ciphers (R), books about Mary Magdalene (M), and books about the ancient Priory of Sion (B). An example redescription for this data set is "books involving Priory of Sion as well as Mary Magdalene are the same as nontravel books describing where popes are interred," or $B \cap M \Leftrightarrow G - Y$. This is an exact redescription and gives two different ways of defining the singleton set {"The Da Vinci Code"}. The basic premise of redescription mining is that object sets that can indeed be defined in at least two ways are likely to exhibit concerted behavior and are, hence, interesting. This problem is nontrivial because we are not directly given the object(s) that must participate in the redescription nor the set-theoretic constructions to be used in combining the given descriptors.

Redescriptions such as $B \cap M \Leftrightarrow G - Y$ are exact. Other redescriptions such as $Y \Leftrightarrow G$ are approximate, since Y and G do not induce the same set. The quality of a redescription can be assessed in terms of the Jaccard's coefficient (the ratio of the size of the common elements to elements on either side of the redescription). In this case, the Jaccard's coefficient is $1/3$. Similarly, the redescription $G \Leftrightarrow R$ also holds with Jaccard's coefficient $1/3$. Noticing that these two approximate redescriptions share a descriptor, we can chain them to form the sequence: $Y \Leftrightarrow G \Leftrightarrow R$, i.e., some London travel books (Y) overlap with books about places where popes are interred (G), some of which are books about ancient codes (R). We refer to such a chain of approximate redescriptions as a *story*. The task of storytelling is, given the end points of the story (in this case, Y and R), to find a path between them through a sequence of intermediaries, each of which is an approximate redescription of its neighbor(s). As another example of a story, one that holds with Jaccard's coefficient $1/2$ at each step, we have $B \Leftrightarrow (G \cap M) \Leftrightarrow R$.

Why is this problem interesting and relevant? Storytelling finds application in many domains such as bioinformatics, computational linguistics, document modeling, social network analysis, and counterterrorism. In these contexts, stories reveal multiple forms of insights. First, since intermediaries must conform to a priori knowledge, we can think of storytelling as a carefully argued process of removing and adding participants, not unlike a real story. Knowing exactly which objects must be displaced, and in what order, helps expose the mechanics of complex relationships. Second, storytelling can be viewed as an abstraction of relationship navigation for propositional vocabularies and offers insights similar to what we expect to gain from techniques such as inductive logic programming applied to multirelational databases. Third, storytelling reveals insight

- *D. Kumar is with Feeva Technology, 500 Howard Street, Suite 425, San Fransisco, CA 94105. E-mail: deept@feeva.com.*
- *N. Ramakrishnan is with the Department of Computer Science, Virginia Tech, Blacksburg, VA 24061. E-mail: naren@cs.vt.edu.*
- *R.F. Helm and M. Potts are with the Department of Biochemistry, Virginia Tech, Blacksburg, VA 24061. E-mail: {helmrf, geordie}@vt.edu.*

Manuscript received 6 May 2007; revised 6 Dec. 2007; accepted 17 Jan. 2008; published online 25 Jan. 2008.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-05-0199.
Digital Object Identifier no. 10.1109/TKDE.2008.32.

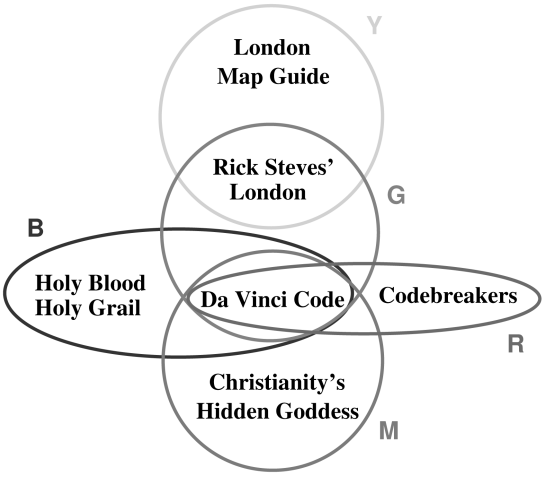


Fig. 1. An example input to storytelling.

into how the underlying Venn diagram of sets is organized, and how it can be harnessed for explaining disjoint connections. In particular, we can investigate if certain sets have a greater propensity for participating in some stories more than others. Such insights have great explanatory power and help formulate hypotheses for situating new data in the context of well-understood processes. Finally, in domains such as bioinformatics, the emergence of high-throughput data acquisition systems (e.g., genome-wide functional screens, microarrays, and RNAi assays) has made it easy for a domain scientist to define vocabularies and sets. We argue that these domains are now suffering from "descriptor overload." Storytelling promises to be a valuable tool to attack this problem and reconcile disparate vocabularies.

Why is this problem difficult? Storytelling is nontrivial because the space of possible descriptor expressions is not enumerable beforehand and, hence, the network of overlap relationships cannot be materialized statically. In a typical application, we have hundreds to thousands of objects and an order of magnitude greater descriptors, with an even larger number of possible set-theoretic constructions made of the descriptors. Effective storytelling solutions must multiplex the task of constructive induction of descriptor expressions with focused search toward the end point of the story.

The main contributions of this paper are twofold: By casting storytelling as a generalization of redescription mining, we achieve a *compositional data mining* approach to storytelling. Specifically, we show how to embed the CARTwheels data mining algorithm for mining redescriptions [2], in an A^* search procedure to compose sequences of redescriptions and realize a story. Second, we showcase three applications of storytelling: a study of word overlaps in large English dictionaries, exploring connections between gene sets in a bioinformatics data set, and relating publications in the PubMed index of abstracts. All of these applications reveal insight into the underlying vocabularies, present significant potential for knowledge discovery, and demonstrate the generality of our algorithms.

This paper also improves upon the preliminary conference version [4] by the use of distance metrics derived from partitions for storytelling search, new bounding techniques that can be used by a heuristic to curtail growth in branching factor, proof of the admissibility of such a heuristic, empirical validation of the scalability of our implementation, and the effectiveness of our heuristic over uninformed search techniques such as breadth first search.

The rest of this paper is organized as follows: Section 2 provides pertinent background on redescriptions and redescription mining, and is necessary before presenting the actual storytelling algorithm. Section 3 demonstrates how to embed the CARTwheels redescription mining algorithm in A^* search, presents a heuristic for use with A^*, and proves its admissibility. This section also covers implementation details such as fast data structures and significance testing for stories. Experimental results are given in Section 4. Connections to related work are drawn in Section 5, and ideas for future work are presented in Section 6.

2 Background

2.1 Formalisms

We begin by formally defining a few key concepts.

Definition 1. *A* set system $(O, \mathcal{S})$ *is a universal set of objects* $O = \{o_1, o_2, \ldots, o_n\}$*, and a collection* $\mathcal{S}$ *of subsets of* O *such that they form a covering of* O*, i.e.,* $\bigcup_i \mathcal{S}_i = O$.

Given a set system, we can think of each element of $\mathcal{S}$ as a Boolean function or "feature" that returns true if a given object from O is present in it, and false otherwise.

Definition 2. *A* descriptor $\mathcal{Z}$ *is a Boolean expression over one or more of the features of* $\mathcal{S}$*. Given a descriptor* $\mathcal{Z}$*, we will denote the set of objects it represents (for a given set system) by* $O(\mathcal{Z})$.

Observe that the elements of $\mathcal{S}$ are trivially descriptors.

Definition 3. *The Jaccard's coefficient* $\mathcal{J}(Z_i, Z_j)$ *between descriptors* $\mathcal{Z}_i$ *and* $\mathcal{Z}_j$ *is given by* $\frac{|Z_i \cap Z_j|}{|Z_i \cup Z_j|}$.

The Jaccard's coefficient is symmetric. It is zero if the descriptors $\mathcal{Z}_i$ and $\mathcal{Z}_j$ are disjoint, and one if they induce the same set of objects.

Definition 4. $\mathcal{Z}'$ *is an exact* redescription *of* $\mathcal{Z}$ *if and only if* $J(\mathcal{Z}, \mathcal{Z}') = 1$ *holds for the given set system.*

The operative phrase here is "for the given set system." We, thus, exclude trivial redescriptions such as $\mathcal{Z}_1 - (\mathcal{Z}_1 - \mathcal{Z}_2) \Leftrightarrow \mathcal{Z}_1 \cap \mathcal{Z}_2$ that hold with Jaccard's coefficient 1 in *all* set systems (i.e., irrespective of how $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are defined).

Definition 5. $\mathcal{Z}'$ *is an approximate* redescription *of* $\mathcal{Z}$ *if and only if* $J(\mathcal{Z}, \mathcal{Z}') < 1$.

Definition 6. *A* story *from descriptor* $\mathcal{Z}_1$ *to descriptor* $\mathcal{Z}_k$ *at Jaccard's threshold* θ *is a sequence of descriptors* $\mathcal{Z}_2, \mathcal{Z}_3, \cdots, \mathcal{Z}_{k-1}$ *such that* $\mathcal{J}(Z_i, Z_j) \geq \theta$, $1 \leq i < k$, $j = i + 1$*. The length of the story is given by* $k - 1$*, i.e., the number of redescriptions involved.*

Observe that since the intermediaries in the story are descriptors, they can be general Boolean expressions. In this

case, to ensure well posedness, we will require that all participating expressions conform to a given bias, e.g., monotone forms, conjunctions, or are otherwise length-limited. More information about the specific bias used in this paper is forthcoming.

The storytelling problem is the following:

> Given a set system $(O, \mathcal{S})$, designated start and end descriptors $\mathcal{Z}_1$ and $\mathcal{Z}_k$, and Jaccard's threshold θ, find all stories from $\mathcal{Z}_1$ to $\mathcal{Z}_k$.

2.2 Generalizations

Although our formalism above focused on sets, we also study multisets in this paper. Here, we consider O to be an ordered set of elements and model each multiset descriptor $(\mathcal{Z}_i)$ as a weighted vector $\mathcal{V}_{Z_i}$ defined over O. The weight corresponding to each element in O could simply be the frequency of that element in the descriptor or it could be a more complicated formulation of the same. For multisets, we use the weighted Jaccard's coefficient as our similarity measure between two descriptors. The weighted Jaccard's $(\mathcal{J}_w)$ between descriptors $\mathcal{Z}_i$ and $\mathcal{Z}_j$ can be defined as follows [5]:

$$\mathcal{J}_w(Z_i, Z_j) = \frac{\sum_{x=1}^{|O|} (\mathcal{V}_{Z_i}[x] * \mathcal{V}_{Z_j}[x])}{\sum_{x=1}^{|O|} (\mathcal{V}_{Z_i}[x])^2 + \sum_{x=1}^{|O|} (\mathcal{V}_{Z_j}[x])^2 - \sum_{x=1}^{|O|} (\mathcal{V}_{Z_i}[x] * \mathcal{V}_{Z_j}[x])}. \quad (1)$$

Notice that the weighted Jaccard's coefficient reduces to the unweighted case if we use 1 as the weight for each element present in a multiset and 0 for elements not in the multiset. The approach we outline for mining stories for simple sets generally holds for the case of multiset descriptors as well. In case any of the steps needs to be handled differently, we outline the details of the alternate approach as well.

2.3 Algorithms for Redescription Mining

There are many algorithms proposed for mining redescriptions, some based on systematic enumeration and pruning (e.g., CHARM-L [3]) and some based on heuristic search (e.g., CARTwheels [2]). These algorithms also differ in their choice of bias (conjunctions in CHARM-L, and depth-limited DNF expressions in CARTwheels). We focus on CARTwheels as it provides the exploratory features necessary to incrementally construct stories.

CARTwheels exploits the fact that redescriptions never occur in isolation but rather in pairs or other groups. For instance, redescription $\mathcal{Z}_i \Leftrightarrow \mathcal{Z}_j$ coexists with $\neg\mathcal{Z}_i \Leftrightarrow \neg\mathcal{Z}_j$. This property is used to search for matching partitions (here, $\{\mathcal{Z}_i, \neg\mathcal{Z}_i\}$ and $\{\mathcal{Z}_j, \neg\mathcal{Z}_j\}$) by growing two binary decision trees (CARTs) in opposite directions so that they are matched at the leaves. The nodes in these trees correspond to Boolean membership variables of the given descriptors so that we can interpret paths to represent set intersections, differences, or complements; unions of paths would correspond to disjunctions. Essentially, one tree exposes a partition of objects via its choice of descriptors (e.g., $\mathcal{Z}_i$) and the other tree tries to grow to match this partition using a different choice of descriptors $(\mathcal{Z}_j)$. If partition correspondence is established, then paths that join define redescriptions. CARTwheels explores the space of possible tree matchings via an alternation process, whereby trees are repeatedly regrown to match the partitions exposed by the other tree.

$$\begin{aligned}
\mathcal{S}_1 &= \{\ o_1, \\
\mathcal{S}_2 &= \{\ o_1, \quad o_2, \quad o_3 \\
\mathcal{S}_3 &= \{\ \quad o_2, \quad\quad o_4 \\
\mathcal{S}_4 &= \{\ \quad\quad o_3, \quad\quad o_5 \\
\mathcal{S}_5 &= \{\ \quad\quad\quad\quad o_5 \\
\mathcal{S}_6 &= \{\ \quad\quad\quad\quad\quad\quad o_6\ \}
\end{aligned}$$

Fig. 2. Example data for illustrating operation of storytelling algorithm.

Unlike previous work, where CARTwheels alternation was configured to enumeratively explore the space of all redescriptions [2], the main contribution of this paper is to demonstrate how we can focus the alternation toward a target descriptor.

3 Designing a Storyteller

We embed CARTwheels inside an A* search procedure, using the former to supply next move operators on search branches to the latter. Each move is a possible redescription to be explored, and a heuristic function evaluates these redescriptions for their potential to lead to the end descriptor of the story. Backtracking happens when a previously unexplored move (redescription) appears more attractive than the current descriptor. The search terminates when we reach a descriptor that is within the specified Jaccard's threshold from the ending descriptor or when there are no redescriptions left to explore.

In this paper, we focus on story length—number of redescriptions to reach the end descriptor—as the primary criterion of optimality, although different criteria might be more suitable in other applications. The story length is useful when we desire explainability of stories: shorter stories are more explainable and appear less tenuous than longer stories. Applications such as MorphWord (covered later), time-ordered news summaries, and causal reasoning between event descriptors are situations where this criterion is relevant. Other criteria that can be considered (but not studied in this paper) include the number of extraneous objects (other than the start and end set) introduced by the story, average Jaccard's coefficient, conformance to a domain theory (e.g., as used in inductive logic programming), or some combination of these factors.

3.1 Working Example

For ease of illustration, consider the artificial example in Fig. 2 with six descriptors $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6\}$ defined over the universal set $O = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ (in a realistic application, the number of descriptors would greatly exceed the number of objects). Our goal is to find a story

between descriptor $\mathcal{S}_1$, corresponding to the set $\{o_1\}$, and $\mathcal{S}_5$, corresponding to the set $\{o_5\}$, such that each step is a redescription that holds with Jaccard's coefficient at least $\theta = 0.5$. In this example, we set the maximum depth of CARTs used to 2.

We begin with a CART that models the start descriptor $\mathcal{S}_1$ by exposing the partition $\mathcal{P}_1 = \{\mathcal{S}_1, \overline{\mathcal{S}_1}\}$, i.e., $\{\{o_1\}, \{o_2, o_3, o_4, o_5, o_6\}\}$. The end descriptor is similarly captured by a CART exposing the partition

$$\mathcal{P}_5 = \{\mathcal{S}_5, \overline{\mathcal{S}_5}\} = \{\{o_1, o_2, o_3, o_4, o_6\}, \{o_5\}\}.$$

We first translate our requirement of minimum Jaccard's coefficient into a threshold on minimum distance between partitions. (This is necessary because, unlike metrics such as entropy gain and Gini coefficient, a constraint on Jaccard's coefficient does not directly characterize probability distributions necessary for incremental induction of decision trees.)

The specific partition-distance metric we use between partitions $\mathcal{P}_i = \{Z_i, \overline{Z_i}\}$ and $\mathcal{P}_{i+1} = \{Z_{i+1}, \overline{Z_{i+1}}\}$ is the López de Mántaras [6] criterion:

$$\begin{aligned}\mathcal{D}(\mathcal{P}_i, \mathcal{P}_{i+1}) = & -\frac{1}{|O|}\left[z_a \log\left(\frac{z_a}{|Z_{i+1}|}\right) + z_c \log\left(\frac{z_c}{|Z_{i+1}|}\right)\right] \\ & -\frac{1}{|O|}\left[z_b \log\left(\frac{z_b}{|\overline{Z_{i+1}}|}\right) + z_d \log\left(\frac{z_d}{|\overline{Z_{i+1}}|}\right)\right] \\ & -\frac{1}{|O|}\left[z_a \log\left(\frac{z_a}{|Z_i|}\right) + z_b \log\left(\frac{z_b}{|Z_i|}\right)\right] \\ & -\frac{1}{|O|}\left[z_c \log\left(\frac{z_c}{|\overline{Z_i}|}\right) + z_d \log\left(\frac{z_d}{|\overline{Z_i}|}\right)\right],\end{aligned} \tag{2}$$

where

$$\begin{aligned} z_a &= |Z_i \cap Z_{i+1}|, \quad z_b = |Z_i \cap \overline{Z_{i+1}}| \\ z_c &= |\overline{Z_i} \cap Z_{i+1}|, \quad z_d = |\overline{Z_i} \cap \overline{Z_{i+1}}|. \end{aligned}$$

In (2), the first four terms correspond to the entropy when $\mathcal{P}_{i+1}$ is used to split $\mathcal{P}_i$. Similarly, the last four terms correspond to the entropy when $\mathcal{P}_i$ is used to split $\mathcal{P}_{i+1}$. Given $|Z_i|$ and $|O|$, the distance value in (2) is dependent on $|Z_i - Z_{i+1}|$ and $|Z_{i+1} - Z_i|$. The Jaccard's coefficient between Z_i and Z_{i+1} can also be calculated using these two quantities as

$$\mathcal{J}(Z_i, Z_{i+1}) = \frac{|Z_i| - |Z_i - Z_{i+1}|}{|Z_i| + |Z_{i+1} - Z_i|}.$$

If we fix the value of $|Z_{i+1} - Z_i|$, where

$$0 \leq |Z_{i+1} - Z_i| \leq \lfloor((1-\theta)Z_i)/\theta\rfloor,$$

there are values of $|Z_i - Z_{i+1}|$, such that

$$\lfloor(1-\theta)Z_i\rfloor \geq |Z_i - Z_{i+1}| \geq 0,$$

and where the Jaccard's coefficient between Z_i and Z_{i+1} is greater than θ.

Fig. 3 shows the behavior of the distance function in (2) for different values of $|Z_i - Z_{i+1}|$ and $|Z_{i+1} - Z_i|$. Here, each column of data points from the x-axis upward represents the distance values as $|Z_i - Z_{i+1}|$ increases for a fixed value of $|Z_{i+1} - Z_i|$. This figure clearly indicates the region

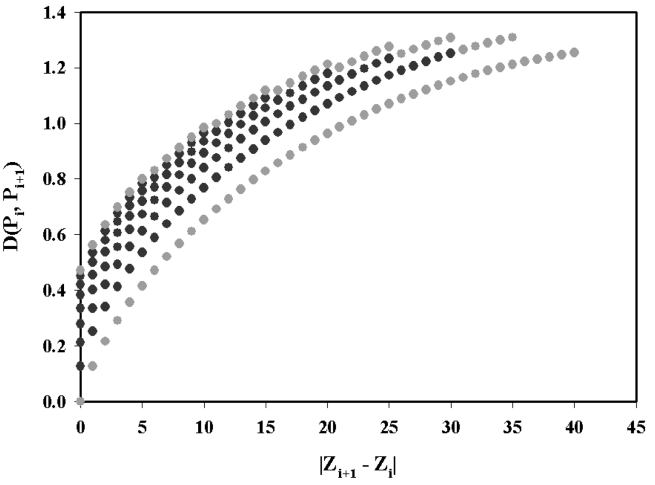


Fig. 3. Behavior of distance metric used for decision tree construction. Each column of data points from the x-axis upward represents the distance values as $|Z_i - Z_{i+1}|$ increases for a fixed value of $|Z_{i+1} - Z_i|$. The lighter data points bound the region in which we need the distance between partitions to lie for the Jaccard's threshold to hold. The parameters used in this plot are $|Z_i| = 10$, $|O| = 100$, $\theta = 0.2$.

(bounded by the points shaded in light color) in which we need the distance between partitions to lie for the Jaccard's threshold to hold. We call the region covered by points in Fig. 3 the region of similar partitions, which gets narrower as more extraneous elements are introduced by the (current) redescription; hence, we can map a Jaccard's coefficient threshold between descriptors into a requirement of distance between two partitions. Apart from this simple relationship between the distance metric and Jaccard's coefficient, another reason why we chose this metric is that it promises to provide a better match between two consecutive partitions given the restricted length of the trees we construct [6].

The general idea now is to consider the partition exposed by the current tree and induce new trees to match this partition. In our previous work [2], we allow these partitions to be composed of arbitrary blocks so that when two trees match, we obtain as many redescriptions as there are matching blocks. In the present work, we focus on only two-block partitions since, by the statement of the story-telling problem, both start and end descriptors induce only two-block partitions and we desire a single chain of redescriptions between them. Thus, for a one-level tree, which has only one descriptor such as Z_i, the partition would be $\mathcal{P}_i = \{Z_i, \overline{Z_i}\}$. For a two-level tree, the partition would correspond to how the four paths in the tree are merged to induce a two-block partition.

Consider the beginning partition $\{\mathcal{S}_1, \overline{\mathcal{S}_1}\}$. In constructing a tree to match this partition, we look for a descriptor inducing a two-block partition such that the distance between that partition and $\{\mathcal{S}_1, \overline{\mathcal{S}_1}\}$ is minimized. If, for one or more descriptors, the distance lies in the similar partitions region, we greedily choose the one which induces a block Z_{i+1} with the highest Jaccard's coefficient with the end point of the story. Once the tree has been constructed in this manner, class assignments at the leaves are made by majority, and paths that lead to a given class are unioned to form redescriptions.

obj.	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	class
o_1	$\surd$	$\times$	$\times$	$\times$	$\times$	$\mathcal{S}_1$
o_2	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\overline{\mathcal{S}_1}$
o_3	$\surd$	$\times$	$\surd$	$\times$	$\times$	$\overline{\mathcal{S}_1}$
o_4	$\times$	$\surd$	$\times$	$\times$	$\times$	$\overline{\mathcal{S}_1}$
o_5	$\times$	$\times$	$\surd$	$\surd$	$\times$	$\overline{\mathcal{S}_1}$
o_6	$\times$	$\times$	$\times$	$\times$	$\surd$	$\overline{\mathcal{S}_1}$

(a)

obj.	$\mathcal{S}_1$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	class
o_1	$\surd$	$\times$	$\times$	$\times$	$\times$	$(\mathcal{S}_2 - \mathcal{S}_3)$
o_2	$\times$	$\surd$	$\times$	$\times$	$\times$	$\overline{(\mathcal{S}_2 - \mathcal{S}_3)}$
o_3	$\times$	$\times$	$\surd$	$\times$	$\times$	$(\mathcal{S}_2 - \mathcal{S}_3)$
o_4	$\times$	$\surd$	$\times$	$\times$	$\times$	$\overline{(\mathcal{S}_2 - \mathcal{S}_3)}$
o_5	$\times$	$\times$	$\surd$	$\surd$	$\times$	$\overline{(\mathcal{S}_2 - \mathcal{S}_3)}$
o_6	$\times$	$\times$	$\times$	$\times$	$\surd$	$\overline{(\mathcal{S}_2 - \mathcal{S}_3)}$

(b)

Fig. 4. (a) Data set to initialize storytelling algorithm. (b) Data set for the second iteration.

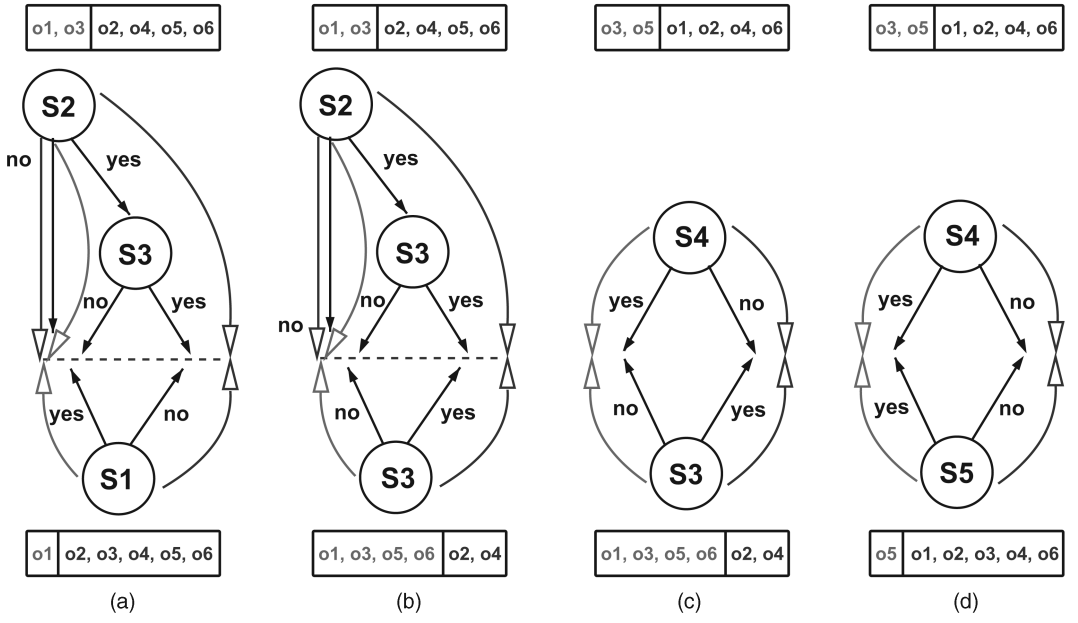


Fig. 5. Storytelling using CARTwheels alternation. Beginning with $\mathcal{S}_1$, the starting descriptor exposed by the bottom tree in (a), the alternation systematically moves toward $\mathcal{S}_5$, the ending descriptor in (d). At each step, we alternately keep one of the trees fixed and grow a new tree to match it. The partition induced by each tree is shown above or below the tree as the case may be. The story mined here is the sequence of redescriptions: $\mathcal{S}_1 \Leftrightarrow (\mathcal{S}_2 - \mathcal{S}_3) \Leftrightarrow \overline{\mathcal{S}_3} \Leftrightarrow \mathcal{S}_4 \Leftrightarrow \mathcal{S}_5$.

For instance, Fig. 5a shows the decision tree we have constructed to match the partition $\{\mathcal{S}_1, \overline{\mathcal{S}_1}\}$. This tree provides the first step in the story to be the redescription $\mathcal{S}_1 \Leftrightarrow (\mathcal{S}_2 - \mathcal{S}_3)$. In this example, we show only one possible "next tree", but in our implementation, we maintain a number of such possible matching trees to simulate a branching process and for potential backtracking. Note that while the current redescription holds with a Jaccard's value of 0.5, the new descriptor does not have any overlap with $\mathcal{S}_5$ (the target).

The alternation process is easily understood by considering the data sets from which each tree is being learned. To initialize the alternation, we prepare a traditional data set for classification tree induction (see Fig. 4a), where the entries correspond to the objects, the class (to be learned) corresponds to membership in the starting descriptor, and the Boolean features are comprised of the remaining descriptors. For the next step in our story, we use partition $\{(\mathcal{S}_2 - \mathcal{S}_3), \overline{(\mathcal{S}_2 - \mathcal{S}_3)}\}$ as the classes to match and consider the data set as shown in Fig. 4b. In constructing the new data set, observe that we ignore the descriptor that is the top-most node (here, $\mathcal{S}_2$) in the decision tree that defines the current partition. This ensures that we do not utilize the same features for matching a partition as those that define the partition! The one-level tree we learn at this stage is shown in Fig. 5b. The redescription of interest here is $(\mathcal{S}_2 - \mathcal{S}_3) \Leftrightarrow \overline{\mathcal{S}_3}$, which also holds with a Jaccard's coefficient of 0.5. Although it introduces the element we seek (o_5), the redescription to the end point of the story, $\overline{\mathcal{S}_3} \Leftrightarrow \mathcal{S}_5$, has only a Jaccard's coefficient of 0.25. We, hence, continue the search and obtain the redescription $\overline{\mathcal{S}_3} \Leftrightarrow \mathcal{S}_4$ which gives us the desired overlap with the target, and our final redescription, namely $\mathcal{S}_4 \Leftrightarrow \mathcal{S}_5$. Our story is thus

$$\mathcal{S}_1 \Leftrightarrow (\mathcal{S}_2 - \mathcal{S}_3) \Leftrightarrow \overline{\mathcal{S}_3} \Leftrightarrow \mathcal{S}_4 \Leftrightarrow \mathcal{S}_5.$$

Before we describe our storytelling algorithm in detail, it is important to recognize the implicit expression bias used by CARTwheels. Since descriptors are obtained by fusing one or more paths in a decision tree, the form of Boolean expressions allowed here can be characterized as depth-limited DNF, i.e., a disjunction over conjunctions

TABLE 1
Storytelling Algorithmic Framework

Input:

1) a domain of objects O;
2) a collection $\mathcal{S}$ of sets defined over O;
3) a designated starting descriptor $X \in \mathcal{S}$; and
4) a designated ending descriptor $Y \in \mathcal{S}$.

Parameters:

1) a threshold θ ($0 < \theta < 1$) denoting the minimum required Jaccard's coefficient for each connection in the story;
2) d (depth of trees) that imposes a bias $\mathcal{B}$ over set expressions defined on $\mathcal{S}$; and
3) branching factor b that restricts the maximum number of possible next states from each state in the A* search.

Output: a story comprising a sequence of intermediaries $Z_1, Z_2, \cdots, Z_k$, such that $X = Z_1$, $Y = Z_k$, $J(Z_i, Z_{i-1}) \geq \theta, 1 < i \leq k$, and each of Z_i's is in the desired bias $\mathcal{B}$.

Initialization:

set open list for A* search $\mathcal{OL} = \{\}$; closed list for A* search $\mathcal{CL} = \{\}$
set classes $\mathcal{C}$ = $\{X, \overline{X}\}$; features $\mathcal{F}$ = $\mathcal{S} - \mathcal{C}$
set dataset $\mathcal{D}$ = construct_dataset(O, $\mathcal{F}$, $\mathcal{C}$)
for ($j = 1$; $j \leq b$; $j = j + 1$)
  set tree t_j = construct_tree($\mathcal{D}$, d, j)
  if (eval(t_j, θ))
    set g_j = 0; h_j = calculate_heuristic_score(t_j, $\mathcal{C}$, Y, θ, $|O|$); s_j = $g_j + h_j$
    add t_j with (s_j, g_j, h_j) as a node in $\mathcal{OL}$

Alternation:

set boolean done = f$alse$
while (($\mathcal{OL}$ is not empty) AND ! done)
  t_N = head($\mathcal{OL}$)
  if (h_N = 0)
    print_story(t_N, $\mathcal{CL}$)
    set done = true
  set classes $\mathcal{C}$ = paths_to_classes(t_N); features $\mathcal{F}$ = $\mathcal{S}$ − top(t_N)
  set dataset $\mathcal{D}$ = construct_dataset(O, $\mathcal{F}$, $\mathcal{C}$)
  for ($j = 1$; $j \leq b$; $j = j + 1$)
    set tree t_j = construct_tree($\mathcal{D}$, d, j)
    if (eval(t_j, θ))
      set g_j = $g_N + 1$; h_j = calculate_heuristic_score(t_j, $\mathcal{C}$, Y, θ, $|O|$); s_j = $g_j + h_j$
      add t_j with (s_j, g_j, h_j) as a node in $\mathcal{OL}$
  delete head from $\mathcal{OL}$

where each conjunction is of length at most the depth of the CART.

3.2 Implementation

The storytelling algorithmic framework is shown in Table 1 and follows the outline of the working example above. The key utility functions are *construct_data set*, which prepares the data set $\mathcal{D}$ at each alternation; *construct_tree*, which is called b (branching factor) times at each step to create trees of desired depth limit d; *eval*, which determines if the Jaccard's coefficient between the current descriptor and the union of the paths leading to it in the current tree have a Jaccard's coefficient higher than or equal to θ; *calculate_heuristic_score*, which computes the heuristic score for each tree used to guide the search; and *print_story*, which prints the story by tracing back the sequence of mined redescriptions.

The choice of b and d can be made based on domain-specific considerations. Smaller values of d use simpler-to-understand descriptors in the story but run the risk of inability to find a story. Similarly, larger values of b offer greater guarantee of completeness of the A^* search but introduce significant penalty in time complexity. Specific examples of how these values are selected and the tradeoff analysis are covered later.

In the outer A^* search procedure, qualified trees are placed in the open list $\mathcal{OL}$ (a priority queue) and considered

TABLE 2
Heuristic for Storytelling A^* Search

calculate_heuristic_score(t_j, C, Y, θ, $|O|$):
set Z_{j-1} = target class from $\mathcal{C}$; Z_j = block from t_j that redescribes to Z_{j-1}
set $f_j = |Z_j \cap Y|$; $e_j = |Z_j - Y|$
set $h_j = 0$
calculate h_j = minpath(f_j, e_j, $|Y|$, h_j, θ, $|O|$)
return h_j

minpath(f, e, $|Y|$, h, θ, $|O|$):
calculate $\theta_Y = f/(e + |Y|)$
if ($\theta_Y \geq \theta$)
return h
else
calculate δf_{max} = min($\lfloor \frac{(1-\theta)(f+e)}{\theta} \rfloor$, $|Y| - f$); δe_{max} = min($\lfloor (1-\theta)(f+e) \rfloor$, $|O| - |Y| - e$)
set $h_{min} = \infty$
for ($i = 0$; $i \leq \delta f_{max}$; $i = i + 1$)
set $done$ = false
for ($k = \delta e_{max}$; $k \geq 0$ and ! $done$; $k = k - 1$)
calculate $\theta_{new} = \frac{f+e-k}{f+e+i}$
if ($\theta_{new} \geq \theta$)
set $done$ = true; h_{curr} = minpath($f + i$, $e - k$, $|Y|$, $h + 1$, θ, $|O|$)
if ($h_{curr} < h_{min}$)
set $h_{min} = h_{curr}$
return h_{min}

in order of their evaluations. If the heuristic evaluation h_N for the currently picked tree t_N is zero, we have arrived at a tree that has sufficient Jaccard's overlap with the end point of the story and we terminate. If h_N is not zero, a new set of classes $\mathcal{C}$ for objects in O is induced using the function *paths_to_classes*, t_N is moved to the closed list, and all trees induced by *construct_tree* are placed in the open list. This process is repeated until there is no tree left in the open list or a story has been found.

Just as the notion of "class" is revised at each alternation, so are the candidate set of "features." Observe that, in the Initialization step, the set of possible features involves all except the starting descriptor. Inside the Alternation subroutine, the candidate set of features is made equal to all except the feature used at the root of the current tree (supplied by the routine *top*).

The heuristic score h_j for tree t_j is combined with cost expended so far (g_j) to arrive at the evaluation criterion s_j. Nodes in $\mathcal{OL}$ are, hence, ordered by s_j. We assume unit cost per redescription so that the story length is the number of redescriptions required to traverse to the ending descriptor. The heuristic function h is designed to systematically never overestimate the number of redescriptions remaining and takes the value of zero for a tree whose partition is within the specified Jaccard's coefficient to the ending descriptor. We now present details of h and prove its admissibility.

3.3 Heuristic

Table 2 outlines the approach to estimate h_j for tree t_j. This algorithm can be understood as follows: Assume that the new descriptor Z_j (provided by tree t_j) has f_j elements in common with the target descriptor Y and e_j elements that do not participate in Y. This means that Z_j must shed enough of the e_j elements and acquire enough of the $|Y| - f_j$ elements in order to have a Jaccard's threshold of $\geq \theta$ with Y. The goal of *calculate_heuristic_score* is to estimate the minimum number of redescriptions required to shed the requisite number among e_j elements and acquire some of the necessary $|Y| - f_j$ elements. The procedure first conservatively estimates if the current discrepancies already correspond to a Jaccard's threshold of $\geq \theta$ with Y, in which case it returns zero. If this is not possible, the procedure estimates the shortest number of steps in which the deletions and additions can happen by a recursive computation. Two extremes are considered at each step—the case where we can acquire as many of the necessary new elements as dictated by θ without any removals and the case where we can shed as many of the unnecessary elements as dictated by θ without any additions. This step provides us with the bounds δf_{max} and δe_{max} in Table 2. We then search combinatorially within these ranges for the maximal number of deletions, for every possible number of additions, such that θ holds, using dynamic programming. The minimum number of redescriptions over all possibilities is then returned.

For a worked-out example, consider Fig. 6. Here, assume that our start descriptor (for the universal set O defined earlier) is defined using the first tree grown in our running example (Fig. 5a), i.e., $\mathcal{S}_2 - \mathcal{S}_3$. The end descriptor is the same as in our running example. In the directed graph shown, each node corresponds to the tuple (f, e). Thus, the

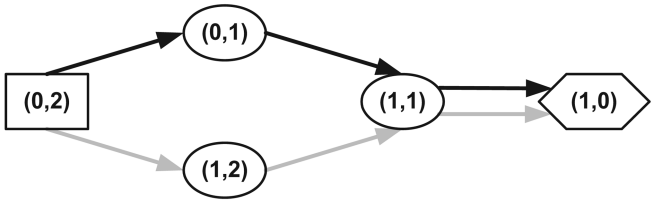


Fig. 6. The working of the heuristic function. The start state (descriptor) is represented by the tuple (0, 2) and the end node by the tuple (1, 0). For $\theta = 0.5$, the next states considered from each state are connected by the edges. The two different sets of edges (top and bottom) represent the paths that could be taken to reach the final state in the minimum (3) number of steps (redescriptions).

start node (shown in square) is set to (0, 2) since it—$\{o_1, o_3\}$—contains none (0) of the required elements and two spurious elements (which are to be discarded). Similarly, the end node (shown in a hexagon) is set to (1, 0). The Jaccard's coefficient threshold is set to 0.5. For (0, 2), $\delta f_{max} = 1$ and $\delta e_{max} = 1$. Thus, the best possible next states correspond to the other ends of the two directed edges starting from (0, 2). However, none of the next two states possible has the required overlap with (1, 0). Hence, we recurse the given problem into a smaller subproblem. As an example, we move to the state (1, 1) from the node (1, 2). This new state has the required overlap with the final node, and thus, we have found a path to the final state as $(0,2) \rightarrow (1,2) \rightarrow (1,1) \rightarrow (1,0)$. The other possible nodes and paths can also be examined in this manner to construct the whole graph shown. From this graph, the minimum path length found is 3, and this is the value returned by our heuristic function for the state (0, 2).

3.4 Proof of Admissibility

To prove that our heuristic is admissible, we must show that it never overestimates the cost (length) of a path from a node to the goal. One way to structure the proof is to characterize the underlying graph or state space in which the search for a path is conducted (each node in the graph is an (f, e) pair as before).

We identify four graphs (see Fig. 7) to help structure the proof:

1. $\mathcal{G}$ is a graph whose nodes are all legal (f, e) pairs and whose edges bring together nodes that can potentially satisfy the Jaccard's threshold. We say "potentially" because it is not possible to ascertain the exact overlap between descriptors given only the (f, e) information. Nevertheless, we know that we need only retain those (f, e) pairs, where $f \leq |Y|$ and $(f + e) \leq |O|$. Similarly, for any two nodes $V_i = (f_i, e_i)$ and $V_j = (f_j, e_j)$ in $\mathcal{G}$, we can calculate the maximum value of Jaccard's coefficient *possible* between them as

$$\mathcal{J}_{max}(V_i, V_j) = \frac{|V_i \cap V_j|}{f_i + e_i + f_j + e_j - |V_i \cap V_j|},$$

   where $|V_i \cap V_j|$ is defined as $\min(f_i, f_j) + \min(e_i, e_j)$. Then, we need to retain only those edges in $\mathcal{G}$ such that $\mathcal{J}_{max}(V_i, V_j) \geq \theta$.
2. $\mathcal{G}_S \subseteq \mathcal{G}$, retaining only those (f, e) pairs from $\mathcal{G}$ that are realizable in bias $\mathcal{B}$.

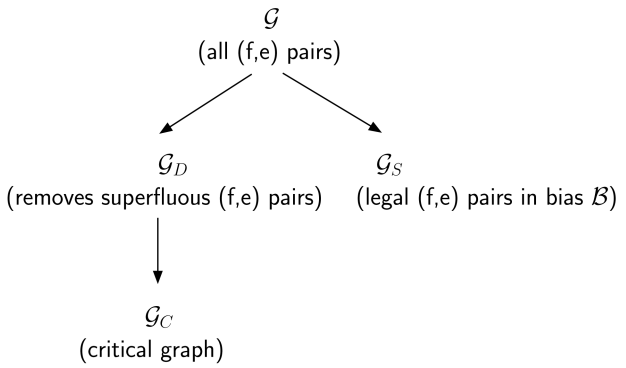


Fig. 7. Sequence of graph transformations used to demonstrate admissibility of heuristic.

3. $\mathcal{G}_D \subseteq \mathcal{G}$, removing "superfluous" edges between (f, e) pairs that are deemed unnecessary to reach the destination from the start of the story. An edge from (f_i, e_i) to (f_j, e_j) is deemed superfluous if either $f_i < f_j$ or $e_i > e_j$.
4. $\mathcal{G}_C \subseteq \mathcal{G}_D$, retaining only those edges from (f_i, e_i) to (f_j, e_j) if, for the given $f_j - f_i$ improvement in desired elements, $e_i - e_j$ is the best reduction in unwanted elements possible (within the constraints of the Jaccard's coefficient).

In more detail, $\mathcal{G}$, $\mathcal{G}_D$, and $\mathcal{G}_C$ are based purely on analytical calculations on (f, e) pairs, without regard to the data set at hand. $\mathcal{G}_S$ is the only graph that is cognizant of the given data set and CARTwheels bias. The crux of the proof depends on the following observation:

> $\mathcal{G}_S$ holds the shortest path identified by CARTwheels alternation, whereas $\mathcal{G}_C$ holds the shortest path as estimated by the heuristic.

Hence, the goal is to show that the shortest path in $\mathcal{G}_C$ cannot be greater than the shortest path in $\mathcal{G}_S$. The following result establishes this:

Lemma 1. *The heuristic defined by* calculate_heuristic_score *is admissible.*

Proof. The heuristic searches in $\mathcal{G}_C$ for a shortest path. It is clear, by construction, that $\mathcal{G}_C$ cannot overestimate the cost of the path with regard to $\mathcal{G}_D$. To understand the shortest paths in $\mathcal{G}_D$, observe that a path in $\mathcal{G}_D$, in its attempt to gain as much of the missing elements, assumes that we do not lose any of the elements already present. In reality, however, we are likely to temporarily lose some of the required elements. However, all that matters is that the heuristic considers the best-case scenario since it provides a lower bound on the actual length. Hence, a shortest path in $\mathcal{G}_D$ (and by extension $\mathcal{G}_C$) underestimates the actual cost in $\mathcal{G}$. Now, the bias imposed by $\mathcal{G}_S$ can only serve to increase the cost by rendering some of the nodes of $\mathcal{G}_D$ unavailable. Furthermore, paths in $\mathcal{G}_S$ might elect to temporarily lose desired elements or might choose suboptimal reductions of unwanted elements. Hence, the estimate obtained by the heuristic is optimistic in all respects and is, thus, admissible. □

The heuristic function defined in Table 2 is applicable for simple set descriptors. In case the descriptors are modeled as multisets, we cannot use the same approach. This is

because, for instance, elimination of different subsets from a given descriptor, even if they are of the same size, will result in different Jaccard's coefficients. This implies that we will have to exhaustively search all combinations for removal and addition of elements to determine the theoretically shortest possible story length. To avoid this, for the case of the weighted Jaccard's coefficient in (1), we use a simpler heuristic function wherein we estimate the maximum weight we can gain/lose at each step. We add *and* remove these maximum weight values from our current document weight. Using this idea, we calculate the number of steps required to gain enough of the weight for the final document and lose enough weight from the current document, to reach a Jaccard's coefficient above the threshold for the final document (observe that this is still admissible but not as good a tracker of the shortest story length).

3.5 Data Structures

The efficient implementation of our storytelling algorithm hinges on data structures for fast estimation of overlaps. This problem has been studied by the database community in various guises, e.g., similarity search [7] and set joins on similarity predicates [8]. Specific solutions advocate the use of signature trees [9], hierarchical bitmap indices [10], and locality sensitive hashing [11], especially the technique of min-wise independent permutations [12] that is particularly geared toward the Jaccard's coefficient. In this paper, we combine an AD-tree data structure [13] with the signature tables [14] approach for efficient similarity search in categorical data.

The signature table is constructed before the Initialization step mentioned in Table 1. Here, objects in the universal set are divided into a predefined number of clusters (c) on the basis of their co-occurrence frequencies. This is achieved by first constructing a graph where each object is a node and objects that co-occur form edges. Each edge is associated with a weight which is the inverse of the co-occurrence frequency for that edge. The weight associated with each node is the sum of the weights of each edge it is a part of. The total weight of the graph is the sum of the weight of all nodes. We set the critical weight of the graph to be the total weight divided by c. For finding each cluster, we begin with the nodes that are a part of the edge that currently has the minimum weight associated with it. We delete these nodes from the graph and add them to the current cluster. The weight of the cluster is now the sum of the weights of nodes it contains (as obtained from the original graph). We continue to add nodes to the cluster that have the minimum weight in an edge associated with any of the objects that already are a part of the cluster. This continues till the weight of the cluster is greater than the critical weight. At this point, we recalculate the critical weight on the basis of the nodes remaining in the original graph and proceed to finding the next set of nodes, till all c clusters have been found.

Each descriptor, originally a binary vector size $|O|$, can now be condensed into a binary vector (the signature) of size c by encoding a 1 for each cluster that has an object present in the descriptor and a 0 for each cluster that has no object in the descriptor.

All descriptors and their co-occurrence frequencies (used in constructing a decision tree of depth more than 1) are also built into an AD-tree at this stage. The descriptors at the top-level of the AD-tree are additionally linked to their signatures. When a similarity search query is issued, only nodes that correspond to signatures of interest need to be investigated. At greater depths in the AD-tree, we can either construct individual signature tables for each node in the AD-tree or we can opt to use a traditional AD-tree node that contains descriptor names and co-occurrence frequencies. In our implementation, we used traditional AD-tree nodes at depth greater than 1.

Using these data structures, we can reduce the number of descriptors searched against at each step and improve the speed of computation of stories. For instance, in the first call to the function *construct_tree*, where we are looking for the best match for the class X from among the descriptor set D, we can reduce X to a vector of size c (X^c). Also, we keep a count of the number of objects in X that belong to each of the c clusters in the form of a frequency vector f^c. The optimistic Jaccard's coefficient $(\mathcal{OJ})$ between X^c and a signature vector V_i^c corresponding to a set of descriptors can then be calculated by the formula

$$\mathcal{OJ}(X^c, V_i^c) = \frac{\sum_{j=1}^{c}(f^c[j] * X^c[j] * V_i^c[j])}{\sum_{j=1}^{c} f^c[j]}.$$

We then compare X^c to all the signature vectors and retain only those for which the optimistic Jaccard's coefficient is above θ. This narrows down our search to only those descriptors that have potential to provide the necessary overlap.

3.6 Complexity and Significance Analysis

The time and space complexities of our approach can be characterized by two search spaces: the space of possible decision trees (CARTs) that are explored at each step of the alternation and the branching space of the A* search itself. Using the AD-tree data structure, the space complexity amounts to $O(|\mathcal{S}|^d)$. In constructing a decision tree using the AD-tree data structure, each node in the decision tree requires, at most, $|\mathcal{S}|$ nodes in the AD-tree to be compared against. Therefore, the time complexity is $O(b^l 2^d |\mathcal{S}|)$. Observe that d is usually fixed to be a constant; hence, the time complexity is typically driven by the branching process of A* search.

The significance of a story is assessed at the level of each redescription participating in the story. To assess the significance of redescription $Z_i \Leftrightarrow Z_{i+1}$, we use the cumulative hypergeometric distribution [15] to determine the probability of obtaining a rate of co-occurrence of Z_i and Z_{i+1} (over the object domain), given their marginal occurrence probabilities, and comparing it to the observed rate of co-occurrence by chance. For simple set descriptors, this amounts to the probability (p) that a randomly constructed Z_{i+1} of size $|Z_{i+1}|$ has an overlap of at least $|Z_i \cap Z_{i+1}|$ with the fixed set Z_i:

$$p(Z_i, Z_{i+1}) = 1 - \sum_{j=0}^{|Z_i \cap Z_{i+1}|-1} \frac{\binom{|Z_i|}{j}\binom{|O|-|Z_i|}{|Z_{i+1}|-j}}{\binom{|O|}{|Z_{i+1}|}}. \quad (3)$$

To account for multiple hypothesis testing, the significance threshold is determined by first characterizing the distribution for all descriptors tested. This is achieved by

fixing Z_i and randomly sampling from the set of available descriptors. For each such descriptor Z_k sampled, we use the formula in (3) to determine the co-occurrence probability of an overlap of at least $|Z_i \cap Z_k|$. We use the q-value approach [16] suggested for estimating false discovery rate to calculate the adjusted significance of the probability obtained for the redescription between Z_i and Z_{i+1}. For all redescriptions considered in our case studies, the q-value threshold was set to 0.01.

For the case of multisets, elements having unequal weights implies that (3) cannot be used for estimating the probability we seek. Therefore, we employ a simulation methodology wherein we randomly generate multisets of size $|Z_{i+1}|$ with each element associated with a weight assigned to it. We calculate the proportion of these randomly generated sets that have an overlap greater than $\mathcal{J}_w(Z_i, Z_{i+1})$ with Z_i to estimate the probability $p(Z_i, Z_{i+1})$. This process needs to be repeated for randomly chosen descriptors to obtain the distribution of probability required for estimating the q-value.

4 Experimental Results

Our three experimental studies are meant to address diverse questions:

1. What is the tradeoff between number and length of stories mined?
2. How does the time taken to find stories scale with the length of the stories mined?
3. Is the node evaluation criterion for storytelling search $(s_j = g_j + h_j)$ more informed than vanilla breadth first search $(s_j = g_j)$?
4. Do the stories discovered by our algorithm shed domain-specific insights?

We undertake systematic studies over many applications to answer the first three questions and provide illustrative examples to address the last question, where appropriate. The first application characterizes word overlaps in large English dictionaries and illustrates scalability of the implementation and how the different parameter settings affect the quality of stories mined. The second application, involving gene sets in bioinformatics, uses decision trees of depth 2 and showcases the constructive induction capabilities of CARTwheels when used for storytelling. This application and the third, which builds stories between PubMed abstracts, also illustrate interesting nuggets of discovered knowledge.

4.1 Word Overlaps

In our first study, we implement storytelling for the MorphWord puzzle wherein we are given two words, e.g., PURE and WOOL, and we must morph one into the other by changing only one letter at a time (meaningfully). One solution is

$$\text{PURE} \rightarrow \text{PORE} \rightarrow \text{POLE} \rightarrow \text{POLL} \rightarrow \text{POOL} \rightarrow \text{WOOL}.$$

Here, we can think of a word as a set of (letter, position) tuples so that all meaningful English words constitute the descriptors. Each step of this story is an approximate redescription between two four-element sets, having three elements in common. It is important to note that words that are anagrams of each other (e.g., "ELVIS" and "LIVES") will *not* have a Jaccard's coefficient of 1, since position is important.

We harvested words of length 3 to 13 words from the Wordox dictionary of English words (http://www.esclub.gr/games/wordox/), yielding more than 160,000 words. Consistent with the MorphWord puzzle, we restrict all CARTs to be of depth $d = 1$ and study the effect of θ and b on the number of stories possible, length of stories mined, and time taken to mine stories. For ease of interpretation, we recast Jaccard's thresholds in terms of the number of letters in common (lc) between two words. Although MorphWord is traditionally formulated with $lc = 1$, we explore higher lc values as well. Due to space restrictions, we present our results on subsets of the master word list, namely, 5 letter words (L_5) and 10 letter words (L_{10}). In each case, we selected 100,000 pairs of words at random and tried to find stories between them, with different lc and b settings. An example story we mined with five letter words (with setting $lc = 3$) is:

$$\text{BOOTH} \Leftrightarrow \text{BOATS} \Leftrightarrow \text{BEAMS} \Leftrightarrow \text{DEADS} \Leftrightarrow$$
$$\text{GRADS} \Leftrightarrow \text{GRADE} \Leftrightarrow \text{CRAZE} \Leftrightarrow \text{CRASH} \Leftrightarrow \text{FLASH}.$$

Fig. 8 (first column) depicts plots of the fraction of stories (out of 100,000) mined with various story lengths as a function of lc, for a branching factor $b = 5$. In these plots, a story length of 0, rather counterintuitively, implies that no story was found for the word pair considered. The critical story length where the majority of stories are mined steadily increases as lc is increased. This is because, as lc is increased, more overlap is required at each step of the story such that it takes longer for one word to morph into another. At the same time, the total number of stories mined decreases as lc is increased, due to the lack of viable redescriptions.

To study the effect of b on the length of stories mined, we focus our attention on lc values of 2 for L_5 and 5 for L_{10}. Fig. 9 (first column) shows plots of the fraction of stories mined with various lengths as a function of b. As before, a path length of 0 in the plots implies that no story was found for the word pair considered. Here, there are qualitative changes between the two data sets (Fig. 9, first column, top and bottom rows).

For L_5, the lc value chosen (2) supports a significant number of short stories. This lc value affords a high number of possible redescriptions per descriptor (about 20-100), making it highly likely that the A* algorithm will follow a path that leads close to the target word. In other words, b does not have as significant an impact for this data set.

For L_{10}, the lc value chosen contributes to a higher probability of longer stories. As a result, the branching factor b plays a crucial role. This is evident in the case of $b = 1$, where the excessively greedy strategy is often rendered futile. As b increases, the chances of going down toward the target word increases, and more stories are mined.

To study the effect of these parameters on the time required to mine stories, we set $b = 5$ as before for understanding the role of lc. We computed the average time taken to mine a story, for various story lengths, across all pairs of words considered. Fig. 8 (second column) shows plots of this average time against story lengths, for different lc values. In these plots, we have normalized the time

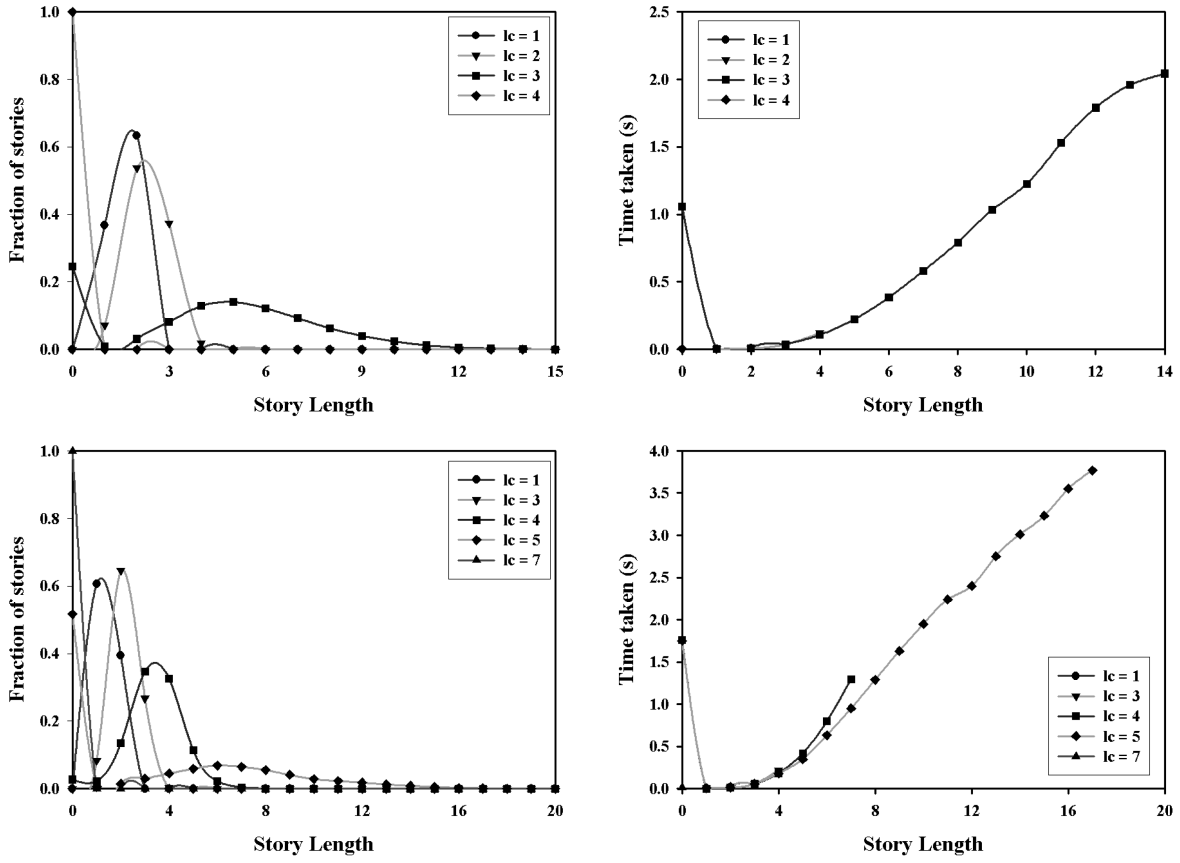


Fig. 8. Word overlap study: (First column) Fraction of stories mined as a function of story length, for different values of lc. (Second column) Average time required to mine stories as a function of story length, for different values of lc. (Top row) Five letter word vocabulary (L_5). (Bottom row) Ten letter word vocabulary (L_{10}).

measurements by calibrating the maximum time to have a value of 1. The actual time taken to find stories ranges from a few seconds to a few minutes (on a 2.3-GHz Apple Xserve G5 computer with 4-GB RAM), depending on the parametric settings as well as the story length. (This holds for the other studies as well.) The plots in Fig. 8 (second column) show that the general behavior in the two graphs is again quite similar. There is a near-linear increase in time required, with steeper increases for lower lc values. This is because the lower lc values cause an increase in the number of possibilities (within the bound of $b = 5$) which must be explored before converging on the shortest path. Also, observe the higher times for story lengths of 0, indicating it takes longer to conclude that stories do not exist. Similar linear trends are observed in time versus the role of b (Fig. 9, second column). Here, steeper profiles are witnessed for higher b values. Once again, this is due to the increase in the number of possibilities, although as Fig. 9, second column (bottom panel) shows, these increases appears to taper off quickly. These figures clearly indicate the underlying tradeoff in mining stories: time versus importance of optimal story lengths.

Fig. 10 highlights the advantage of using the heuristic function described in Table 2. Here, the top row corresponds to plots for L_5 with $lc = 3$ and $b = 10$, and the bottom row corresponds to L_{10} with $lc = 5$ and $b = 10$. The plot on the left for L_5 shows the behavior of the effective branching factor for both BFS and heuristic search. Here, there is a vast difference in the branching factor initially but this difference tends to decrease as the story length increases. This behavior is primarily because as the story length increases, we are more and more likely to encounter nodes that have very few redescriptions (1 or 2) associated with them. Our choice of lc being so high also ensures that the number of redescriptions is low. The branching factor is not binding in these cases. Correspondingly, the plot on the right shows the time comparison of heuristic search against BFS. It is important to note that the time taken for the heuristic search also includes the time taken to compute the heuristic. This plot shows that even though the effective branching factor becomes quite similar for the two cases as story length increases, there is still significant difference between the time taken in mining stories with the heuristic search requiring lesser time.

A more pronounced effect of the heuristic can be seen in the plots from L_{10}. Here, there is a vast difference between the heuristic and BFS for both the effective branching factor and time taken in searching for all story lengths. This is because, for our choice of lc, we find stories that are not very long. This also decreases the chances of encountering nodes with very few redescriptions.

4.2 Gene Sets

In our second case study, we mine stories among descriptors defined over gene sets in the budding yeast *S. cerevisiae*. We draw our descriptors from various bioinformatics vocabularies (e.g., the Gene Ontology (GO), microarray experiment clusters, and experiment ranges) as done in previous work

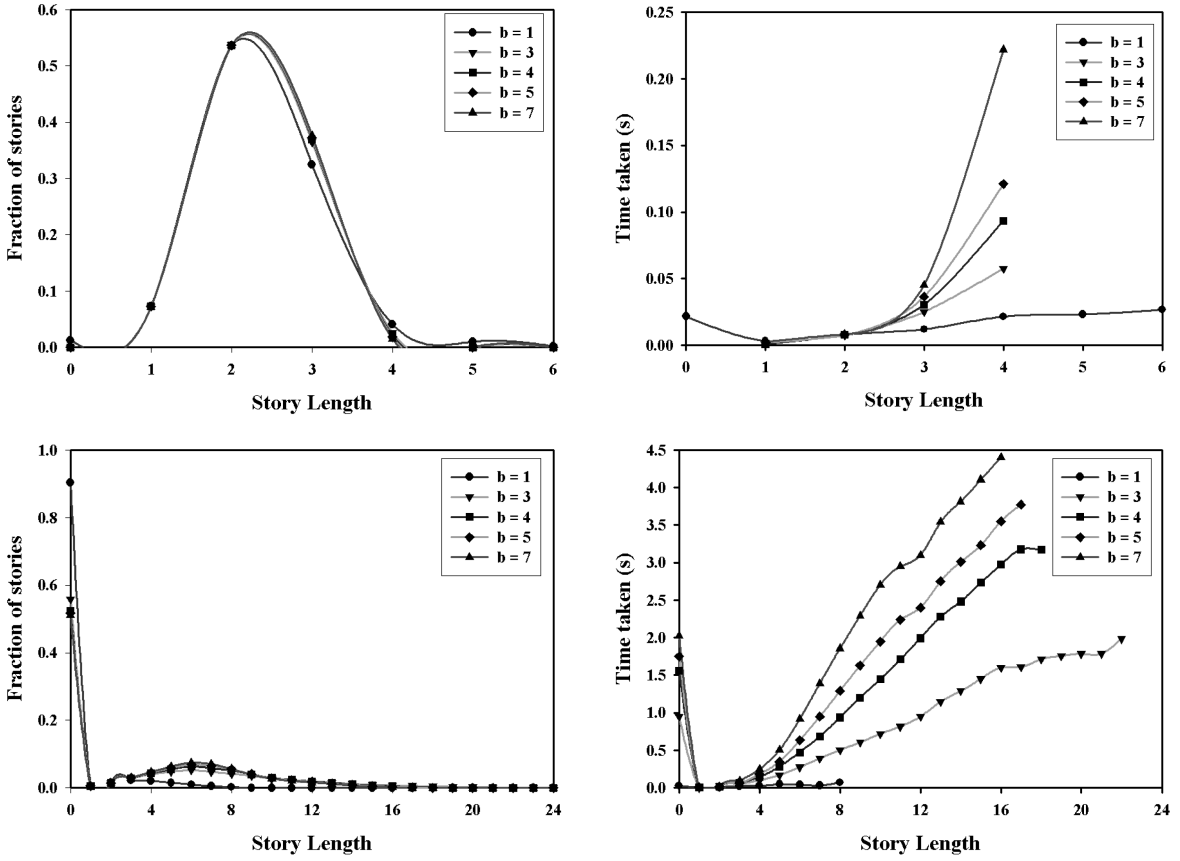


Fig. 9. Word overlap study: (First column) Fraction of stories mined as a function of story length, for different values of b. (Second column) Average time required to mine stories as a function of story length, for different values of b. (Top row) Five letter word vocabulary (L_5). (Bottom row) Ten letter word vocabulary (L_{10}).

(see [2] and [3]). An example of significant story, between the GO categories protein modification and hexokinase, mined for $\theta = 0.5$, $b = 5$, and $d = 2$ is shown in Fig. 11. Observe that the second descriptor in the story involves a set intersection performed by CARTwheels. A unifying feature that links the genes in this story is their common role in nutrient control and carbohydrate metabolism, particularly metabolism of glucose-phosphate. Considering the three genes in the first descriptor, YKL035W is involved in the reversible conversion of glucose-1-phosphate to UDP-glucose via UTP; YJL164C is a cyclic adenosine monophosphate (cAMP)-dependent kinase and binds both YFL033C (glucose repressed, nutrient control) and YIL033C (glycogen accumulation); and YGL158W is a kinase that binds YGL115W (release from glucose repression). Two new genes enter the story with the first redescription, namely, YCL040W (involved in phosphorylation of glucose) and YFR053C (a hexokinase also involved in the phosphorylation of glucose in the glycolysis pathway). In traversing the second redescription, two additional genes appear: YDR516C is involved in phosphorylation of glucose and, most importantly, also binds YCL040W (which is present in earlier redescriptions). YGR052W is a mitochondrial serine/threonine kinase of unknown function. Through the thread of the story, we predict that YGR052W may also be involved in an aspect of glucose metabolism and/or nutrient control.

4.3 PubMed Abstracts

For our final case study, we consider the more than 140,000 publications about yeast in the PubMed index and focus on finding stories between publication abstracts. Each abstract is, hence, a descriptor over terms/keywords. We restrict our CARTs to be of depth 1 and also adopt the weighted Jaccard's measure, as defined in (1), that is more suited to measuring similarity between bags.

To generate keywords, we focused on the 3,756 abstracts containing the keywords "*yeast* AND *stress*". For each of these papers, we used the corresponding title and abstract and split the text into words delimited by space. We removed completely numeric or special character words as well as a set of stop words from these words. We used Porter's stemming and manual inspection to group words that share similarity together. We calculated the IDF values for each word (w) that appears in any document as the log_2 of the ratio of the total number of documents to the number of documents in which w appears. Next, we computed the $TF_w.IDF_w$ for each word that appears in a paper for each of the papers and eliminated 95 percent of these values by setting a threshold of 7 on this value. The words that have $TF_w.IDF_w$ values above 7 were retained for each paper along with their frequency values. This resulted in a total of 6,821 unique words in our study.

For document Z_i, the vector $\mathcal{V}_{Z_i}[w]$ in the definition of weighted Jaccard's in (1) is given by $\mathcal{V}_{Z_i}[w] = TF_w.IDF_w$. Two examples of significant stories we mined using this

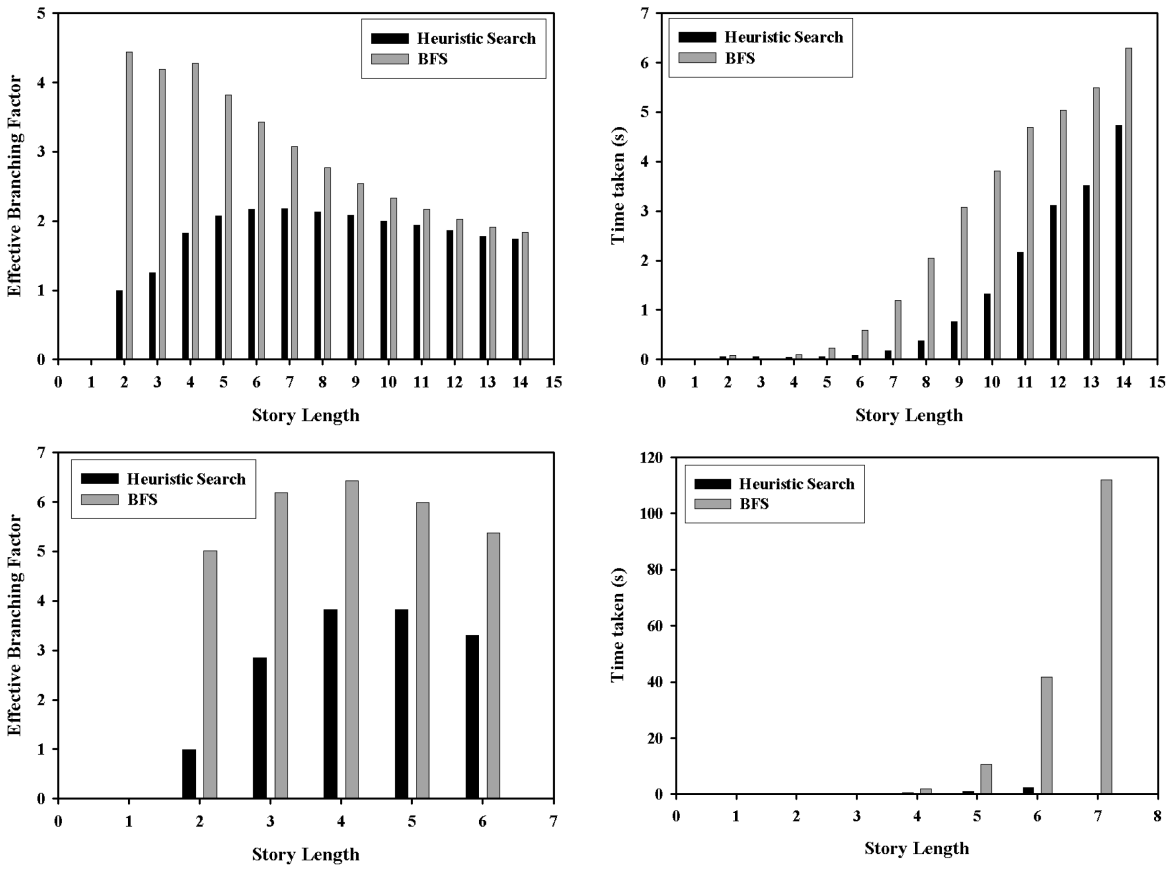


Fig. 10. Word overlap study: Comparison of the behavior of heuristic-based search with BFS for (top) L_5, with $lc = 3$ and $b = 10$ and (bottom) L_{10}, with $lc = 5$ and $b = 10$. The plots on the left compare the average value of the effective branching factor for various story lengths. The plots on the right compare the time taken using the two search strategies.

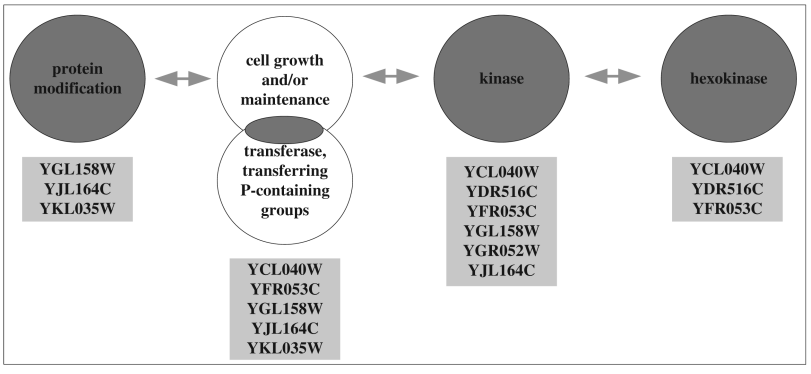


Fig. 11. A significant story among gene sets from protein modification to hexokinase.

function are given in Figs. 12 and 13 (the PubMed IDs and publication dates are given alongside).

The first story (see Fig. 12), mined with $\theta = 0.2$, $b = 5$, begins with a high-throughput experiment that links chemical stress to gene expression in *Saccharomyces cerevisiae*, and ends with heat stress transcription factors in tomato. The "story line" was initiated through comparisons between oxidative and heavy metal stresses. This led to a paper identifying a gene from *Candida sp.* that was expressed when the cells are exposed to cadmium but not copper, mercury, lead, or manganese. Interestingly, a BLAST search for the encoded protein sequence indicates that the protein is novel. The link between tomato heat stress transcription factors and a cadmium-specific gene with no known match in the current databases was through work with the fission yeast *Schizosaccharomyces pombe*, where a study looked specifically at heat and cadmium stress responses. This story, hence, illustrates the key players in the systems biology of related chemical stresses.

For our next example, we mined for stories that follow a strict sequential order of publication year. One example here is shown in Fig. 13, with settings $\theta = 0.4$, $b = 5$. The featured compound in this story—cAMP—is a signaling molecule found in all forms of life. In yeast, it serves as a relay for glucose levels, effecting distinct responses in accord with nutrient need and availability. This story, presented "backward in time" in Fig. 13, starts with a paper that addresses specific changes in gene transcription

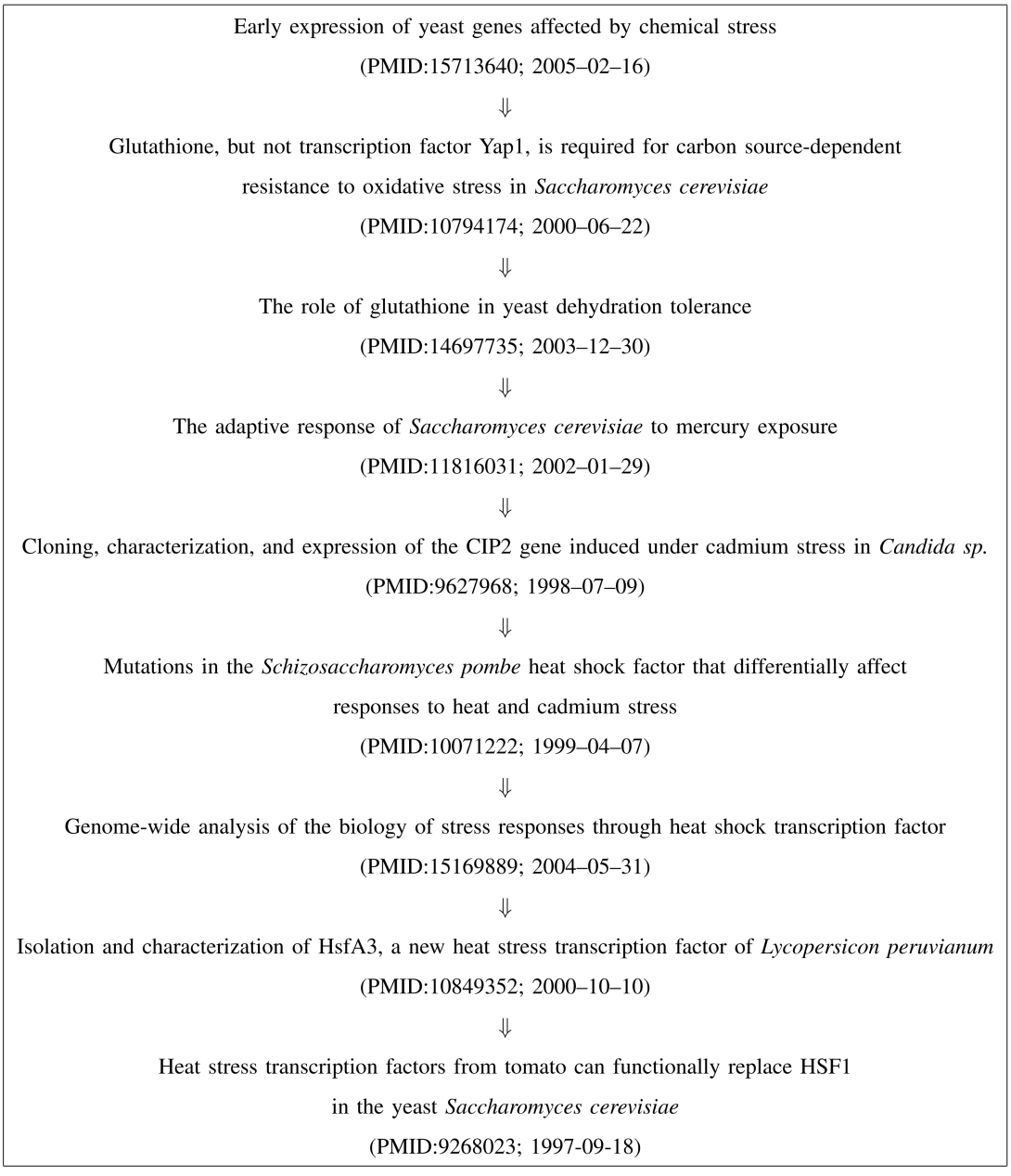


Fig. 12. An example of significant story among PubMed abstracts relating chemical stresses.

modulated by cAMP levels in *Schizosaccharomyces cerevisiae*. It was connected with a paper that also addressed the relationship between nutrients and cAMP, but with a different yeast (*Saccharomyces cerevisiae*) and with a different emphasis (partners upstream and downstream of where cAMP intersects the pathway). The third paper in the story describes the relationship between a serine/threonine protein kinase (Snf1) and nutrient levels, and how it is related to AMP concentrations (the degradation product of cAMP), while the fourth paper links catalase gene expression to cAMP. Together these four papers provide a continual story line on how yeast responds to changes in nutrient levels. Interestingly (at the time of this writing), Paper #4 has been cited 82 times, Paper #3 114 times, and Paper #2 182 times (Paper #1 is too new to be heavily cited). However, the only connection between them in the citation indices is that Paper #2 has referenced Paper #4.

5 Related Research

We survey related work in various categories.

In *information visualization* (e.g., see [17]), storytelling has been viewed not as a data mining tool but as an information organization tool based on narrative structures from real life. The emphasis of software developed here is to provide templates and diagrammatic semantics that make the *manual* process of constructing stories easier, whereas we focus on *automated* approaches to mine stories.

In *topic tracking* [18], the goal is to postprocess search results into story lines by analyzing bipartite graphs of document-term relationships. Here, a story is a thread of related documents with temporal as well as semantic coherence. These works are focused on *unsupervised* discovery of all threads, whereas we focus on *directed* (but not necessarily temporal) stories between given start and end points.

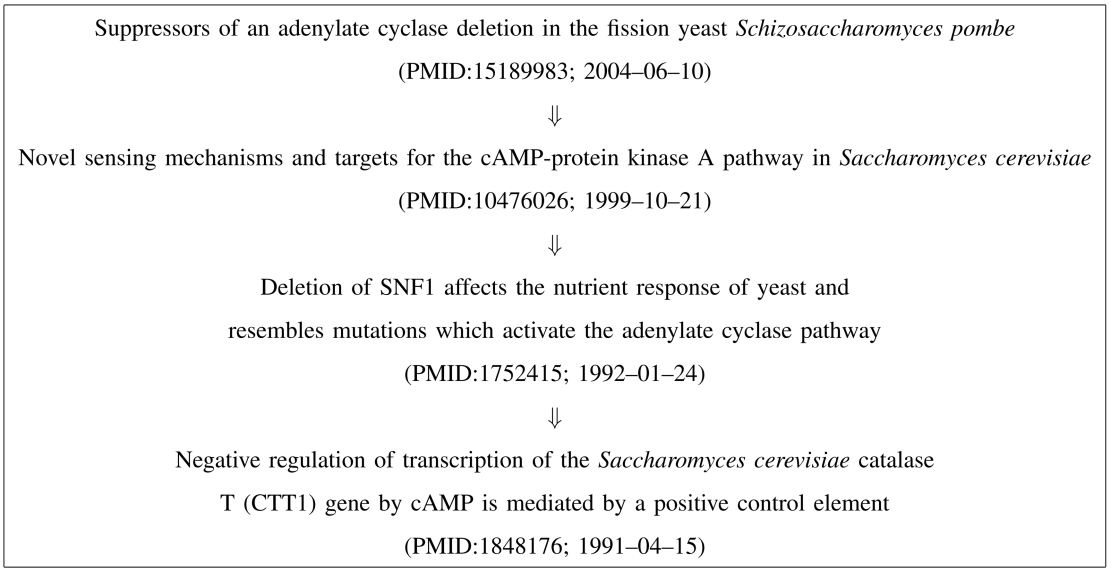


Fig. 13. An example of significant story among PubMed abstracts around cAMP levels.

Link mining [19] begins with data that can be modeled as a collection of links and, in this sense, storytelling can be approached as a problem of analyzing overlap relationships expressed as a graph. However, as stated earlier, such graphs are hard to materialize in entirety and, hence, storytelling must multiplex the task of positing intermediaries with focused search toward the end point of the story.

Storytelling is closest in spirit to *literature-based discovery systems* such as Swanson's Arrowsmith [20]. In 1985, Don Swanson, quite by accident, connected two different pieces of information across medical literature that led him to the hypothesis that magnesium deficiency may play a role in certain types of migraine, a result since subsequently proven. The Arrowsmith project aims to automate this process by looking for relationships among articles in biomedical literature. In storytelling, we extend this idea to find longer chains of relationships by bringing set-theoretic constructions into the mining process.

Finally, in *language modeling research* such as the "one story one flow" hypothesis [21], a story is a hidden Markov model (HMM) that helps identify segments of news stories that follow each other. Our goal is to present a generalized framework for storytelling in generalized set systems, not just documents. Nevertheless, new similarity functions based on language modeling can be utilized in our framework, besides the standard (weighted) Jaccard's coefficient.

6 Discussion

By defining stories as chains of redescriptions, we have been able to design a storytelling algorithm as A^* search around the outputs of a redescription mining algorithm. We have demonstrated the scalability of this approach using the Word overlaps case study and showcased its potential for knowledge discovery using the gene sets and PubMed abstracts case studies.

Storytelling can serve multiple uses besides data mining from set systems. It can be used as an information exploration tool for large document collections, as a postprocessing approach for search results, and to chain together research results from multiple publications. All of these next-generation applications can be viewed as forms of "combinatorial information integration," and storytelling promises to be a valuable algorithmic engine behind such applications.

In future work, we aim to investigate other metrics for evaluating stories besides story length, e.g., based on the number of objects temporarily brought into the story, the story's conformance to prior background knowledge, or using overlap metrics that better mirror a domain scientist's conception of set similarity. We also aim to explore connections to works that characterize the structure of partitions [22], [23] and investigate whether story lines can be designed around paths in such discrete structures. We also intend to generalize from propositional to predicate vocabularies and cast storytelling in the context of relational redescriptions. This will help provide structured stories that follow a template of connections. Our eventual goal is to establish storytelling as an important tool for reasoning with data and domain theories.

Acknowledgments

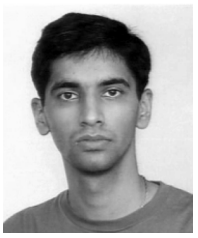
A preliminary version of this paper appeared as a short paper in the *Proceedings of the 12th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '06)*. This work was done while D. Kumar was with the Department of Computer Science at Virginia Polytechnic Institute and State University.

References

[1] L. Parida and N. Ramakrishnan, "Redescription Mining: Structure Theory and Algorithms," *Proc. 20th Nat'l Conf. Artificial Intelligence (AAAI '05)*, pp. 837-844, 2005.

[2] N. Ramakrishnan, D. Kumar, B. Mishra, M. Potts, and R. Helm, "Turning CARTwheels: An Alternating Algorithm for Mining Redescriptions," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '04)*, pp. 266-275, 2004.

[3] M. Zaki and N. Ramakrishnan, "Reasoning About Sets Using Redescription Mining," *Proc. 11th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '05)*, pp. 364-373, 2005.
[4] D. Kumar, N. Ramakrishnan, M. Potts, and R. Helm, "Algorithms for Storytelling," *Proc. 12th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '06)*, pp. 604-610, 2006.
[5] D. Grossman and O. Frieder, *Information Retrieval: Algorithms and Heuristics*. Springer, 2004.
[6] R. López de Mántaras, "A Distance-Based Attribute Selection Measure for Decision Tree Induction," *Machine Learning*, vol. 6, pp. 81-92, 1991.
[7] A. Nanopoulos and Y. Manolopoulos, "Efficient Similarity Search for Market Basket Data," *The VLDB J.*, vol. 11, no. 2, pp. 138-152, 2002.
[8] S. Sarawagi and A. Kirpal, "Efficient Set Joins on Similarity Predicates," *Proc. ACM SIGMOD '04*, pp. 743-754, June 2004.
[9] N. Mamoulis, D. Cheung, and W. Lian, "Similarity Search in Sets and Categorical Data Using the Signature Tree," *Proc. Ninth IEEE Int'l Conf. Data Eng. (ICDE '03)*, pp. 75-86, 2003.
[10] M. Morzy, T. Morzy, A. Nanopoulos, and Y. Manolopoulos, "Hierarchical Bitmap Index: An Efficient and Scalable Indexing Technique for Set-Valued Data," *Proc. Seventh East-European Conf. Advances in Databases and Information Systems (ADBIS '03)*, pp. 236-252, Sept. 2003.
[11] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via Hashing," *Proc. 25th Int'l Conf. Very Large Data Bases (VLDB '99)*, pp. 518-529, Sept. 1999.
[12] A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher, "Min-Wise Independent Permutations," *J. Computer and System Sciences*, vol. 60, no. 3, pp. 630-659, June 2000.
[13] A. Moore and M. Lee, "Cached Sufficient Statistics for Efficient Machine Learning with Large Datasets," *J. Artificial Intelligence Research*, vol. 8, pp. 67-91, 1998.
[14] C. Aggarwal, J. Wolf, and P. Yu, "A New Method for Similarity Indexing of Market Basket Data," *Proc. ACM SIGMOD '99*, pp. 407-418, 1999.
[15] S. Tavazoie, J. Hughes, M. Campbell, R. Cho, and G. Church, "Systematic Determination of Genetic Network Architecture," *Nature Genetics*, vol. 22, no. 3, pp. 213-215, 1999.
[16] J. Storey and R. Tibshirani, "Statistical Significance for Genome-Wide Experiments," *Proc. Nat'l Academy of Sciences*, vol. 100, pp. 9440-9445, 2003.
[17] A. Kuchinsky, K. Graham, D. Moh, A. Adler, K. Babaria, and M. Creech, "Biological Storytelling: A Software Tool for Biological Information Organization Based Upon Narrative Structure," *ACM SIGGROUP Bull.*, vol. 23, no. 2, pp. 4-5, Aug. 2002.
[18] R. Guha, R. Kumar, D. Sivakumar, and R. Sundaram, "Unweaving a Web of Documents," *Proc. 11th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '05)*, pp. 574-579, 2005.
[19] L. Getoor, "Link Mining: A New Data Mining Challenge," *ACM SIGKDD Explorations Newsletter*, vol. 5, no. 1, pp. 84-89, 2003.
[20] D. Swanson and N. Smalheiser, "An Interactive System for Finding Complementary Literatures: A Stimulus to Scientific Discovery," *Artificial Intelligence*, vol. 91, no. 2, pp. 183-203, 1997.
[21] P. Fung and G. Ngai, "One Story, One Flow: Hidden Markov Story Models for Multilingual Multidocument Summarization," *ACM Trans. Speech and Language Processing*, vol. 3, no. 2, pp. 1-16, July 2006.
[22] M. Meila, "Comparing Clusterings by the Variation of Information," *Proc. 16th Ann. Conf. Learning Theory (COLT '03)*, pp. 173-187, 2003.
[23] D. Simovici and S. Jaroszewicz, "An Axiomatization of Partition Entropy," *IEEE Trans. Information Theory*, vol. 48, no. 7, pp. 2138-2142, 2002.



Deept Kumar received the BS degree in chemical engineering from the Indian Institute of Technology, Mumbai, in 1998 and the MS degree in environmental engineering and the PhD degree in computer science in 2007 from Virginia Polytechnic Institute and State University, Blacksburg. After completion of his undergraduate studies, he was with Infosys Technologies Ltd., Bangalore, as a software engineer for a year. He is currently a senior data analyst for a start-up operating out of San Francisco.

Naren Ramakrishnan received the PhD degree in computer science from Purdue University, in 1997. He is an associate professor of computer science and a faculty fellow at Virginia Polytechnic Institute and State University. He also serves as an adjunct professor at the Institute of Bioinformatics and Applied Biotechnology (IBAB), Bangalore, India. His research interests include computational science, problem-solving environments, mining scientific data, and information personalization. He currently serves on the editorial board of *Computer* and is a member of the IEEE Computer Society.

Richard F. Helm received the BS degree from the College of Environmental Science and Forestry, State University of New York, Syracuse, in 1982 and the PhD degree from the University of Wisconsin, in 1987. He was a postdoctoral researcher at both Oregon State University and the USDA-ARS Dairy Forage Research Center (Madison, Wisconsin) prior to becoming an assistant professor in the Department of Wood Science and Forest Products, Virginia Polytechnic Institute and State University (Virginia Tech) in 1992. In 2004, he transferred to the Department of Biochemistry, Virginia Tech, where he is currently an associate professor. His research interests include analytical tools to answer biochemical questions (vigen.biochem.vt.edu). He is presently the director of the Virginia Tech Mass Spectrometry Incubator (www.mass.biochem.vt.edu).

Malcolm Potts received the BSc degree (with honors) in botany, the PhD degree in microbiology, and the DSc degree from Durham University, United Kingdom, in 1995. He is currently on leave as a lead scholar in the Department of Biological and Environmental Sciences, Qatar University, State of Qatar. He is a professor of biochemistry and a codirector of the Virginia Tech Center for Genomics, Virginia Polytechnic Institute and State University, Blacksburg. He completed his postdoctorals at Hadassah Medical School, Jerusalem; the Heinz Steinitz Marine Biology Laboratory, Elat University, Israel; Karl von Ossietsky University, Oldenburg, Germany; and Florida State University. His research interests include structural, physiological, and molecular basis for environmental stress tolerance—especially desiccation tolerance of cyanobacteria.