Structure-based Mining of Hierarchical Media Data, Meta-Data, and Ontologies

K. Selçuk Candan   Jong Wook Kim   Huan Liu   Reshma Suvarna

Computer Science and Eng. Dept,

Arizona State University, Tempe, AZ 85287

{candan, jong.wook.kim huan.liu, reshma.suvarna} @asu.edu

ABSTRACT

Users now have access to unprecedented amounts of media data, but it is increasingly difficult to integrate relevant media from multiple and diverse sources. The functioning of a multimodal integration system requires metadata, such as ontologies, that describe media resources and media components. Such metadata are generally application dependent and this can cause difficulties when media needs to be shared across application domains. There is a need for a mechanism that can relate the common and uncommon terms and media components. In this paper, we develop an algorithm to mine and automatically discover mappings in hierarchical media data, meta-data, and ontologies, using the structural information inherent in these types of data. We evaluate the performance of this algorithm for various parameters using both synthetic and real data collections and show that the structure based mining of relationships provides high degrees of precision.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - Data Mining; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms and Topics

Algorithms; experimentation

Keywords

Retrieval and mining of semantics; extracting and mining semantics from multimedia databases; knowledge discovery in XML, hierarchical multimedia, and ontologies; multi-modal integration; structure-based mining

1. INTRODUCTION

Universality, i.e., the need for accessing media, collected and indexed independently by various applications and organizations, necessitates uniform organization schemes that would allow easy access and integration of media. Instead, mostly media is available to users and applications in diverse structures and formats. Furthermore, considering the multitude and diversity of these applications, it is not viable to expect a global unifying scheme.

A semantic network of media, wherein different applications can exchange information and integrate multimodal data requires the information about each media to be represented in a detailed and structured manner. To enable such information exchanges various hierarchical metadata frameworks have been proposed. For instance, Resource Description Framework (RDF) [1] is such an effort supported by the World Wide Web Consortium. RDF aims at providing a means for the description of metadata, in an organized, informative, searchable, and accessible manner. RDF schemas are implemented for representing multimedia data like image, audio, and video files. For instance, an RDF-based system is presented in [2], where RDF schemas provide a non-visual description of photos. This description is split into three different parts:

- Dublin Core [3] schema is used for identifying the photograph and describing properties like creator, editor and title.

- Technical schema is used for capturing technical data about the photo and the camera such as the type of camera, type of film, scanner and software used for digitization.

- Content schema is used for categorizing the subject of the photo by means of a controlled vocabulary. This schema allows photos to be retrieved based on such characteristics as portrait, group portrait, landscape, architecture, sport, etc.

RDF and similar metadata description frameworks provide a common language through which meta-data, such as media and application ontologies, are exchanged. This enables software systems to create a uniform structure to represent and organize data, which renders the integrated data manageable and retrievable. Consequently, many multimedia standards define objects as a structured collection of media objects.

The metadata (such as content descriptors and feature names), used to describe resources in RDF and other metadata description languages, are defined by various communities. The metadata creators, depending on the specific application, culture, use different terms for similar concepts. For the functioning of an

automated multimodal media integration system, the semantics behind the metadata terms used by various authors and communities should be mined and automatically related. Hence, a mechanism to mine and relate semantically similar but syntactically different metadata is required

In this paper, we develop algorithms to automatically mine concept mappings in hierarchical media data, metadata, and ontologies. We propose a solution which mines such relationships using the inherently available structural information. We use Multidimensional Scaling [3,12,13,14] to map the nodes between the two different but similar structures (multimedia hierarchies, ontologies, or namespaces) such that the syntactically different but semantically similar components map to each other.

In Section 2, we present our motivation. In Section 3, we define the problem formally and in Section 4, we propose a solution and develop an efficient algorithm. Then, in Section 5, we evaluate the performance of the algorithm for various parameters and show that it is very effective in addressing this challenge.

2. MOTIVATIONS
In this section, we present two related motivations for this work. This first one, integration of resource description framework (RDF) specifiec multimedia resources, requires mining of mapping of hierarchical namespaces (or ontologies). The second application, XML multimedia document matching and integration, requires mining and mapping of components (elements or attributes) in various multimedia objects.

2.1 Integration of Resource Description Framework (RDF) described Media Resources
If application and media content experts could easily associate metadata with each resource they create, then this metadata could be used by access and integration engines to increase their efficiency and precision. In order to enable this, the metadata format used by different applications must be compatible.

Within the context of web information integration, many proposals were made to the World Wide Web Consortium (W3C) for representation of Web-related metadata. Initial solutions were based on the <META> tag of the HTML. Currently, many companies, such as Microsoft, IBM, Motorola, Netscape, Nokia, OCLC, are actively participating in the field of metadata framework developments. In 1997, Netscape submitted a proposal, titled ``Meta Content Framework'', to W3C [5]. The two principles on which the meta content framework (MCF) is based are (1) there is no distinction between the representation needs of data and metadata, and (2) for interoperability and efficiency, schemas for different applications should share as much as possible in the form of data structure, syntax, and vocabulary. The culmination of various frameworks was the Resource Description Framework or RDF [6]. RDF provides application developers with a foundation for the description of metadata for the next generation of interoperable applications[1].
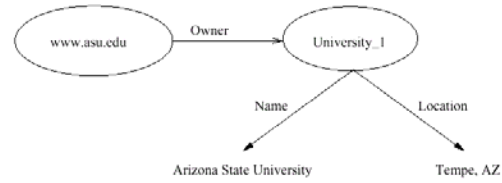

Figure 1. Example RDF statement

Ontologies, formalisms that define the relationships among terms in a given application, describe the context in which metadata is applied. They are used to link, compare, and differentiate information provided by various application resources. RDF provides a rich data model where entities and relationships can be described. The relationships in RDF framework are first class objects, which means that relationships between objects may be arbitrarily created and be stored separately from the two objects. This nature of RDF is very suitable for dynamically changing, distributed, shared nature of the Web. It is designed to provide a framework that ensures interoperability between metadata frameworks.

The metadata (property names) used to describe resources are generally application dependant and must be associated with RDF schema. This, however, can cause difficulties when RDF descriptions need to be shared across application domains.

For example, Figure 1 represents an RDF statement for resource **www.asu.edu**. The property (metadata) used to define the resource University_1 are, *Name* and *Location*. However, the property *Location* can be defined in some other application domain as *Address*. Although, the semantics of both property names are the same, syntactically they are different. In general a property name may have a broader or narrower meaning depending upon the needs of particular application domains. In order to prevent such conflicts and ambiguities, the terminology used by each application domain must be clearly identified.

RDF uniquely identifies property names by using the Namespace mechanism [7]. A namespace can be thought of as a context or an ontology that gives a specific meaning to what might otherwise be a general term. It provides a method for unambiguously identifying the semantics and conventions governing the particular use of property names by uniquely identifying the governing authority of the vocabulary. Thus using namespaces RDF provides ability to define and differentiate semantics among communities.

Although with the help of namespaces, we can uniquely identify and relate the metadata to a particular governing authority or a community, there is no straightforward way to map and relate terms or properties among different communities.
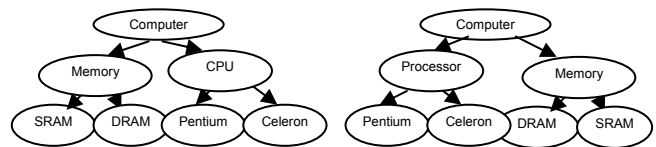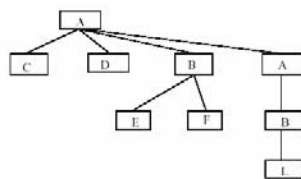

Figure 2. Similar hierarchical namespaces

Consider the two hierarchical namespaces provided in Figure 2 (the hierarchy usually corresponds to the concept/class hierarchy) of the underlying domains. As it is implied by the similar structures of these namespace hierarchies, the terms *Processor* and *CPU* are semantically related. Therefore, if the user integrates two data domains each using one of these two namespaces, whenever a query is issued using the property name *CPU*, the content having the property name *Processor* should also be retrieved.

Automatic mapping of the semantically similar but syntactically different terms from the different namespaces is one of the necessities for integration of content from independently created data sources. An automated mechanism needs to be devised that relate the common and uncommon terms of various metadata communities.

2.2 Matching of Extensible Markup Language (XML) specified Media Objects

Many multimedia standards define objects as structured collections of media data. XML description of such multimedia objects and structures is very common. Examples include virtual reality modeling languages (X3D), media content description frameworks (MPEG7), e-commerce web documents, and geographic information systems. Extensible Markup Language (XML) [8] defines a generic syntax used to mark up data with simple, human readable tags. It provides a standard format for computer documents. As shown in Figure 3, an XML document is a tree-like structure, whose structure may be defined through a Document Type Definition (DTD) or through an XML-Schema. XML is very flexible; its attributes and sub-elements can be either missing or repeated. DOM [9] and LORE [10] are two well-known tree-based data models for XML documents. To store a XML document in its tree form, each node of the tree corresponds to an element or an attribute of an element in the XML document. The root node contains the document's root. A child node corresponds to a subelement or an attribute of the parent node. For each child of a node, besides the pointer to the child, there is a tag in the node that indicates the name of the child node. If the child is a subelement, the name is its element tag. If the child is an attribute, the name is the attribute name.


```
<!ELEMENT A(C*,D?,B*,A*)> <!ELEMENT B(E?,F?,L?)>
<!ELEMENT C(#PCDATA)> <!ELEMENT D(#ANY)>
<!ELEMENT E(#CDATA)> <!ELEMENT F(#ANY)>
<!ELEMENT L(#CDATA)>
```
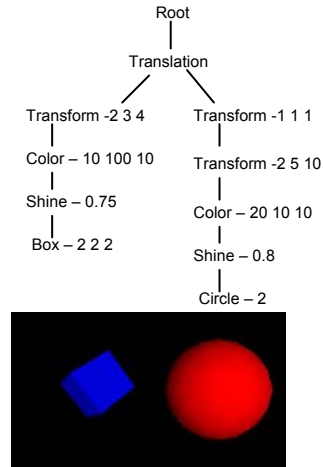
Figure 3. An example XML document and the corresponding DTD

Figure 4. An example X3D document and the corresponding 3D world

XML became the *de facto* standard for multimedia data representation. For example, X3D [11], a file format and related access services for describing interactive 3D objects and worlds, is based on XML. X3D nodes are expressed as XML elements, i.e., tagged names (see Figure 4 for an example X3D document).

XML tags can be used for describing the data in the form of a hierarchical structure. This provides flexibility and expressive power to the data description framework, but it also complicates the integration task as different domains could have different sets of rules, tags, and properties to represent the same data. Although, whenever they are available, DTDs and namespaces provide information about the structure of the XML files, not all XML documents have associated DTDs. In fact, one of the main reasons why XML is becoming a de facto information exchange standard is that each XML document (even when it does not have an associated DTD) is self-describing: the hierarchical structure and the tags in this hierarchy gives information about the relationships between the tags; i.e., the data elements and their attributes. Consequently, given two similar multimedia objects, where different tag names are used to denote similar concepts, it should be possible to make the association between these tags using an analysis of the hierarchical XML document structures. structures. For example, in Figure 5, nodes labeled *F* and *X* are likely to correspond to each other.
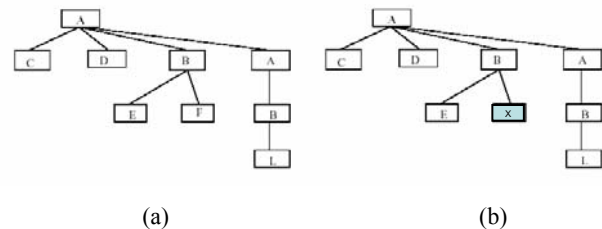

(a)   (b)

Figure 5. Two similar XML documents; the node labels *F* and *X* are likely to denote semantically similar elements

3. PROBLEM STATEMENT

The problem we address in this paper is to mine mappings between the nodes of hierarchical media data, metadata, and ontologies. The main observation is that the structural similarity between two given trees (such hierarchical media objects, XML documents, or name spaces) can provide clues about the semantic relationships between their nodes.

In general, the nodes in the two trees can be divided into common and uncommon nodes. The common nodes are those shared by the two trees and can either have the same labels or (in the case of multimedia data) they may have application dependent features that provide high degrees of similarity [25]. In this paper, we do not focus on how common nodes are discovered. Our aim is to relate the uncommon nodes of a two given hierarchical structures. Therefore, formally, we can state the problem as follows:

 Given

- two trees, *T1(V1,E1)* and *T2(V2,E2)*,

where *V* denotes the nodes in the tree and *E* denotes the edges between the nodes,

- a partial mapping, *M,* between the nodes in *V1* and *V2*, (we call those nodes in *V1* and *V2* that have a mapping, *the common nodes*) and

- two unmapped nodes v_i in *V1* and v_j in *V2,*

compute the similarity between v_i and v_j.

For example, given the two trees in Figure 5, the user or the content-integrator might want to find which node in the first tree corresponds to the node labeled X in the second tree (In this example. purely based on the structures of the two trees, we can conclude that X corresponds to F in the first tree).

The use of structural information for mining of semantic relationships is not new. We used structural information available on the web for mining web document associations, summarizing web sites, and answering web queries [22,23,24]. [17,18,19,20] have used the language taxonomies and IS-A hierarchies to define the similarity and distance between terms in natural language. These mainly rely on the observation that given a tree, *T(V,E)* and two nodes, *a* and *b*, on the tree, we can compute a distance, *d(a,b),* between the nodes by considering the structure of the tree, for instance by counting the number of edges between them. *The main challenge we face in this paper for finding the similarity between two nodes in two **different** trees, on the other hand, is that there is no common structure to compare these two nodes*; since the two trees may have arbitrarily different structures, finding a mapping between the nodes is not trivial.

4. PROPOSED APPROACH

In order to match two nodes in two different trees, we need to find a mapping such that the distance values in two trees between the common nodes are preserved as much as possible. In this paper, we address this challenge by mapping the two trees into a common space using the matching nodes and comparing the unmapped nodes in this common space. The proposed solution can be broken down into four steps:
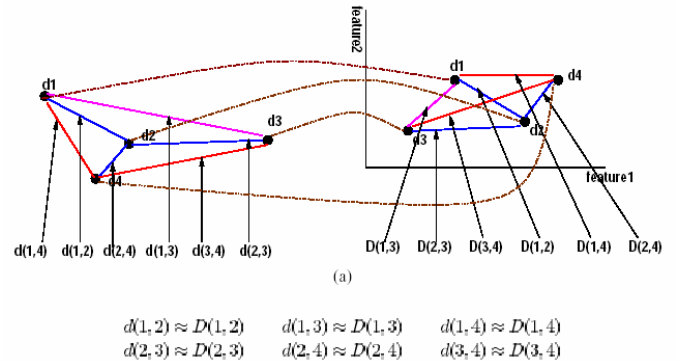
1. Map the nodes of *T1* and *T2* into two multi dimensional spaces, *S1 and S2,* both with the same number of dimensions.

2. Identify transformations required to align the space *S1* with the space *S2* such that the *common nodes* of the two trees are as close to each other as possible in the resulting aligned space.

3. Use the same transformations to map the *uncommon nodes* in *S1* onto *S2*.

4. Now that the nodes of the two trees are mapped into the same space, use clustering and nearest-neighbor algorithms to find the related *uncommon nodes* in the two trees.

4.1 Step I: Map both Trees into Multi-Dimensional Spaces

We map the trees based on the common nodes. For example, the common nodes of two given namespaces might include the shared terms like *University* and *College* that are *known* to denote similar concepts.

Multi-Dimensional scaling (MDS) is a family of data analysis methods, all of which portray the structure of the data in a spatial fashion [3,12,13,14]. MDS is used to discover the underlying spatial structure of a set of data items from the distance information among them.

MDS works as follows, it takes as inputs (1) a set of N objects, (2) a matrix of N x N, containing pair wise distance values, and (3) the desired dimensionality k. Given these inputs, MDS tries to map each object into a point in the k-dimensional space. The mapping process of documents, given their distances with respect to one another, is shown in Figure 6.

$$d(1,2) \approx D(1,2) \qquad d(1,3) \approx D(1,3) \qquad d(1,4) \approx D(1,4)$$
$$d(2,3) \approx D(2,3) \qquad d(2,4) \approx D(2,4) \qquad d(3,4) \approx D(3,4)$$

Figure 6. MDS mapping of four data points onto a two dimensional space

The criterion for the mapping is to minimize a *stress* value which is calculated as

$$stress = \sqrt{\frac{\sum_{i,j}(d'_{ij} - d_{ij})^2}{\sum_{i,j} d^2_{ij}}}$$

where *dij* is the actual distance between two nodes *vi* and *vj* and *d'ij* is the distance between the corresponding points *pi* and in the k-dimensional space. Thus if we can maintain the distance between *pi* and *pj* the same as the distance between *vi* and *vj* then *dij = d'ij* for all pairs of nodes, then the stress is 0, which is the ideal mapping.

MDS initially starts with a configuration of points, called *initial configuration*. In this work, we assume MDS uses a random configuration. Thus, MDS initially starts with a random configuration of points. It then applies the some form of steepest descent iteratively to minimize the stress. Once the inputs are mapped into a k-dimensional space, one can use a multi-dimensional index structure to do nearest-neighbor and range searches.

In a tree-structured data, for example in a namespace, similar or related nodes are closer to each other and have less number of edges between them than the dissimilar nodes. The closer the nodes the shorter the distance between them in distance matrix [17,18,19,20], hence similar or related nodes are mapped closer to each other in a multidimensional space. In other words, MDS maps the similarity between points: similar nodes are mapped closer to each other and dissimilar ones are mapped far off from each other.

4.2 Step II: Find a Transformations Required to Map the Common Nodes of the Two Trees Closer to Each Other on the Space

Once both trees are mapped onto two separate k-dimensional spaces, we need to relate the common nodes of the two trees. To achieve this, we identify transformations required to map the common nodes from both trees to each other as close as possible in a shared space. In order to match the common nodes, we use the Procrustes alignment algorithm [15,26]. Given two sets of points, the Procrustes algorithm uses linear transformations to map one set of points on the other set of points. The general Procrustes algorithm seeks the isotropic dilation and the rigid translation, reflection and rotation needed to best match one configuration to another [15]. In our case, the inputs to the algorithm are the nodes (terms) common to *T1* and *T2*.

4.3 Step III: Use the Same Transformations to Map the Uncommon Nodes

The previous step returns the transformations required to modify the given spaces such that the common nodes of both trees conform to each other as much as possible. This matching between the common nodes of both trees can, then, be used to define the similarity between the uncommon nodes. Using the transformations identified in the previous step, the uncommon nodes in two trees are mapped into the space in terms of their distances from the common nodes in respective trees. The uncommon nodes of both trees that are approximately at the same distance or at the same distance range from the common nodes in their respective trees are likely to be similar and will be mapped close to each other in the shared k-dimensional space.

4.4 Step IV: Use Clustering to Find the Related Uncommon Nodes from the Two Trees

At this point we have two trees whose nodes are mapped onto a shared k-dimensional space such that the common nodes are close to each other in the space. Hence, we can use clustering and nearest-neighbor approaches to identify related uncommon nodes.

To retrieve the related points from the multi-dimensional space, we use a *k-means* [16] based clustering algorithm, which requires

centroids to be given to form clusters around. We use the nodes of one trees, as the centroids for the clustering and we use the distance in the Euclidean space to achieve clustering.

As a result, returned clusters contain the node in one tree specified as the centroid and one or more nodes from the other tree that are closest in the shared space. Thus, in the form of a cluster, we have pairs of nodes from two different trees that are similar to each other.

5. EXPERIMENTAL EVALUATION

In this section, we provide an experimental evaluation of the proposed approach for mining mappings between the nodes of hierarchical media data, metadata, and ontologies. In order to evaluate the proposed approach and to observe the effects of various data parameters (like the number of nodes in the two trees and their degrees or fanouts), we needed a large number of trees. Furthermore, we needed to be able to vary these parameters in a controlled manner to observe the performance under different conditions. Therefore, we systematically generated a large number of tree-structured data (i.e., the ground truth) with varying parameters and use these trees in our initial experimental evaluation. After observing the effectiveness of our algorithm using this ground truth, we also used a real collection of data and verified our results.
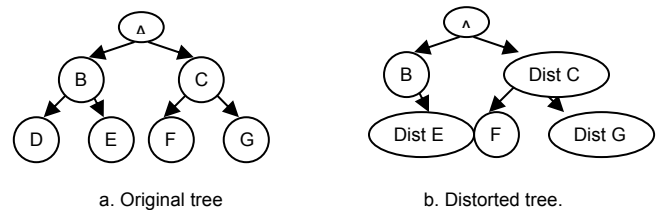

a. Original tree   b. Distorted tree.

Figure 7. Tree distortion process

5.1 Generating Ground Truth

The challenge addressed in this paper is to relate nodes of two trees that can differ from each other in terms of the number of nodes, density of nodes, and simply the node labels. Therefore, to generate two related but different trees, we

1. pick an original tree, and then

2. distort the original tree by relabeling existing nodes, deleting nodes existing nodes in, and adding new nodes to the original tree to generate a distorted tree (Figure 7).

Thus, the original and the distorted trees act as two similar trees, which are different in terms of the number of nodes and the labels of some of the nodes. The two trees have some nodes that are common (undistorted) and some nodes that are uncommon

5.1.1 Synthetic Tree Generation for Controlled Experiments

For the first set of experiments, where input data trees are generated in a controlled manner, we developed a tree generation program which creates a tree randomly based on two parameters: number of nodes and maximum fan out (degree) of the tree. For

our experiments, we have generated original trees with the configuration shown in Table1.

Table 1. Parameters for tree generation

Number of nodes in the tree	25, 50, 100 and 200
Fan out (degree)	1, 2, 4, 8 and 16

We generated (4*5 =) 20 sets of trees with a distinct combination of number of nodes and the fanout value. In our experiments, we used 5 different seed values for every combination; therefore, we have a total of 100 original trees for experiments. We report these results in Sections 5.3, 5.4, and 5.5.

5.1.2 Experiments with Real Trees
In addition to synthetics trees, we also run our experiments with the TreeBank data set [21], which has a deep recursive structure (whereas our synthetic trees were mostly balanced). We report these results in Section 5.6.

5.2 Terminology
Following are the terms used in discussing and explaining the experiment results:
- **Number of Nodes:** The total number of nodes in the original tree
- **Number of nodes that are mapped:** number of nodes in the original tree + the number of nodes in the distorted tree.
- **Correct mapping:** When a given query node of a given tree does map to the corresponding node of the other tree, then the mapping is said to be correct mapping.
- **Erroneous mapping:** When a given query node of a given tree does not map to the corresponding node of the other tree, then the mapping is said to be an erroneous mapping and each erroneous mapping has a degree of error, *err*:
  - mapping to a sibling of the correct node [*err*=1],
  - mapping to the parent node of the correct node (the correct node does not have a sibling) [*err*=1]
  - mapping to the parent node of the correct node (the correct node has at least one sibling) [*err*=2]
  - mapping to the sibling of the parent [*err*=3],
  - mapping to a distant node [*err*=4], and
  - no mapping.
- **Error Percentage:** This is the ratio of the erroneous mappings in the number of mappings returned. Note that, at most one of the mapped nodes can be a correct map.
  *(# erroneous mappings / # of nodes mapped)*100*
- **Precision:** The precision is measured as

$$\frac{m_1 + m_2 + \ldots + m_k}{k}$$

where *k* is the number of nodes returned and

$$m_i = \frac{1}{1+err_i}$$

is the degree of matching of node n_i in the result; i.e., a node with a lower error contributes more to the precision. Note that if the algorithm does not return any matches for a given node, then the corresponding precision is defined as 0.

5.3 Experiment I – Label Differences
In the first set of experiments, we aimed to see the performance of the proposed algorithm when the structures of the trees are identical, but some of the nodes are labeled differently.

For every original synthetic tree, we have generated 4 distorted trees using the *node rename* operation. The level of distortions experimented with are 5%, 25%, 45% and 65%. Therefore, we have a total of (100*4 =) 400 test cases. For every query node, the implementation returns the matching nodes from the other tree. For every original tree we run four tests. Figures 8, 9, and 10 provide the following observations:

- As the distortion increases the error also increases. Most common errors are due to nodes that are mapped to a sibling of the correct node.
- The error pattern observed is similar in case of trees with a total number of nodes 25, 50, 100, and 200.
- As the fan out increases, the error also increases. For trees with fanout 1 or 2, no errors are observed.
- The precision is close to perfect (1.0) for low renaming distortions. For heavy renaming (65%), the weighted precision can drop slightly for trees with large fanouts.
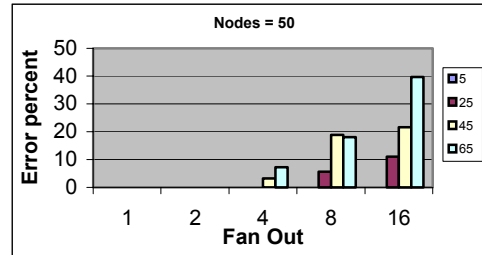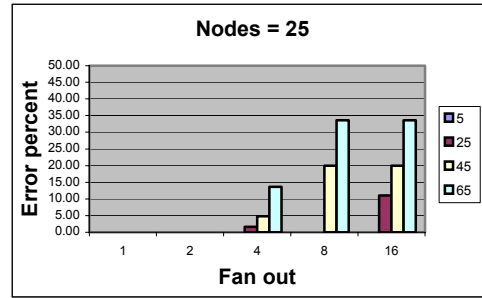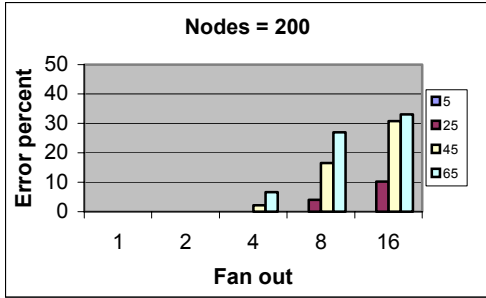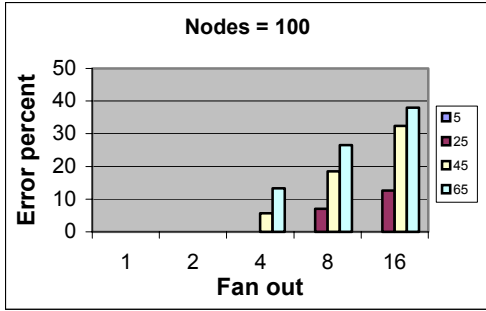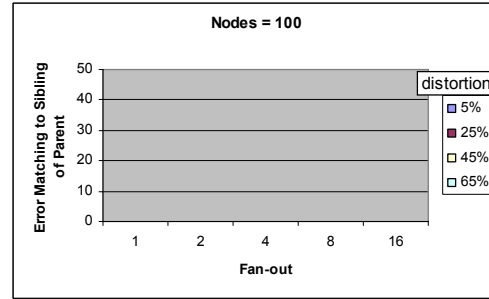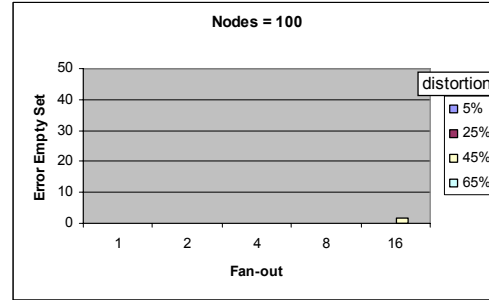
Figure 8. Percentage mapping error in Experiment I
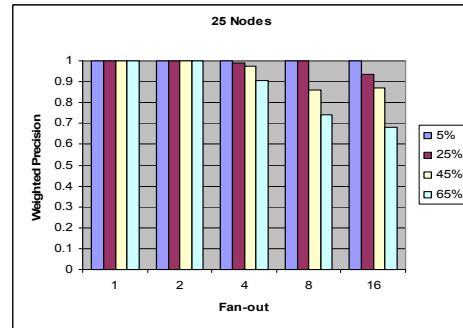

Figure 9. Types of the errors for Experiment I
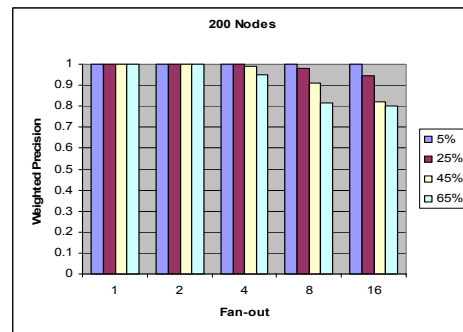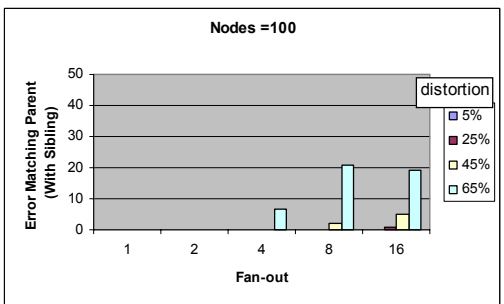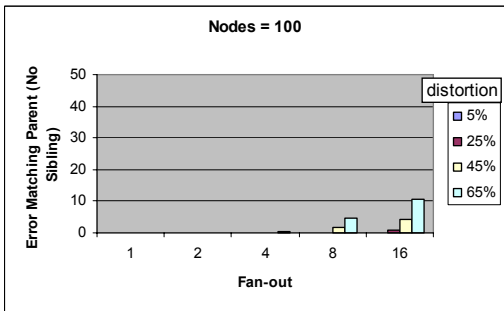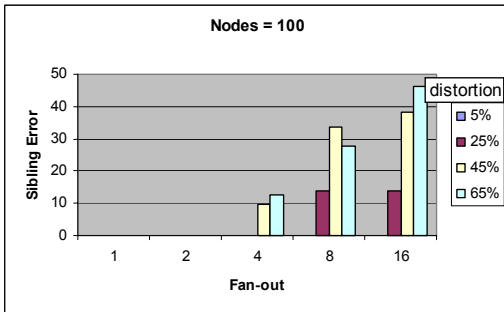

Figure 10. Weighted precision in Experiment I

5.3.1 The effect of distortion

The similarity mapping of the nodes of the original tree and the distorted tree is based upon two things: The distances between the nodes in each tree and the distances between the common nodes in both trees. The higher the number of common nodes between the two trees, the more similar the two trees are. Hence resulting mappings between the distorted nodes and the original nodes are

better. Increase in the distortion lessens the number of common nodes between the two trees; as a result, the error rate gets higher.

5.3.2 The effect of fanout

For trees with maximum fanout 2, there is a high probability of correct mapping when only one of the two siblings is mislabeled. However, when the fanout is higher, the probability that siblings (especially the leaf siblings that are structurally identical to each other) will be erroneously mapped to each other increases. Hence, the rate of correct mapping decreases when the fanout increases.

5.3.3 Types of errors

Figure 9 presents different types of errors for trees with 100 nodes (the results are similar in other tree sizes as well). In Experiment I, the only operation that causes distortion is "renaming". Hence, although the names of some of the nodes are modified, the structure of the distorted tree is maintained. Consequently, in most erroneous cases, the distorted nodes are simply mapped to a sibling of the correct node.

5.3.4 Precision

Figure 10, on the other hand, takes into account the *degree of match* even for those nodes that do not perfectly match the requested node. From this figure, it is again clear that the result precision is close to perfect (1.0) for low renaming distortions. In the case of heavy renaming (65%), the weighted precision drops as the fanout increases. However, the degree of drop is not significant, which means that even when the algorithm cannot find a perfect match, it returns a node close to what was expected.

More importantly, the results show that as the number of nodes in the tree increases, the weighted precision significantly improves. This shows that, as the number of available nodes increase, the distance-based mapping of nodes into the search space becomes more and more robust and this leads into better matches.

5.4 Experiment II – Structural Differences

In Experiment II, we used combinations of addition, deletion, and rename operations to generate distortions in the original synthetic trees. This enabled us to observe the performance when the structures of the trees are also variable. For each of the 100 original trees, we apply 3 levels of distortions:

- 15% (5% of addition + 5% of deletion + 5% of rename),
- 30% (10% addition + 10% deletion + 10% rename),
- 45% (15% addition + 15% deletion + 15% rename).

Figures 11, 12, and 13 provide the following observations:

- As the distortion increases, the error also increases. The most significant error is *empty matches*.
- *Unlike* Experiment I, when the fanout is 1 or 2, the error percentage is the highest. The error percent drops sharply for fanout value of 4 and it remains more or less constant as fanout increases.

We examine different effects in detail next.

5.4.1 The effect of distortion

In Experiment II, we used a combination of addition, deletion, and rename operations to generate distortion. The normal observation in this experiment is that almost 75% of the test-runs return results that do not exhibit entirely correct mapping. These errors are due to the increase in the overall distortion. The higher the distortion is, the lower is the number of common nodes between the two trees; hence, the greater is the probability of wrong mapping. The type of distortion is also a key factor that influences the error proportion. As expected, the change in the tree structure (due to additions and deletions) has a negative effect on the error percentage.

5.4.2 The effect of fanout

In trees with lower fanouts, each node is closely related (short distanced) to a few nodes. Each of these close nodes is highly important in achieving a correct mapping. If any of these close nodes is deleted, then the given node looses an important distance information. Hence, it becomes difficult to exactly map the node. As a result, either the node does not get mapped to any other node of the corresponding tree or it maps to the parent of the correct node (Figure 11). On the contrary, if the tree has high fanout, each node has a large number of siblings with which it is closely related. Even if one of these nodes is deleted, there are many other nodes for the given node to relate to. Although, there is an increased probability that the given node wrongly maps to a sibling, there is a relatively high probability of correct mapping.

5.4.3 Precision

Figure 13 shows the weighted precision obtained by the proposed algorithm in the case of a combination of distortions. From this figure, it is clear that the result precision is large for large fanouts. An increase in the number of nodes in the tree, on the other hand, has different effects depending on the fanout of the nodes. If the fanout is low, a larger tree actually means a significant drop in the precision. If the fanout is large, however, a higher number of nodes in the tree actually improve the precision. This is in accord with the drop in error in combined distortions (Figure 11).
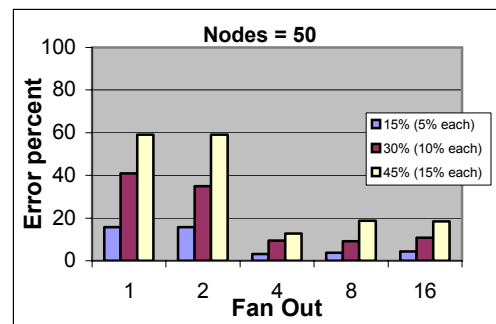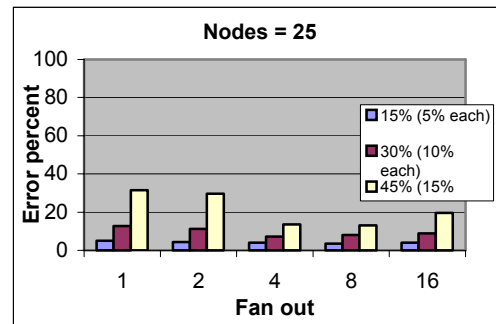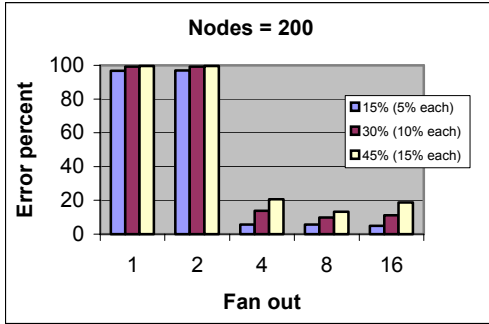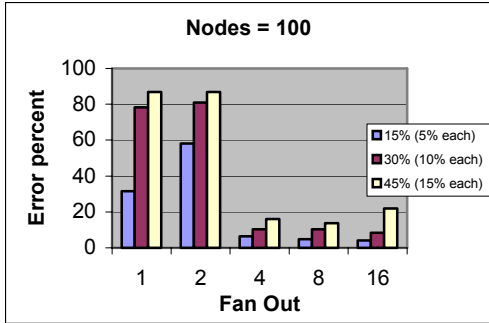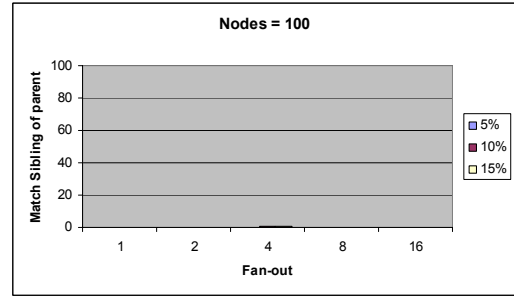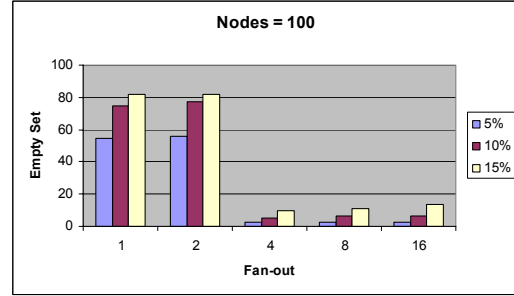
Figure 11. Percentage mapping error in Experiment II
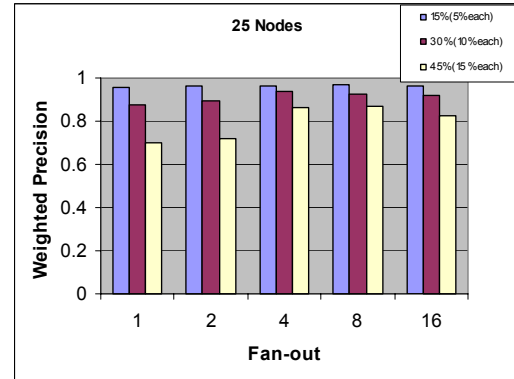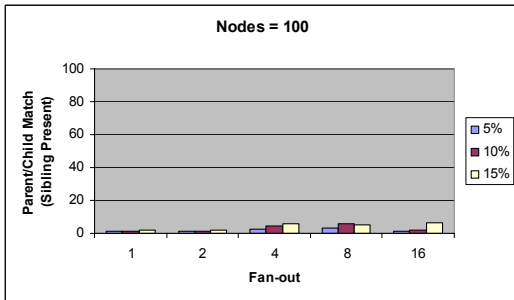

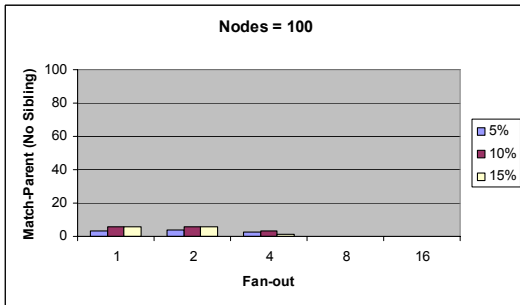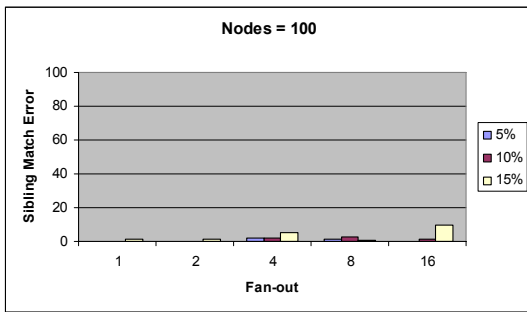Figure 12. Types of the errors for Experiment II
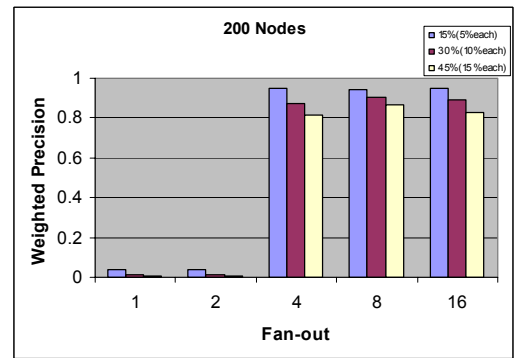
Figure 13. Weighted precision in Experiment II

5.5 Experiment I vs. Experiment II

As expected, better results were observed in Experiment I as compared to Experiment II. In case of Experiment I, the only distortions were caused by renaming. Thus, the structures of the

original and distorted tree were the same, resulting in better matches.

In these two experiments, we observed significantly different behaviors when it comes to the effects of fanouts.

- In Experiment I, smaller fanout means smaller chance of mapping a node to the sibling of the correct node. Hence, a smaller fanout translates into a smaller error rate.
- In the case of Experiment II, on the other hand, when the fanout is very low, the overall tree structure could be drastically changed by a small amount of node deletions and additions. Since the proposed algorithm is based on the structure of the, the resulting error rate was considerably high in cases with low fanout. For large fanouts, however, too much renaming is more detrimental than structural change, as without enough matching nodes.

5.6 Experiment III: TreeBank Collection

In addition to the synthetic trees we used in Experiments I and II, we also run additional experiments with the TreeBank data set available at [21]. The deep recursive structure of this data set (maximum depth 36, average depth, 7.87), in contrast to the mostly balanced structures we used in experiments with synthetic trees, also provides opportunities for additional observations. For the experiments with real data, in order to observe the effects of distortion, we clustered the trees in the collection based on their numbers of nodes. Therefore, for instance, if we wanted to observe the precision of our algorithm for trees with 100 nodes, from the collection we selected trees that have around 100 nodes. Then, we applied various types of distortions on these trees as explained earlier.

5.6.1 Experiment I: Label Differences on TreeBank Data

Figure 14 shows the weighted precisions obtained by the proposed algorithm in experiments with TreeBank data (with only node relabelings). The results show that the proposed algorithm is very robust with respect to relabeling errors in real data. Even when 65% of the nodes are relabeled, the approach is able to identify the correct node with upto 90% precision. When we compare the results presented in this figure with the results obtained using synthetic trees (Figures 10 and 11), we see that for large fanouts, the precision the algorithm provides on real data is significantly larger (up to 90% precision even with 65% relabelings) than the precision obtained on synthetic tree sets (60% precision with 65% relabelings).

We observed that for trees with 200 nodes around 70% of the errors were due to nodes that did not match any other node, 14% of the errors were due to nodes that matched their siblings, and another 14% were due to nodes that matched sibling of their parents. This is in contrast with the results with synthetic data (Figure 9) where the no mapping errors were close to 0. Nevertheless, the overall precision was higher than the case for synthetic data; i.e., when there were nodes that are returned in the result set, the errors of these nodes were closer to 0.

5.6.2 Experiment II: Structural Differences on TreeBank Data

Figure 15 shows the weighted precisions obtained by the proposed algorithm in experiments with TreeBank data (all types of distortions, including additions and deletions of nodes, are allowed). From this figure, it is clear that the precision behavior of the proposed algorithm in real data matches the precision behavior obtained using the synthetic tree set we have used in the previous experiments (Figures 14 and 15). As expected, due to structural distortions, the weighted precisions are lower than the case for only relabelings, but they are above 80% even with 45% combined distortions.

5.7 CONCLUSIONS

The functioning of an automated multimodal media integration system requires access to metadata that describe the individual media resources. The metadata are generally application dependent. Therefore, an automated media integration mechanism needs to mine and relate the common and uncommon components. In this paper, we develop algorithms to automatically discover mappings in hierarchical media data, metadata, and ontologies. The proposed algorithm uses the structural information to mine and map the semantically similar but syntactically different terms and components. We extensively evaluated the performance of the algorithm for various parameters and showed that the algorithm is very effective in achieving a high degree of correct matches.
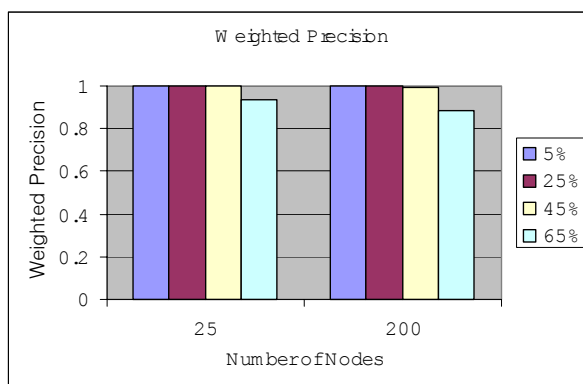

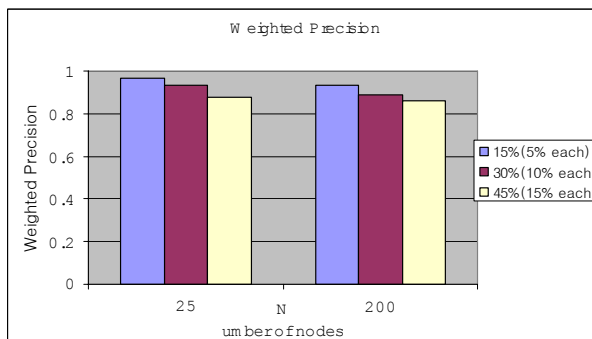**Figure 14. Weighted precision in experiments with TreeBank data (only rename distortions)**

Figure 15. Weighted precisions in experiments with TreeBank data (with structural distortions)

6. REFERENCES

[1] D.Brickley and R.Guha. Resource Description Framework (RDF) schema specification,2000. http://www.w3.org/TR/RDF-schema.

[2] Y. Lafon and B. Bos. Describing and Retrieving Photos using RDF. 2000. http://www.w3.org/TR/photo-rdf

[3] Dublin Core Initiative And Metadata Element Set, http://dublincore.org

[4] J.B. Kruskal. Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis, Psychometrika, 29(1):1-27, March 1964.

[5] R.V. Guha and T. Bray. Meta Content Framework Using XML. 1997. http://www.w3.org/TR/NOTE-MCF-XML-970624

[6] O.Lassila. Introduction to RDF Metadata. 1997. http://www.w3.org/TR/NOTE-rdf-simple-intro

[7] Namespaces in XML. http://www.w3.org/TR/REC-xml-names

[8] Extensible Markup Language (XML). http://www.w3.org/TR/REC-xml

[9] Document Object Model (DOM). http://www.w3.org/DOM/

[10] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A Database Management System for Semistructured Data. SIGMOD Record, 26(3):54-66, September 1997.

[11] Extensible 3D (X3D™) Graphics. http://www.web3d.org/x3d.html

[12] J.B. Kruskal. Nonmetric MultiDimensional Scaling: A Numerical Method, Psychometrika, 29(2):115-129, June 1964.

[13] W.S. Torgerson. *Multidimensional scaling: I. theory and method,* Psycometrika, 17:401-419, 1952.

[14] J.B. Kruskal and Myron Wish. *Multidimensional scaling,* SAGE publications, Beverly Hills, 1978.

[15] J. Gower. Generalized procrustes analysis. Psychometrika, 40:33–51, 1975.

[16] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations, Proc. 5th Berkeley Symp. Math. Statist. Prob., 1967, Vol. 1, pp. 281-297.

[17] P. Resnik. Sematic Similarity in a Taxanomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. Journal of Artificial Intelligence Research, 11, pp.95-130, 1999.

[18] P. Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxanomy. IJCAI, pp. 448-453, 1995.

[19] Rada, R., Mili, H., Bicknell, E., and Blettner, M. Development and Application of a Metric on Semantic Nets. IEEE Transactions on Systems, Management, and Cybernetics, 19(1), pp. 17-30, 1989.

[20] Lee, J., M. Kim, and Y. Lee, Information Retrieval Based on Conceptual Distance in IS-A Hierarchies. Journal of Documentation, 1993. 49(2): p. 188-207.

[21] University of Pennsylvania Treebank Project collection at http://www.cs.washington.edu/research/xmldatasets/www/repository.html

[22] W.-S. Li, K.S. Candan, Q. Vu, and D. Agrawal, Query Relaxation by Structure and Semantics for Retrieval of Logical Web Documents, TKDE 14(4): 768-791, 2002.

[23] K.S. Candan, Wen-Syan Li: Discovering Web Document Associations for Web Site Summarization. DaWaK 2001: 152-161

[24] K.S. Candan, W.-S. Li: Using Random Walks for Mining Web Document Associations. Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pp. 294-305, 2000.

[25] K.S. Candan, W.-S. Li. On Similarity Measures for Multimedia Database Applications. Knowledge and Information Systems 3(1): 30-51, 2001.

[26] D.G. Kendall. Shape Manifolds: Procrustean metrics and complex projective spaces. *Bulletin of the London Mathematical Society*, 16:81–121, 1984.