WIM: an Information Mining Model for the Web

Ricardo Baeza-Yates[1] Álvaro R. Pereira Jr[2] Nivio Ziviani[3]

[1]Center for Web Research, CS Dept.
Univ. of Chile – Santiago, Chile
&
ICREA Research Professor
Tech. Dept., Pompeu Fabra Univ.
Barcelona, Spain
ricardo.baeza@upf.edu

[2,3]Department of Computer Science
Federal Univ. of Minas Gerais
Belo Horizonte, Brazil
{alvaro,nivio}@dcc.ufmg.br

Abstract

This paper presents 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. The Web data in the warehouse is modeled as a graph, 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. In this paper we present the model components, propose a set of eleven operators 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. As practical examples, we present views for clustering nodes and for identifying related item sets.

1 Introduction

In the last years, the amount of accessible digital information has grown enormously. Estimations say that between 1999 and 2004 the humankind has published in the Web a volume of data equivalent to all the information generated before 1999. The World Wide Web (or just Web) can be seen as a textual and multimedia distributed database that reaches hundreds of terabytes. This amount of data represents new challenges in many areas in computer science, particularly to database, information retrieval, data mining and Web mining areas [5, 3].

The Web is a collection of semistructured documents. Most documents are of in HTML, with tags containing meta information about pieces of the document. There are three distinct data types in the Web: semistructured multimedia content (or Web pages), hyperlink (or just link) structure and usage data in the form of Web logs [30]. A system used for storing, retrieving and managing large amounts of any type of data is called a data warehouse. A particular data warehouse containing data from the Web is known as a Web warehouse.

In this paper we present a model to mine information in applications involving graph analysis. Our model is referred to as WIM – Web Information Mining – model. The motivation behind the model is to have a platform for intensive but simple Web mining processing. Currently, most Web mining applications need *ad-hoc* solutions which are not easy to maintain and scale. Our goals are to provide a modular, flexible, extensible, and scalable testbed, with the disadvantage of a potential performance lost due to its generality. Nevertheless, we believe that our approach will allow faster analysis and hence, more results.

To demonstrate the model characteristics we use a Web warehouse. In the Web warehouse, Web data is modeled as a graph, where nodes represent Web pages and edges represent hyperlinks. WIM models the Web pages, the link structure and logs as objects that represent some view of a Web warehouse, manipulated by a formal algebra with eleven operators. The objects are instances of classes that incorporates sets of elements. Physical objects contain attributes directly obtained from Web pages and links.

This work is organized as follows. Section 2 presents details of the WIM model. Section 3 introduces the primitive operators. Section 4 presents practical examples, showing different views for two distinct applications: clustering nodes and identification of related item sets in the Web

warehouse. Section 5 presents related work. Finally, Section 6 presents the conclusions of our work.

2 The Model

To fulfill our main goals, that is, modularity, flexibility, extensibility, and scalability; we decided to use an object oriented approach for the model, coupled with the support of efficient internal algorithms.

The WIM model components are classes, objects, physical objects, operations and views, as shown in Figure 1. *Super Node* is the superclass and *Node*, *Relation* and *Log* are subclasses of *Super Node*. The WIM model is based on the object oriented paradigm. *Objects* are sets representing instances of some class and *Physical Pages*, *Physical Links* and *Physical Logs* are physical objects of the classes *Node*, *Relation* and *Log*, respectively. The objects are modified by operations performed on the objects. *Views* are queries performed by operations on sequences of objects and *Dynamic Operations* are views with generic input objects and parameters.

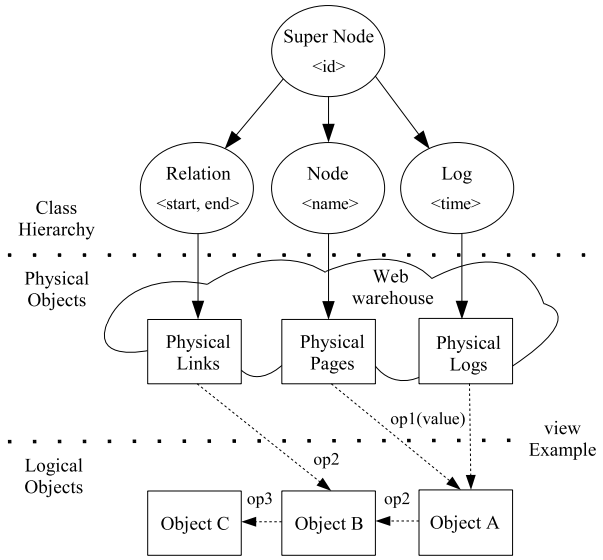

Figure 1. Main components of the model.

Every *object* has the attribute id as an internal reference to its elements. The attribute $name$, defined in the class *Node* is an external reference to the node. The class *Relation* allows relationships among object nodes of other classes. Objects of the class Relation, with attributes $start$ and end, store start and end nodes of each relation element. The WIM model supports hypergraphs [12]. In this case, other attributes apart from $start$ and end are required to store the other nodes involved in each hyperedge. In this paper, we often refer to elements of the classes Node and Relation by *nodes* and *relations*, respectively. The subclass *Log* associates information of Web logs to object elements. Objects of this class have the attribute $time$, referred to the time stamp of every event in the log.

An *object* is always a set of elements. As instances, the objects *physical pages* and *physical links* represent, respectively, all the Web pages and links among pages in a given Web warehouse. 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 in the page, the page text, title, among other things. Objects have *Dynamic Attributes* [4], allowing the addition or deletion of attributes when an operator is applied.

The *operations* are functions that modify the object characteristics, generating a new object or returning a value. 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 [14], depending on time and space costs and on the application. Materialized objects can be used in any other view as input objects. In Figure 1, the view $Example$ would be:

$$Object_A = op_1(Physical\ Pages, Physical\ Logs, value);$$
$$Object_B = op_2(Physical\ Links, Object\ A);$$
$$Object_B = op_3(Object\ B);$$

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 $op1$. By generalizing the input objects and parameters of the view, the following dynamic object is able to be used in any other view:

$$Result = dynamicOperator(O_{ClassNode},\ O_{ClassLog},\ O_{ClassRelation},\ value);$$

The set of operations that we describe next were designed after studying several normal Web mining applications and extracting the basic needs of them.

3 Primitive Operations

We call *primitive operations* the operations that are important to the WIM model and that cannot be a view, that is, they cannot be implemented by using a sequence of other operators. The WIM algebra is composed of eleven primitive operators, divided into three categories: the extended relational algebra operators (containing five operations), the

graph operators (containing four operations) and other operators (the content similarity operator and the calculus operator). The three categories of operators are presented in the following three sections.

3.1 Extended Relational Algebra Operators

The extended relational algebra operators are used to manipulate object attributes, with extensions of the traditional relational algebra operators including set operators. They are five operators: Select, Project, Merge, Join and Order.

Select: this operator selects tuples of the object. Two options are possible: $value$ and $top\text{-}k$. The option $value$ allows the selection of tuples that satisfy a condition for a given attribute. It is also possible to combine various conditions by means of logical operators (AND, OR, NOT). With the option $value$ the operator works as the traditional relational algebra operator [14]. $Top\text{-}k$ returns only a given number of tuples in the top ranking (according to an attribute or similarity measure) of an ordered list.

Project: this operator returns an object with a given set of attributes. Two options are possible: $normal$ and $relation$. With the option $normal$, the operator works as the traditional relational algebra operator with the same name. $Relation$ is used to project all the relations of an hypergraph. The Project operator can return "bags" [14], with multiple occurrences of a tuple.

Merge: with this operator three options are possible: $union$, $intersection$ and $difference$. The option $union$ returns all the tuples that occur in any object involved. $Intersection$ returns only the tuples that occur in the two objects involved. $Difference$ returns an object with the tuples that occur in an object A, excepting those ones that also occur in an object B. Merge can be applied to bags, where the amount of duplicated tuples into the object returned vary according to the merging option.

Join: with this operator four distinct options are possible: $objects$, $bags$, $non\text{-}directed$ and $disjoin$. Join $objects$ works as the Join operator of the relational algebra, returning an object with the attributes of two input objects, considering the tuples with the same value for a given attribute (normally the identifier). Join $bags$ converts a bag to a set, that is, without duplications of tuples. A new attribute $quantity$ is inserted, to count the amount of previous identical tuples. *Non-directed* is used in relation objects to eliminate the direction of the relations, what is done by just deleting the relations with values $start$ and end inverted. The $disjoin$ option is used to separate items that are arrays, where every element of the array generates a new tuple into the object. An implicit array separator may be indicated. For example, applying Join($disjoin$) to the tuple $\{1;\ (a,\ b,\ c)\}$ results in the tuples $\{1;\ a\}$, $\{1;\ b\}$ and $\{1;\ c\}$.

Order: this operator sorts the tuples in $increasing$ or $decreasing$ order by one or more attributes.

3.2 Graph Operators

The graph operators are used in applications where the information of both nodes and edges of the graph are important, such as singular value decomposition (as *Pagerank* [22]). There are four operators: Add relation, Link distance, Connected Nodes and Singular Value Decomposition.

Add Relation: in this operator, if the input object belongs to the class relation, it adds new relations and the graph returned is a hypergraph. If the input object belongs to the class node, the input object is converted to the class relation. The following options are available, for both relation and node input objects: $fixed$, $cross\ product$ and $same\ attribute$. $Fixed$ means that two given attributes are the start and end attributes of the new object. In this case, the operator may be unary or binary. If unary, one of the start or end attributes may be created for a constant value. $Cross\ product$ converts all distinct node pairs of the object in relations, similarly with a Cartesian product of the object with itself, excluding self loops. $Same\ attribute$ means that a relation is inserted among every pair of nodes with the same attribute value, for a given attribute. For example, if we have the tuples $\{1;\ a\}$, $\{1;\ b\}$, $\{2;\ c\}$ and $\{2;\ d\}$, the application of the current operator with the option $cross\ product$ results in the non-directed relations $a-b$, $a-c$, $a-d$, $b-c$, $b-d$ and $c-d$. With the option $same\ attribute$ the resulting relations are only $a-b$ and $c-d$, since there is no cross product among elements with different values for the first attribute (values 1 and 2). The direction can also be considered. For example, if the operator Add Relation is used with $same\ attribute$ and $directed$, for the instance example above, the following directed relations would be returned: $a \rightarrow b$, $b \rightarrow a$, $c \rightarrow d$ and $d \rightarrow c$. The resulting object discards the other attributes related to the nodes, since the values are related to only one object, not to every object of the relation.

Link Distance: this operator is also a way for adding new relations, which can be done according to methods like $co\text{-}citation$ [29], $bibliographic\ coupling$ [19] or $distance\text{-}k\ transitivity$ [15]. A new attribute $distance$ is added to store the previous distance among nodes of the new relation object. The operator is recursive for co-citation and bibliographic coupling, but it is also possible to perform partial link distance. In this case, relations are inserted among nodes that are distant from a given maximum number of relations. The operator can be applied to every relation of an object or to only a subset of elements. When applied to a subset of elements, a list of nodes must be informed and the resulting object would contain only relations involving the nodes of this list. Distance-k transitivity can be forward,

backward or both. The option "both" works as an union of the options forward and backward. It is also possible to perform transitive closure.

Connected Nodes: this operator identifies the $connected$ components, the $strongly$ connected components or the $minimum\ spanning\ tree$ of a graph [12], always adding a new attribute $subgraph$ to indicate the subgraph number for which the relation belongs to. The operator works by deleting edges of the graph, for any of the three options.

Singular Value Decomposition: when this operator is applied to an object of the class relation, it returns an object of the class node with one more attribute according to the $Pagerank$ [22], $Authority - Hub$ [20], or any other measure defined by matrix algebra.

3.3 Other Operators

In this third category we have two isolated operators: Content Similarity and Calculus. The Content Similarity operator identifies the content similarity among connected nodes [24]. The Calculus operator is directly applied to given attributes of objects, performing mathematical and statistical calculus.

Content Similarity: if this operator is applied to a relation object, compare a given textual attribute of the start and end nodes of every relation. As the information of the nodes are carried by objects of the class node, objects of this class must be informed. The direction of the relation can be used for containment comparison. 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. The comparison processes may be independent for distinct applications. For Web applications some comparison processes could be:

- *tf-idf*: term frequency and inverse document frequency [28];
- *Shingles*: it uses similar passages of text [10];
- Terms comparison: it uses attributes such as title, URL text, similar items of the list, anchor text, or mail-to address.
- User-defined function: this function may use both relation and node objects (for example, a given ranking algorithm for searching).

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, for the nodes existing in the two objects. If unary, four options are possible: $count$, $constant$, $normalize$ and $function$. $Count$ returns the quantity of tuples in the object. With the option $constant$, the sum, difference, multiplication or division is used with a constant value in one of the two operands. Sum and difference operations may be applied to text attributes, respectively, for appending or for deleting some text of a given attribute. The option $function$ allows the application of specific functions for a given attribute and returns a value. The function can be user defined or pre-defined, as the $statistical$ functions. The statistical measures are $mean\ average$, $geometric\ average$, $standard\ deviation$, $mode$ and $median$. The last two measures can be applied to non numerical attributes.

4 Examples of Applications

In this section we present two applications that show the expressive power of the WIM model. Before entering in the applications we show how useful operations can be defined. 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.

In figures that represent views and dynamic operators we use boxes to represent objects and arrows to represent an operation. The object linked from an arrow is an input for the current operation. The object which an arrow links is the output of that operation. The internal object labels represent the history of operators applied to objects in the view. For example, an object has a label $O_{RN.CN}$ if the operator "Connected Nodes" (CN) was applied to the input object of the view, and later the operator "Relation to Node" (RN) was applied. The label of an operation represents the operator used and its parameters. An arrow with a dashed line do not represent an operation. It means that the input object stores a constant value that is used as a parameter in an operator (as in in Figure 4). If an object is represented with a double line in its base (such as the two end objects of the view Format List in Figure 5), the object is materialized.

Figure 2 shows the dynamic operator ***Relation to Node***, 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 apart from $start$ and end in the projection step. The remaining parameters used are fixed, which means that the user of the dynamic operator cannot modify them. This occurs with the two first parameters of the operator Project. Note that this dynamic operator cannot be applied to a relation object that represents a hypergraph.

In the following sections we present views for clustering nodes and for identifying related item sets.

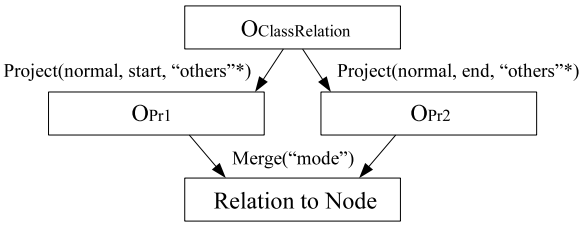

Figure 2. The dynamic operator Relation to Node.

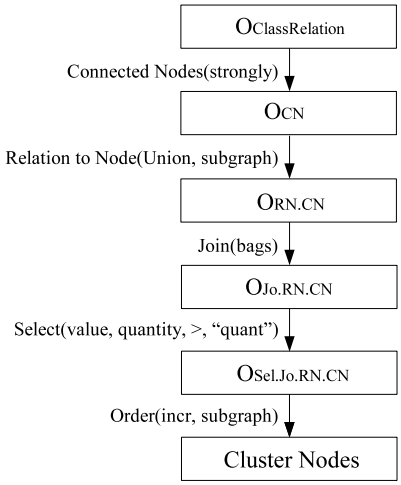

Figure 3. The dynamic operator Cluster Nodes.

4.1 Clustering Nodes

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* [18]. Related to graph applications, the clustering activity consists of partitioning the graph. In this section we present two views for graph partitioning.

The first view, *Cluster Nodes*, groups nodes according to the presence of a relation – a very simple but useful heuristic for Web graph applications (non-labeled graph). 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 that each node belongs to.

Figure 3 shows the view Cluster Nodes as a dynamic operator, built for future use in other views. It means that the input of the view is generic. Thus, it is allowed that other objects belonging to the *class relation* is the input object. Some internal attributes may be variables, like the attribute *quantity* of the operator Select.

The dynamic operator works as follows. Initially the operator Connected Nodes is applied to an object of the class relation. The strongly connected components are identified by inserting a new attribute $subgraph$ to store the subgraph number which the relation belongs to. Next, a dynamic operator Relation to Node is used to convert the object from the class relation to the class node. The conversion consists of considering every node in the $start$ and end nodes of the input object as a node in the output object. The parameter $Union$ means that the object $O_{RN.CN}$ contains all the nodes existing in the attributes $start$ and end from the input object O_{CN}. The parameter $subgraph$ indicates that the attribute $subgraph$ is kept in the resulting operator $O_{RN.CN}$.

Next, Join is used with the option $bags$. Thus, object $O_{Jo.RN.CN}$ contains the same elements than object $O_{RN.CN}$, but without replication of tuples and with a new attribute $quantity$, to store the number of identical tuples (for every distinct tuple) in the previous object $O_{RN.CN}$. Select is used to consider only the clusters with a minimal number of nodes (the constant user-defined $quant$), what might be important in Web warehouse. At the end the nodes are ordered by the attribute $subgraph$, just to keep the elements of each cluster together. 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.

It is possible to use the dynamic operator just presented to cluster nodes in labeled graphs. The simplest way might be to perform the operator Select before the operator Cluster Nodes, eliminating edges that do not reach a given threshold. The second view is a more sophisticated solution for labeled graph partitioning. Some algorithms for clustering on labeled graphs use the minimum spanning tree and thresholds to eliminate edges [13]. The mean average of all the edges weights, or functions involving this statistical measure are often used as threshold.

Figure 4 shows the view *Improved Cluster*. Initially the mean average for the attribute $weight$ is calculated and stored as a constant value in the object $O_{meanAverage}$. By the operator Connected Nodes, only the edges belonging to the minimum spanning trees are maintained. The object O_{CN} has a new attribute $subgraph$, that identifies the subgraph number which each edge belongs to, since various minimum spanning trees can be generated. In the next step, Select is used to exclude edges that have the weight less than the mean average. Now the minimal spanning trees are partitioned. The operator Connected Nodes is applied again to rearrange the new subgraphs, associating a new subgraph

number to every subgraph. Relation to Node returns the list of nodes and its subgraph number.

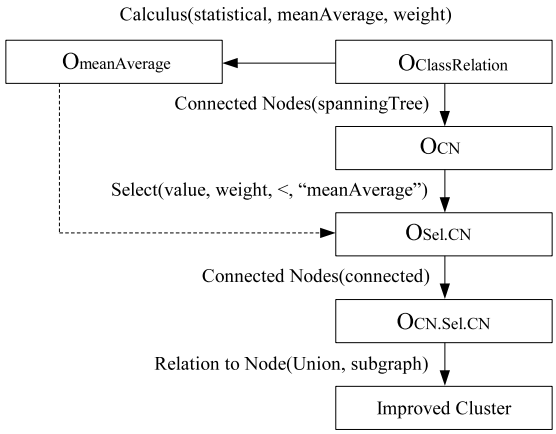

Figure 4. The view Improved Cluster.

4.2 Related Item Sets

In this section we present views to find related item sets in HTML lists, using 'LI' tags. We allow a related item set to be composed by items that not necessarily have occurred *allways* together in some lists. They may occur frequently *in pairs*. Note that we use the expression *item set* instead of *itemset*, since the second one is defined in data mining as being the items that occur together. Views for mining frequent *itemsets* are presented in [23].

Preliminarily we present in Figure 5 the dynamic operator *Format List*, required for the views involving sets in this section, with the function of preparing the items of lists to be used as nodes. The operator Format List works as follows. Initially given attributes are projected. We are using only HTML lists for simplification, but it would be interesting to use other evidences as the items of menus. The Join operator is used to separate the elements of the list, where each item generates a tuple. The attributes $itemId$ and $item$, added by the operator Join, are projected and represent one of the two operator outputs. This object is required to retrieve the item name, since the other object returned do not possess this information. The operator Node to Relation is applied with the parameter $sameAttr$, where every pair of nodes with the same id becomes a relation. The object returned contains, for every list considered in the input node object (object of the class node), relations among every pair of items of the list.

Figure 6 presents the view Sets, constructed to identify the related item sets. Applied to a physical pages object containing an attribute with the items list, the operator Sets returns a node object where each node represents an element of the sets identified. The heuristic used in the Sets view is

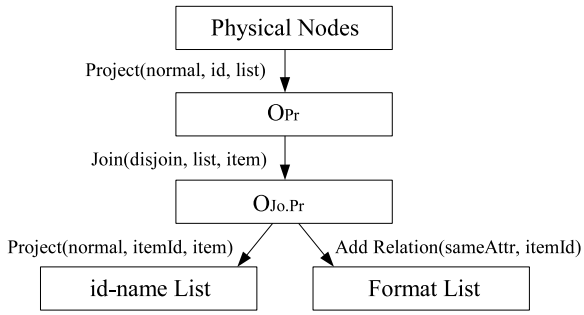

Figure 5. The dynamic operator Format List.

the following: a related item set contains item pairs that occurred together frequently in lists of the Web warehouse, privileging also individually the most frequent item pairs.

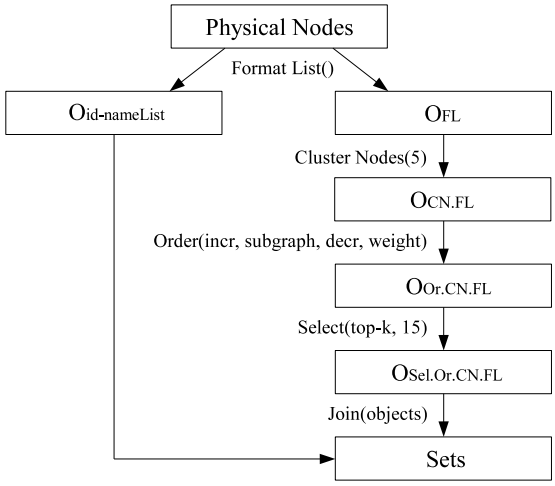

Figure 6. The view Sets.

The view Sets shown in Figure 6 works as follows. Initially the dynamic operator Format List is used to prepare the data in the physical object. The operator Cluster Nodes groups the nodes, considering the pairs that occur together at least five times, number that can be adjusted. The nodes are ordered initially by the subgraph number and later by the weight inserted by the previous operator Node to Relation. The weight represents the amount of relations preliminarily existent among the nodes. The top 15 distinct elements are selected. The object returned is finally joined with one object that contains the item names. The joining step is required because in the operator Node to Relation above only the reference of the item ($itemId$) was considered. If desired, the selection and joining steps would be performed in a loop, in order to return several most frequent related lists.

Now suppose we want to make suggestions of similar items, based on one or more input items. A similar application is Google Sets [16]. The view *Search Set* is shown in

Figure 7. The heuristic used in the view is similar to the one used in the previous view Sets, however the sets considered are limited by the query terms and other terms that have a "link distance" relation to the query terms.

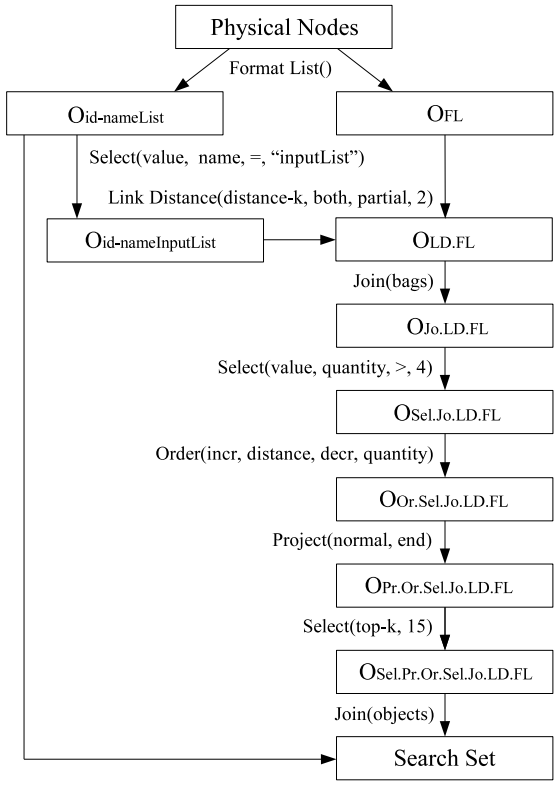

Figure 7. The view Search Set.

Figure 7 presents the view Search Set. From the object $O_{id-nameList}$, Select is used to obtain the references and names of the input item list $inputList$, informed by the user. The operator Link Distance uses the nodes in $O_{id-nameInputList}$ to perform distance-k transitivity on nodes. The object $O_{LD.FL}$ is a bag [14] and contains only relations involving the items of the query (present in the object $O_{id-nameInputList}$), including the relations just inserted by the operator Link Distance. The operator Join is used and a new attribute $quantity$ is added, with the amount of similar elements in the bag, for each different element. For pruning, only elements that appeared at least five times are selected. The elements are ordered initially by the distance among the nodes and later by the amount of similar elements. In object $O_{Or.Sel.Jo.LD.FL}$ the $start$ attribute contains the nodes of the query, so only the end attribute is projected. The top 15 distinct nodes are selected and next the names of the resulted nodes are obtained from the object $O_{id-nameList}$, by the operator Join.

5 Related Work

The Web is viewed as a directed graph in [21], 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 [21], [8] introduces Whoweda (Warehouse of Web Data), 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. In [7] new operators, as pack (for group objects) and sort are introduced for Whoweda, using interesting examples. The operator join for Whoweda is introduced in a more recent work [9]. WIM differs of Whoweda in many aspects. WIM is based on the object oriented paradigm. WIM operators work in a higher level than Whoweda operators. Whoweda has operators for visualization, what is not defined in the WIM.

The main and basic concepts in Web mining are well explained in [30]. Most of the efforts in Web mining for Web warehouse are focused on the XML technology, as the Xylene project [1], a dynamic warehouse for XML data. A recent work, [6], uses frequent itemsets to treat the template detection problem. Specially in Web usage mining [11], algorithms based on usage, content and structure information to automatically identify interesting knowledge in usage have been developed. As an example, they performed the algorithms to discover interesting frequent itemsets.

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 [17] 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. The graph algebra introduced in [26] promises to reduce the effort in view specifications by lowering the number of queries needed to define a view. As an example, a view to solve the minimal covering graph problem is constructed.

There is a number of papers related to the implementation and complexity analysis of algorithms related to graph algebra, data and web warehouse, and web mining. We highlight [2], that discuss the pros and cons of materialize the views for future use. The operator Project often results in bags, specially when applied to relation objects. The use and implementation of bags in Web warehouses is discussed in [27].

Raghavan and Garcia-Molina [25] 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. Thus, the Web warehouse is modeled as a collection of pages and links, with associated page and link attributes. The model incorporates the ranking of pages and links, allowing queries using different similarity measures, as TF-IDF or *Pagerank*. A snapshot of the Web warehouse is called a Web relation. By applying specific operators on a Web relation, a new Web relation is generated. This work is closer to WIM than other works introduced in this section, with some important distinctions. In WIM we extend the relational algebra and the algebra used in [25], allowing operations for relation objects. We also extend the concept of node, which is not limited to web pages, and we allow hypergraphs.

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 Web as the main application. However, the model can be applied to other graph structures such as bibliographic databases where links are citations, or telephone transactions where pages are client information and links are phone calls.

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 of the key characteristics of the WIM model: *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; *broadness*, since many new views were created without changing the set of operators; and the *applicability* of the model for graph problems and problems that can be converted in graph problems. We stand out the importance of the graph operators, mainly that ones to convert objects between distinct classes. The conversion operators give flexibility to the model, allowing the implementation of heuristics as the one used in the views for finding related item sets.

As extensions of the examples shown in this paper, we have developed views for applications related to data mining and information retrieval [23]. Related to data mining, we have developed a view for finding frequent itemsets for association rule mining. Related to information retrieval, we have developed views for performing a simple query, for clustering the query results and for improving the quality of the ranking.

Considering all the views developed, all the eleven primitive operators have been used. Most of them have been used in at least three different views. We also highlighted the importance of dynamic operators. Some of them are frequently used in the views, as the Relation to Node and Cluster Nodes.

Acknowledgements

This work was supported by GERINDO Project–grant MCT/CNPq/CT-INFO 552.087/02-5 (A. Pereira Jr and N. Ziviani), CYTED VII.19 RIBIDI Project, Grant P04-067-F of the Millennium Scientific Initiative, Mideplan, Chile (R. Baeza-Yates), and CNPq Grants 30.5237/02-0 (N. Ziviani) and 14.1636/2004-1 (A. Pereira Jr).

References

[1] S. Abiteboul, S. Cluet, G. Ferran, and M. C. Rousset. The xyleme project. *Computer Networks*, 39(3):225–238, 2002.

[2] S. Abiteboul, J. McHugh, M. Rys, V. Vassalos, and J. L. Wiener. Incremental maintenance for materialized views over semistructured data. In *24rd International Conference on Very Large Data Bases (VLDB'98)*, pages 38–49. Morgan Kaufmann Publishers Inc., 1998.

[3] R. Baeza-Yates. Information retrieval in the web: beyond current search engines. *International Journal on Approximated Reasoning*, 34(2-3):97–104, 2003.

[4] R. Baeza-Yates, T. Jones, and G. Rawlins. New approaches to information management: Attribute-centric data systems. In *7th International Symposium on String Processing and Information Retrieval (SPIRE'00)*, pages 18–27, A Coruña, Spain, September 2000. IEEE Computer Science Press. (Invited paper).

[5] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press/Addison-Wesley, England, 1999.

[6] Z. Bar-Yossef and S. Rajagopalan. Template detection via data mining and its applications. In *Eleventh International Conference on World Wide Web (WWW'02)*, pages 580–591. ACM Press, 2002.

[7] S. S. Bhowmick, S. K. Madria, W.-K. Ng, and E.-P. Lim. Data visualization in a web warehouse. In *ER Workshops*, pages 68–80, 1998.

[8] 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.

[9] 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.

[10] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic clustering of the web. In *Sixth International World Wide Web Conference (WWW'97)*, pages 391–404, 1997.

[11] R. Cooley, P.-N. Tan, and J. Srivastava. Discovery of interesting usage patterns from web data. In *Revised Papers from the International Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*, pages 163–182, 2000.

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

[13] M. Gaertler. Clustering with spectral methods. Master's thesis, Universitt Konstanz, 2002.

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

[15] G. H. Gonnet and R. Baeza-Yates. *Handbook of algorithms and data structures: in Pascal and C*. Addison-Wesley Longman Publishing Co., Boston, MA, second edition, 1991.

[16] Google. http://labs.google.com/sets, May 2005.

[17] 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.

[18] J. Han and M. Kamber. *Data mining: comcepts and techniques*. Morgan Kaufmann, United States of America, 2001.

[19] M. M. Kessler. Bibliographic coupling between scientific papers. American Documentation, 1963.

[20] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[21] 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.

[22] 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.

[23] 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.

[24] A. R. Pereira Jr and N. Ziviani. Retrieving similar documents from the web. *Journal of Web Engineering (JWE)*, 2(4):247–261, november 2004.

[25] 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.

[26] E. A. Rundensteiner. Tools for view generation in object-oriented databases. In *Second International Conference on Information and Knowledge Management (CIKM'93)*, pages 635–644. ACM Press, 1993.

[27] W.-K. N. E.-P. L. S. Bhowmick, S.K. Madria. Web bags: are they useful in a web warehouse? In *Fifth International Conference of Foundation of Data Organization (FODO'98)*, pages 210–220, Kobe, Japan, 1998.

[28] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[29] H. G. Small. Co-citation in the scientific literature: A new measure of relationship between two documents. *Journal of the American Society for Information Science*, 24:265–269, 1973.

[30] 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.