Computing and Informatics, Vol. 33, 2014, 1065–1094

XML AND FUZZY-BASED TWO VARIOUS KNOWLEDGE RETRIEVAL METHODS IN EDAPHOLOGY

Anantaraman Meenakshi

Department of Computer Science and Engineering
K. L. N. College of Information Technology
Sivagangai District, Tamil Nadu, India
e-mail: ameenakshiphd@gmail.com

Vasudev Mohan

Department of Mathematics
Thiagarajar College of Engineering
Madurai, Tamil Nadu, India
e-mail: vmohan@tce.edu

Abstract. In this paper, we propose a proficient method for knowledge retrieval in edaphology to assist the edaphologists and those who are related with agriculture in a big way. The proposed method mainly consists of two phases of which the first one is to build the knowledge base using XML and the latter part deals with information retrieval using fuzzy search. Initially, the relational database is converted to XML database. This paper discusses two algorithms, one is when the soil characteristics are given as input to have the plant list and in the other, plant names are given as input to have the soil characteristics suited for the plant. While retrieving the query result, the crisp numerical values are converted to fuzzy value using the triangular fuzzy membership function and matched to those in database. Those which satisfy are added to the result list and subsequently, the frequency is found out to rank the result list so as to obtain the final sorted list. Performances metrics are used in order to evaluate the method and compared to baseline paper to identify the number of plants retrieved, ranking efficiency, and computation time and memory usage. Results obtained proved the validity of the method and the method obtained the average computation time of 0.102 seconds and average memory usage of 2 486 Kb, which are all far better than our previous method results.

Keywords: Knowledge management, XML, knowledge retrieval, soil, edaphology, fuzzy search

1 INTRODUCTION

Today, access to information through Web data plays a significant role. Although facing a quick growing flood of information on the World Wide Web, we observe a rising need for advanced tools that direct us to the kind of information that we are looking for [1] Retrieval results of main search engines are increasing every day. Mostly, general terms searches frequently wind up with over one million results. Generally, the keyword-matching mechanisms are used in IR techniques. If one topic has different syntactic representations, the information mismatching problem may occur as in this case [2]. "Data mining" and "knowledge discovery" are the examples that are referred to the same topic [23]. "If data mining is used to search documents containing knowledge discovery", it may be missed by keyword-matching mechanism. Information overloading is the problem which occurs, when one phrase is having different semantic meanings. A common example is the query, "apple", which may mean apples, the fruit, or iMac computers. This search results may be mixed by much useless information [3, 4, 5]. If we know that a user needed information about "apple the fruit" but not "iMac computer", we can deliver more useful and meaningful information and thus the information needed by the user could be better captured. In order to satisfy user information needs in a better way, the current IR models need to be enhanced [6].

For supporting the future generations of the Web, the growth and evolution of the Web makes knowledge retrieval systems necessary, in particular, text mining, and knowledge based systems formulate the implementation of such systems in practice [7]. Knowledge Management (KM) is an intelligent process by which the raw data is gathered and is transformed into information elements. These information elements are then accumulated and organized into context-relevant structures [8, 22]. KM is intended to approve ongoing business success all the way through a formal, structured initiative to brighten the creation, distribution, or use of knowledge in an organization [9]. In information sciences to illustrate different levels of abstraction in human centered information processing, the data-information-knowledge-wisdom hierarchy is used. Data Retrieval Systems (DRS), such as database management systems, are well appropriate for the storage and retrieval of structured data [10]. Web search engines such as Information Retrieval Systems (IRS) are very helpful in searching the significant documents or web pages that include the information necessary for a user. In order to extract the useful knowledge, a user must read and analyze the relevant documents [11].

Significantly, the way in which the information on soil is acquired and managed and is changed by increasing the amount of numerical data combined with fast development of new information processing tools. Tree Analysis (TA) is a modeling technique that is being used increasingly. TA has numerous advantages that appear to suit well soil-landscape modeling applications [12]. Non-parametric is one of the most interesting features, which means that no assumption is made regarding variable distribution. It avoids variable transformation caused by bi-modal or skewed histograms, which are frequent in soil class signatures. The field of knowledge management is both innovative and highly volatile. Even as we were capable to find many accepted articles on knowledge management and some overviews, all deals with comparatively small subsets to the range of the work we establish, referred to as knowledge management [13]. Overviews of the current state and direction of knowledge management were unfortunately unable to find, therefore much of the effort was placed on understanding the status and direction of knowledge management development under the statement that knowledge-based systems will eventually need to be integrated into a larger knowledge management system [14].

1.1 Edaphology

Edaphology is about the influence of soils on living things, mainly plants. It also deals with the study of how soil influences man's use of land for plant growth as well as man's overall use of the land. Agricultural soil science is the general subfield within edaphology (known by the term agrology in some regions) and environmental soil science. (Pedology deals with pedogenesis, soil morphology, and soil classification). Soil science is the technical study of soil as a natural resource on the surface of the earth together with soil formation, classification and mapping; physical, chemical, biological, and fertility properties of soils; and these properties in relation to the use and management of soils. Sometimes terms such as pedology refer to branches of soil science (formation, chemistry, morphology and classification of soil) and edaphology (influence of soil on organisms, especially plants), are used as synonymous with soil science. The diversity of names associated with this discipline is related to the various associations concerned. In reality, engineers, agronomists, chemists, geologists, geographers, ecologists, biologists, microbiologists, sylviculturists, sanitarians, archaeologists, and specialists in regional planning, all contribute to further knowledge of soils and the development of the soil sciences. How to preserve soil and land in a world with a growing population, possible future water crisis, increasing per capita food consumption, and land degradation are the concerned factors raised by soil scientists.

1.2 Need for Knowledge Retrieval in Soil Database

As the plants demand varying quantities of diverse nutrients at different stages of growth, the preservation of fertility at the appropriate level in the soil and the selection of suitable vegetation type for the soil are especially vital for cropping. Therefore, in taking care of plants the knowledge of deficiency/excess of the nutrients in the soil is very significant. The large quantity of data and the multiple areas of expertise that are indispensable for soil exploration generate a massive volume of knowledge. This factor highlights the need for designing an efficient system to adjust, standardize, manage, retrieve and process soil information in order to attain improved productivity in agriculture.

The characteristics and the information about the soils collected by edaphologists are utilized to have input relational database. The input database has two tables of which one is plant description table which contains attributes that describe the plants and the other contains the soil characteristics, which includes the soil attributes. The tables are initially converted to XML database using plant identification number attribute in both the tables as the foreign key. The proposed method discusses two algorithms. One is to find the plants suited to the input soil characteristics and the other is to find the soil characteristics needed for the input plant name. Both the algorithm makes use of fuzzy search and ranking to have the results. In fuzzy search initially the numerical crisp values are converted to fuzzy values using the fuzzy triangular membership function and then compared with the database to have the results. After converting to fuzzy values, ranking process is done by finding the frequency in order to have the final result list in response to the query.

The main contributions of our proposed technique are:

- Conversion of relational database to XML so that information retrieval happens in a faster and easier way.
- Use of fuzzy search which adds to having a greater flexibility and having better query results.
- We discuss two algorithms of which in the first one, soil characteristics are inputted to have the plants satisfying the query and in the second one, plant name is inputted to have the soil characteristics best matched to the plant.
- We compute the performance metrics having the attributes: number of plants retrieved, ranking efficiency, computation time and memory usage in order to evaluate the method.
- We make a detailed study by comparing our proposed method to our previous method [16].

The rest of the paper is organized as follows: A brief review of researches related to the proposed technique is presented in Section 2. Section 3 describes proposed method for fuzzy-based knowledge retrieval in edaphology. The detailed experimental results and discussions are given in Section 4. The conclusions are summed up in Section 5.

2 REVIEW OF RELATED WORKS

Literature presents many works for information/knowledge storing and retrieval process of various application related database. Here, we review the literature based on the works available in knowledge management [10, 19], knowledge representation [15, 16] and the application of retrieval process in various domains, like soil analysis [16], petrographic analysis [21] and libraries [17]. The works presented in [10, 19] use concept map for knowledge management. Accordingly, Irfan et al. [10]

proposed a method that provided qualitative approach for enhancing the existing conceptual model for knowledge processing to do transformation. Modified knowledge management process transformed the heterogeneous data into a uniform format and was further integrated in expert warehouse concept. On the other hand, Tergan [19] has analyzed the impending of digital concept maps for supporting processes of individual knowledge management. The concept maps utilized had the potential to promote spatial learning strategies by visualization of the knowledge and support processes of individual knowledge management, for instance, the acquisition, organization, representation, (self-)evaluation, communication, localization, and utilization of knowledge. Moreover, they had the potential to represent and make accessible the conceptual and content knowledge of a domain, and information associated to it.

The works given in [15, 16] present the techniques for representing the information into different views on knowledge base. In accordance, Farenhorst and de Boer [15] described four main views on architectural knowledge based on the results of a systematic literature review. Based on software architecture and knowledge management theory, they defined four main categories of architectural knowledge, and discussed four distinct philosophies on managing architectural knowledge. Similarly, Velasquez and Palade [20] have designed a Knowledge Base (KB), which includes a database-type repository for maintaining the patterns, and rules, as an independent program that consults the pattern archive. In the architecture, an artificial system or a human user could consult the KB so as to improve the relation between the web site and its visitors. The architecture was tested with data from a Chilean virtual bank, which proved the efficiency of the approach.

In [16, 17, 21] unique applications, such as edaphology, petrographer system and academic libraries has been taken by the authors to retrieve the significant information from the database based on knowledge base. In edaphology, Meenakshi et al. [16] presented an efficient tree-based system for knowledge management. The system assisted edaphologists and an agricultural expert in obtaining the right crops/plants for the given soil characteristics. The characteristics and the information about the soils collected by edaphologists were utilized in the design of the presented system. The proposed system was composed of two phases, namely knowledge representation and knowledge retrieval. Firstly, a knowledge base was constructed by modeling the domain knowledge collected by edaphologists using the tree data structure. A novel algorithm was devised for effective knowledge retrieval from the modeled knowledge base, and subsequently, for the given soil characteristics, that provided with a set of plants/crops to be cultivated in that soil for better productivity from the constructed knowledge base.

To aid petrographic analysis and interpretation of oil reservoir rocks, Abel et al. [21] have presented the petrographer system, an intelligent data base application, and also data management by making use of resources both from knowledge system technology and database technology. The petrographer system developed was a structure closely coupled with a relational database system, which acts as a warehouse for the knowledge base and the user data, and an object oriented component, which preferably conserves the semantics of data and creates inferences. Incapable of improving reference services in academic libraries, Ralph and Ellis [17] investigated the use of the knowledge base of question point as a knowledge management tool. It would benefit librarians therefore, if they use a knowledge management tool that could capture and store their communal knowledge for future use. This study has explored the librarians' perceptions of the benefits and problems of using the knowledge base of question point and its impact on reducing response time and duplication.

3 PROPOSED METHOD FOR FUZZY-BASED KNOWLEDGE RETRIEVAL IN EDAPHOLOGY

In this section, we discuss the proposed efficient technique for knowledge management in edaphology by making use of XML and fuzzy search logic. These two features constitute to building a proficient system which gives edaphologists a solid edge when it comes to storing and retrieving informational knowledge in the concerned domain which ultimately results in having an increased productivity from the agricultural lands. This is the fact that right crop for the right soil can serve the best results. The soil is characterized by many parameters including the mineral and chemical compound content in the soil. For having the optimum outcome from the agriculture lands, the soil characteristics and the depth play a major role. In order to model and develop the relational database we make use of soil characteristics collected by edaphologists. The proposed technique mainly consists of two sections of which the first one is to build the knowledge base using XML and the latter part deals with information retrieval by searching using fuzzy logic. Figure 1 shows the block diagram of the proposed method. The proposed technique consists of two sections:

- creation of XML database
- information retrieval by searching using fuzzy logic.

3.1 Creation of XML Database

The primary step of the knowledge management system is to develop and model the domain knowledge or information collected from edaphologists. The optimal modeling of the information is of paramount importance as the system performance based on the effective management and retrieval of information directly depends on it. In general, proficient data structures like K-graphs [15, 18] are chosen for knowledge modeling. In [18], we make use of the tree data structure for knowledge representation which is almost like the K-graph and can be defined as an acyclic connected graph with one parent node and each node having a set of zero or more children nodes. In our proposed technique, we are improving it and use XML which

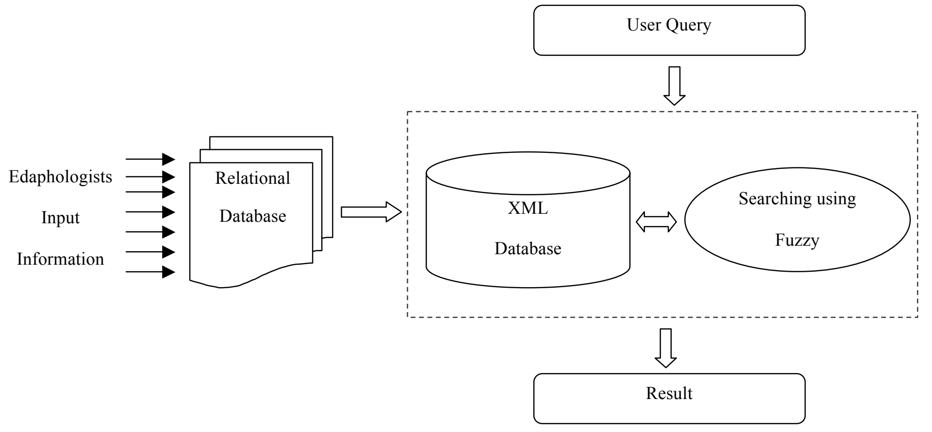

Figure 1. Block diagram of the proposed technique

ends up in attaining better results. For the purpose, we convert relational database into XML.

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and also machine-readable. XML is widely used for the representation of arbitrary data structures. The main advantage of using the XML is the flexibility, accessibility and portability it offers. The most beneficial matter in using XML is the improved speed and performance when compared to tree structure. Also the use of XML reduces the time incurred information retrieval.

Id	name	Geology	Taxonomy
0001	Prosophis juliflora, Cyprus sp., Hariyali,	Clay	Fine, montmorillonitic, isohyperthermic, noncalcareous, Chromic Haplusterts
0003	Palmyrah	Granite	Fine, mixed, isohyperthermic, noncalcareous, Typic Rhodustalfs
0017	Eucalyptus, Palmyrah, Neem, Tamarind	Laterite	Fine, mixed, isohyperthermic, noncalcareous, Typic Haplustepts
0019	Palmyrah, Neem	Granite	Clayey, mixed, isohyperthermic, noncalcareous, Lithic Haplustepts
0021	Palmyrah, Prosophis juliflora	Granite	Fine, mixed, isohyperthermic, noncalcareous, Typic Haplustalfs
0023	Neem, Palmyrah, Prosophis juliflora, Ta	Sand	Loamy-over-sandy, mixed, isohyperthermic, noncalcareous, Typic Ustifluvents
0024	Neem, Prosophis, Tamarind	Granite	Sandy, mixed, isohyperthermic, calcareous, Typic Ustorthents
0032	Palmyrah	Sand	Sandy, mixed, isohyperthermic, noncalcareous, Aquic Ustipsamments
0035	Neem, Palmyrah	Granite	Fine, mixed, isohyperthermic, calcareous, Calcic Haplustepts
0037	Neem, Prosophis juliflora	Western Ghats	Fine, mixed, isohyperthermic, noncalcareous, Typic Haplustepts
0039	Neem, Palmyrah, Tamarind	Granite	Fine, mixed, isohyperthermic, noncalcareous, Typic Haplustepts
0041	Palmyrah, Neem, Accacia	Granite	Clayey-skeletal, mixed, isohyperthermic, noncalcareous, Typic Haplustepts
0042	Ipomea, Thespesia populanea, Vagai	Clay	Fine, mixed, isohyperthermic, noncalcareous, Typic Rhodustalfs
0045	Palmyrah, Neem, Prosophis juliflora	Granite	Loamy, mixed, isohyperthermic, noncalcareous, Lithic Ustorthents
0047	Eucalyptus, Vagai	Laterite	Fine, mixed, isohyperthermic, noncalcareous, Fluventic Haplustepts
0050	Prosophis juliflora, Neem, Vetiver	Granite	Clayey-skeletal, mixed, isohyperthermic, noncalcareous, Lithic Ustorthents
0051	Prosophis juliflora, Palmyrah	Granite	Fine-loamy, mixed, isohyperthermic, noncalcareous, Fluventic Haplustepts
0055	Prosophis juliflora, Palmyrah, Tamarind	Granite	Fine-loamy, mixed, isohyperthermic, noncalcareous, Typic Haplustepts

Figure 2. Example of the Plant table

Id	Depth	Desc1	Clay	Silt	Sand	PH	EC	Ca	Mg	Na	K	P2O5	K2O
0001	0-13	Dark brown (10 YR 4/3); sandy clay; m	38.46	15.84	45.70	8.30	1.30	11.80	4.10	3.59	0.40	10.00	120.00
0001	13-65	Dark grayish brown (10 YR 4/2); sandy	41.92	18.07	40.01	8.90	1.50	13.87	4.30	4.07	0.46	10.00	80.00
0001	65-184	Very dark grayish brown (10 YR 3/2); c	40.84	21.90	37.26	8.90	1.40	14.20	3.90	4.09	1.02	0	0
0003	0-18	Red (2.5 YR 4/6); sandy clay loam; mo	24.60	15.2	60.2	7.36	0.04	8.0	1.5	0.68	0.32	29.45	228.00
0003	18-42	Red (2.5 YR 4/6); sandy clay loam; mo	25.60	16.8	57.60	7.11	0.04	10.0	2.0	0.73	0.35	27.37	376.00
0003	42-80	Dark yellowish brown (10 YR 4/6); cla	38.14	24.2	37.66	6.85	0.09	15.5	1.5	2.59	6.70	29.41	415.00
0003	80-140	Dark yellowish brown (10 YR 4/6); cla	44.60	25.6	29.80	6.81	0.08	18.5	1.5	4.25	4.62	15.60	425.00
0017	0-10	Dark yellowish brown (10 YR 4/4); san	38.24	13.64	48.12	6.61	0.03	5.50	8.50	0.46	0.33	16.58	69.00
0017	10-29	Brownish yellow (10 YR 6/6); sandy cl	36.92	14.06	49.02	7.02	0.03	5.50	5.00	0.51	0.25	9.50	42.00
0017	29-57	Brownish yellow (10 YR 6/6); clay; mo	43.82	20.90	35.28	7.12	0.02	6.00	4.00	0.46	0.25	10.68	46.00
0017	57-87	Yellowish brown (10 YR 5/4); clay; mo	46.16	19.56	34.28	7.18	0.03	6.50	4.00	0.47	0.25	12.67	53.00
0017	87-1000	Weathered parent	0	0	0	0	0	0	0	0	0	0	0
0019	0-12	Very dark grayish brown (10 YR 3/2); c	44.60	20.24	35.16	8.19	0.07	31.00	6.50	0.36	0.28	20.78	122.00
0019	12-25	Very dark grayish brown (10 YR 3/2); g	48.40	21.26	30.34	8.33	0.05	32.00	8.00	0.38	0.08	19.82	141.00
0019	25-39	Dark brown (10 YR 4/3); gravelly clay;	50.00	24.24	25.76	7.76	0.07	33.50	8.00	0.46	0.05	22.55	69.00
0019	39-1000	Weathered Granitic	0	0	0	0	0	0	0	0	0	0	0
0021	0-18	Reddish brown (5 YR 4/4); sandy clay	18.60	13.00	68.40	7.12	0.06	5.50	2.50	4.27	1.12	3.72	103.00
0021	18-37	Reddish brown (5 YR 4/4); clay; mode	31.14	18.80	50.06	6.77	0.02	10.00	8.00	4.25	1.09	6.77	100.00
0021	37-78	Dark reddish brown (5 YR 3/4); clay; m	34.60	19.25	46.15	6.72	0.02	10.50	12.00	3.75	1.17	4.67	108.00
0023	0-17	Brown (10 YR 5/3) clay loam; weak, fi	35.00	22.50	42.50	8.00	0.85	14.50	3.26	3.34	0.58	2.80	174.00
0023	17-39	Pale brown (10 YR 6/3) clay loam; wea	38.00	19.50	42.50	8.30	0.86	15.81	3.51	3.52	0.62	3.50	158.00
0023	39-42	Yellowish brown (10 YR 5/4) sand; we	4.85	7.65	87.50	6.90	0.10	2.15	0.86	0.45	0.54	4.00	56.00
0023	42-96	Yellowish brown (10 YR 5/4) sand; we	13.70	5.00	81.30	7.00	0.05	4.59	1.53	0.35	0.20	2.00	35.00

Figure 3. Example of the Soil Characteristics table

Initially, the knowledge is stored in the relational database with the inputs from edaphologists. Here, it comprises of two tables of which the first one contains the plant details and the other the soil description. The plant details table consists of plant names, geology and taxonomy corresponding to the plant ID. Figure 2 shows an example of plant table having the attributes plant identification number I, name *Na*, geology *Ge* and taxonomy *Ta*. We can see that a plant can have multiple plant IDs and the geology and taxonomy vary accordingly. The description table contains the plant ID, depth and the description of the soil. It also has the values of various parameters like clay, silt, sand, pH, electrical conductivity, calcium, magnesium, sodium, potassium, phosphorus pent-oxide, potassium oxide. Here, we can see that the soil characteristics for the plant ID changes with the depth and because of that, each plant ID has more than one soil characteristics attached to it. Figure 3 gives an example of soil characteristics table S having attributes of plant identification number I, depth D, description G, clay *Cl*, silt *Sl*, sand *Sa*, hydrogen ion concentration H, electrical conductivity E, calcium *Ca*, magnesium M, sodium *Ns*, potassium *Pt*, phosphorous pent oxide *Ph* and potassium oxide *Po*.

The first process in the paper is to store the data from two tables in the XML format. For the same, we select plant ID I as the foreign key to join both the tables. Here, the data is converted to the XML format and then retrieved accordingly to the search query. During the conversion of the relational database to the XML structure, a tree like structure is built with the use of tags. Here, first the plant ID is taken and it acts like the parent tag. In each plant ID, complete details are added in pattern having the details from both the tables corresponding to the plant ID. First the attributes from the plant table are added to the XML. Here first the name, then geology and taxonomy are given tags and are added to the structure. Then soil descriptions are added to structure corresponding to the plant ID. A single plant may have more than one plant ID associated with it and also many soil characteristics attached to it as the soil characteristics vary with the depth. In each soil characteristics the depth, description, clay, silt, sand, pH and the chemical element contents are given. A separate description tag is created for each soil characteristics column in the characteristics table and a plant ID will have more than one of these description tags. After creating the complete structure for a plant ID, the structure for the next plant ID is made. Likewise for all the plant IDs in the table, the procedure is followed to get the final XML structure. In the XML every detail related to a single ID is stored first and after completing it, it will move to the other plant IDs. N is the total number of plant identification numbers in the tables.

For each I_j, where $0 < j \leqslant N$,
Find Na, Ge, Ta from P where $I = I_j$.
Store in XML
Find D, G, Cl, Sl, Sa, H, E, Ca, M, Ns, Pt, Ph and Po from S
where $I = I_j$
Store in XML

It can be noted that there will be only one row in the plant table corresponding to the plant ID whereas there will be many rows corresponding to the plant ID in the soil characteristics table with the depth as the soil characteristics required by the plant changes. Figure 4 shows the example of the XML structure for edaphology.

3.2 Information Retrieval Using Fuzzy Search

From the knowledge base which is stored in XML format, we need to extract information in the best possible manner in order to aid the edaphologists in the best way. For this extraction of knowledge, we make use of the fuzzy search by which we can retrieve the information in a more flexible manner compared to the conventional methods and also results in having less time incurred. The advantage with the fuzzy search is based on minimization of the marginal values and the flexibility which results in faster and better execution. The paper discusses two search scenarios, one with the soil characteristics for the input plant name and the other with the soil characteristics for the plant input. In both cases, we make the fuzzy search. Fuzzy search deals with having fuzzy description instead of crisp values and in here mostly description crisp values are converted into fuzzy sets based on certain parameters. The fuzzy sets count to three which proves ideal in easy searching and also in obtaining results with a faster timing which is of vital importance. The fuzzy sets are designed considering the highest and lowest values in the discrete crisp values and are based on the triangular fuzzy membership function. The retrieval of information is done accordingly from the XML based on the input query, be it the plant name or the soil description.

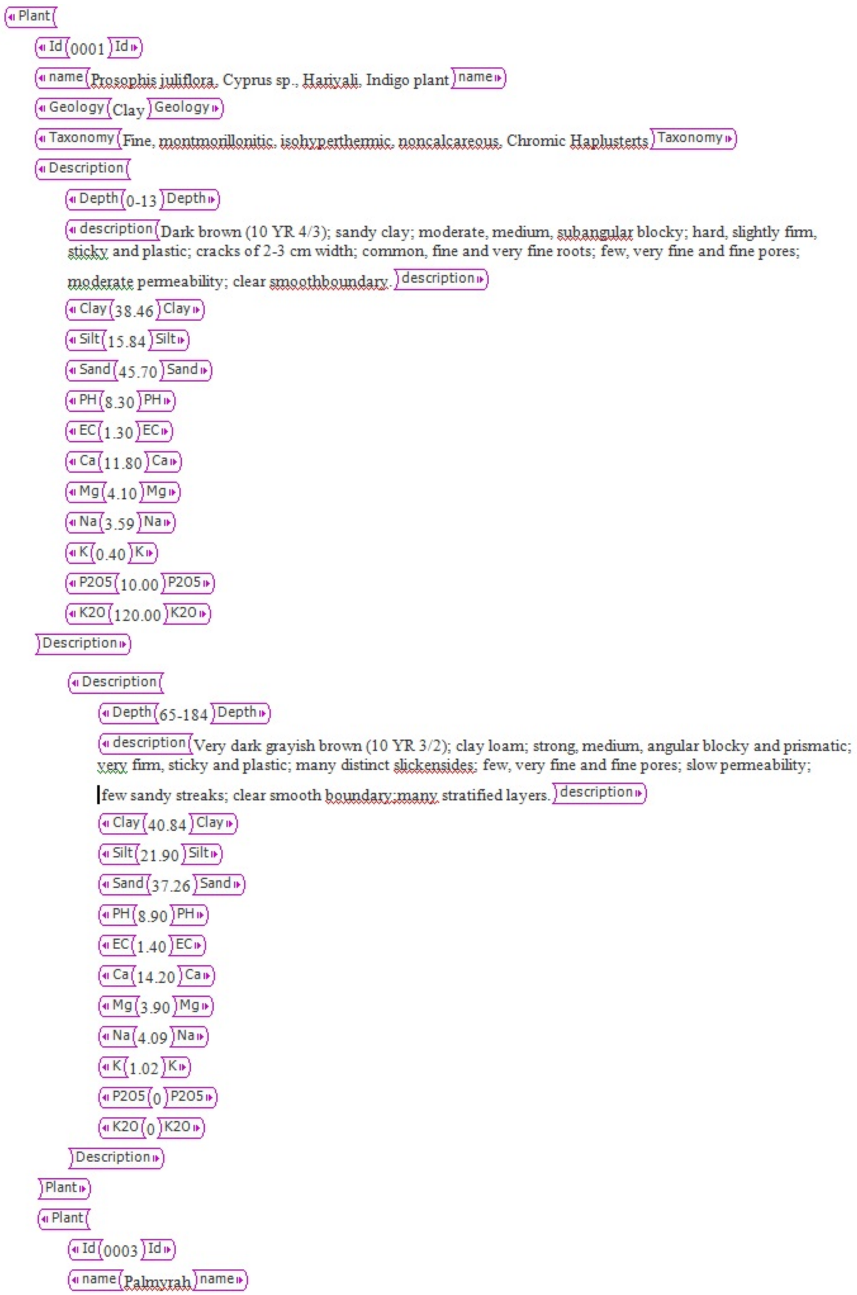

Figure 4. Example of the XML structure

Fuzzy search incorporates flexibility to the search which is important considering the edaphology domain. It is because a plant survives a range of values for the attributes rather than a precise single value. For example, a particular plant A is said to grow in nine meters depth with particular soil characteristics. When the query is given for the plant having the same soil characteristics but with a depth of eight meters, it will miss out on this plant A. But in reality, soil characteristics for a depth eight meters and soil characteristics for the same plant at nine meters will be similar and can be treated as one. Thus, incorporating fuzzy logic adds more flexibility to the search and matches with real life scenario. The information retrieval has three main steps:

- converting attributes to the fuzzy sets
- searching in the corresponding node and retrieval of plants
- ranking based on frequency.

The three steps are explained in a detailed manner in the next part. The results are taken from the ranked results to obtain the plant or the soil characteristics required. As discussed in the earlier part the search happens in two cases.

Case 1: Getting the plant based on the soil description (Algorithm 1). Getting the ideal plant for the available soil description is of vital importance as the plant grows and plant output directly depends on the soil characteristics. Having the right soil characteristics for the right plant will provide the best results and this can be made possible having the right answers to the search queries seeking the best plant that can be planted on the soil having the said attributes. One or more soil characteristics can be given as inputs to have the results having the list of plants suitable for the said conditions. As mentioned above, information retrieval to have the plant list based on the input soil characteristics is a three step procedure which includes a) converting attributes to fuzzy sets, b) searching the plants and getting the result list and c) ranking based on frequency. Figure 5 shows the block diagram of Algorithm 1.

a) **Converting attributes to the fuzzy sets.** First of all the crisp values of the input soil characteristic attributes are converted to the fuzzy set based on the value. Normally, the fuzzy sets are three in number where the first one-third will come in the first fuzzy set, the second one-third is in the second fuzzy set and the last one-third is in the last fuzzy set. Here the first fuzzy set is termed low, the second fuzzy set is termed medium and the last fuzzy set is termed high.

The method is improved having overlapping functions by having fuzzy triangular member in-order to improve flexibility. The depth, clay, silt, sand, pH, electrical conductivity, calcium, magnesium, sodium, potassium, phosphorus pent-oxide, and potassium oxide values (D, Cl, Sl, Sa, H, E, Ca, M, Ns, Pt, Ph, Po) have the crisp values that are converted to the fuzzy set. The other text inputs like name, geology, taxonomy and the description forms the text inputs (G, Na, Ge

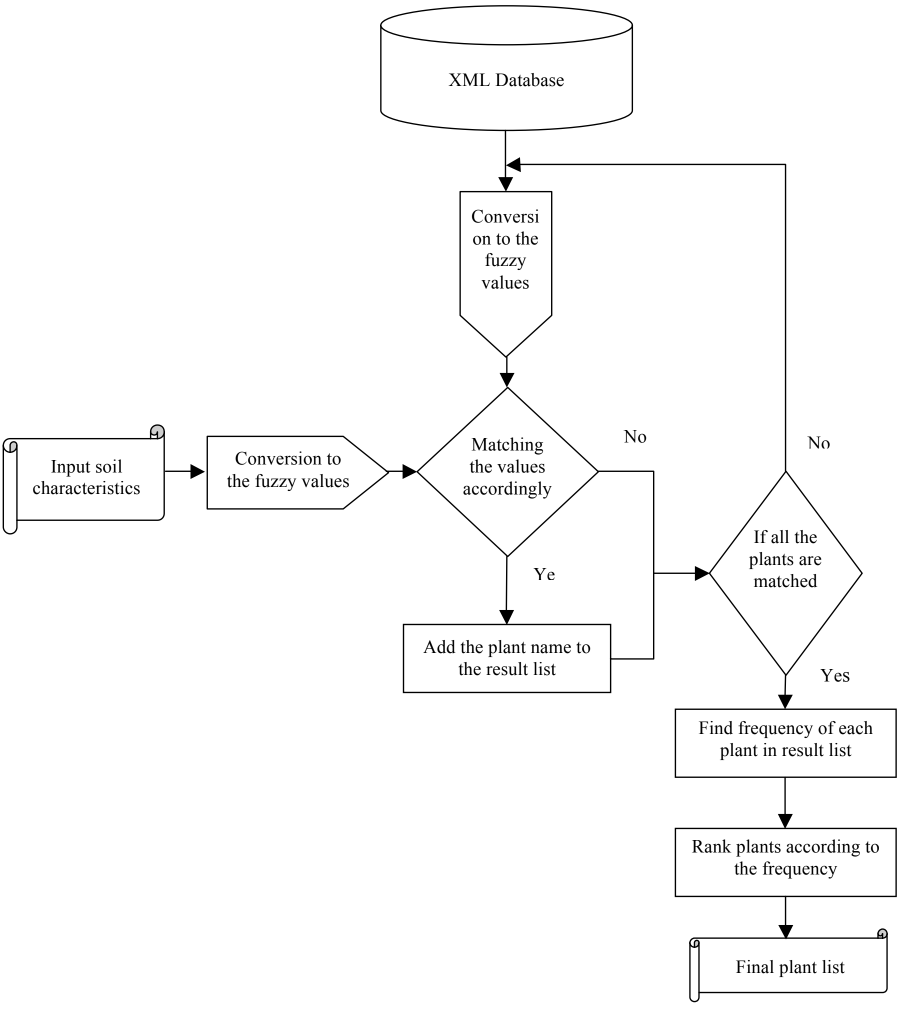

Figure 5. Block diagram of Algorithm 1 (Getting the plant list for the given soil conditions)

and *Ta*) are not changed and are compared in the text format during the search operation.

For each I_j, where $0 < j \leqslant N$,
For every attribute (*D*, *Cl*, *Sl*, *Sa*, *H*, *E*, *Ca*, *M*, *Ns*, *Pt*, *Ph* and *Po*) where $I = I_j$,
convert to fuzzy F_D, F_{Cl}, F_{Sl}, F_{Sa}, F_H, F_E, F_{Ca}, F_M, F_{Ns}, F_{Pt}, F_{Ph} and F_{Po}
For other attributes *G*, *Na*, *Ge* and *Ta*, No change
The conversion to the fuzzy is based on the fuzzy triangular membership values

Crisp Values	**Fuzzy Value**
Minimum − 33.33 % of Maximum	Low
33.33 % − 66.66 % of Maximum	Medium
66.66 % − Maximum	High

Table 1. The conversion to fuzzy values

discussed in the previous section. Here the conversion of the values is into three fuzzy sets HIGH, MEDIUM and LOW.
For each I_j , where $0 < j \leqslant N$,
For every element Ej where $E = \{D, Cl, Sl, Sa, H, E, Ca, M, Ns, Pt, Ph, Po\}$, convert to LOW, MEDIUM or HIGH fuzzy set where each of it is defined by the triangular membership function.

Fuzzy triangular membership function. The attributes having numerical values in the XML database are transformed into the fuzzy sets using the triangular membership function. Membership functions can either be chosen by the user arbitrarily or be designed using machine learning methods like artificial neural networks, genetic algorithms and others. There are different shapes of membership functions; triangular, trapezoidal, piecewise-linear, Gaussian, bell-shaped, etc. Here, we have chosen the triangular membership function in which a, b and c represent the x coordinates of the three vertices of a fuzzy set A (a: lower boundary and c: upper boundary where membership degree is zero, b: the centre where membership degree is 1). One of the key issues in all fuzzy sets is how to determine fuzzy membership functions,

- The membership function fully defines the fuzzy set.
- A membership function provides a measure of the degree of similarity of an element to a fuzzy set.
- Membership functions can take any form, but there are some common examples that appear in real applications.

The formula used to compute the membership values is depicted as below,

$$f(x) = \begin{cases} 0 & \text{if } x \leqslant a \\ \frac{x-a}{b-a} & \text{if } a \leqslant x \leqslant b \\ \frac{c-x}{c-b} & \text{if } b \leqslant x \leqslant c \\ 0 & \text{if } x \geqslant c \end{cases} \tag{1}$$

Figure 6 shows a triangular membership function for a single fuzzy set. Here, we can see that at 'a' and 'c' the value is zero and it reaches steadily to a maximum of value one at the centre point 'b' between 'a' and 'c'.

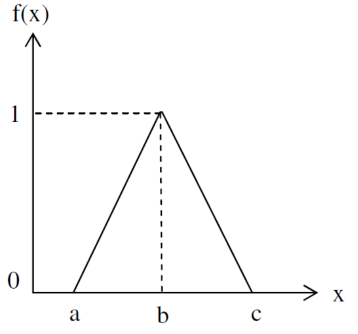

Figure 6. Triangular membership function

Figure 7 shows the plot considering all the three membership functions of having overlapping values. Here, the curves for low, medium and high are shown for the attribute, say depth.

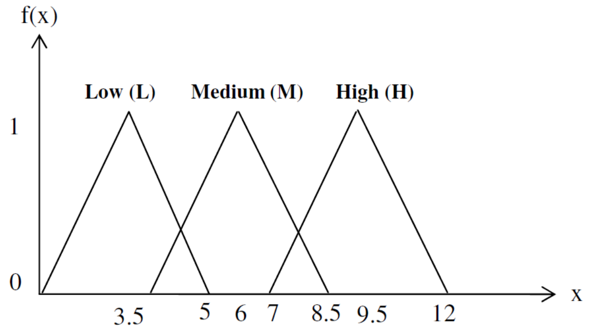

Figure 7. Triangular membership function with defined parameters and their values

By using the fuzzy membership formula, we have transformed the numerical attributes into the fuzzy sets.

b) **Searching in the corresponding node and retrieval of plant lists.** After converting to the fuzzy sets, the searching process happens where the information is retrieved according to the input query and the searching happens in the node of the XML corresponding to the input query attributes. For example, when a depth of eight meters is given as the input, first it is converted to fuzzy set and then all the plants that have the same fuzzy set are found out by searching in the depth node. For the searching, we compare using the string compare function comparing the input attribute fuzzy word to others in the database under the same root node. If a range is given instead of a single value as the word, it is too converted to the fuzzy set. The plants that satisfy the input condition are found out and listed. The searching happens inside the XML database with the use of fuzzy search where initially the values are converted to the fuzzy values. For a description of depth giving arbitrary value D_i, we have to convert it to fuzzy value and do the search in the database under the fuzzy values for the node depth.

For an input D_i, convert to Fuzzy F_{Di},
For each I_j, where $0 < j \leqslant N$,
Search in root node depth if $F_{Di} = F_D$, then select the corresponding Na,
Add Na to the result list R

For those having the same fuzzy depth values in the database, the corresponding plant names are added to the result list. The same process happens for all cases $\{D, Cl, Sl, Sa, H, E, Ca, M, Ns, Pt, Ph, Po\}$ where some soil characteristics is given as input X_i where the values are converted to the fuzzy values F_{xi} and compared with the fuzzy root nodes in the XML database $\{F_D, F_{Cl}, F_{Sl}, F_{Sa}, F_H, F_E, F_{Ca}, F_M, F_{Ns}, F_{Pt}, F_{Ph}, F_{Po}\}$. Those which satisfy the conditions are noted and added to the result list R, where $R = \{Na_1, Na_2, \ldots, Na_k\}$, where k is the total number of results in the list which contains the names of the plant Na which satisfies the condition. When there are multiple input conditions, then names of the plants which satisfy all the input conditions are only added to the list.

For an input X_i and Y_i convert to Fuzzy F_{Xi} and F_{Yi}
For each I_j, where $0 < j \leqslant N$,
Search in root node depth if $F_{Xi} = F_x$ and $F_{Yi} = F_y$, then select the corresponding Na
Add Na to the result list R.
X_i and Y_i are the input conditions, F_x and F_y are the fuzzy values from the database corresponding to the X and Y nodes.

c) **Ranking based on the frequency and fuzzy value.** After the search, we get the plant list having the plant names which satisfy the conditions. In the list, plant names will appear in many places and will look random. In order to have a better understanding and also to know the best plant that is suitable for the given conditions we have to arrange it in the best possible way. For this purpose, we find out the number of times the plant appears in the list or rather the frequency of the plant in the list. The frequency of the plant directly gives the direct knowledge how well that plant can grow in the said conditions. Better the frequency, better the chance of the plant growing well under the conditions. Hence, we rank the plants based on the frequency of the plant and its fuzzy value to get the final list.

From the result list R, we have to find the most appropriate answers for the input conditions. For the purpose, the frequency of each plant in the list is the total number of results in the list.

For each Na_i in R, $0 < i \leqslant k$
If $Na_j = Na_i$, for $0 < j \leqslant k$
$C_i = C_i + 1$
Then, $S_i = \frac{1}{C_i} \sum_{j=1}^{C} F(C_j)$.

Here C_i is the frequency of the i^{th} name in the result list R and S_i is the final fuzzy score of the i^{th} plant name. After finding out the fuzzy score of each plant, the list is sorted accordingly so that the plant with maximum fuzzy score comes first. Let m be the number of unique plant names in the list.

For Na_i in R, $1 < i \leqslant m$
sort in descending order with respect to S_i.

For given input soil conditions, the plants in the top of the list will yield good results and this knowledge will prove beneficial for edaphologists. Hence the plants fit for the given conditions are obtained.

Case 2) Getting the soil characteristics based on the input plant name (Algorithm 2). For any given plant, it grows well to particular soil characteristics, so getting the right soil characteristics for the given plant is of highest importance. With the variation of the soil characteristics, the output growth of the plant varies drastically, so for any edaphologists it is great benefit to know the soil characteristics for the given plant. One or more soil conditions may be associated to the same plant, so it is necessary to find the best soil conditions that fit the plant.

Information retrieval to have the soil characteristics based on the input plant is a three step procedure which includes a) converting attributes to fuzzy values, b) retrieval of soil characteristics list and c) getting the best soil characteristics for the input plant. Figure 8 shows the block diagram of Algorithm 1.

a) **Converting attributes to the fuzzy values.** In this process we first transform the crisp numerical values $\{D, Cl, Sl, Sa, H, E, Ca, M, Ns, Pt, Ph, Po\}$ to fuzzy values $\{F_D, F_{Cl}, F_{Sl}, F_{Sa}, F_H, F_E, F_{Ca}, F_M, F_{Ns}, F_{Pt}, F_{Ph}, F_{Po}\}$ by means of triangular fuzzy membership function as in the other case. The values are changed to low, medium and high fuzzy sets. The text inputs *G*, *Na*, *Ge* and *Ta* are not changed and remain the same.

b) **Retrieval of soil characteristics list.** A plant will appear many times in the database and there will be more than one soil characteristics attached to it, so it

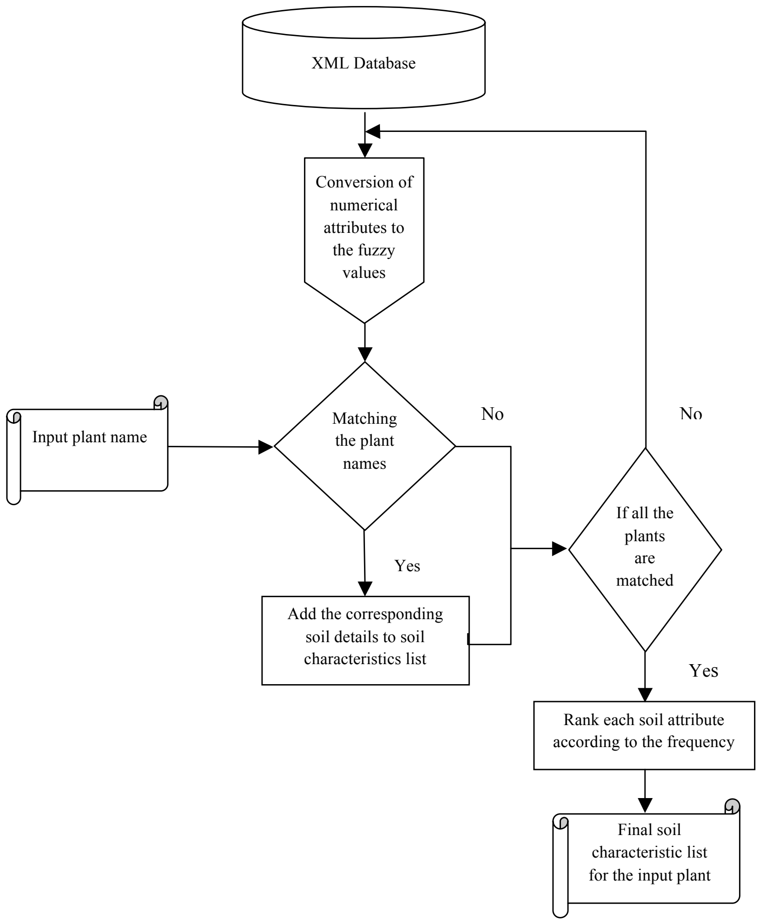

Figure 8. Block diagram of Algorithm 2 (getting the soil characteristics for the given plant)

is very important to get the best characteristics that match the input plant. For the same, we search for all the columns in the table linking to the input plant name and get all the attribute values from the list which will be in the fuzzy format. Here all soil characteristics corresponding to input plant are found out from the database, so that every attribute will have multiple answers as there will be more than one characteristics linked to the plant and it is absolutely necessary to find out the best characteristics for each attribute.

For every I_j, $0 < j \leqslant N$,
If $Na_j = Na$,
SELECT all description for Na_j from data base and add to result list R.

The result list R will have fields of I, F_D, F_{Cl}, F_{Sl}, F_{Sa}, F_H, F_E, F_{Ca}, F_M, F_{Ns}, F_{Pt}, F_{Ph}, F_{Po}, G, Na, Ge and Ta and each field will have more than one value.

c) **Getting the best soil characteristics for the input plant.** After the search, we get the list having the soil characteristics which match the input plant. Here, there will be more than one soil characteristics that match the concerned plant so it is necessary to find the best soil conditions that match the plant. In order to accomplish the task, we find out the number of times the particular soil characteristic appears in the list; thus, for every attribute we find the frequency of the characteristics and select the one having the highest frequency. The best soil characteristics will be the results for each attribute having the highest frequency thus, in order to find the most accurate value for the field we find the frequency of each field. Here g is the number of fields in the list. Considering an arbitrary field z_i, we find the frequency of the result values.

For each Z_i in R, $0 < i \leqslant g$
If $F_j = F_n$, for $0 < j \leqslant k$, $j <> n$
$C_i = C_i + 1$

Here C_i is the frequency of i^{th} fuzzy value in the result list R in the field Z_i. After finding out the frequency for each fuzzy value of the field associated with the plant input, the list is sorted accordingly so that in each field the fuzzy value that is most redundant with maximum frequency comes first. Let m be the number of unique fuzzy values for each field in the list.

For Z_i in R, $1 < i \leqslant m$
sort in Descending Order with respect to C_i.

Hence we get the soil characteristics that are most suited for the plant and will aid the edaphologists in the best way. The knowledge will end up in having maximum results from the plant.

4 RESULTS AND DISCUSSION

This section presents the results and discussion of our proposed method for knowledge retrieval in edaphology. Here, we evaluate both the algorithms used in the search operations where in the first one, plant list for the input conditions is found out and in the other one, the soil characteristics list for the input plant name is found out from the XML database. We also compare this paper to our baseline paper with the help of the performance metrics obtained in response to various user input queries. The obtained data are analyzed with the help of bar charts which prove the validity of our proposed technique.

4.1 Experimental Set Up and Dataset Description

The proposed technique is implemented in JAVA on a system having 4 GB RAM and 2.10 GHz Intel i-5 processor. Initially, the domain knowledge collected from edaphologists is modelled into a knowledge base, which acts as the input data set. The input database consists of two tables, of which one is the plant list table and the other the soil characteristic table. The two tables are linked by the foreign key plant identification number. There are 148 plant IDs in the database, in each plant table there are four attributes and in soil characteristics table there are 15 attributes. The plant table attributes are plant identification number, name, geology and taxonomy. The soil characteristics table attributes are plant identification number, depth, description, clay, silt, sand, hydrogen ion concentration, electrical conductivity, calcium, magnesium, sodium, potassium, phosphorous pent oxide and potassium oxide. The input database is stored in a file and later converted to XML database, from where the results are searched in reference to the user input query.

4.2 Performance Metrics

In order to find the performance and to evaluate our proposed method, we make use of certain parameters that constitute the performance metrics. Selection of performance metrics parameters is of high importance as it should give a clear-cut idea of how well the method works when compared to other existing technologies and also should be able to validate the effectiveness of the method. In this paper, we make use of four parameters that form the evaluation metrics.

Number of plants retrieved: The input to the method will be a user query which will have the soil characteristics and the output will be the plant list which will have the names of plants that satisfy the input user query. The parameter "number of plants retrieved" is the number of plants in the plant list. As the number of plants retrieved increases, the effectiveness of the plant retrieval method also increases.

Ranking efficiency: The plant list will have many plants that satisfy the input conditions which are subsequently ranked. Ranking is done so that the most appropriate plants for the input soil conditions come on top in the plant list; so the ranking procedure is of vital importance because the best fitting plants should come on the top. In our method, we rank based on the frequency count and fuzzy score. Similarly we perform the ranking for the soil characteristics list in response to the input plant name. Here the ranking is done for each individual attribute in the soil characteristic list to get the best fitting soil characteristics list for the input plant.

Computation time: Computation time refers to the time incurred between the input query and the output list. The input query may be soil characteristics or a plant name and the output will be the plant list or the soil characteristics list accordingly. Reduction of the computation time shows better and faster processing of the query. Our method had a great advantage in reducing the computation time as we are using the fuzzy search method.

Memory usage: The amount of memory used up while executing the query is known as the memory usage. Having a lesser memory usage will validate the effectiveness of the method.

4.3 Experimental Sample Results

In our method for knowledge retrieval in edaphology, we make use of two algorithms. In the first one, we input the soil characteristics to get the plant list that satisfies the input condition. For performance analysis, the experimentation has been performed with 50 queries, but the result has been provided here for six queries only. Sample input and corresponding output are given in Table 2. The table only shows the top 10 results of the total 44 plant names retrieved by the algorithm.

Input Query	**Output**
Description=Dark brown, Clay = 65.25, Silt = 20, Sand = 25, PH = 9.5, EC = 2, Ca = 5.1, Mg = 3.5, Na = 8.25, K = 6.7, P2O5 = 116, K2O = 340, depth = 18 − 35	Total process time is: (sec) 0.163, Total taken memory (kb): 2 489, 1. Prosophis, value = 0.8254 2. Bonassus, value = 0.7928 3. Wetland weeds, value = 0.6974 4. Grasses, value = 0.6165 5. Cassia, value = 0.6076 6. Jatropha, value = 0.5722 7. Acacia arabica, value = 0.5404 8. Accacia arabica, value = 0.5356 9. Palmyrah, value = 0.5164 10. Prosophis juliflora, value = 0.5124

Table 2. Sample table for Algorithm 1

Input Query	**Output**
Plant name: Prosophis	Total process time is: (sec) 0.094, Total taken memory (kb): 2 490, depth = 25 − 47 description = brown Silt = 57 Sand = 97.38 PH = 12.41 EC = 1.18 Ca = 49 Mg = 40.5 Na = 14.25 K = 7.7 P2O5 = 316 K2O = 745

Table 3. Sample table for Algorithm 2

In Algorithm 2, the plant name is given as query to obtain the soil characteristics list best fitting for the plant. The sample input plant name and corresponding output obtained is given in Table 3.

4.4 Performance Analysis of Algorithm 1 (Getting the Plant Based on the Soil Description)

In this section, we discuss the detailed analysis of Algorithm 1 where the soil characteristics are given as the user query and plant list that fit the query is the output. For the analysis, we test by having six different queries and we evaluate the algorithm using the performance metrics. The six queries used for the testing are given in Table 4.

In the analysis, we make use of the performance metrics parameters of the number of plants retrieved, computation time and memory usage. Tables 5 and 6 show the values obtained for different metrics attributes for different queries for the proposed method and the baseline method. Figures 9, 10 and 11 show the chart graph for number of plants retrieved, computation time and memory usage for various queries for the two methods.

Next we evaluate our proposed method by comparing the results of our method to the baseline paper with the help of evaluation metrics.

Query1	Query2	Query3
Description = Dark brown Clay = 65.25 Silt = 20 Sand = 25 PH = 9.5 EC = 2 Ca = 5.1 depth = 18–35	Depth = 25–55 Description = Yellow Clay = 5.25 Silt = 45 Sand = 63.8 PH = 10.41 EC = 2.18 Mg = 23.5	Depth = 24–45 Description = Red Clay = 54.5 Silt = 31 Sand = 63.38 PH = 5.81 EC = 1.18 Na = 10.25
Query4	**Query5**	**Query6**
Description = brown Clay = 68.25 Depth = 30–48 Sand = 97.38 EC = 1.16 Ca = 86 Mg = 40.5 Na = 16.25	Depth = 27–45 Clay = 67.27 Silt = 57 PH = 11 EC = 1.20 Ca = 49 Mg = 45.5 K = 10.7	Description = Dark blue Clay = 66.27 Silt = 65 Sand = 99.3 PH = 12.41 Ca = 56 Mg = 44.5

Table 4. Queries for Algorithm 1

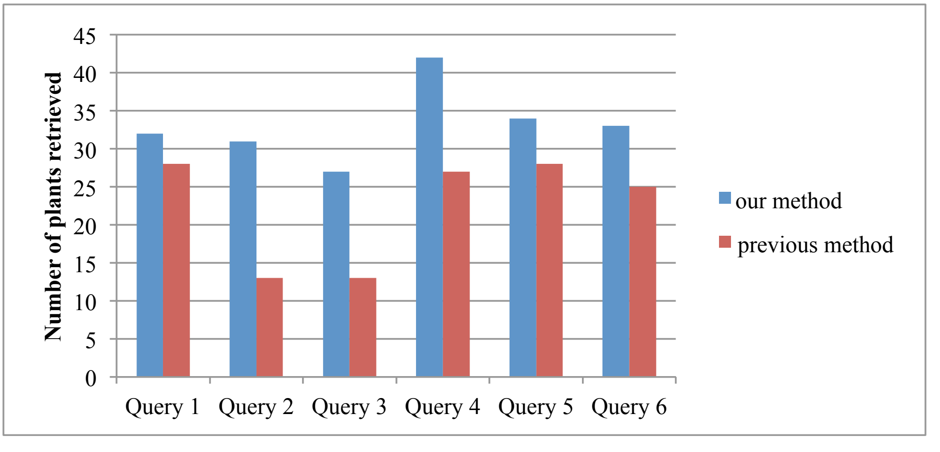

Figure 9. Chart showing the number of plants retrieved for various queries by the two methods

Performance Metrics	Query1	Query2	Query3	Query4	Query5	Query6
No of Plants Retrieved	32	31	27	42	34	33
Computation Time (s)	0.102	0.103	0.102	0.105	0.098	0.104
Memory Usage (Kb)	2 486	2 487	2 487	2 487	2 485	2 486

Table 5. Table showing performance metrics values for input query for our method

Performance Metrics	**Query1**	**Query2**	**Query3**	**Query4**	**Query5**	**Query6**
No of Plants Retrieved	28	13	13	27	28	25
Computation Time (s)	1.072	1.026	1.03	1.045	1.029	1.092
Memory Usage (Kb)	2 483	4 997	7 625	3 225	4 657	13 785

Table 6. Table showing performance metrics values for input query for our previous method

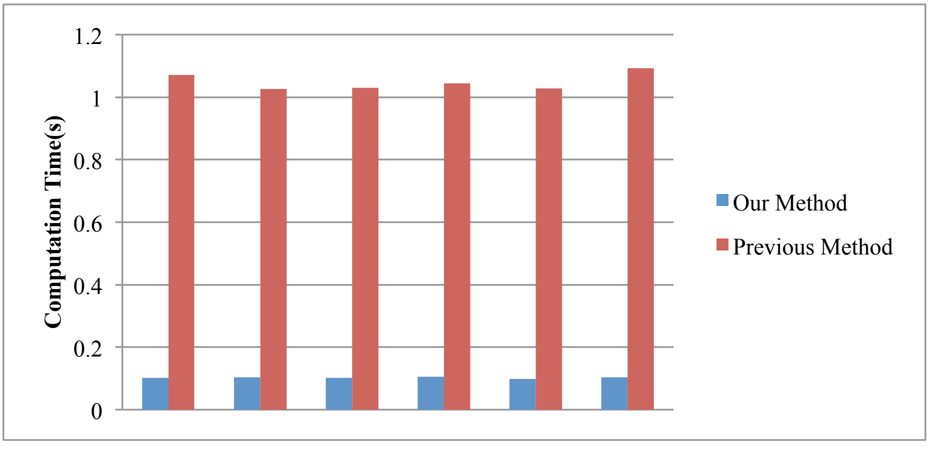

Figure 10. Chart showing the computation time for various queries by the two methods

Inferences from Figures 9, 10 and 11

- The figures plot the metrics values obtained for six different queries given as input. Here in all cases the plot is done for both the proposed method and the existing baseline method.
- Figure 9 shows the plot of the number of plants obtained by the two methods in response to the six queries given as input. From the plots, it is very clear that the proposed method achieves a better number of plants for all the input queries.
- From the analysis, we found that our method could retrieve a total of 199 plants for the six input queries when compared to 134 plants retrieved by the baseline method. The results show that our proposed method was able to retrieve 33 plants for a query on average when compared to 22 for the baseline method. The analysis proves higher efficiency of our method to retrieve plants for the input query.
- Figure 10 plots the computation time for two methods for all the six input queries. The computation time taken by our method is very low when compared to the baseline method.

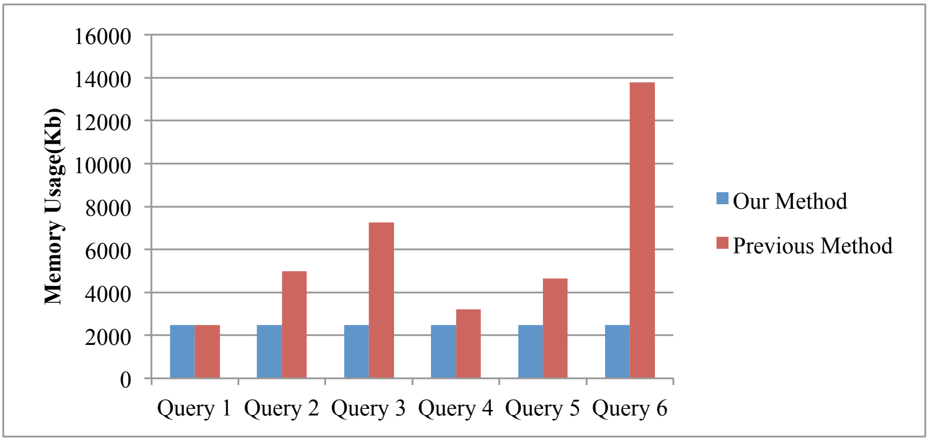

Figure 11. Chart showing memory usage for various queries by the two methods

- From the analysis made on the obtained time values, the total time for computation of six queries came to 0.663 seconds, taking an average of 0.102 second for one query. The time is far below when compared to the computation time for the baseline method which came to 6.29 seconds for six queries taking an average of 1.05 second for a query.
- Figure 11 plots the memory space utilized by the two methods. Our method proves efficient by taking lesser memory space when compared to the baseline method.
- The total memory utilized by our method came to 14 918 kB, taking an average of 2 486 kB per query whereas the total memory came to 36 412 in case of baseline method having an average of 6 068 kB per query.

Analysis using Ranking Efficiency. When a query having certain soil characteristics is given to the method, it outputs plants that satisfy the conditions. Both our previous and proposed methods yield a number of plants in response to the query; but our proposed method has an upper hand as we are ranking the results and finding out the best plants for the soil characteristics, and this can be shown from having the ranking efficiency analysis done for each query. In the graphs, top K results of our results are compared with all the results of the previous method for the same query. We find the results which are common to both and also those which are missing from the previous method output list. If any plant which is included in our plant list (considering top K results only) is missing from the output list, it clearly shows the efficiency of our method. This we find out using the intersection operator.

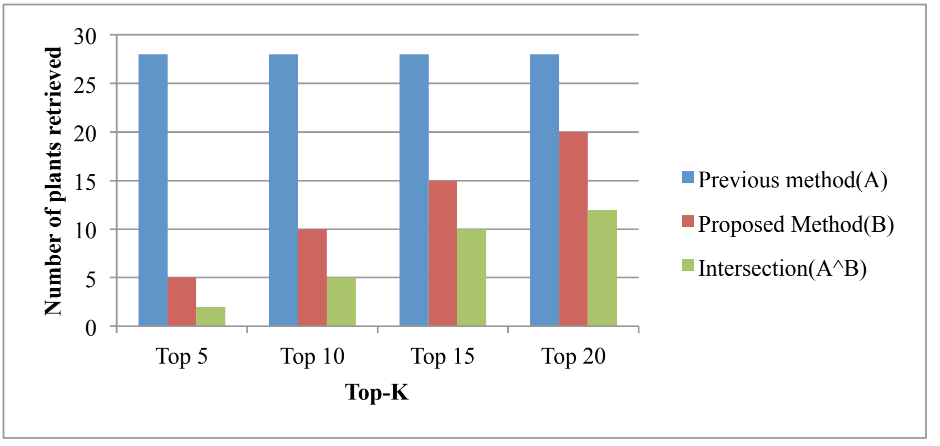

Figure 12. Chart showing the ranking efficiency for Query 1

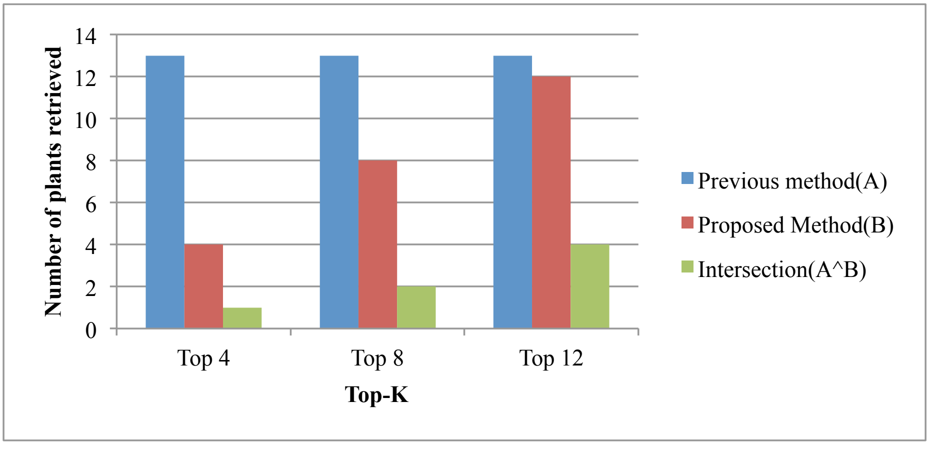

Figure 13. Chart showing the ranking efficiency for Query 2

Analysis of the figures 12–17.

- All the figures show the ranking efficiency in response to six queries. Each plot is of the number of plants retrieved by our previous method (A), our method (B) and also the number of plants that comes in common in the list $A\hat{}B$.
- For each query, we consider different top K results where the K takes values $\{5, 10, 15, 20\}$ for query 1, 4, 5 and 6. For queries 2 and 3, K takes values $\{4, 8, 12\}$.

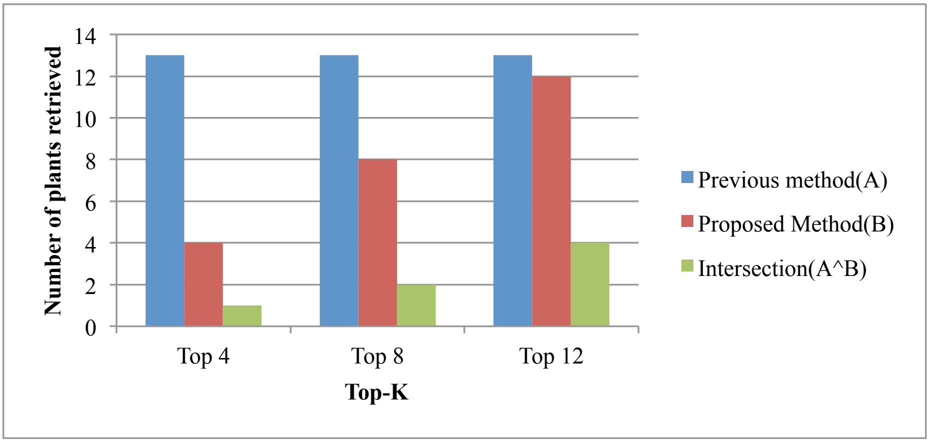

Figure 14. Chart showing the ranking efficiency for Query 3

- For every case, we can find values attained for $A \wedge B$ less than K. This means that our method was able to have the results which the previous method was not and this directly shows the efficiency of our method.

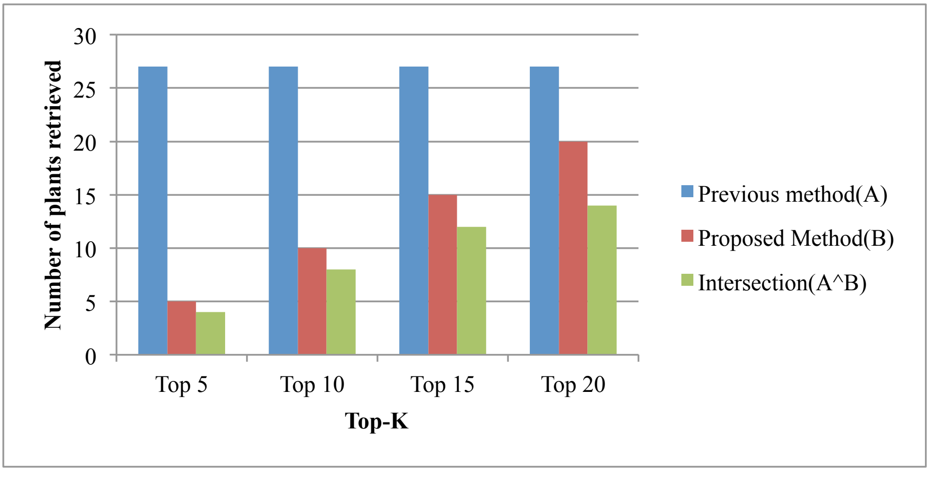

Figure 15. Chart showing the ranking efficiency for Query 4

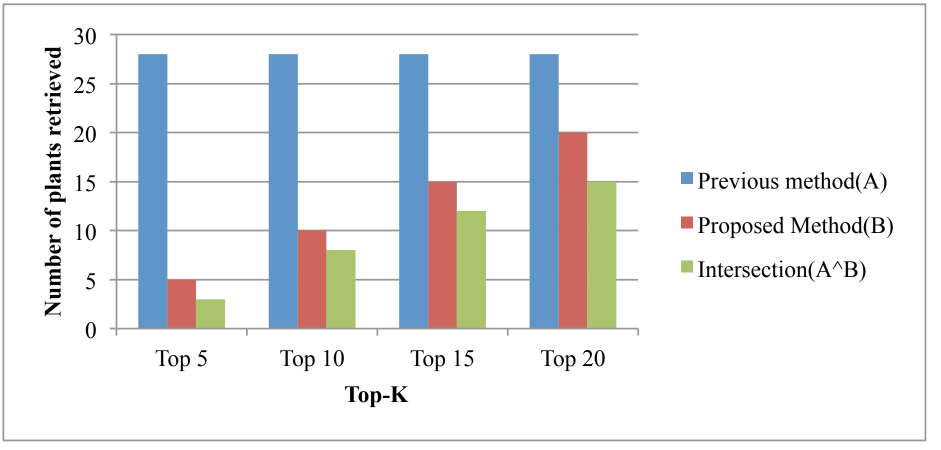

Figure 16. Chart showing the ranking efficiency for Query 5

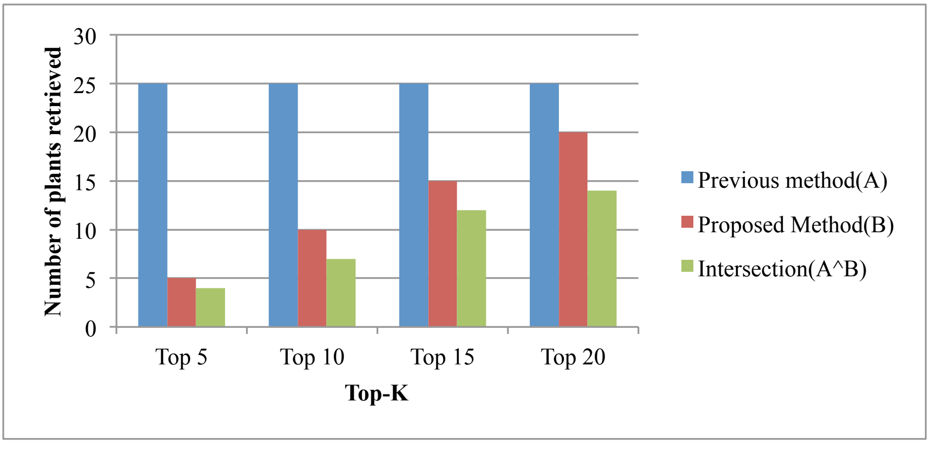

Figure 17. Chart showing the ranking efficiency for Query 6

5 CONCLUSION

Efficient knowledge retrieval in edaphology helps edaphologists and agriculturists in having the right crop for the right soil which ultimately increases the output. This paper discusses an efficient way to retrieve knowledge using two algorithms. Here, initially, the relational database is converted to the XML from which information retrieval is by using fuzzy search. The first algorithm is used when the soil characteristics are inputted to have the plant list and in the other algorithm, plant names are inputted to have the soil characteristics suited for the plant. Subsequently, result list is ranked by frequency thus obtaining the final sorted list used in order to evaluate the method that is made using performance metrics parameters such as the number of plants retrieved, ranking efficiency, computation time and memory usage. The method was also compared with our previous methods. The results obtained proved the validity of the method and the method obtained average computation time of 0.102 seconds and average memory usage of 2 486 Kb, which all are far better than the previous method results.

REFERENCES

[1] Koester, B.: Conceptual Knowledge Retrieval with FooCA: Improving Web Search Engine Results with Contexts and Concept Hierarchies. In: Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining, Lecture Notes in Computer Science, Springer 2006, Vol. 4065, pp. 176–190.

[2] Newman, B. D.—Conrad, K. W.: A Framework for Characterizing Knowledge Management Methods, Practices and Technologies. Proceedings of the Third International Conference on Practical Aspects of Knowledge Management, 2000, pp. 30–32.

[3] Li, Y.—Yao, Y. Y.: User Profile Model: A View from Artificial Intelligence. Proceedings of 3rd International Conference on Rough Sets and Current Trends in Computing, 2002, pp. 493–496.

[4] Li, Y.—Zhong, N.: Web Mining Model and Its Applications for Information Gathering. Knowledge-Based Systems, Vol. 17, 2004, pp. 207–217.

[5] Li, Y.—Zhong, N.: Mining Ontology for Automatically Acquiring Web User Information Needs. IEEE Transactions on Knowledge and Data Engineering, Vol. 18, No. 4, 2006, pp. 554–568.

[6] Tao, X.—Li, Y.—Nayak, R.: A Knowledge Retrieval Model Using Ontology Mining and User Profiling. Integrated Computer Aided Engineering, Vol. 15, No. 4, 2008, pp. 1–24.

[7] Yao, Y.—Zeng, Y.—Zhong, N.—Huang, X.: Knowledge Retrieval (KR). Proceedings of IEEE International Conference on Web Intelligence, 2007, pp. 729–735.

[8] Apistola, M.—Lodder, A. R.—Mommers, L.: A Knowledge Management Exercise in the Domain of Sentencing Towards an XML Specification. Proceedings of the Second International Workshop on Legal Ontologies, Amsterdam, 2001, pp. 48–57.

[9] Denning, S.: The Role of ICTs in Knowledge Management for Development. The Courier ACP-EU, 2002, No. 192, pp. 58–61.

[10] Irfan, R.—Shaikh, M. U.: Enhance Knowledge Management Process for Group Decision Making. Proceedings of World Academy of Science, Engineering and Technology, 2009.

[11] Whittaker, J.—Burns, M.—Van Beveren, J.: Understanding and Measuring the Effect of Social Capital on Knowledge Transfer Within Clusters of Small-Medium Enterprises. Proceedings of the 16th Annual Conference of Small Enterprise Association of Australia and New Zealand, 2003.

[12] Grinand, C.—Arrouays, D.—Laroche, B.—Martin, M. P.: Extrapolating Regional Soil Landscapes from an Existing Soil Map. Sampling Intensity, Validation

Procedures, and Integration of Spatial Context. Science Direct Journal, Vol. 143, 2008, pp. 180–190.

[13] Bui, E. N.—Henderson, B. L.—Viergever, K.: Knowledge Discovery from Models of Soil Properties. Ecological Modelling, Vol. 191, 2006, No. 3-4, pp. 431–446.

[14] Bui, E. N.: Soil Survey as a Knowledge System. Geoderma, Vol. 120, 2004, Iss. 1-2, pp. 17–26.

[15] Farenhorst, R.—de Boer, R. C.: Knowledge Management in Software Architecture: State of the Art. Chapter 2. In: Babar, M. A., Dingsoyr, T., Lago, P., van Vliet, H. (Eds.): Software Architecture Knowledge Management, Springer 2009, ISBN: 978-3-642-02373-6.

[16] Meenakshi, A.—Mohan, V.: An Efficient Tree-Based System for Knowledge Management in Edaphology. European Journal of Scientific Research, Vol. 42, 2010, No. 2, pp. 253–267.

[17] Ralph, L. L.—Ellis, T. J.: An Investigation of a Knowledge Management Solution for the Improvement of Reference Services. Journal of Information, Information Technology, and Organizations, Vol. 4, 2009, pp. 17–38.

[18] Yang, Q.—Yin, J.—Ling, C.—Pan, R.: Extracting Actionable Knowledge from Decision Trees. IEEE Transaction on Knowledge and Data Engineering, Vol. 19, 2007, No. 1, pp. 43–56.

[19] Tergan, S.-O.: Digital Concept Maps for Managing Knowledge and Information. Lecture Notes in Computer Science, Springer 2005, Vol. 3426, pp.185-204.

[20] Velasquez, J. D.—Palade, V.: A Knowledge Base for the Maintenance of Knowledge Extracted from Web Data. Knowledge-Based Systems, Vol. 20, 2007, No. 3, pp. 238–248.

[21] Abel, M.—Silva, L. A. L.—De Ros, L. F.—Mastella, L. S.—Campbell, J. A.—Novello, T.: PetroGrapher: Managing Petrographic Data and Knowledge Using an Intelligent Database Application. Expert Systems with Application, Vol. 26, 2004, No. 1, pp. 9–18.

[22] Wang, N.—Du, H.—Xu, B.—Dai, G.: Compact Indexes Based on Core Content in Personal Dataspace Management System. Computing and Informatics, Vol. 33, 2014, No. 2, pp. 281–302.

[23] Atmani, B.—Beldjilali, B.: Knowledge Discovery in Database: Induction Graph and Cellular Automaton. Computing and Informatics, Vol. 26, 2007, No. 2, pp. 171–197.

Anantaraman Meenakshi received her Bachelor's degree in physics in May 1995, Masters Degree in computer applications in June 1998 both from Madurai Kamaraj University and Master of Engineering in computer science and engineering in June 2005 from Anna University, Chennai. In July 2014, she received her Ph. D. in information and communication engineering from Anna University, Chennai, where she has been doing research in knowledge engineering for the past six years. She presented more than 10 papers on national and international conferences and published 5 papers in international journals. She has more than 16 years of experience in teaching. Currently she is a Professor in the Department of Computer Science and Engineering in K. L. N. College of Information Technology, Sivagangai District, Tamil Nadu, India.

Vasudev Mohan obtained his Doctor degree in applied mathematics from Madurai Kamaraj University, Tamil Nadu, India. He has more than 35 years of experience in research and teaching. Currently he works as a Professor and Head of the Department, and as Dean for Planning and Administration at Thiagarajar College of Engineering, Madurai, Tamil Nadu, India. His research interests include graph theory, artificial intelligence and finite state automata. He presented more than 10 papers on national and international conferences and published 20 papers in international journals.