8th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
S u c e a v a, R o m a n i a, M a y 25 – 27, 2 0 0 6

TOWARDS INTEGRATING DECISION TREE WITH XML TECHNOLOGIES

Diana GOREA[1], Sabin Corneliu BURAGA[2]
"Al. I. Cuza" University of Iasi
str.G-ral Berthelot nr.16, RO-700483 Iasi
[1]dgorea@infoiasi.ro, [2]busaco@infoiasi.ro

Abstract. *The paper proposes a method for efficiently store collections of multi-purpose decision trees within a native distributed XML database. The predictive information for building the XML decision trees is gathered through Web mining techniques and methodologies. In order to share data from heterogeneous sources, the model employs semantic Web languages to describe and represent data sources. The use of a native XML database system provides robust storage and manipulation capabilities of XML decision trees according to a logical model mapping. The classification of real data can be obtained by issuing queries over XML decision trees, using specific XML-based query processing capabilities.*
Keywords: *decision tree, XML, distributed XML database system, semantics.*

Introduction

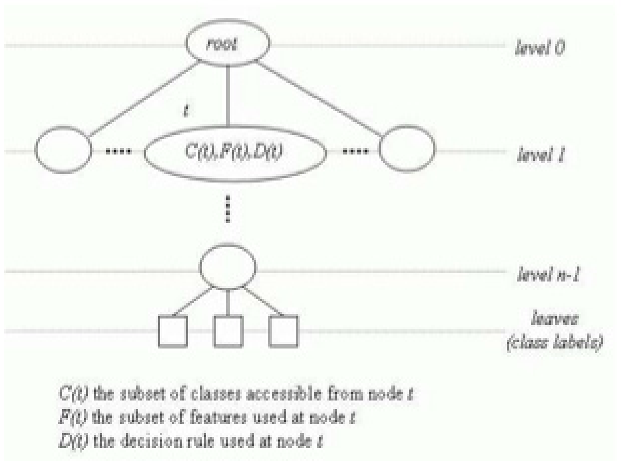
Decision and regression trees are the hierarchical approach to decision support making methods and are used successfully in many various areas like medical diagnosis, agent learning, risk assessment, radar signal classification, commercial and banking applications, strategy games, policy assessment, expert systems and speech recognition, to name only a few.
Usually, the decision trees are built to support one or more decisions. In the latter case, we may also consider a continuous exploitation of a decision tree by various beneficiaries. As a consequence, in the context of a distributed system, we may consider a frequently revised distributed repository of multipurpose decision trees.
In the context of World-Wide Web space, a decisional system can function as a group of Web services in order to be invoked by other Web applications. In this case, an XML-based approach in storing decision trees could be more flexible and useful than classical representations (the XML format of decision trees can be viewed as a serialization mechanism of information or knowledge exchanged by decision-making components, e.g. Web agents or services).
More interesting approach is to use the decision trees into semantic Web applications. In this case, a distributed native XML-based decisional system can play an important role, because it can offer semantic Web services for making decisions within complex Web applications, such as multi-agent systems or Grids.
The decision rules incorporated by decision trees offer a superior layer of the actual semantic Web layers (metadata, schema, and ontology layers) [11] and can be easily expressed by XML constructs.

After providing some details regarding the formal model of decision trees and their use in classification, in section 3 we'll propose an XML-based format for storing decision trees within a XML database system and the extensions of this format to incorporate various metadata and ontological assertions. Section 4 will present how we build XQuery assertions to make queries over XML decision trees.

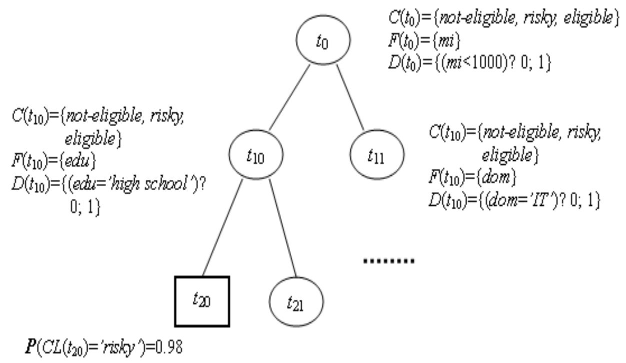
Decision Tree Classifier Model

First, we'll present some general information regarding the model of a decision tree classifier.
Let X be a q-dimensional vector called pattern whose components are called features or attributes. The instances (samples) of X are represented by attribute value pairs.
If the features of X are elements in a totally ordered set, X is called ordered or numerical pattern, otherwise it is called a categorical pattern. In the case of ordered pattern, the features may have continuous or discrete values.

Let *Y* be a discrete set of values, called class labels. In the simplest case – the Boolean classification – *Y* has exactly two elements.
Let *L* be the set of instances. *L* is partitioned into two subsets:
– L_1 is the subset of instances that are mapped into a corresponding class label and it is called training data. The training data may contain instances with missing values, when values for some attributes are unknown. The training data is used to build the decision tree.
– L_2 is the subset of instances for which the class label has to be found, which is known as test data.
A decision rule is a function that maps an instance into a subset of class labels. Decision rules are used at each internal node of a decision tree to split the instance according to the values of a selected subset of features. A decision tree is an ordered tree in which each internal node t is labelled with the sequence ($C(t)$, $F(t)$, $D(t)$) and each leaf is labelled with a class label. General structure of a decision tree is depicted in figure 1.


root
level 0
t
C(t),F(t),D(t)
level 1
level n-1
leaves
(class labels)
C(t) the subset of classes accessible from node t
F(t) the subset of features used at node t
D(t) the decision rule used at node t
Figure 1. General structure of a decision tree.

The decision tree learning is a method for approximating a discrete-valued target function. The misclassification rate of the target function is the likelihood of obtaining the wrong class label for an instance. If target function has values in a continuous set, the method is called regression tree learning. A decision tree can be seen as a set of sequences of decision rules. A sequence of decision rules corresponds to a path from the root to a leaf. The label of the leaf represents the output of the target function that takes a test instance as input. This way the outcome of a global complex decision (the target function) can be approximated by the union of simpler local decisions (the decision rules) made at each level of the tree.
For example, in figure 2 we have a fragment of a decision tree that can be used in a banking application to predict behaviour of a prospective client when according a loan.


t_0
$C(t_0)$={not-eligible, risky, eligible}
$F(t_0)$={mi}
$D(t_0)$={(mi<1000)? 0; 1}
$C(t_{10})$={not-eligible, risky, eligible}
$F(t_{10})$={edu}
$D(t_{10})$={(edu='high school')? 0; 1}
t_{10}
t_{11}
$C(t_{10})$={not-eligible, risky, eligible}
$F(t_{10})$={dom}
$D(t_{10})$={(dom='IT')? 0; 1}
....
t_{20}
t_{21}
$P(CL(t_{20})$='risky')=0.98
Figure 2. Example of a decision tree.

Construction Methods

Design of Optimal Decision Trees Given a set of labelled training instances L_1 – for which the classification is known – and a set of classes *Y*, the problem is to find a decision tree *T* that optimizes a cost function. The problem may have multiple solutions. The performance of a decision tree in further classifying the training instances strongly depends on how well the tree is designed.

Resulting directly from the above, a good design of a decision tree must assure the balance between the accuracy and efficiency measures.
Tree construction can be divided into following tasks: (1) choosing the appropriate tree structure, (2) choosing the feature subsets to be used at each internal node and (3) choosing the decision rule to be used at each internal node.
Designing an optimal decision tree means maximizing the mutual information gain at each level of the tree. The problem of designing an optimal decision tree is a NP-complete problem [18], motivating the necessity for finding efficient heuristics for constructing near optimal decision trees.

Tree construction According to [28], various heuristics for constructing a decision tree may be categorized in: the bottom-up approach [21], the top-down approach [29, 30, 33, 34], the hybrid approach [19], the growing-pruning approach [4, 25, 26], the entropy reduction approach [32]. Some of the methods separate the tree structure construction task from the feature or decision rule selection at each node, while others tend to combine these three tasks.

Some of the algorithms that assure the balance between the accuracy and efficiency measures were implemented by the majority of the decision support software and are briefly presented in the following paragraphs.

CHAID [18] and Exhaustive CHAID [3] are algorithms suitable for both classification and regression trees and are used especially for large datasets.

CART (classification and regression trees) incorporates the growing-pruning method suggested in [4] – it is considered computationally expensive, because in the pruning phase it generates a sequence of trees from which the one that minimizes the misclassification rate is chosen.

In [24] a growing-pruning algorithm, called ID3, is proposed, in which training data is initially placed in the root and then repeatedly split into partitions by the value of a selected feature at each node until no further splitting is possible or the class label is the same for all the instances in the current node. The order of attributes is selected by calculating the value of entropy function. The pruning process uses another dataset, called validation set, in order to improve the performance of the decision tree when it moves from the training data, where the classification is known, to real-world applications, where the classification has to be predicted.

C4.5 is a software extension to the basic ID3 algorithm proposed in [25] to address the following issues not dealt with by ID3: avoid over fitting of data by determining how deeply to grow a decision tree, handling attributes with missing or continuous values, handling attribute with different costs, choosing an attribute selection measure, and improving computational efficiency.

An improvement of C4.5 is C5 which provides more efficiency and a few additional facilities like variable misclassification costs, adding new data types, defining attributes as functions of other attributes, and giving support for sampling and cross-validation.

Regarding feature selection at a certain node, we distinguish two main approaches: an univariated test, when the decision rule at the node depends on a single feature (most of the presented methods), and multivariate test [24] – in this case, the decision rule depends on more features.

Incremental Restructuring When training data arrives in continuous or periodic flow, it is more reasonable to update the existing tree than to rebuild it from scratch. In [31] it is described how to update decision trees incrementally as more data is made available, by presented a method for mapping a decision tree and a new training data set into a new tree. Transforming one tree into another requires the ability to efficiently restructure a decision tree. To do this, a transposition operator is defined that transforms the tree into another one that also incorporates the new set of instances. After the transposition operation is accomplished, the node decision rules have to be rearranged according to the feature selection metric that was used initially to construct the tree.

Other approaches can be consulted in [10, 16, 22].

Storing Decision Trees in Distributed XML Databases

Most of available implementations for decision tree construction algorithms provide a model for serializing decision trees for storing or exporting to another application.

The XML (Extensible Markup Language) [2] meta-language is suitable for storing decision trees. Moreover, XML has become the universal standard for information interchange between applications; thereby, it can be also used for transferring decision trees between different applications (Web agents, Web services, etc.).

In the context of once-built multiple-used decision trees, it makes sense to efficiently store them in order to further classify data samples. Furthermore, if we consider decision trees that are constructed from Web data (eventually, already adnotated in a ontological manner) and serve more than one beneficiary, then it would be helpful for those consumers – human users, Web services, agents, etc. – to obtain on-line predictions for their input data samples.

Our proposal is to build a distributed XML database of decision trees, taking advantage of the robust storage and manipulation capabilities of the XML decision trees according to a logical model mapping offered by various native XML database management systems. Basically, an XML decision tree is obtained by converting the output of a decision tree construction tool into XML and stored according to a logical model.

First of all, we propose an XML-based format, in order to store a decision tree. Each tree has a list of features denoted by `<feature-list>` element and a list of classes expressed by `<class-list>` construct. To express the type of features, we can use the XML Schema [15] data types (e.g., `xs:integer`) and/or categories – such categories are ontological constructs declared by RDF/OWL assertions. To assure the model generality, an upper-level ontology, such as ABC [20], can be used.

Each `<node>` element of the XML document corresponds to a node in the decision tree and contains information about the subset of features that the decision rule at the node depends on (`<features>` element), the subset of classes accessible at the node (`<classes>` element) and the decision rule selected at the node, which can be modelled as one or more tests over selected features (for this, we use `<decision-rule>` construct). The leaf nodes are represented as `<node>` elements with a single sub-element corresponding to the predicted class label.

For instance, the XML decision tree can be used in a banking application to predict behaviour of a prospective client when according a loan. This approach was proposed in [17]. To keep the simplicity of presentation, we depict a binary tree, given that any ordered tree can be uniquely transformed into an equivalent binary tree [27].

Using the proposed XML constructs, such a document has the following form:

```
<decision-tree type="binary">
<feature-list>
<feature ID="mi" name="monthly-income"
type="continous"/>
<feature ID="dom" name="domain"
type="categorical" />
<feature ID="edu" name="education"
type="categorical" />
<!-- other features -->
</feature-list>
<class-list>
<class ID="c1" name="eligible" />
<class ID="c2" name="risky" />
<class ID="c3" name="not-eligible" />
</class-list>
<node leaf="no">
<features><feature IDREF="#mi"
/></features>
<classes>
<class IDREF="#c1" />
<class IDREF="#c2" />
<class IDREF="#c3" />
</classes>
<decision-rule
direction-if-true="left">
<test comparison="less-than-equal-to">
<range IDREF="#mi" />
<cutpoint value="1000" />
</test>
</decision-rule>
<sub-trees>
<node leaf="no">
<features>
<feature IDREF="#edu" /></features>
<classes>
<class IDREF="#c1" />
<class IDREF="#c2" />
<class IDREF="#c3" />
</classes>
<decision-rule
direction-if-true="left">
<test comparison="equal">
<range IDREF="#edu" />
<selected-value value="high school"
/></test>
</decision-rule>
<sub-trees>
<node leaf="yes">
<class-label IDREF="#c2"
probability="98" /></node>
<node leaf="no"><!-- continuing to decide
--></node>
</sub-trees>
</node>
<node leaf="no"><!-- right sub-tree
--></node>
</sub-trees>
</node>
</decision-tree>
```

Adding Metadata and Ontological Assertions

In the above example, we can enumerate the following main ontological categories: monthly income (its individuals have numerical values), domain and education (both of them have a categorical type), and loan conditions (the individuals of this ontological class have values “eligible”, ”risky” and ”not-eligible”).

Because XML format permits mixed vocabularies (different XML constructs), we can extend the proposed representation by providing metadata and ontological assertions. Using Metadata Each feature and decision rule may have attached various metadata via RDF (Resource Description Framework) [23] assertions. This can be very useful for decision trees that are used in the context of e-commerce

by comparison or recommender Web agents. For example, if one of the features denotes an e-shop, then we can use DCMI (Dublin Core Metadata Initiative) [35], RSS (RDF/Rich Site Summary) [11] or other kinds of metadata constructs to enrich the decision tree with relevant information that can further be used.

If the decision tree is used by a planning component of a multi-agent system or Grid, we can add different metadata regarding spatial and temporal relationships established between various software components of that application. For this, we can use XFiles [5] and TRSL (Temporal Relation Specification Language) metadata languages, following our previous research (please, consult [5, 6, 8, 9]).

Using ontological assertions – via RDF [22] or OWL [12] languages –, decision trees can be (automatically) transformed to be used in different contexts, according to user/application needs.

In this manner, a decision making Web-based tool can use various Web mining techniques and methodologies to build XML decision trees, which contains ontological categories of features, classes and decision rules. Because our proposed model implies the XML format, RDF/OWL-based documents can be used as predictive information by interrogating semantic Web indexing and retrieval engines, such as SWOOGLE [13].

To attach various metadata/ontological declarations to monthly income, we can use:

```
<rdf:Description rdf:about="#mi">
<!-- mountly income is expressed in USD
and has integer values -->
<currency>USD</currency>
<rdfs:range rdf:resource="xs:integer" />
</rdf:Description>
```

One important aspect is to enrich decision trees with ontological constructs in order to use them in the context of planning process within a multi-agent system or Grid. Because agents expose communication and collaborative characteristics, a suitable ontological model can be KAD (Knowledge-Argument-Decision) [14].

In our case, argumentative discourses are accomplished by agents themselves, nor by humans. At a syntax level, the ontological model is expressed by OWL declarations “embedded” into XML decision trees. Following [1, 7], agents can exchange – e.g., for planning purposes – XML-based information (in this case, knowledge) within the multi-agent system. A similar approach could be adopted for Grid applications.

Obtaining Predictive Information via XQuery

Using the proposed approach, obtaining a classification for a test instance is equivalent to issuing a query over a XML document that holds a decision tree. Queries are expressed in XQuery [36], a flexible query language for XML data.

The XQuery query takes as parameters a set of values corresponding to the features in the test data and retrieves the `<class-label>` sub-element of a `<node>` element corresponding to a leaf node in the decision tree. It performs a tree traversal in depth applying at each node the decision rule for the actual parameters and then following the branch that corresponds to the outcome. The process continues recursively until a leaf node is reached.

The features in a test data instance that represents the input for a classification query are also modelled as a XML element as follows:

```
<test-instance><feature
name="monthly-income" value="1500" />
<feature name="education" value="PhD" />
<feature name="domain" value="IT" />
<!-- other features --></test-instance>
```

An XQuery query that obtains a classification for a test instance will call the `recursive_traversal()` function having two parameters: `<node>` (the decision tree root node) and `<test-instance>`.
The `apply_decision_rule()` function returns a Boolean value after evaluating the decision rule at the current node.
This function depends on the decision problem that must be resolved (of course, using the metadata and ontological assertions included in tree, the function could imply different strategies in order to correctly evaluate and apply the decision rule).

```
declare function recursive_traversal
($node as element(), $test as element())
as element()* {
if ($node/@leaf = "yes") then return
$node/class-label
if (apply_decision_rule ($node, $test))
then recursive_traversal
($node/sub-trees/node[position() = 1])
else recursive_traversal
($node/sub-trees/node[position() = 2])
}
```

Related Work

There are a number of commercial decision tree software products available on the market. Among other models that aid in decision-making, they basically provide a the decision tree construction based on data stored in a database or spreadsheet and the decision tree analysis to obtain all possible outcomes and probabilities, to facilitate the selection of the best course of action when facing complexity and uncertainty.
Some of existing software decision tools provide export of decision trees in XML format, in order to integrate the results with other information management and decision support components. XQuery is supported only by various database engines or native database management systems and is not integrated into standard decision applications.

Our model differs from other existent proposals in that it not only proposes storage of generated XML decision trees, but also integrates XML technologies (such as querying XML documents or attaching metadata and ontological assertions) into decision tools, in order to offer support for semantic Web decisional applications.

Conclusions and Further Work

The paper presented an XML-based model for storing and maintaining multipurpose decision trees to be used in the context of semantic Web applications. A distributed native XML database system, such as Berkeley DB-XML or Mark's Content Information Server, can be adopted to efficiently store and retrieve such as trees by using XQuery constructs.

Our proposal is flexible enough to permit additional metadata or/and ontological assertions to be included into XML decision tree – augmenting features, classes or decision rules – in order to give support for reasoning.

This approach has advantage in the context of using Web mining techniques to retrieve, in an intelligent manner, predictive information (knowledge) for building collections of XML decision trees.

The metadata expressed in RDF graphs can be obtained with queries expressed in SPARQL query language [37]. New RDF graphs can be constructed as the result of querying existing information.

Our further focus is to refine proposed XML model by expressing decision rules via RuleML [11] or related languages, investigating a practical implementation in the context of predictive and planning activities, following the ideas presented in the paper.

References

[1] Alboaie, S., Buraga, S., Alboaie, L. (2004) *An XML-based Serialization of Information Exchanged by Software Agents*, International Informatica Journal, 28 (1), 13–18

[2] Bray, T. (ed.) (2004) *Extensible Markup Language (XML) – version 1.0 (Third Edition), W3C Recommendation*, Boston http://www.w3.org/TR/REC-xml

[3] Biggs, D., de Ville, B., Suen, E (1991) *A method of choosing multiway partitions for classification and decision trees*, Journal of Applied Statistics, 18 (1), 49–62

[4] Breiman, L. et al. (1984) *Classification and regression trees*, Belmont, Wadsworth

[5] Buraga, S. C. (2002) *A Model for Accessing Resources of the Distributed File Systems*, Lecture Notes in Computer Science – LNCS 2326, Springer-Verlag, 224–230

[6] Buraga, S. C., Alboaie, L. (2004) *A Metadata Level for the tuBiG Grid-aware Infrastructure*, Proceedings of SYNASC04 International Symposium, Mirton, 535–546

[7] Buraga, S. C., Alboaie, S., Alboaie, L. (2003) *The Use of XML Technologies for Exchanging Information within a Multi-Agent System*, International Scientific Journal of Computing, 2 (3)

[8] Buraga, S. C., Ciobanu, G. (2002) *A RDF-based Model for Expressing Spatio-Temporal Relations between Web Sites*, Proceedings of the 3rd International Conference on Web Information Systems Engineering, IEEE Computer Society Press

[9] Buraga, S. C., Gabureanu, P. (2003) *A Distributed Platform based on Web Services for Multimedia Resource Discovery*, Proceedings of the 2nd International Symposium on Parallel and Distributed Computing, IEEE Computer Society Press

[10] Crawford, S. L. (1989) *Extensions to the CART algorithm*, International Journal of Man-Machine Studies, 31, 197–217

[11] Daconta, M., Obrst, L., Smith, K. (2003) *The Semantic Web*, Wiley

[12] Dean, M., Schereiber, G. (eds.) (2003) *OWL Web Ontology Language Reference*, W3C Recommendation, Boston http://www.w3.org/TR/owl-ref/

[13] Ding, L. et al., *A Search and Metadata Engine for the Semantic Web*, Proceedings of 13th ACM Conference on Information and Knowledge Management, ACM Press, 2004

[14] Evangelou, C., Karacapilidis, N., Khaled, O. A. (2005), *Interweaving knowledge management, argumentation and decision making in a collaborative setting: the KAD ontology model*, International Journal of Knowledge and Learning, 1 (1-2), 130– 145

[15] Fallside, D. (ed.), *XML Schema Reference, W3C Recommendation*, Boston, 2001: http://www.w3.org/TR/xmlschema-1/

[16] Fisher, D. (1996), *Iterative optimization and simplification of hierarchical clusterings*, Journal of Artificial Intel ligence Research, 4, 147–178

[17] Gorea, D., Felea, V. (2005), *A Machine Learning Approach in Establishing Reliability of Customers in Banking Applications*, International Conference of Information in Economy, Bucharest, Roumanie,

[18] Hyafil, L., Rivest, R. L. (1976), *Constructing optimal binary decision trees is NP- complete*, Information Processing Letters, 5 (1), 15–17

[19] Kass, G. V. (1980), *An Exploratory Technique for Investigating Large Quantities of Categorical Data*, Journal of Applied Statistics, 29 (2), 119–127

[20] Lagoze, C., Hunter, J. (2001), *The ABC Ontology and Model*, Journal of Digital Information, 2 (2)

[21] Landeweerd, G. et al. (1983), *Binary tree versus single level tree classification of while blood cells*, Pattern Recognition, 16, 571–577

[22] Lovell, B. C., Bradley, A. P. (1996), *The multiscale classifer*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 18, 124–137

[23] Manola, F., Miller, E. (eds.), *RDF (Resource Description Framework) Primer*, *W3C*

Recommendation, Boston, 2004: http://www.w3.org/TR/rdf-primer/

[24] Murthy, S. K., Kasif, S., Salzberg, S. (1994) *A system for induction of oblique decision trees*, Journal of Artificial Intelligence Research, 2 –32

[25] Quinlan, J. R. (1986) *Induction of Decision Tree*, Machine Learning, 1, 81–106[26] Quinlan, J. R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann

[26] Rounds, E. (1980) *A combined non-parametric approach to feature selection and binary decision tree design*, Pattern Recognition, 12, 313–317

[27] Rounds, E. (1980) *A combined non-parametric approach to feature selection and binary decision tree design*, Pattern Recognition, 12, 313–317

[28] Safavian, S. R., Landgrebe, D. (1991) *Survey of Decision Tree Classifier Methodology*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 3, 660–674

[29] Sethi, I. K., Sarvarayudu, G. (1982) *Hierarchical classifier design using mutual information*, IEEE Transactions on Pattern Analysis and Machine Intel ligence, 4, 441–445

[30] Swain, P., Hauska, H. (1977) *The decision tree classifier design and potential*, IEEE Transactions on Geoscience Electronics, 15, 142–147

[31] Utgoff, P.E., Berkman, N.C., Clouse, J.A. (1997) *Decision Tree Induction Based on Efficient Tree Restructuring*, Machine Learning, 29, 5–44

[32] Wang, Q., Suen, C. (1987) *Large tree classifier with heuristic search and global training, IEEE Transactions on Pattern Analysis and Machine Intelligence, 9, 91–102*

[33] Wu, D., Landgrebe, D, Swain, P. (1975) *The decision tree approach to classification*, Technical Report RE-EE 75-17, School of Electroning Engineering, Purdue University, Lafayette

[34] You, K., Fu, K. (1976) *An approach to the design of a linear binary tree classifier*, Proceedings of the 3rd Symposium on Machine Processing of Remotely Sensed Data, Purdue University

[35] * * *, *Dublin Core Metadata Initiative* http://www.dublincore.org/

[36] Boag S., Chamberlin D., Fernandez M., Florescu D., Robie J, Simeon J. (2005) *XQuery Language - W3C Candidate Recommendation*, http://www.w3.org/TR/xquery

[37] Prud'hommeaux, E., Seaborne, A. (2006) *SPARQL Query Language for RDF - W3C Working Draft 20 February 2006*, http://www.w3.org/TR/rdf-sparql-query/