# Toward Approximate GML Retrieval Based on Structural and Semantic Characteristics

Joe Tekli[1], Richard Chbeir[1], Fernando Ferri[2], and Patrizia Grifoni[2]

[1] LE2I Laboratory UMR-CNRS, University of Bourgogne
21078 Dijon Cedex France
{joe.tekli,richard.chbeir}@u-bourgogne.fr
[2] IRPPS-CNR, via Nizza 128, 00198 Roma, Italy
{fernando.ferri,patrizia.grifoni}@irpps.cnr.it

**Abstract.** GML is emerging as the new standard for representing geographic information in GISs on the Web, allowing the encoding of structurally and semantically rich geographic data in self describing XML-based geographic entities. In this study, we address the problem of approximate querying and ranked results for GML data and provide a method for GML query evaluation. Our method consists of two main contributions. First, we propose a tree model for representing GML queries and data collections. Then, we introduce a GML retrieval method based on the concept of tree edit distance as an efficient means for comparing semi-structured data. Our approach allows the evaluation of both structural and semantic similarities in GML data, enabling the user to tune the querying process according to her needs. The user can also choose to perform either *template* querying, taking into account all elements in the query and data trees, or *minimal constraint* querying, considering only those elements required by the query (disregarding additional data elements), in the similarity evaluation process. An experimental prototype was implemented to test and validate our method. Results are promising.

**Keywords:** GML Search, Ranked Retrieval, Structural & Semantic Similarity, GIS.

## 1 Introduction

In recent times, the amount of spatial data, available in standalone as well as web-based Geographic Information Systems (GISs), is becoming huge and accessible to users who are generally non-experts. Most of the time, such users query data without a deep knowledge about the spatial domain they want to query, or they may not know how to formulate meaningful queries, resulting in a reduction of the quality of the query results. In order to overcome such limitations, the introduction of some query relaxation mechanisms, by which approximate and ranked answers are returned to the user, represents a possible solution. The need of answers that approximately match the query specified by the user requires the evaluation of similarity.

Another important new trend in GISs is the adoption of XML-based formats, particularly GML (Geography Mark-up Language) [18] as the main standard for exchanging geographic data and making them available on the Web. This language is based on W3C's XML (eXtensible Mark-up Language) encoding, as an efficient and widely accepted means for (semi-structured) data representation and exchange. In fact, a geographic entity in GML, consists of a hierarchically structured self-describing piece of geographic information, made of atomic and complex features (i.e., containing other features) as well as atomic attributes, thus incorporating structure and semantically rich data in one entity. Hence, the problem of evaluating GML similarity in order to perform approximate querying, can be reduced to that of performing XML-based search and retrieval, considering the nature and properties of geographic data and data requests.

A wide range of algorithms for comparing semi-structured data, e.g., XML-based documents, have been proposed in the literature. These vary w.r.t. the kinds of XML data they consider, as well as the kinds of applications they perform. On one hand, most of them make use of techniques for finding the edit distance between tree structures [3, 17, 26], XML documents being modeled as Ordered Labeled Trees (OLT). On the other hand, some works have focused on extending conventional information retrieval methods, e.g., [1, 6, 8], so as to provide efficient XML similarity assessment. In this study, we focus on the former group of methods, i.e., edit distance based approaches, since they target rigorously structured XML documents (i.e., documents made of strictly tagged information, which is the case of GML data, cf. Section 3) and are usually more fine-grained (exploited in XML structural querying [23], in comparison with *content-only* querying in conventional *IR* [22]). Note that information retrieval based methods target loosely structured XML data (i.e., including lots of free text) and are usually coarse-grained (useful for fast simple XML querying, e.g., keyword-based retrieval [8]).

Nonetheless, in addition to quantifying the structural similarities of GML features, semantic similarity evaluation is becoming increasingly relevant in geospatial data retrieval as it supports the identification of entities that are conceptually close, but not exactly identical. Identifying semantic similarity becomes crucial in settings such as (geospatial) heterogeneous databases, particularly on the Web where users have different backgrounds and no precise definitions about the matter of discourse [21]. Thus, finding semantically related GML modeled items, and given a set of items, effectively ranking them according to their semantic similarity (as with Web document retrieval [13]), would help improve GML search results.

In this study, we present the building blocks for a GML retrieval framework, evaluating both *structural* and *semantic* similarities in GML data, so as to produce approximate and ranked results. Our query formalism is based on approximate tree matching as a simple and efficient technique to query GML objects. It allows the formulation of *structure-and-content* queries with only partial knowledge of the data collection structure and semantics. In addition, our method allows both *template* and *minimum constraint* querying. According to the latter interpretation, the GML answer entity could contain additional elements w.r.t. those required by the query, such elements being disregarded in similarity evaluation. Yet, following the former strategy, all query and data elements are equally considered. The user can also tune the GML similarity evaluation process, by assigning more importance to either structural or semantic similarity, using an input structural/semantic parameter.

The remainder of the paper is organized as follows. Section 2 briefly reviews the state of the art in GML search methods and related XML similarity issues. Section 3 discusses the background and motivations of our study. In Section 4, we develop our GML approximate query evaluation approach. Section 5 presents our preliminary experimental tests. Section 6 concludes the paper and outlines future research directions.

## 2 State of the Art in GML and XML Retrieval, and Related Issues

Conventional geographic information science and retrieval have been concerned with managing and searching digital maps where geometry plays a major role (e.g., spatial browsing, exact querying based on geographic coordinates, …) [11]. Nonetheless, little support has been provided for managing geographic information based on text, in which references to locations are primarily by means of place names and textual descriptions, in addition to the geospatial data itself [10]. In this context, very few approaches have been proposed for GML-based geographic data search and retrieval in particular.

The few existing methods for managing and querying GML-based geographic information have tried to map GML data to classic spatial DBMS (e.g., Oracle Spatial, DB2 Spatial, PostGIS, …), e.g., [19, 27, 31]. This is connected with the genesis of GML, which was born as an interchange format for heterogeneous geographic database systems. Such methods usually underline the semi-automatic mapping of the GML application schema (describing the geographic data) to a bunch of object/relational schemas. They extend XML data storage in traditional DBMS to consider geospatial properties of GML (e.g., adding dedicated structures for storing geographic coordinates). Having mapped the GML data into object/relational DB structures, corresponding geographic data can be hence processed for classic DB querying. While such techniques might be efficient w.r.t. storage and indexing, they are limited to exact querying and retrieval functions, and do not allow approximate and ranked search results.

On the other hand, approximate querying methods developed for XML, e.g., [1, 6, 7, 14], cannot be straightforwardly applied to GML. First, such methods would have to consider the semantic richness of text-based geographic data in order to perform relevant GML querying. Second, most of these methods are based on underlying IR-concepts (most address the INEX evaluation campaigns) and target loosely structured XML (including lots of free text). They can be generally criticized for not sufficiently considering the structural properties of XML [20] (and consequently GML, which usually underlines rigorously structured data, due to the structured nature of geographic information).

Some methods have tackled the problem of searching rigorously structured XML data, by exploiting the concept of approximate tree matching [9, 23, 24]. In [23], the author propose an approach based on tree edit distance for evaluating the similarity between XML query and data trees. Similarity is evaluated in terms of the total cost needed to transform the query tree into one embedded in the data tree, and is used to rank the results. In a subsequent study [24], the authors propose to combine approximate tree embedding with TF-IDF ranking in evaluating query and data tree similarity. The classical notion of term (as a piece of text) is extended to structural term (as a sub-tree). Methods in [23, 24] only focus on the structural features of XML, disregarding semantic similarity. A method based on a similar approximate tree embedding technique is provided in [9] for querying MPEG-7 XML documents. Here, the authors introduce semantic similarity assessment between query and data tree labels, based on a dedicated knowledge base describing MPEG-7 concepts. Yet, the approach does not produce ranked results (it returns a Boolean value indicating whether the query tree is embedded or not in the data tree).

## 3 Background and Motivation

### 3.1 A Glimpse on GML

The Geography Markup Language (GML) [18] is an XML encoding for the transport and storage of geographic information, where real world entities can be represented as sets of GML features. Geometric features are those with properties that may be valued geometrically (e.g., types *Point*, *Line*, *Polygon…*, with geometric coordinates designating their positions, extents and coverage). Remaining non-geometric features provide textual descriptions of the geographic entity at hand. For instance, to model a *Hotel* in GML, one would define non-geometric features, such as *Name (Text)*, *Rank (Number)*, *Address (Text)*, … and geometric ones, e.g., *Location (Point)*, *Area (Polygon)*, …

Features/attributes and corresponding types, in a given GML modeled entity, are defined via the GML application schema to which the document, containing the GML entity model, conforms. The GML schema defines the features and attributes of the GML documents they describe, as well as their structural dispositions and the rules they adhere to in the documents. Similarly to schemas in traditional DBMS, GML schemas are valuable for the protection, indexing, retrieval and exchange of corresponding documents [18]. Figure 1 shows a sample GML document and part of its corresponding GML schema.

```
  <?xml version="1.0">
  <xmlns:gml="http://www.opengis.net/gml" City.xsd …>
  <City name= "Rome">
 <ArtisticGuide>
     <Monuments>
         <Cathedral name= "St Peter">
             <Style>Renaissance</Style>
             <Location>
                 <Point>
                     <Coordinates>
                         <Latitude>
                             <Degrees>41</Degrees>
                             <Minutes>52</Minutes>
                         </Latitude>
                             …
                          </Coordinates>
                 </Point>
             </Location>
             …
         </Cathedral>
     </Monuments>
  </ArtisticGuide>
  </City>
```

```
     <?xml version="1.0">
     <xs:schema xmlns:gml="http://www.opengis.net/gml" …>
 <element name="City" type=="CityType"/>
     <complexType name="CityType">
    <sequence> <element name = "ArtisticGuide" type ="ArtisticGuideType"/>…
        </sequence>
        <attribute name="Name" type="String">
     </complexType>
     <complexType name="ArtisticGuideType">
        <sequence> <element name = "Monuments" type=="MonumentsType"/>
    </sequence>
     </complexType>
     <complexType name="MonumentsType">
       <sequence> <element name="Cathedral" type="CathedralType"/> …
       </sequence>
     </complexType>
     <complexType name="CathedralType">
       <sequence>
         <element name="Style" type="String">
         <element name="Location" type="LocationType"> …
       </sequence>
       <attribute name="Name" type="string"/>
     </complexType> …
```

**Fig. 1.** Sample GML document and part of the corresponding GML application schema

## 3.2 Querying GML Data

In order to allow efficient approximate and ranked GML querying on the Web, we underline the need for a technique to search GML data where users can express queries in the simplest form possible, taking into account the structured nature of GML, in a way that less control is given to the user and more of the logic is put in the ranking mechanism to best match the user's needs. In other words, we aim to simplify, as much as possible, the query model and predicates (developed in the following section) without however undermining query expressiveness. In this context, we distinguish between two different kinds of GML queries, i) *template* where the user specifies a sample snapshot of the GML data she is searching for (e.g., a piece of map, providing a somewhat complete description of the requested data), or ii) *minimal constraint* where the user only identifies the minimal requirements the data should meet in order to belong to the query answer set (e.g., providing a small description, or an approximate location to pinpoint a given geographic object). For instance:

- $Q_1$: *"Find Churches built prior to 1600".*
- $Q_2$: *"Find Cities containing gothic churches".*
- $Q_3$: *"Pinpoint Locations of churches in the city of Rome".*
- $Q_4$: *"Find all restaurants situated at latitude 41 degrees 55 minutes North, and longitude 12 degrees 28 minutes East".*

While queries $Q_{1\text{-}3}$ are solely user-based, queries combining minimal constraint user preferences and geo-coordinates could be equally relevant. Consider for instance query $Q_4$ where the user searches, via her mobile GPS device, for restaurants in the vicinity of her current location. Such queries could also be viewed as of partial *Template* style, due to the presence of rather detailed geographic information provided by the GPS device.

Note that in this study, we do not aim to define a GML querying language (i.e., a formal syntax following which a GML query should be written), but rather the underlying querying framework. The user could formulate the query using plain text (guided by a dedicated GUI), or via some predefined syntax (e.g., a GML document fragment, or a pictorial representation converted into GML [5]) to be represented in our query model.

In addition, we emphasize on the need to consider the semantic meaning of GML entity descriptions and corresponding textual values, as a crucial requirement to perform approximate GML querying. For instance, a user searching for *cities* with *cathedrals* of *Gothic* style (query $Q_1$), would naturally expect to get as answers *cities*, *counties* or *towns* containing either *basilicas*, *cathedrals*, *churches* or *temples* which are of either *Gothic*, *Medieval* or *Pre-renaissance* styles, ranked following their degrees of semantic relevance to the original user request. The impact of semantic similarity on approximate GML querying is further discussed in the following section.

# 4 Proposal

## 4.1 GML Data and Query Models

As shown above, geographic entities in GML represent hierarchically structured (XML-based) information and can be modeled as Ordered Labeled Trees (OLTs)

[28]. In our study, each GML document is represented as an OLT with nodes corresponding to each subsumed feature and attribute. Feature nodes are labeled with corresponding element tag names. Feature values (contents) are mapped to leaf nodes (which parent nodes are those corresponding to their features' tag names) labeled with the respective values. To simplify our model, attributes are simply modeled as atomic features, corresponding nodes appearing as children of their encompassing feature nodes, sorted by attribute name, and appearing before all sub-element siblings.

Values could be of different types (*text*, *number*…), and user derived types could also be defined [18]. In the following, and for simplicity of presentation, we consider the basic *text*, *number* and *date* types in our discussion (from which derive most data-types, including geometric ones, e.g., *point*, *polygon*…). Note that our GML tree model itself, and the GML querying approach as a whole, are not bound to the types above, and could consider any other data-type, as we will show subsequently.

**Definition 1 - GML Tree:** Formally, we represent a GML document as a rooted ordered labeled tree $G = (N_G, E_G, L_G, T_G, g_G)$ where $N_G$ is the set of nodes in $G$, $E_G \subseteq N_G \times N_G$ is the set of edges (feature/attribute containment relations), $L_G$ is the set of labels corresponding to the nodes of $G$ ($L_G = Fl_G \cup Fv_G \cup Al_G \cup Av_G$ such as $Fl_G$ ($Al_G$) and $Fv_G$ ($Av_G$) designate respectively the labels and values of the features (attributes) of $G$), $T_G$ is the set of data-types associated to the feature and attribute nodes of $G$ ($T_G= \{GeoEntity\} \cup FT \cup AT$, having $FT = AT = \{Text, Number, Date\}$), and $g_G$ is a function $g_G : N_G \rightarrow L_G, T_G$ that associates a label $l \in L_G$ and a data-type $t \in T_G$ to each node $n \in N_G$. We denote by $R(G)$ the root node of G, and by $G' \lhd G$ a sub-tree of G ●

**Definition 2 - GML Tree Node:** A node $n$ of GML tree $G = (N_G, E_G, L_G, T_G, g_G)$ is represented by a doublet $n = (l, t)$ where $l \in L_G$ and $t \in T_G$ are respectively its label and node data-type. The constituents of node $n$ are referred to as $n.l$ and $n.t$ respectively (Figure 2) ●

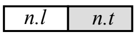

**Fig. 2.** Graphical representation of GML tree node *n*

Value data-types in our GML tree model are extracted from the corresponding GML schema. In other words, in GML tree construction time, the GML document and corresponding schema are assessed simultaneously so as to build the GML tree. Textual values are treated for stemming and stop word removal, and are mapped to leaf nodes of type *Text* in the GML tree. Numerical and date values are mapped to leaf nodes of types *Number* and *Date* respectively. As for the GML feature/attribute nodes themselves, they are assigned the data-type *GeoEntity*, their labels corresponding to the geographical entity names defined in the corresponding GML schema. To model the GML data repository, we connect all GML trees to a single root node, with a unique label (e.g., *'Root'*).

Consider for instance the GML data repository in Figure 3. It is made of two GML document trees describing *City* geographic entities (cf. extracts of GML document and schema in Figure 1). Geometric coordinates are depicted for the geographic entity

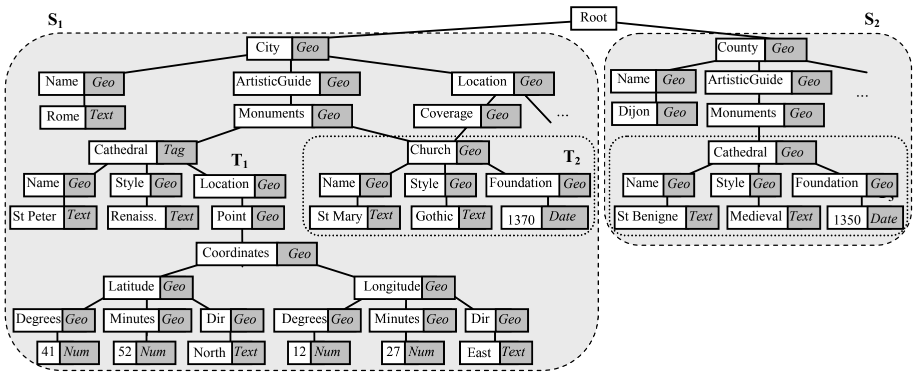

**Fig. 3.** Extract of a sample GML data repository (*Geo* stands for *GeoEntity*)

describing *St Peter cathedral* in *Rome* (latitude and longitude coordinates), and are omitted for remaining GML entities for clarity of presentation. Recall that most geographic data-types can be expressed in terms of basic types *Text*, *Number* and *Date*, which is the case of element *Point* (of derived GML *PointType*).

On the other hand, our definition of a GML query is simple and consists of a GML tree, similarly to GML documents, with special leaf nodes to represent query predicates. A query with an *Or* logical operator is decomposed into a *disjunctive normal form* [23], and is thus represented as a set of GML trees, corresponding to the set of conjunctive queries.

**Definition 3 - GML Query:** It is expressed as a GML tree, $Q = (N_Q, E_Q, L_Q, T_Q, g_Q, n_d)$ (cf. Definition 1) encompassing a *distinguished* node $n_d$ underlining the matches in the data tree that are required as answers to the query (i.e., the query's return clause). The query's root node $R(Q)$ designates its search scope/context. Its set $T_Q$ encompasses the *GeoEntity* type for distinguishing GML geographic entities, and predicate types $P\_t_i$ corresponding to every GML value data-type $t_i$ considered in the GML data model (e.g., $T_Q = \{GeoEntity\} \cup \{P\_Text, P\_Number, P\_Date\}$) ●

**Definition 4 - GML Query Node:** It is a GML tree node (cf. Definition 2) with additional properties to represent predicates. With $n.t = P\_t_i$ (predicate corresponding to GML data-type $t_i$), the node's label $n.l$ underlines a composite content made of the predicate operator $n.l.op$ and value $n.l.val$ (e.g., leaf node $Q_1[2]$ of query $Q_1$ in Figure 4 is of $Q_1[2].l.op$ = '<' and $Q_1[2].l.val$ = '*1600*', having $Q_1[2].t = P\_Date$, which underlines that the predicate value '*1600*' is of type *Date*) ●

Note that each data-type has its own set of operators (e.g., $\{=, <, \leq, >, \geq\}$[1] for numbers and dates, and {=, *like, …*} for text). GML query trees, corresponding to the sample queries provided in Section 3.2, are depicted in Figure 4. Recall that query trees can be constructed via a dedicated GUI, which would suggest, on-the-fly, the list of possible query nodes following the context of the query at hand.

[1] The *difference* operator ($\neq$) is omitted due to its particular processing (to be addressed in an future study).

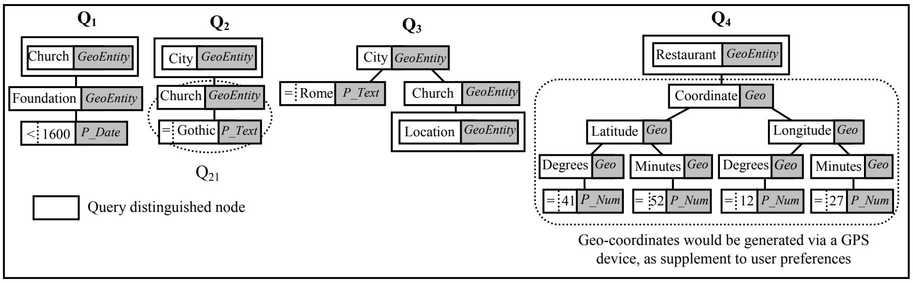

**Fig. 4.** Sample GML query trees

To the exception of the *containment* topological operator implicitly encoded in the GML hierarchy itself (cf. queries $Q_2$ and $Q_3$), we do not consider explicit spatial and temporal operators (e.g., *far*, *near*, *adjacent to*, …) in our current approach. These underline composite computational operations (e.g., a location point is *near* another location point if their distance, computed based on their coordinates, is below a certain threshold) and would induce more complex GML document and query graph structures instead of simple trees (introducing different kinds of cross links connecting GML entities). Recall that our current study sets the foundations toward approximate GML querying, to be consequently extended in addressing spatio-temporal relations.

**Definition 5 – Predicate Satisfaction:** Given a predicate GML query node $q_i$, and a data node $s_j$, $s_j$ satisfies $q_i$ ($s_j \models q_i$) if:

- The data node type corresponds to that of the query ($s_i.t \approx q_i.t$, i.e., $\forall\, t_r \in \{Text, Number, Date\}$, $q_i.t = P\_t_r \wedge s_j.t = t_r$),
- The data node label $s_j.l$ verifies the logical condition defined by $q_i.l$ ●

For instance, leaf node *$T_2[6]$* of data tree $T_2$ in Figure 3, having *$T_2[6].l$ = '1350'* and *$T_2[6].t$ = 'Date'*, satisfies predicate node *$Q_1[2]$* of query $Q_1$ in Figure 4, with *$Q_1[2].l$ = '<1600'* and *$Q_1[2].t$ = 'P_Date'*.

**Definition 6 - GML Query Scope:** Given a GML query *Q*, the scope of *Q* is identified by its root node *R(Q)*, and corresponds to the GML data sub-trees, in the data repository, having identical or semantically similar enough root nodes as that of the query. ●

We assume that the user defines, with the query, the kind of GML data she is looking for, i.e. the scope/context of her query. If for instance the root of the query is labeled *Restaurant*, then GML data in the context of GML data entity *Restaurant*, or semantically similar entities such as *Pizzeria*, *Bar*, … would naturally interest the user.

**Definition 7 - *Template* and *Minimal constraint* querying:** A GML query *Q* could be either evaluated as a i) *template* of the GML data the user is searching for, ii) or could represent the *minimal constraints* the data should meet to belong to the query answer set. In the former case, all GML query and data nodes would be considered in query/data similarity evaluation. Following the latter strategy, only elements required by the query tree are taken into account in query/data similarity evaluation, additional elements in the data tree being disregarded in the evaluation process. ●

Note that geographic queries most likely follow the *minimal constraint* style, the user usually specifying her information needs in the simplest form possible (cf. queries $Q_1$, $Q_2$ and $Q_3$ in Figure 4). Nonetheless, *template* querying could be particularly useful in *search-by-document* and *search-by-image* systems for instance, where the query could be a whole geographic document or a piece of map the user is searching for in the geographic repository. A *template* style query could be any of the sub-trees $S_1$, $S_2$, $T_1$… in Figure 3.

## 4.2 GML Query Evaluation

The goal of this work is to develop a method for searching a GML data repository in order to identify portions of data that exactly or approximately match user requests. Having modeled both GML data and queries as trees, GML query evaluation can be reduced to the problem of searching the various data sub-trees, in the data repository, corresponding to the query's search scope (i.e., with matching root nodes), identifying those that share structural and semantic similarities with the query tree. The result of the query would be a set of data nodes matching the query's distinguished node, ranked by the similarity degree between the query tree and corresponding data candidate answer sub-tree. Thus, we propose a GML querying framework based on the concept of tree edit distance as a widely known and efficient means for comparing XML-based tree structures [3, 4, 17]. In addition to evaluating GML data structure, our method also integrates semantic similarity assessment [12, 13], so as to capture the semantic meaning of GML element labels/values.

A simple motivating example, underlining the need to consider semantic similarity in GML querying, is that of evaluating query $Q_2$ of Figure 4, against the GML data repository *G* in Figure 3. Using structural-only similarity evaluation, one can realize that the only GML data tree to (actually) fulfill the data request of $Q_2$ (searching for *Cities* containing *Cathedrals* of *Gothic Style*) is $S_1$ (describing the *City* of *Rome*, containing data tree $T_2$ describing the *St. Mary Church* which is of *Gothic* style). That is due to the structural similarity between sub-tree $Q_{21}$ and $T_2$. However, one can recognize that data tree $S_1$ (describing the *County* of *Dijon*) also fulfills the data request of query $Q_2$, since it contains tree $T_3$ (describing the *St. Benigne Cathedral* which is of *Medieval* style). This answer goes undetected using structure-only similarity evaluation, since the semantic similarity between *City*/*County*, *Church*/*Cathedral* and *Gothic*/*Medieval* are missed.

In summary, our GML querying method consists of three main components: i) *CAT Identification* component for identifying GML data Candidate Answer Trees (following the query's scope), ii) *GML Tree Comparison* component for evaluating the structural and semantic similarity between the query tree and each of the candidate answer trees, iii) and the *Query Answer Identification* component for recognizing the GML data elements, in each data candidate answer tree, to be returned to the user (following the query's distinguished node). The overall system architecture is depicted in Figure 5.

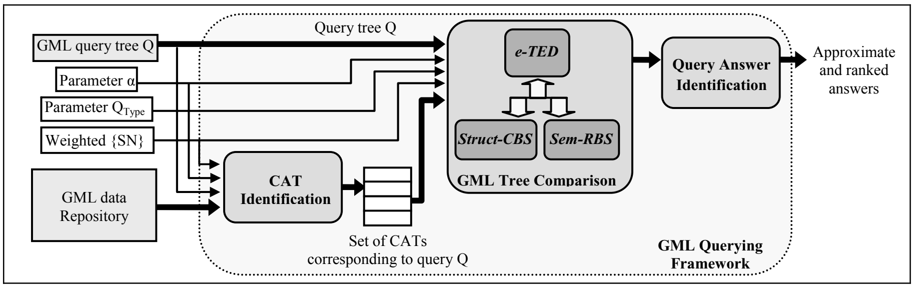

**Fig. 5.** Simplified activity diagram of our GML querying approach

### 4.2.1 GML Candidate Data Tree Identification

The first step in assessing a query is to identify its search scope. Following the traditional IR logic, whole physical files are considered as candidate answers. Nonetheless, GML documents differ in their granularity: some documents may contain information about *monuments*, others about *cities* containing hundreds of *monuments* (cf. Figure 3).

Obviously, it is not relevant to retrieve the entire *city* when the user is searching for certain *monuments*. Hence, the GML query search scope should be identified dynamically, w.r.t. the query at hand.

Following our GML data and query model, the query scope (cf. Definition 6) can be identified as the set of GML data sub-trees (which we identify as Candidate Answer Trees, *CAT*s), in the data repository, having identical, or semantically similar enough, root nodes as that of the query (i.e., same/similar label, with the same data-type). Consider for instance query $Q_1$, searching for *churches* that have certain characteristics. When considering root node identity, query $Q_1$'s *CAT*s would be all data sub-trees having root node label *Church*, i.e., data tree $T_2$. When taking into account semantic similarity, $Q_1$'s *CAT*s would also encompass $T_1$ and $T_3$ of root nodes *Cathedral*. With queries $Q_2$ and $Q_3$, answer candidates would be data sub-tree $S_1$ (of root node *City*) when considering node identity, and would include $S_2$ (of root node *County*) when considering semantic similarity.

**Definition 8. Candidate Answer Tree:** Given a GML node similarity measure $Sim_{GML}$, reference semantic networks $\{SN\} = \{SN_{Geo}, SN_{Text}\}$ for evaluating the semantic similarity between GML *GeoEntity* and *Text* node labels, and a semantic similarity threshold α, the set of candidate answer trees $Q_{CAT}$, for a given query $Q$, in a GML data repository $G$, $Q_{CAT} = \{S \,/\, S \lhd G \wedge ((R(Q) = R(S)$ if $\alpha = 1) \vee Sim_{GML}(R(Q), R(S), \{SN\}) \geq \alpha$ otherwise$)\}$ ●

Our semantic similarity threshold also serves as a structural/semantic similarity parameter, underlying the extent of structural/semantic similarity considered while identifying candidate answers. It allows the user to assign more importance to the structural or semantic characteristics of GML data in answering the query at hand.

- For $\alpha = 1$, only *CAT*s with root nodes identical to that of the query are the only ones considered. This corresponds to purely structural querying.

- For $0 < \alpha < 1$, *CAT*s with root nodes of semantic similarity higher than $\alpha$ are considered. As $\alpha$ decreases, the size of the answer set $Q_{CAT}$ will increase, following the semantic similarities between query and *CAT* root nodes.
- For $\alpha = 0$, all data sub-trees in the GML data repository are considered as *CAT*s.

Parameter $\alpha$ is exploited throughout the querying framework to determine the amount of structural/semantic similarity considered in query/*CAT* comparison (cf. Section 4.2.2).

As for GML node similarity $Sim_{GML}$, it is evaluated w.r.t. the nodes' constituents, i.e. their labels and types and is developed subsequently.

### 4.2.2 GML Tree Comparison

Having identified the set of *CAT*s corresponding to the query at hand, the GML tree comparison component evaluates the structural and semantic similarity between the query tree and each of the *CAT*s, so as to provide the user with approximate and ranked results.

Our GML query/*CAT* tree comparison component combines and extends two recent approaches that target XML structure and semantic similarity respectively [25, 26]. It consists of four main modules for: i) identifying the *Structural Commonality Between two XML Sub-trees* (*Struct-CBS*) [26], ii) quantifying the *Semantic Resemblance Between two XML Sub-trees* (*Sem-RBS*) [25], and iii) computing *Tree Edit Distance* (*TED*). In short, the *TOC* algorithm makes use of *Struct-CBS* [26] and *Sem-RBS* [25] to structurally and semantically compare all sub-trees in the GML query tree and data tree (*CAT*) being compared. The produced sub-tree similarity results are consequently exploited as edit operations costs (node update, node insertion, tree insertion…) in an extension of Nierman and Jagadish [17]'s main edit distance algorithm. Here, *e-TED* identifies our extended edit distance algorithm (Figure 5).

Hence, the inputs to the GML tree comparison component are as follows:

- The GML query tree and data tree (*CAT*) to be compared,
- Parameter $\alpha \in [0, 1]$ enabling the user to assign more importance to the structural or semantic aspects of the GML query and data trees (*CAT*),
- Parameter $Q_{Type}$ enabling the user to chose between *template* or *minimal constraint* querying.
- Reference semantic networks $\{SN\}=\{SN_{Geo}, SN_{Text}\}$ to be utilized for semantic similarity evaluation.

The GML tree comparison component outputs the similarity (edit distance) value between the pair of query tree and data tree (*CAT*) being compared, based on the sum of corresponding edit operations costs. Hereunder, we first i) develop the GML node semantic similarity measure $\text{Sim}_{\text{GML}}$ exploited in computing edit operations costs, and then ii) show how the main tree edit distance algorithm *e-TED* exploits edit operations costs, and considers both template and minimal constraint querying in the GML query/data comparison process. Note that we skip the details concerning the inner-workings algorithms *Struct-CBS* and *Sem-RBS* mentioned above, since they have been thouroughly described in [25, 26].

*4.2.2.1 GML Node Similarity Measure*

As shown in Section 4.1, GML data (query) node labels either consist of geographic entity names, i.e., nodes of type *GeoEntity*, or geographic feature/attribute values (predicates), mainly *Text*, *Number* and *Date* (cf. Definitions 2 and 5). Obviously, it is meaningless to compare nodes encompassing different types of data (e.g., *GeoEntity* names with nodes bearing information of type *Date* or *Number*). Hence, we compute GML node similarity between corresponding node labels, given that the concerned nodes are of matching data-types, making use of similarity measures dedicated to the data-types at hand. We particularly focus on the semantic similarity $Sim_{Sem}$ between nodes bearing conceptual information, i.e., nodes of types *GeoEntity* and *Text*, where information can be described via groups of concepts, organized in knowledge bases or semantic networks. Here, exsiting semantic similarity measures (e.g. Lin [12], Wu and Palmer [29]…) could be exploited, taking into account the concerned reference semantic network:

- We define $SN_{Geo}$ as a semanitc network describing the semantic relations between the different geographical entities defined in the GML application schema (describing the data at hand), and exploit it in evaluating semantic similarity between *GeoEntity* node labels,
- We also exploit $SN_{Text}$ as a more generic semantic network describing concepts found in everyday language (e.g., WordNet [15]), to compare GML element/attribute textual values.

As for *Number* and *Date* labels, they bear non-conceptual information, i.e., information that cannot be described with concepts, organized in knowledge bases. Various methods for comparing such non-conceptual information has been addressed in classic database systems [16], e.g.:

$$Sim_{Number}(n_1.l,\ n_2.l) = 1 - \frac{|n_1.l - n_2.l|}{|n_1.l| + |n_2.l|}$$

A similar (yet more intricate) variation could be exploited for comparing dates. Formally, given a GML query node qi and a data node sj, and considering the basic data-types mentioned above (GeoEntity, Text, Number and Date):

$$Sim_{GML}(q_i, s_j, \{SN\}) = \left\{ \begin{array}{lll} 1 & if & s_j \models q_i \\ Sim_{Sem}(q_i.l,\ s_j.l,\ SN_{Geo}) & else\ if & q_i.t = s_j.t = \text{‘GeoEntity’} \\ Sim_{Sem}(q_i.l.val,\ s_j.l,\ SN_{Text}) & else\ if & q_i.t = P\_t_r \ \wedge\ s_j.t = \text{‘Text’} \\ Sim_{Number}(q_i.l.val,\ s_j.l) & else\ if & q_i.t = P\_t_r \ \wedge\ s_j.t = \text{‘Number’} \\ Sim_{Date}(q_i.l.val,\ s_j.l) & else\ if & q_i.t = P\_t_r \ \wedge\ s_j.t = \text{‘Date’} \\ 0 & & otherwise \end{array} \right\} \quad (1)$$

Recall that $s_i \models q_i$ underlines predicate satisfaction (Definition 5), i.e., query and data nodes are of corresponding types $q_i.t \approx s_j.t$, such as the data node satisfies the logical condition specified by the query. Similarity is obviously maximal (=1) when the data node satisfies the query's predicate node. If both query and data nodes underline the same data-type, similarity is evaluated following the corresponding similarity measure. Yet, if data-types are different, similarity is minimal (=0). Note that additional data-types could be considered in the same manner, by exploiting corresponding similarity measures.

#### *4.2.2.2 Edit Operations Costs*

Here, we provide the cost scheme for the *update* operation, as an example on how GML node similarity is exploited in computing edit operations costs. Remaing tree edit operations costs (i.e., node insertion/deletion, and tree insertion/deletion) follow similar costs schemes, integrating structural and semantic similarity scores accordingly. Given a GML query node $q \in Q$ (Definition 4) and GML data tree node $s \in S$ (Definition 2), the cost of the update operation *Upd(q, s)* applied to $q$ and resulting in GML query node $q'$ such as *(s=q' if q.t='GeoEntity')*$\vee$*(s* $\models$ *q' otherwise)* (i.e., if $q$ is of type predicate, $P\_t_i$), would vary as:

$$\text{Cost}_{\text{Upd}}(q, s, \alpha, \{SN\}) = \begin{bmatrix} 1- (1-\alpha)\times \text{Sim}_{\text{GML}}(q, s, \{SN\}) & \text{if } ((q \neq s) \wedge (s \not\models q)) \\ 0 & \text{otherwise} \end{bmatrix}$$

Parameter α is the structural/semantic parameter utilized in the *CAT Identification* component to assign more importance to either structural or semantic similarities:

- For α = *1*, only label equality/difference is considered in computing edit operations costs. Consequrently, *e-TED* will be considering the structural similarity between the query sub-tree $Q_{Sb}$ (rooted at node *q*) and the CAT tree.
- For α = *0*, label semantic similarity is considered between corresponding GML node and *CAT* node labels. Hence, *e-TED* will evaluate the structural and semantic similarity between the sub-tree $Q_{Sb}$ (rooted at *q*) and the CAT tree.

#### *4.2.2.3 TED Algorithm Extended to Template/Minimal Constraint Comparisons*

In short, *e-TED* starts by computing the cost of updating the root nodes of the trees being compared (Figure 6, line 4). Then, it computes the costs of deleting every first level sub-tree in the query tree (lines 5-10), and those of inserting every first level sub-tree in the *CAT* data tree (lines 10-16). Here, both structural and semantic similarity evaluation are considered when assigning edit operations costs (as briefly described above, [25, 26]).

On one hand, all (structurally and/or semantically weighted) operations are considered when performing *template* querying (i.e., all query/data tree elements are considered, which comes down to the classic *TED* formulation [17]). On the other hand, to allow *minimal constraint* querying, our *e-TED* disregards node and tree insertion operations in the computation process (Figure 6, lines 9 and 17). In other words, all additional elements in the *CAT* data tree will be disregarded in computing similarity, only considering those required by the query. Consequently, the algorithm recursively computes all combination of insertion, deletion and update operations to identify those yielding the minimum edit distance, i.e., the minimum cost edit script (lines 11-20). For instance, the result of comparing query $Q_3$ with data sub-tree $S_1$, following the minimal constraint strategy, is depicted in Figure 7. Here, only nodes required by the query are considered in the computation process, additional data nodes being disregarded (cf., edit distance mappings and mapping scores,

```
Algorithm e-TED()
Input: Query Tree Q and data tree S, parameter $\alpha$ for structural/semantic weighting, $Q_{Type}$
parameter, weighted semantic networks {SN}
Output: Edit distance between Q and S
Begin
M = Degree(Q)                 // The number of first level sub-trees in Q.          1
N = Degree(S)                 // The number of first level sub-trees in S.          2
Dist [][] = new [0...M][0…N]                                                        3
Dist[0][0] = $Cost_{Upd}$(R(Q), R(S), α, {SN})     //Update operation                4
For (i = 1 ; i ≤ M ; i++)  { Dist[i][0] = Dist[i-1][0] + $Cost_{DelTree}(Q_i)$ }     5
For (j = 1 ; j ≤ N ; j++)                                                           6
    {                                                                               7
      If ($Q_{Type}$=’Template’)  {Dist[0][j] = Dist[0][j-1] + $Cost_{InsTree}(S_j)$ }   8
      Else {Dist[0][j] = Dist[0][j-1]}          // $Q_{Type}$ = ‘Minimal Constraint’     9
    }                                                                               10
For (i = 1 ; i ≤ M ; i++)  {                                                        11
      For (j = 1 ; j ≤ N ; j++)  {                                                  12
          Dist[i][j] = min{                                                         13
                Dist[i-1][j-1] + TED($Q_i$, $S_j$),         //Dynamic programming     14
                Dist[i-1][j] + $Cost_{DelTree}(Q_i)$,                                15
                If ($Q_{Type}$=’Template’)  { Dist[i][j-1] + $Cost_{InsTree}(S_j)$ }    16
                       Else { Dist[i][j-1] }     // $Q_{Type}$=‘Minimal Constraint’      17
                }                                                                   18
      }                                                                             19
  }                                                                                 20
Return  Dist[M][N]        // Sim =1 / (1 + Dist))                                   21
End
```

**Fig. 6.** Edit distance algorithm *TED*

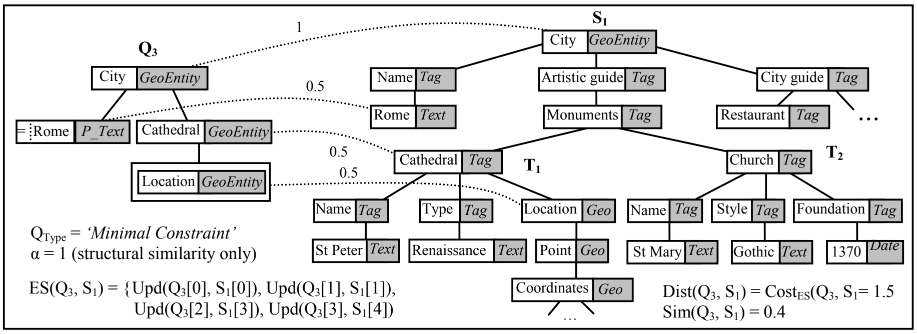

**Fig. 7.** GML query/CAT tree mappings

computational details being omitted due to space limitations). Note that with $Q_{Type}$= *‘Template’*, all additional nodes in $S_1$ (i.e., $S_1[6]$, $S_1[7]$, ...), would have to be considered in the similarity evaluation process, which would drastically decrease the similarity value.

Recall that similarity is computed based on the sum of the minimum cost edit operations corresponding to the query and *CAT* trees, i.e., inverse of edit distance (cf. Figure 6 line 21, and Figure 7 for computation example).

### 4.2.3 GML Query Answer Identification Component

The GML *Query Answer Identification* component underlines the elements in the data tree (*CAT*) which are to be returned to the user. These correspond to the nodes (along with their sub-trees), in the data tree, that match the GML query's *distinguished node* (cf. Definition 3). Such matches could be identified following a post-processing of the results (i.e., edit operations and mappings) produced by the *GML Tree Comparison component.*

In fact, one of the main advantages of using tree edit distance is that along the similarity (distance) value, a mapping between the nodes in the compared trees is provided in terms of the edit script, allowing the identification of correspondences between elements of the query tree and data tree (*CAT*) being compared. Consider for instance the edit distance mappings between GML query tree $Q_3$ and data CAT $S_2$, depicted in Figure 7. The number next to each mapping link designates its mapping score, which is inversely proportional to the cost of the corresponding edit operation. Consequently, mappings reveal the data node matching the distinguished query node, in our case $S_1[10]$=(*'Location', GeoEntity*). Hence data node $S_1[10]$ is returned to the user, along with its sub-tree (i.e., the geographic coordinates of the *St Peter* cathedral in *Rome*).

In the case where multiple data nodes match the query's distinguished node, we simply identify those with the highest mapping scores, i.e., those corresponding to the most relevant mappings. Note that a dedicated threshold, specifying the minimum acceptable mapping score for a node to be considered as a relevant match to the query's distinguished node, can be considered. In addition, when the query's distinguished node is the same as its root node (e.g., queries $Q_1$ and $Q_2$), its matching node in the data *CAT* would be none other than the data tree's root node itself. Thus, the whole data tree would be returned to the user.

## 5 Experimental Evaluation and Validation Tests

We have implemented our GML query evaluation approach in the *XS3* prototype system[2].Hereunder, we provide preliminary *precision* and *recall* results w.r.t. a select collection of GML queries (including $Q_1$, $Q_2$, $Q_3$ and $Q_4$ of Figure 4) applied on a GML data repository constructed based on geographic data sampled from Wikipedia (Figure 3). The current data repository includes geographic information concerning 40 major historical and artistic monuments in the cities of *Rome, Dijon* and *Sao Paolo*. Ten queries were considered, distributed equally between *minimal constraint* ($Q_{1\text{-}3}$ and $Q_{5\text{-}6}$) and partial *template* ($Q_4$ and $Q_{7\text{-}10}$) styles. Queries were first manually evaluated, identifying the set of relevant answers for each query, ranked following their order of relevance w.r.t. to the user (three different test subjects, one doctoral student and two post-doctoral researchers, were involved in the experiment). Manual answers were mapped to system generated ones so as to compute precision (*PR)*, recall (*R*) and F-measure values (*F-value*) accordingly.

Results in Figure 8 depict overall *PR*, *R* and *F-value* results for each query. These underline our approach's applicability and potential in identifying relevant answers to

[2] Available online at `http://www.u-bourgogne.fr/DbConf/XS3`

simple GML queries. Note that in our evaluation, we adopted the *range query* formalism without however utilizing a predefined similarity threshold in identifying answers. We rather selected the whole set of ranked system generated answers (CATs) bound by the least similar relevant one, i.e., the last answer (CAT) to actually correspond to a user defined answer (which similarity value was considered as the *range query* threshold). This allowed us to verify the performance of our method in selecting relevant answers (achieving high *recall*, crucial for any method to be admissible in search applications [22]), and most importantly its effectiveness in filtering out non-relevant ones (*precision*).

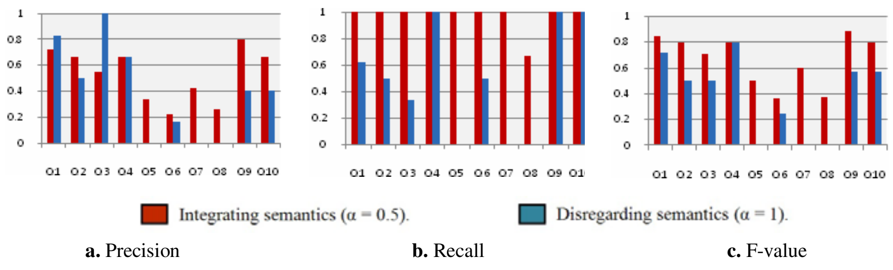

**Fig. 8.** Precision (*PR*), Recall (*R*) and F-measue (*F-Value*) results

High *recall* was particularly achieved when integrating semantic similarity evaluation, whereas quite a few relevant answers were missed when disregarding semantics (Figure 8.b). Semantic similarity evaluation also seemed crucial in amending *precision* (Figure 8.a). Queries $Q_5$, $Q_7$ and $Q_8$ are typical examples, where all relevant answers were missed by the system, when disregarding semantics (*PR*=*R*=*F-value*=0). However, the impact of semantic similarity evaluation seemed to decrease when searching geographic data based on their geometric attributes (e.g., coordinates) rather than textual descriptions, which was expected.

On the other hand, a major difference between the results achived with and without semantic similarity evaluation is *relevance ranking*. While single answers were usually obtained (for each query) when disregarding semantics and relying solely on GML data structure, the integration of semantic similarity resulted in the generation of a ranked set of answers, underlining their semantic similarities w.r.t. the geo-concepts in the query at hand. Ranking results are depicted in the *PR/R* graphs of Figure 9. Figures 9.a and 9.b show rather regular *PR/R* curves (*precision* decreasing gradually with the increase of *recall*), with queries $Q_1$ and $Q_3$ (Figure 9.a) clearly reflecting higher retrieval quality than their counterparts in Figure 9.b. Nontheless, some queries (e.g., $Q_5$, $Q_7$ and $Q_9$ of Figure 9.c) underlined relatively poor ranking capabilities, the system identifying and ranking non relevant answers (CATs) prior to relevant ones (*precision* starting at zero, and then increasing gradually w.r.t. *recall*, as relevant answers are added to the answer set). Further experiments are being conducted to analyze this effect, making use of dedicated relevance ranking metrics such as Kendall's tau and Spearman's footrule [2].

In addition, we have conducted timing experiments to verify the time complexity of the query evaluation process. Results show that the approach is linear in the size of

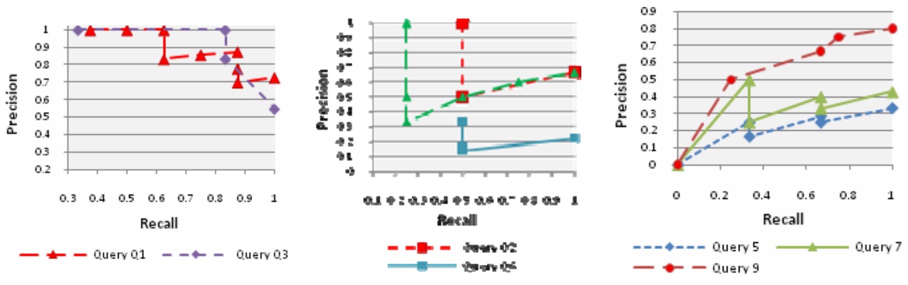

**Fig. 9.** *PR/R* graphs, obtained with semantic similarity evaluation

each of the query/data trees, as well as the size of the reference semantic network, when semantics comes to play, i.e., $O(|Q| \times |CAT| \times max(|SN_{Geo}|, |SN_{Text}|))$. Compexity graphs were omitted due to lack of space. Details concerning all experimental results are available online[3].

## 6 Conclusion

GML has been gaining growing attention as an effective means for geographic data representation and exchange in GISs on the Web. In this paper, we introduce the building blocks for an approximate GML retrieval method, considering both structural and semantic features of GML data, in the query evaluation process. Our query formalism is based on approximate tree matching as a simple and efficient technique to query GML. It allows the formulation of *structure-and-content* queries with only partial knowledge of the data collection structure and semantics, and enables both *template* and *minimum constraint* querying.

Preliminary experiments are promising, and underline the impact of semantic similarity on the query evaluation process. We are currently expanding our data testbed, in order to conduct more extensive experiments, also testing the ranking capabilities of the proposed methods using dedicated relevance ranking metrics such as Kendall's tau and Spearman's footrule [2]. We are also developing a web-based GUI to support the user in formulating queries, dynamically suggesting, following the corresponding input GML schema, the list of possible query nodes following the context of the query at hand. Considering spatio-temporal relations and predicates remains an obvious upcoming step. In this context, it might be interesting to extend our tree model to a more generic graph model, encompassing spatio-temporal links between geographic features, and thus try to adapt our tree edit distance algorithm accordingly.

## References

[1] Amer-Yahia, S., Lakshmanan, L., Pandit, S.: FleXPath: Flexible Structure and Full-Text Querying for XML. In: Proc. of the ACM Inter. Conf. on Management of Data (SIGMOD), pp. 83–94 (2004)

[2] Bar-Ilan, J.: Comparing rankings of search results on the Web. Information Processessing and Management (41), 1511–1519 (2005)

[3] At `http://www.u-bourgogne.fr/DbConf/GMLSearch`

[3] Chawathe, S.: Comparing Hierarchical Data in External Memory. In: Proceedings of VLDB, pp. 90–101 (1999)

[4] Dalamagas, T., et al.: A Methodology for Clustering XML Documents by Structure. Information Systems 31(3), 187–228 (2006)

[5] Ferri, F., Grifoni, P., Rafanelli, M.: The Management of Spatial and Temporal Constraints in GIS using Pictorial Interaction on the Web. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 92–105. Springer, Heidelberg (2004)

[6] Fuhr, N., Großjohann, K.: XIRQL: A Query Language for Information Retrieval. In: Proc. of the ACM-SIGIR Conference, pp. 172–180 (2001)

[7] Grabs, T., Schek, H.-J.: Generating Vector Spaces On-the-fly for Flexible XML Retrieval. In: Proc. of ACM SIGIR Workshop on XML and Information Retrieval, pp. 4–13 (2002)

[8] Guo, L., et al.: XRANK: ranked keyword search over XML documents. ACM SIGMOD, 16–27 (2003)

[9] Hammiche, S., et al.: Semantic Retrieval of Multimedia Data. In: ACM MMDB Workshop, pp. 36–44 (2004)

[10] Jones, C., Purves, R.: Geographic Information Retrieval. J. of Geo. Info. Science 22(3), 219–228 (2008)

[11] Larson, R.: Geographic Information Retrieval and Spatial Browsing. In: GIS and Libraries: Patrons Maps and Spatial Information, pp. 81–124 (1996)

[12] Lin, D.: An Information-Theoretic Definition of Similarity. In: Proc. of the ICML Conference, pp. 296–304 (1998)

[13] Maguitman, A., et al.: Algorithmic Detection of Semantic Similarity. In: WWW Conference, pp. 107–116 (2005)

[14] Marian, A., et al.: Adaptive Processing of Top-k Queries in XML. In: ICDE Conference, pp. 162–173 (2005)

[15] Miller, G.: WordNet: An On-Line Lexical Database. International Journal of Lexicography 3(4) (1990)

[16] Motro, A.: Vague: A User Interface to Relational Databases that Permits Vague Queries. ACM Transactions on Office Information Systems 6(3), 187–214 (1988)

[17] Nierman, A., Jagadish, H.V.: Evaluating structural similarity in XML documents. In: Proc. of the ACM WebDB Workshop, pp. 61–66 (2002)

[18] Open Geospatial Consortium. Geography Mark-up Language, `http://www.opengeospatial.org/standards/gml`

[19] Paul, M., Gosh, S.K.: An Approach for Geospatial Data Management for Efficient Web Retrieval. In: Proc. of the 6th IEEE International Conference on Computer and Information Technology (2006)

[20] Pokorny, J., Rejlek, V.: Databases and Info. Systems, Frontiers in Artificial Intelligence and Applications. In: Barzdins, J., Caplinskas, A. (eds.) A Matrix Model for XML Data, pp. 53–64. IOS Press, Amsterdam (2005)

[21] Rodriguez, M.A., Egenhofer, M.J.: Comparing Geospatial Entity Classes: an Asymmetric and Content-Dependent Similarity Measure. Journal of Geographical Information Science 18(3), 229–256 (2004)

[22] Salton, G.: The SMART Retrieval System. Prentice Hall, New Jersey (1971)

[23] Schlieder, T.: Similarity Search in XML Data Using Cost-based Query Transformations. In: Proc. of the International ACM WebDB Workshop, pp. 19–24 (2001)

[24] Schlieder, T., Meuss, H.: Querying and Ranking XML Documents. Journal of the American Society for Information Science, Special Topic XML/IR 53(6), 489–503 (2002)

[25] Tekli, J., Chbeir, R., Yetongnon, K.: Extensible User-based Grammar Matching. In: ER Conf., pp. 294–314 (2009)

[26] Tekli, J., Chbeir, R., Yetongnon, K.: Efficient XML Structural Similarity Detection using Sub-tree Commonalities. In: Brazilian Symposium on Databases (SBBD) and SIGMOD DiSC, pp. 116–130 (2007)

[27] Torres, M., et al.: Retrieving Geospatial Information into a Web-Mapping Application using Geospatial Ontologies. In: Nguyen, N.T., Grzech, A., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2007. LNCS (LNAI), vol. 4496, pp. 267–277. Springer, Heidelberg (2007)

[28] World Wide Web Consortium. The Document Object Model (DOM) (May 2009), `http://www.w3.org/DOM`

[29] Wu, Z., Palmer, M.: Verb Semantics and Lexical Selection. In: Proc. of the 32nd Annual Meeting of the Associations of Computational Linguistics, pp. 133–138 (1994)

[30] Zhang, Z., Li, R., Cao, S., Zhu, Y.: Similarity Metric in XML Documents. In: Knowledge Management and Experience Management Workshop (2003)

[31] Zhu, F., Guan, J., Zhou, J., Zhou, S.: Storing and Querying GML in Object-Relational Databases. In: Proc. of the 14th Annual ACM Inter. Symp. on Advances in Geographic Information Systems, pp. 107–114 (2006)