WIM: an Information Mining Model for the Web
(Extended Abstract)

Ricardo Baeza-Yates[1]
[1]Center for Web Research, CS Dept.
University of Chile – Santiago, Chile
&
ICREA Research Professor
Tech. Dept., Pompeu Fabra Univ.
Barcelona, Spain

Álvaro R. Pereira Jr[2] Nivio Ziviani[2]
[2]Department of Computer Science
Federal University of Minas Gerais
Belo Horizonte, Brazil
{alvaro, nivio}@dcc.ufmg.br

Abstract

This paper presents an extended abstract of a model to mine information in applications involving Web and graph analysis, referred to as WIM *(Web Information Mining) Model. We demonstrate the model characteristics using a Web warehouse, where nodes represent Web pages and edges represent hyperlinks. In the model, objects are always sets of nodes and belong to one class. We have physical objects containing attributes directly obtained from Web pages and links, as the title of a Web page or the start and end pages of a link. Logical objects can be created by performing predefined operations on any existing object. WIM has a concise set of eleven operators. In this paper we summarizes the model components and give examples of views. A view is a sequence of operations on objects, and it represents a way to mine information in the graph.*

1 Introduction

The Web is a collection of semistructured documents. Most documents are of the type HTML, with tags containing meta information about pieces of the document. In this paper we present an extended abstract of a model to mine information in applications involving Web and graph analysis, referred to as *WIM* (Web Information Mining) Model. There are three distinct data types in the Web: semistructured multimedia content, hyperlink (or just link) structure and usage data in the form of Web logs [11]. WIM models these three data types as objects that represent some view of a Web warehouse, manipulated by a formal algebra with distinct operators.

Our algebra is composed of two categories of *primitive operators*: extended relational algebra operators and graph operators, and two other operators. The extended relational algebra operators are used to manipulate the content of databases, with extensions of the traditional relational algebra operators including set operators. The graph operators are used in applications where the information of both nodes and edges of the graph is important, such as singular value decomposition (as *Pagerank* [8]).

This work is organized in the following sections: Section 2 presents details of the model. The primitive operators are presented in Section 3. As practical examples, we show in Section 4 different views for two distinct purposes. Related works are presented in Section 5. Section 6 contains the conclusions of our work.

2 The Model

The information mining model for the Web, WIM, is based on the object oriented paradigm. Objects are instances of classes and incorporate sets of elements. We apply the model to the Web graph, where the elements are initially web pages and web links. The objects are modified by operations performed on the elements. Main components of the model, as well as the relations among them, are shown in Figure 1 and explained below. More details about WIM can be found in [1].

Super Node is the superclass. *Node*, *Relation* and *Log* are subclasses of *Super Node*. Every *object* has the attribute id as an internal reference to its elements. The class *Relation* allows relationships among object nodes of other classes. In spite of most applications demand graphs such that each edge connects two nodes, WIM supports hypergraphs [4]. In this paper we often refer to elements of the classes Node and Relation by *nodes* and *relations*, respectively. The class log is not used in applications shown in this paper, but it is exemplified in [1].

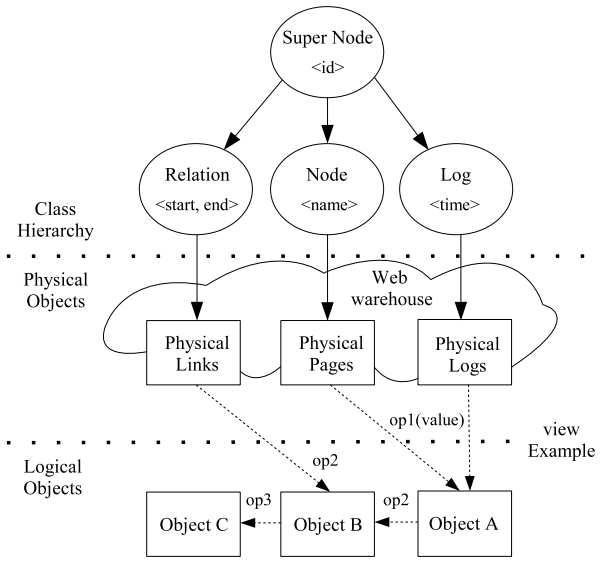

Figure 1. Main components of WIM.

Objects are sets of elements representing instances of some class. Objects *physical pages* and *physical links* represent, respectively, all the Web pages and links among pages in a given web warehouse crawled. These objects have physical attributes that can be directly obtained from the elements of the object. Some examples of attributes for the object physical pages are: the page URL, the page length, the time of the last modification on the page, the time that the page was crawled, the email addresses that are pointed in the page, the page text, title, etc.

The *operations* are functions that modify the object characteristics, generating a new object or returning a value. We identify by *primitive operations* the operations that are important to the model and that cannot be implemented by using other existing operations. A *view* is a query performed by a sequence of operators and objects, where the operators are applied to input objects or to objects returned by other operators. It returns a special object which is the result of the view. Objects returned by views can be *materialized* or not [5], depending on time and space costs and on the application. Materialized objects can be used in any other view as input objects.

Dynamic Operations are views with generic input objects and parameters, normally useful in other views. In the view $Example$ above, the input objects are Physical Pages, Physical Logs and Physical Links. Only one extra parameter is required: $value$, for $op1$.

3 Primitive Operations

In this section we present a summary of the WIM primitive operators, with their main characteristics. Further details about the operators can be found in [1].

3.1 Extended Relational Algebra Operators

Select: select tuples of the object. Two options are possible: $value$ and $top\text{-}k$. The option $value$ allows the selection of only tuples that satisfy a condition for a given attribute. $Top\text{-}k$ returns only a given number of tuples in the top ranking, for an ordered list.

Project: return an object with only a given set of attributes. Two options are possible: $normal$ and $relation$. With the option $normal$, Project works as the traditional relational algebra operator with the same name. $Relation$ is used to project all the relations of an hypergraph.

Merge: three options are possible: $union$, $intersection$ and $difference$. The $union$ returns all the tuples that occur in any object involved. The $intersection$ returns only the tuples that occur in the two objects involved. The $difference$ returns an object with the tuples that occur in an object A, excepting those ones that also occur in an object B.

Join: Four distinct options are possible: $objects$, $bags$, $non\text{-}directed$ and $disjoin$. Join $objects$ works as the Join operator of the relational algebra. Join $bags$ converts a bag to a set, that is, without duplications of tuples. A new attribute $quantity$ is insert, to count the amount of previous same tuples. *Non-directed* is used in relation objects to eliminate the direction of the relations. The $disjoin$ option is used to separate items that are arrays, where every element of the array generates a new tuple into the object.

Order: sort the tuples in $increasing$ or $decreasing$ order by one or more attributes.

3.2 Graph Operators

Add Relation: if the input object belongs to the class relation, add new relations. The graph returned is a hypergraph. If the input object belongs to the class node, convert it to the class relation. The following options are available, for both relation and node input objects: $fixed$, $cross\ product$ or $same\ attribute$. $Fixed$ means that two given attributes are the start and end attributes of the new object. $Cross\ product$ converts all distinct node pairs of the object in relations, similarly with a cartesian product of the object with itself. $Same\ attribute$ means that a relation is inserted among every pair of nodes with the same attribute value, for a given attribute.

Link Distance: also a way for adding new relations, that can be done according to the methods: $co\text{-}citation$, $bibliographic\ coupling$ or $distance\text{-}k\ transitivity$. A new attribute $distance$ is added to store the previous distance among nodes of the new relation object.

Connected Nodes: identify the $connected$ components, the $strongly$ connected components or the $minimum\ spanning\ tree$ of a graph [4], always adding a new attribute $subgraph$ to indicate the subgraph number for which the relation belongs to.

Singular Value Decomposition: applied to an object of the class relation, the operator returns an object of the class node, with one more attribute according to the $Pagerank$, $Authority$, Hub, or any other measure defined by matrix algebra.

3.3 Other Operators

Content Similarity: if applied only to a relation object, compare a given textual attribute of the start and end nodes of every relation. If applied to a node object, the operator is used to calculate the similarity of a given $query$ and a textual attribute of a node object. A new attribute $similarity$ is added.

Calculus: this operator can be $binary$ or $unary$. If binary, it returns an object with the sum, $difference$, $multiplication$ or $division$ of the values or content (if the attribute is non numerical) of two given attributes. If unary, four options are possible: $count$, $constant$, $normalize$ and $function$. $statistical$ functions are allowed.

4 Some Examples

Some useful operations were not defined as operators. One example is an operation to change an object from the class relation to the class node. This is an example of operator that is not primitive, since it can be constructed by using the existing operators. Figure 2 shows the dynamic operator *Relation to Node*, that is formally defined as:
$Result_{ClassNode} = RelationToNode(O_{ClassRelation}, mergeMode, otherAttr^*)$;
where: $mergeMode$ is one of the possible options for the primitive operator Merge. The parameter $otherAttr$ allows the addition of other attributes than $start$ and end, in the projection step. The remaining parameters used are fixed, what means that the user of the dynamic operator cannot modify them.

4.1 Clustering

A cluster is a collection of data objects that are *similar* to one another within the same cluster and are *dissimilar* to the objects in other clusters. The process of grouping a set of objects into classes of similar objects is called *clustering*. Related to graph applications, the clustering activity consists of partitioning the graph. The dynamic operator

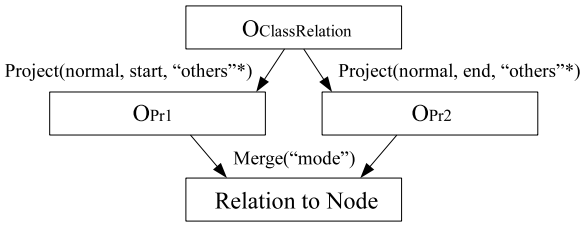

Figure 2. The dynamic operator Relation to Node.

Cluster Nodes groups nodes according to the presence of a relation. The application of Cluster Nodes to one object of the class relation results in an object belonging to the class node, with the addition of the attribute $subgraph$. This new attribute stores the cluster number which each node belongs to. Figure 3 shows the dynamic operator Cluster Nodes.

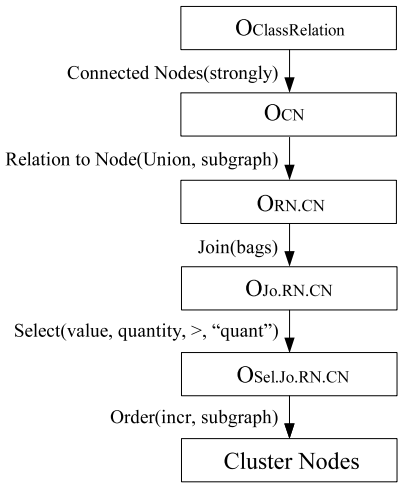

Figure 3. The dynamic operator Cluster Nodes.

The dynamic operator works as follows. Initially the operator Connected Nodes is applied to an object of the class relation. The attribute $subgraph$ is added and stores the subgraph number which the relation belongs to. Following the dynamic operator Relation to Node is used with the option $union$. Join and Select are used to consider only the clusters with a minimal number of nodes (the constant user-defined $quant$). At the end the nodes are sorted using the operator Order.

Note that the object returned contains the list of node numbers and the subgraph number which the nodes belong to. Most views that use this operator may need to merge this result with an object of the class node, returning the names or other information of the clustered nodes. Is is possible

to use the dynamic operator just presented to cluster nodes in labeled graphs. The simplest way may be to perform the operator Select before the Cluster Nodes, eliminating edges that do not reach a given threshold.

4.2 Similar Pages

We desire to construct a view to identify similar pages in a Web warehouse. The most important decision required is related with the space of the comparison. If we want to identify similar pages, in some instant we will need to compare a subset of pages from the data warehouse. We identify the following extreme possibilities: to compare all the documents of the data warehouse would be very inefficient, and to compare only the pairs of documents that have a physical linkage may not return a good result. Figure 4 shows the view *Text Similarity* for the second possibility.

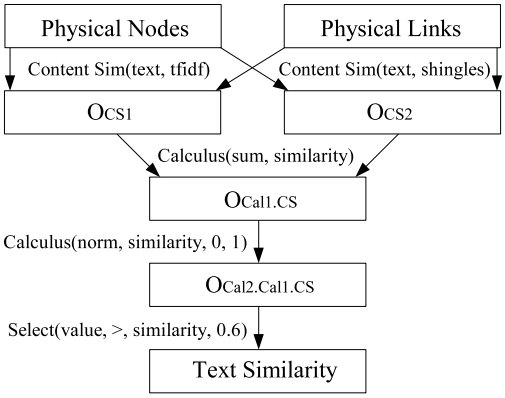

Figure 4. The view Text Similarity.

The view Text Similarity is explained in the following. Initially the operator Content Similarity is used twice, with different similarity measures. In this step, several other comparison heuristics could be used and merged. A new attribute is added to the objects generated, that is, the similarity measure for the two nodes compared. The two similarity values are summed and later normalized between the interval from 0 to 1. The pairs of nodes (the relations) with similarity greater than 60% are returned. The dynamic operator *Text Similarity* can be defined according to the view just presented, for future use:
$Result_{ClassRelation} = TextSimilarity(O_{ClassNode}, O_{ClassRelation}, similarityValue)$;

For Web applications using a very large data warehouse, the list of pairs with similar pages might not be interesting, since it might also be very large. It is interesting to identify the largest similar page clusters. This task is considered in the next view. As mentioned before, one problem of the view Text Similarity shown in Figure 4 is that the answer space is reduced, excluding pages that are similar. It is important to compare not only pages that have a link relation, but all pairs of pages that occur together in the same strongly connected component in the Web graph. Thus, every node of each strongly connected component is compared with every other node of its component. The view *Text Clusters* is shown in Figure 5.

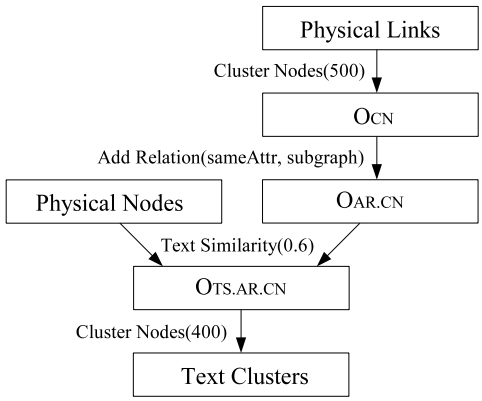

Figure 5. The view Text Clusters.

The view Text Clusters is explained next. The object Physical Links is an input parameter of the dynamic operator Cluster Nodes. The object O_{CN} belongs to the class nodes, and contains all the strongly connected components with more than 500 nodes. The operator Add Relation is used to establish a relation among every possible pair of nodes that belong to the same subgraph. The object $O_{NR.CN}$ is returned and used as parameter of the dynamic operator Text Similarity, together with the object Physical Nodes. Since we are interested in the clusters of similar objects, the dynamic operator Cluster Nodes is applied again. The view Text Clusters contains all the clusters of similar pages with more than 400 nodes.

Please note that in the view Text Clusters we use an heuristic to increase the space of the comparison, that is, to compare every pair of nodes of each strongly connected component. Another possible heuristic is to apply partially the operator Link Distance to the Web graph, with three possibilities: co-citation, bibliographic coupling and distance-k transitivity, and later merging the objects returned. The few steps just described would substitute the first application of the operator Cluster Nodes shown in Figure 5.

5 Related Work

The Web is viewed as a directed graph in [7], using a database perspective. It uses SQL-like query language for the Web search and also retains the Web topological structure to construct a Web algebra to manipulate objects of the model. As an extension of [7], [2] introduces Whoweda, a

project aiming to manage and access heterogeneous information on the Web. Design and research issues in a Web warehouse system are discussed, including algebraic operators for Web information access and manipulation, Web data visualization, and Web knowledge discovery. The operator join for Whoweda is introduced in a more recent work [3].

Related to graph applications, some works propose extensions for the relational algebra model, changing or introducing new operators and elements specifically for graphs. The graph application used in [6] is the electrical network maintenance system. The operators are divided in binary and unary. Examples of binary operators are union and intersection, and examples of unary operators are select and project.

Raghavan and Garcia-Molina [10] recently worked on complex expressive Web queries for a Web warehouse. They view a web warehouse simultaneously as a collection of Web documents, as a navigable directed graph, and as a set of relational tables storing Web pages properties. This work is closer to WIM than other works introduced in this section, with some important distinctions. In our model we extend the relational algebra and the algebra used in [10], allowing operations for relation objects. We also extend the concept of node, which is not limited to web pages, and use hypergraph.

6 Conclusions

We have introduced a model to mine information in Web and graph applications, referred to as *WIM* (Web Information Mining) Model. According to the model, the nodes are represented by an object of the subclass *Node* and the edges by an object of the subclass *Relation*. In this work we focused on the World Wide Web as main application. However, WIM can be applied in other graph applications as bibliographic databases where links are citations, or telephone transactions where pages are client information and links are phone calls. As extensions of the examples shown in this paper, we have developed views for applications related to data mining and information retrieval [9].

In the model, a view is a sequence of operations on objects, and it represents a way to mine information in the graph. As practical examples, we developed views for clustering and for identifying similar pages in a Web warehouse. The examples demonstrated some important characteristics of WIM as: *modularity*, since a view can be converted in a dynamic operator and used in other views; *flexibility*, since we can have distinct views to solve the same problem, and we can adjust internal parameters for the views; and the *applicability* of the model for graph problems.

Acknowledgements

This work was supported by GERINDO Project–grant MCT/CNPq/CT-INFO 552.087/02-5, CYTED VII.19 RIBIDI Project, P01-029F of the Millennium Scientific Initiative, Mideplan, Chile, and CNPq Grants 30.5237/02-0 (Nivio Ziviani) and 14.1636/2004-1 (Álvaro R. Pereira Jr).

References

[1] R. Baeza-Yates, A. R. Pereira-Jr, and N. Ziviani. WIM: an information mining model for the web. Technical Report RT.DCC – 006/2005, Department of Computer Science, Federal University of Minas Gerais, Belo Horizonte, Brazil, April 2005. http://www.dcc.ufmg.br/˜alvaro/bpz05.pdf.

[2] S. S. Bhowmick, S. K. Madria, W.-K. Ng, and E.-P. Lim. Web warehousing: Design and issues. In *ER Workshops*, pages 93–104, 1998.

[3] S. S. Bhowmick, W.-K. Ng, S. K. Madria, and M. K. Mohania. Constraint-free join processing on hyperlinked web data. In *4th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'2000)*, pages 255–264. Springer-Verlag, 2002.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. MIT Press/McGraw-Hill, San Francisco, CA, 1990.

[5] H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database Systems: The Complete Book*. Prentice Hall, October 2001.

[6] A. Gutirrez, P. Pucheral, H. Steffen, and J.-M. Thvenin. Database graph views: A practical model to manage persistent graphs. In *20th International Conference on Very Large Data Bases*, pages 391–402, Santiago, Chile, september 1994.

[7] W.-K. Ng, E.-P. Lim, C.-T. Huang, S. Bhowmick, and F.-Q. Qin. Web warehousing: An algebra for web information. In *Advances in Digital Libraries Conference (ADL'98)*, pages 228–237. IEEE Computer Society, 1998.

[8] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the Web. Technical Report CA 93106, Stanford Digital Library Technologies Project, Stanford, Santa Barbara, January 1998.

[9] A. R. Pereira-Jr and R. Baeza-Yates. Applications of an information mining model to data mining and information retrieval tasks. In *First DEXA International Workshop on Integrating Data Mining, Databases and Information Retrieval (IDDI'05)*, Copenhagen, Denmark, August 2005. IEEE Computer Society Press. To appear.

[10] S. Raghavan and H. Garcia-Molina. Complex queries over web repositories. In *Very Large Data Bases (VLDB'03)*, pages 33–44, Berlin, Germany, September 2003.

[11] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: discovery and application of interesting patterns from web data. *ACM SIGKDD Explorations*, 1(2):12–23, january 2000.