ONTLOGY BASED AGRO ADVISORY SYSTEM

M. Tech. Dissertation

Submitted in partial fulfillment of the requirements
of the degree of

Master of Technology

by
Sonali Sahni
(Roll No: 09305912)

under the guidance of
Prof. N. L. Sarda

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Powai, Mumbai - 400076
Abstract

Queries are expressed by farmers related to their crop in natural language which are usually answered by human experts. It is desired to enable the data or knowledge captured in the system to understand the query as exactly as farmers see and ask questions. The designed system is an Agro Advisory System, which is an ontology based knowledge system. Ontology is designed for Cotton which contains detailed information about the crop. This is stored as a model in Oracle database. Farming activities of the farmer are also captured and stored in database. Farmer’s query is posted over a graph, which is generated from the ontology model and answer to the query is set of paths in the graph satisfying the user query intent. Farmer query is a keyword based query which is filtered based on various aspects like, context information about the farmer, weather predictions, etc. Taking all these parameters into consideration, query is answered which is specific to the farmer.
Acknowledgment

I would like to express my deep gratitude for my guide Prof. N. L. Sarda, who has always been making things simple to understand. Without his deep insight into this domain and his valuable time for this M.Tech. Project, it would not have been possible for me to move ahead properly. He has been remarkable in his attempt to keep me motivated in this project and has always tried to improve me with proper feedback.

Also, I would like to thank my parents who have always believed in me. Without their blessings, it would not have been an easy task. Finally, I would like to thank my friends for their encouragement and suggestions that they gave about my M.Tech. Project.

Sonali Sahni
IIT Bombay

June 26, 2012
# Table of contents

Abstract i

Acknowledgment ii

List of Figures v

List of Tables vi

1 Introduction 1

2 Literature Survey 3

3 Ontology - Based Querying 6
   3.1 System Architecture ............................................. 7
       3.1.1 Query Interface ........................................... 7
       3.1.2 Query Engine ............................................... 7
       3.1.3 Database .................................................. 7

4 Crop Ontology 9
   4.1 Agropedia Indica ................................................ 9
   4.2 Knowledge Models or Ontologies ................................. 9
   4.3 The Foundational Agriculture Crop Ontology .................. 10
       4.3.1 Relations in a Generic Crop Ontology .................... 10
   4.4 Cotton Ontology ............................................... 10
       4.4.1 Disease Ontology .......................................... 13
   4.5 Ontology Storing and Querying ................................. 14

5 Information in Database 15
   5.1 Past Data of Farmer ............................................. 15
# TABLE OF CONTENTS

5.2 Current Data .................................................. 15
5.3 Data for Weather Prediction ................................. 16
5.4 Mapping Ontology and Databases ......................... 16

6 Implementation Approach ................................. 18
   6.1 Keyword - Based Query ................................ 18
   6.2 Context - Based Query ................................ 18
   6.3 Search Algorithm ........................................ 18
      6.3.1 Mapping Keywords on Ontology ................... 20
      6.3.2 Infer Farmer Activity from Database ............... 20
      6.3.3 Infer Information related to Farmer's Context .... 20
      6.3.4 Infer Dependent Information ....................... 21
      6.3.5 Classifying Nodes and Edges on Graph ............. 21
      6.3.6 Calculate Paths ..................................... 22
      6.3.7 Ranking of Paths .................................... 23
   6.4 Description of Algorithm ............................... 24
   6.5 Ontology Validation ..................................... 28
      6.5.1 Steps for validation ................................ 28
      6.5.2 Observations ........................................ 30

7 Results .......................................................... 31

8 Conclusion and Future Work ............................... 37

Appendix .......................................................... 38
List of Figures

3.1 System Architecture ............................................. 7
4.1 Activity Ontology (Generic) ..................................... 11
4.2 Diseases Ontology (Generic) ................................. 11
4.3 Diseases Ontology (Specific) ................................. 12
5.1 Database Schema for farmer activity .......................... 15
5.2 Database Schema for weather prediction ....................... 16
6.1 Overview of Search Algorithm ................................. 19
6.2 Calculating Paths on Ontology ................................. 22
6.3 Observations for Ontology Validation ......................... 30
A1 Results - Example Query#1 - Case 1 ......................... 38
A2 Results - Example Query#1 - Case 2 ......................... 39
A3 Results - Example Query#1 - Case 3 ......................... 40
A4 Results - Example Query#1 - Case 4 ......................... 41
A5 Results - Example Query#2 - Case 1 ......................... 42
A6 Results - Example Query#2 - Case 2 ......................... 43
A7 Results - Example Query#3 ................................. 44
# List of Tables

<table>
<thead>
<tr>
<th>Table Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Mapping - ontology and database</td>
<td>16</td>
</tr>
<tr>
<td>6.1</td>
<td>All matchings from ontology</td>
<td>24</td>
</tr>
<tr>
<td>6.2</td>
<td>Final matches for keywords</td>
<td>24</td>
</tr>
<tr>
<td>7.1</td>
<td>Parameter details - Example Query#1 - Case 1</td>
<td>31</td>
</tr>
<tr>
<td>7.2</td>
<td>Parameter details - Example Query#1 - Case 2</td>
<td>33</td>
</tr>
<tr>
<td>7.3</td>
<td>Parameter details - Example Query#1 - Case 3</td>
<td>33</td>
</tr>
<tr>
<td>7.4</td>
<td>Parameter details - Example Query#1 - Case 4</td>
<td>34</td>
</tr>
<tr>
<td>7.5</td>
<td>Parameter details - Example Query#2 - Case 1</td>
<td>35</td>
</tr>
<tr>
<td>7.6</td>
<td>Parameter details - Example Query#2 - Case 2</td>
<td>35</td>
</tr>
<tr>
<td>7.7</td>
<td>Parameter details - Example Query#3</td>
<td>36</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The intent of developing an Ontology Based Agro Advisory System emerged from the idea of designing an expert system for farmers to pose their queries to the system and getting the best solutions to their problems.

Farmers express their queries in a natural language which are usually answered by human experts. It is desired to enable the data or knowledge captured in the system to understand the query as exactly as farmers see and ask questions. Hence a need arises for developing a knowledge-based system which captures significant aspects of the reality that farmers are likely to mention, that is, answer to the queries dealing with what farmers mean and not what farmers say. The initial idea came from the work on the proposal of Agro Advisory System [7] which is a query-answering support for farmers. It is nothing but an ontology based knowledge system. Knowledge acquisition is done with the aid of Agro experts.

In this project we extend that idea to design Ontology Based Agro Advisory System. Ontology is designed for Cotton and the concepts which have been elaborated are soil and climatic conditions for cotton; varieties of cotton; disease affecting the crop, reasons for their occurrence, their symptoms shown, and cure for those diseases; similarly information on pests attacking the crop, their precautions and cure. This is constructed using Protégé and is stored as a model in Oracle database.

Agro Advisory system cares to store and maintain farming activities of the farmer time to time. Information regarding farmers and their farming practices for past 5 different years is stored in the database for three districts of Punjab.

The system provides an interface which is capable of querying ontologies, created using Web Ontology Language (OWL) [3]. These are stored in the Oracle database as models. The system has been coded using Jena API for Java. Graph is generated for the ontology and querying is performed by mapping user query keywords over the graph. Query is answered by returning set of paths which maximum satisfy the matched keywords. Query search over the ontology is aided by providing contextual information about the farmer, i.e., his location, crop sown, activities performed by him, stage of the crop, etc. and weather predictions. These set of information act as a filtering criteria for better answering the user query.

We present an approach for designing an agro advisory system which answers farmer
queries using background knowledge available in ontologies. Making use of the contextual information present in the database, we answer the queries which are specific to the farmer. The power of keyword based queries over ontology is reflected which takes into account the use of annotation properties for identifying information types in the ontology. This aids in better interpretation of the query as compared to applying direct SPARQL queries over the ontology. Also, ranking of the interpreted queries is done based on various factors which is not taken care by SPARQL.

In further chapters, we present the related work carried out in this area in Chapter 2. The need of ontology-based querying system is expressed in Chapter 3 along with the system architecture. This is followed by the ontology concepts and standards described in Chapter 4. This section also talks in detail about cotton ontology, its storage and querying. Chapter 5 details about the information stored in database, that is farmer activities and weather prediction data and also how this information assists in ontology-based querying. Chapter 6 explains the query engine which is the core part of the system. It also states the search algorithm for querying ontology systems followed by ontology validation part. Implementation results are produced in Chapter 7 followed by Conclusion and Future Work in Chapter 8.
Chapter 2

Literature Survey

Agro Advisory systems have been developed in the past which help in advising the farmer for their crop related queries.

Initial work was carried out at IIT Bombay for developing a discussion forum for farmers, namely aAQUA [13], which is an online multilingual, multimedia Question and Answer based community forum. In a typical aAQUA thread, a farmer submits a problem, and agriculture experts or other farmers provide solutions to it.

eSagu [14] is another advisory system where agricultural expert delivers the expert advice at regular intervals (once in one or two weeks) to each farm by getting the crop status in the form of digital photographs and other information. It is a query-less system and provides agro-advice even without the farmer asking a question by following a proactive approach and avoiding problematic situations.

mKRISHI [12] is a mobile agro advisory system which provides the farmer with audio-video facilities on a mobile phone to express their queries to experts with minimal use of text. It adapts event-based experiential approaches to analyze data from different sources and develop techniques to store and visualize all this multimedia data which holds relevant to the context. The drawbacks behind each of these approaches is the presence of an agro-expert for looking into the farmer’s problems.

The approach presented in this paper mainly concerns with developing an independent interface where farmer can query and obtain results for the query specific to his context. Crop information is stored in a knowledge model known as ontology and queries are answered by searching over these ontology graphs. Ontologies are increasingly being used to build applications that utilize domain-specific knowledge. The major challenge in handling ontology-based semantics queries is how to store and access the ontology data, which are often represented as graphs. The increasing availability of this semantic data offers opportunities for semantic search engines. However, formal query like SPARQL [5], hinders the casual users in expressing their information need. The user thus needs to know the complex syntax of the formal query. Moreover, the user also needs to know the underlying schema and the literals expressed in the ontologies.

Keyword interfaces is one solution to this problem. They are easier to handle for casual
users. Users are very familiar with these interfaces due to their widespread usage. Compared to formal queries, keyword queries are simple queries with set of keywords which can be simply set of words entered by user for expressing his information need.

Since keyword interfaces are suitable for casual users, many studies have been carried out to bridge the gap between keyword queries and formal queries, especially in the information retrieval and database communities [6, 8]. There also exist approaches that specifically deal with keywords interfaces for semantic search engines. Natural language interfaces offer end-users a familiar and convenient option for querying ontology-based knowledge bases. They hide the formality of ontologies and query languages from end-users by offering them a familiar and easier way of query formulation. Many approaches for converting keyword queries to formal query languages are presented for natural language interfaces to ontologies. Some of them are based on either syntactic input [9] and some are based on keyword or unrestricted input [11, 10, 16, 18]. However users feel more comfortable with keyword input because it does not restrict users to follow grammatical rules hence require less time and effort in formation of a query.

Based on the type of input to natural language interfaces, the first category is the one that accepts the syntactic input and parse the user question to find out the RDF statement matches for the question. Querix [9] is an ontology-based question answering system that translates generic natural language queries into SPARQL. It is a domain-independent NLI that requires full English questions as query language. Also, it limits them to a given set of six sentence beginnings. In case of ambiguities, Querix relies on clarification dialogues with users. In this process users need to disambiguate the sense from the system-provided suggestions.

The recent systems based on keywords are SemSearch, Q2Semantic and an approach proposed by [16]. SemSearch [11] is a concept based system with a Google-like Query Interface. It deals with keyword queries in semantic search engines. The template-based approach discussed fixes the possible interpretations and thus, cannot always capture the meaning intended by the users. In SemSearch, keywords are mapped to elements of triple patterns of predefined query templates. These templates fix the structure of the resulting queries. However, all interpretations of the keywords cannot be captured by such templates. Also, since queries with more than two keywords lead to a combination of different possible interpretations, a large number of templates would be needed. SemSearch restricts the user to separate subject and other keywords with a colon.

Some complete graph based processing approaches [16, 18] are introduced. These approaches construct a graph for the keywords mapped resources and then convert the graph to the formal query. A more generic graph-based approach has been proposed by [16] to explore all possible connections between nodes that correspond to keywords in the query. This way, all interpretations that can be derived from the underlying RDF graph can be computed. But [16] is
based on the assumption that the ontology entities to which keywords map to are connected via paths up to a given depth $d$. Hence it explores the graph to a limited depth to avoid complexity.

As constructing and manipulating the complete graph for all resources is time consuming, Q2Semantic [18] has presented a time effective approach by first constructing a subgraph for the mapped resources, apply a process to compact it and then perform rest of the actions on that compact graph. But Q2Semantic accepts certain ontological resources, i.e., current approach of Q2Semantic support keywords that match only literals and concepts contained in the RDF data.

We have adopted a similar graph based approach by exploring all connections between user selected nodes. Nodes exploration is carried on a derived subgraph, applying searching techniques over the graph and calculating paths from it.
Chapter 3

Ontology - Based Querying

Quite a lot of services are already available in India in order to help farmers find information on their activities and products.

However, none of the current services make extensive use of the new semantic techniques, i.e., no semantic searches are available allowing concept-based searches, language-independent platforms for knowledge navigation, no inference or reasoning capabilities are available.

A system is proposed which handles farmer related query without the need of agro-expert handling them at the other end. Farmers as end users can post their observations about the crop to the system and the system in return advices them for the action to be taken against the problem or provides information on the query.

The knowledge which the system requires to store is comprehensive information about the crop. The knowledge given by the agro-expert needs to be acquired by the system for making it independent of answering queries without the aid of an agro-expert. Along with this, system should also be capable of capturing context information. This can be information about farmer’s context, context information on weather, history about diseases, history about the insecticides, etc.

The knowledge of the agro-experts is acquired as knowledge models known as ontologies which are capable of storing the complete information related to the crop. Knowledge seen and exhibited by an agro-expert can be equally depicted by the ontology. It depends upon how well it is captured. This encourages the existence of a system independent of an agro-expert competent of satisfying the end users. Apart from the information related to the crop stored as a knowledge model, contextual information is also stored in the database. One of the context is the information of the farmer, which details about the various activities performed by the farmer time to time. This can be captured only when the farmer registers about his activity done on the crop to our system. Another context information captured is the weather prediction which forecasts the climatic conditions in the region and help in suggesting the farmer better.

Hence arises the need on designing an ontology-based querying system which maps the information asked by the user on the knowledge stored in the ontology, and with the assistance of the contextual information stored in the database about the crop, weather, geographic details, etc., advice or suggestion is made to the user. Overview of system architecture is given below.
3.1 System Architecture

System consists of the following components:

3.1.1 Query Interface

This is an interface where farmer logs in and can post his farming related query. Query can be an observation made by him, information which he wants to seek, or an action/suggestion oriented query. The query made by the farmer can be an attribute-value pairs or it may be a keyword-based query. This query is parsed, tokenized and mapped to resources in the ontology.

3.1.2 Query Engine

This is the core of the system which handles farmer query. Farmer query can be of the form attribute-value pairs or it can be a keyword based query. Query is mapped to the ontology resources which can be any of classes, instances, object/datatype property or literals. Based on the selected resources, dependent information is extracted and also marked on the ontology. Secondly, farmer’s current activities are queried from the database and suggestions are made for what action should be taken by him depending upon the weather forecast and past activities performed, if any.

3.1.3 Database

This is the repository which holds complete information about the crop in the form of ontology. This ontology is stored as a graph in database. Also, past records of the farmers are present as relational data which give us the farming practices done by them. Current farming operations are also stored in separate tables. Weather forecast for the next 5 days are gathered from
3.1 System Architecture

meteorological department and captured in the database. Based on these generic rules and observations are made which help in validating the ontology. Advice is generated by the system based on the current farming which helps him to take precautionary measures in his practice.
Chapter 4

Crop Ontology

4.1 Agropedia Indica

agropedia [1] is an online knowledge repository for information related to agriculture in India. Formerly titled as Agropedia Indica, the development of the project was commenced by team involving various institution of India and headed by Indian Institute of Technology Kanpur (IIT Kanpur).

The agropedia project provides a national reference portal for Agriculture in India, providing domain-specific and user specific services. Agriculture Knowledge repositories of universal knowledge models and localized content is made available for a variety of users, with appropriate interfaces.

agropedia content is semantically cataloged. This is done with the help of knowledge models (KMs) made available by IITK.

agropedia contains a repository of knowledge models. Knowledge models help in mapping the knowledge and tagging content of this site so that the information is semantically searchable.

4.2 Knowledge Models or Ontologies

Knowledge models are mainly used to navigate agricultural knowledge and to organize and search agricultural content. These KMs are the structural representation of knowledge by using symbols to represent pieces of knowledge and relationships between them, which can be used to connect seamlessly to the knowledge base in agropedia using semantic tools.

IIT Kanpur, with the help of an agricultural experts, have organized the KMs in the following ways: a top level generic map for crops (which is called the “Foundational Agricultural Crop Ontology”), and many specialized maps based on specific topics, such as rice, diseases, pesticides, etc. [15].

IIT Kanpur has made its contributions in knowledge models for nine selected crops - Chickpea, Groundnut, Litchi, Pigeon pea, Rice, Sorghum, Sugarcane, Vegetable pea, and Wheat [2]. We focus on giving a wider picture of Cotton crop providing information in different areas.
4.3 The Foundational Agriculture Crop Ontology

The Foundational Agricultural Crop Ontology contains information which is common to every crop type [15]. This information includes:

- origin - defines the geographical areas specific to it
- environmental information - includes information about climate, soil
- varieties and cropping systems
- botanical description
- production practices - comprises of production technologies and protection technologies
- post production practices - related to harvesting, threshing, post harvest technologies, marketing, etc.

4.3.1 Relations in a Generic Crop Ontology

Four different types of relationships are captured in the generic crop ontology. They are defined as:

- “are”, which correspond to the subclassOf
- “is a”, which represent an instance of a concept
- DataTypeProperty, which link individuals to data values
- ObjectProperty, which link individuals to individuals

Both owl:ObjectProperty and owl:DatatypeProperty are subclasses of the RDF class rdf:Property. DataTypeProperties and ObjectProperties may be assigned with a meaningful name.

4.4 Cotton Ontology

Knowledge about the cotton crop is available with us in the form of Ontology. Ontology formally represents knowledge as a set of concepts within a domain, and the relationships between those concepts. Ontologies are the structural frameworks for organizing information and are used in the Semantic Web as a form of knowledge representation about the world or some part of it.

The concepts which have been elaborated for cotton are the soil and climatic conditions for cotton, recommended varieties of cotton which can be location specific, disease affecting
the crop, reasons for their occurrence, their symptoms shown, and cure for those diseases; similarly information on pests attacking the crop, their precautions and cure. The ontology also contains knowledge concepts on the various activities of farming, like, hoeing, sowing, irrigation, fertilizing, spraying and harvesting, along with their timelines, see Figure 4.1

![Figure 4.1: Activity Ontology (Generic)](image)

Ontology can be classified as generic ontology and specific ontology. Generic ontology gives details of the practices of any crop. Basically it defines the classes of the ontology and the relations existing between them. It explains how any two classes are related and the properties existing for any class. Crop ontology is a generic ontology. Figure 4.2 depicts Disease ontology. Specific ontology is the ontology built over the generic ontology and defines the instances of the ontology classes related by properties defined by the generic ontology. Cotton ontology is a specific ontology. Figure 4.3 explains specific ontology for Disease. Some of the components of OWL ontology are defined below:

![Figure 4.2: Diseases Ontology (Generic)](image)
4.4 Cotton Ontology

Figure 4.3: Diseases Ontology (Specific)

**Classes**
Classes describe concepts in the domain. Eg. Disease, Symptom, Damage, Prevention_Measure etc. They can be seen as orange colored concepts in Figure 4.2 and Figure 4.3.

**Instances**
Instances can be defined to be objects of the class. They exhibit all the properties which their class has. Eg. For the classes defined above, instances can be Bacterial_Blight, Leaf_Stunting, Fruit_Shedding, Avoid_American_Cotton, etc. respectively. They are represented as green colored concepts in Figure 4.3.

**Object Property**
Relation existing between any two classes is defined as an object property. Any object property has a domain class (from which class) and a range class (to which class). Eg. causes(Disease, Damage), has_Symptom(Disease, Symptom). causes and has_Symptom are object properties from Disease to Damage and Symptom respectively.

**Datatype Property**
Datatype property of any class is a property which gives values for instances of that class. It is a relation between class and a datatype value (String, Integer, Float, etc.). Eg. quantity(Spraying, String) in Figure 4.1.
4.4 Cotton Ontology

Property – Domain and Range
Properties relate individuals from a domain to individuals in a range. Domain and range are axioms used in the inference process. Eg. For the given relation has_Symptom, Disease is defined as domain and Symptom as range.

Literals
This is the value to which any datatype property matches. Eg. quantity(Spraying, String) has mapping as quantity(Confidor, 40). Here “40” is the literal value.

4.4.1 Disease Ontology
Disease Ontology gives complete knowledge about the diseases affecting the crop. It gives along with information relevant to it like Symptom affecting a disease, Damage caused to the crop, Stage at which it occurs, etc. This information gives a better knowledge what could be the causes of an observation made by farmer and based on state of his crop, what are the suggested control measures.

Stage
Stage class defines various stages in a crop life cycle. This plays a vital role in providing context information about the crop. The diseases are associated with a stage which informs what could be the period of occurrence of a disease. Taking this into account, farmer specific queries can be answered well.

Control Measure
Different control measures defined for the crop are also either stage specific of season specific. This also adds to context information about the crop which aids in answering queries depending upon what is the suitable action to be taken by farmer for current state.

As illustrated above, the ontology stores complete knowledge related to the crop as classes, instances, object/datatype property and literals. Some of the knowledge stored would require immediate actions to be taken by farmers and advice should be generated for such a knowledge. This is mentioned as actionable knowledge. For e.g., knowledge about Disease, Pest, Symptom, etc. would require to take some control measures. Hence this is actionable knowledge. Other kind of knowledge stored deals with information about the crop and would just give him details about the crop. E.g., Variety, Stage, Part, etc. of the crop. This knowledge distinction is due to the fact that while the farmer queries for any knowledge, actionable knowledge holds a priority over the general information. The actionable knowledge is associated with some annotation properties which are explained in later sections. This marks some classes to
be actionable ones.

Ontology also depicts knowledge as spatial and temporal aspects. Spatial aspects deal with the geographic information like soil, climate requirements, etc. of the crop. The temporal aspects mainly concern stages of the crop, seasons of performing an activity like, during which period, etc. name.

4.5 Ontology Storing and Querying

OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web [3]. Due to the standardized representation of OWL ontologies in RDF [4], which comprises of subject-object-predicate tuples, it necessitates extracting ontologies from OWL files and use database management systems for storing ontologies in databases.

The Simple Protocol and RDF Query Language (SPARQL) [5] is a SQL-like language for querying RDF data. SPARQL queries can be executed from inside a JAVA application using the specialized Jena library methods. Each SPARQL request starts with PREFIXes which denote the namespaces used in the query afterwards. An example query for getting varieties of cotton is:

```
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#
PREFIX owl: http://www.w3.org/2002/07/owl#
PREFIX node: <http://www.owl-ontologies.com/Ontology1328892204.owl#>
SELECT ?object
WHERE { node:Cotton node:has_Variety ?object }
```

The first lines defines namespace prefix, and the next lines use the prefix to express a RDF graph to be matched. Identifiers beginning with question mark ? identify variables.

The ontology is stored as RDF triples in database. Jena provides with APIs for modeling it to a graph with nodes and edges. Ontology is hence expressed as a graph with subject and object as nodes and predicates as edges. Traversing this graph and establishing relations between nodes would result in a path of meaningful information for the crop ontology.
Chapter 5

Information in Database

5.1 Past Data of Farmer

Data from various villages of Punjab have been collected and is then used for analysis. The data has been collected from three districts of Punjab. It contains information regarding farmers and their farming practices in 5 different years. Analysis of this data is performed in various aspects to validate details and find patterns in farming practices of farmers. This analysis would help in answering to queries raised by farmers for their specific crops. Observations made for their past records can be verified and queries be answered. This can be seen in Figure 5.1.

Figure 5.1: Database Schema for farmer activity

5.2 Current Data

Current practices of farmer are captured time to time and are monitored periodically. Activities of the farmers at various stages are recorded, i.e. when the crop was sown, its variety, location, etc. Based on the past practices, farmer may be advised on managing current crop cycle.
5.3 Data for Weather Prediction

Weather forecast for the next 5 days are gathered from meteorological department and captured in the database. Mainly data about rainfall, humidity and temperature is captured.

The weather data about rainfall, temperature and humidity is automatically extracted from http://www.imd.gov.in/section/nhac/distforecast/ for each of the three districts of Punjab, i.e., Ferozepur, Muktsar and Mansa. The current system mainly uses the rainfall data and stores it in a separate table RAINFALL which helps in adding on to farmer’s irrigation activity. Also, it is measured that in near future if rain is predicted, farmer is warned in advance to either perform the activity post rains or perform it beforehand, as appropriate.

5.4 Mapping Ontology and Databases

The activity information captured for the farmer is stored in various tables in the database. This information is directly or indirectly mapped to the classes and properties in the ontology. An ontology class mainly maps to a table in the database whereas the properties of a class map to attributes of the table.

<table>
<thead>
<tr>
<th>Activity</th>
<th>DB_Table</th>
<th>Date_of_Appln</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Irrigation</td>
<td>irrigation_type_canal</td>
<td>date_of_application</td>
<td>null</td>
</tr>
<tr>
<td>Irrigation</td>
<td>irrigation_type_tubewell</td>
<td>date_of_application</td>
<td>null</td>
</tr>
<tr>
<td>Nitrogen</td>
<td>urea_appln</td>
<td>date_of_usage</td>
<td>quantity</td>
</tr>
</tbody>
</table>

Direct mapping of information implies that ontology class/property is directly mapped to database table name/attribute name. Indirect mapping implies that the mapping is actually a function where database attribute name maps to a derived function of combination of properties of the ontology class. Below is the table which stores the mapping information, see Table 5.1. First column is the activity which is stored in the ontology. Second column is the database
5.4 Mapping Ontology and Databases

table name to which it maps to. And henceforth, the attribute name is the property of the ontology and the value in the table is the value of the attribute to which that property maps to. E.g. For the activity “Nitrogen”, ontology class name is defined as “Nitrogen” whose mapped database table name is “urea_appln”. The mapping of their properties and attributes is as, “quantity” as ontology property of class “Nitrogen” maps to “quantity” field in “urea_appln” table. “Date_of_Appln” defined here is an indirect mapping. This is not a direct mapping in ontology and is derived as a function of ontology properties \((sowing\_date + frequency \times interval)\). This calculated property maps to “date_of_usage” field of “urea_appln” table.
Chapter 6

Implementation Approach

6.1 Keyword - Based Query

The ontology stored in the OWL file is converted into a graph structure and stored in database in the form of triplets. These triplets are of the form node-edge-node where any node represents a class or an instance and the edges represent the properties/relation between classes.

A query from a user can be answered by mapping the keywords over this graph structure and returning a sub graph which maximum satisfies these selected keywords. Hence the challenge is answering a keyword query over a graph structured RDF data. An algorithm is proposed below which finds out the optimal answer to the query.

6.2 Context - Based Query

While answering to a farmer based on the observations made by him in the query interface, analysis is done on his current activities, i.e., when did he sow his latest crop, his crop is at what stage, what all activities he has performed and what next is to be done by him. This is supported by the observations from the weather predictions on rainfall, temperature and humidity for the next 5 days captured which also help in suggesting the farmer for what needs to be done and care to be taken.

Suggestions are made to him based on the activity information stored in ontology and weather predictions made.

6.3 Search Algorithm

Farmer posts his query over the query interface which is handled as a graph based search. Nature of the query could be set of observations made by him related to the crop. Simplest form of presenting his observations could be set of <attribute-value> pairs, where an attribute could be the context and value is the observation made by him. Farmer may or may not be aware of the pre-defined attribute names. Hence, we offer him to perform keyword query where the system captures the observations made by the farmer and best maps it to the attributes it corresponds to. Some example queries posted by farmer could be:
6.3 Search Algorithm

“Leaf turning yellow to red”,
“Varieties of cotton”

First query is a kind of observation-based query where farmer posts his field observations. Also it is an action query where a farmer would seek an advice from the system. The second query is an information-based query where the farmer wishes to know some general information about the crop.

Ontology is stored as a graph in the database. Generic ontology $O_G$ defines a graph which consists of classes, their object properties and datatype properties. Classes form the nodes of a graph and edges are comprised of object property and/or datatype property. Specific ontology $O_S$ consists of generic ontology with instances of classes and relations (object and datatype properties) between them. Here, nodes of the graph are classes, instances and literal values; whereas the edges are object property and/or datatype property.

The various steps involved in answering the query are described in detail in the below sections as (i) identifying keywords, (ii) finding farmer’s activity information, (iii) finding information as per farmer’s context, (iv) finding dependent information, (v) classification of keywords as nodes and edges on graph, (vi) calculating paths on graph and (vii) ranking of paths. See Figure 6.1. Before performing query search, preprocessing is done for resources in $O_S$ by categorizing them into sets of classes, instances, objectProp, dataProp and literals.

![Figure 6.1: Overview of Search Algorithm](image-url)
6.3 Search Algorithm

6.3.1 Mapping Keywords on Ontology

User query consists of keywords $k_1, k_2, ..., k_n$ which can be mapped to different concepts on the ontology. First step in processing user query is identifying the keywords as one or more of classes, instances, objectProp, dataProp and literals. Keywords are identified to be one or more of the above category and added to sets defined, $C$ for classes, $I$ for instances, $O$ for objectProp, $D$ for dataProp and $L$ for literals. Keywords identified to be one or more of these categories are mapped first based on the exact match of the keyword. If no match is found, then we search for synonyms of the keyword by making use of the WordNet [17].

$$K = k_1, k_2, ..., k_n, \text{ set of user query keywords}$$
$$C = k_{c1}, k_{c2}, ..., \text{ where } k_{ci} \in K$$
and similarly for other sets $I, O, D$ and $L$

Thus, here we convert query $K$ into $<C, I, O, D, L>$

6.3.2 Infer Farmer Activity from Database

Apart from the observations made by the farmer about the crop and posted as a query on the system interface, the system checks for various activities of the farmer. The ontology consists of different activities of the crop, their interval and frequency of application, critical stages of application, etc. System checks whether the farmer has performed all activities till date on time. Different kinds of responses are returned for this.

- If the activity is on time, system returns that activity is on time. In addition it suggests the next date of activity.
- If some activity is not performed, system checks whether that has any adverse affect on the crop. If no, then it just suggests the farmer to perform the activity.
- If yes, then it reports lack of activity and also its harm on crop. Related information is marked on the ontology which help in answering the query better. As an example, for an observation “Boll shedding” made by farmer, if lack of irrigation is seen, then it is registered as one of the causes. Such instances of the ontology are captured in a set of activityI.

6.3.3 Infer Information related to Farmer’s Context

Information queried or observed by the farmer is many times dependent on the farmer’s context and answer to the query may vary from farmer to farmer. This is nothing but the contextual information related to the farmer which helps in further filtering the query results. This is also represented in the ontology by annotation property :context. The information stored in the
ontology is dependent on some of the factors like the stage of the crop of the farmer, the ongoing season, etc. according to which suitable suggestions are made for his query. This information is inferred and marked as selected on the ontology graph, which acts as a filtering criteria for getting relevant answer to user query.

An example of context information can be an observation made by farmer as “Boll Shedding”. Some of the possible reasons for this are “Lack of Irrigation” at Flowering/Boll development stage, “Lack of Nitrogen” at Flowering stage, “Attack of American Bollworm” during the season from Sep-Oct. Based on the activities of the farmer and the season/stage of the crop, this information is filtered and appropriate information is marked as related information. This can vary from farmer to farmer as for one farmer it can be due to lack of irrigation, and for the other one it could be due to attack of American bollworms.

This context information is added to set contextI containing instances.

6.3.4 Infer Dependent Information

For the set of nodes identified in the previous step, we find the related information from the ontology. Here, we identify if there is any dependent information which needs to be given apart from the one queried by the user. For e.g. If the user inputs any Symptom observed by him, related information which we can give him regarding this is Disease and their Control Measure. Dependent information is represented in the ontology by associating annotation properties :dependentClass with the classes. These dependent classes are captured in a set dependentC.

6.3.5 Classifying Nodes and Edges on Graph

For answering the query as different paths on the graph, nodes and edges need to be marked on the graph. For the various classes and instances identified in previous steps, nodes and edges are marked selected on the graph from these sets. Set of selected nodes SN and a set of selected edges SE are made. Output of this step is <SN, SE> which are sets of selected nodes and edges.

Graph based search can be performed either on the generic ontology graph OG or specific ontology graph OS. In order to perform the search efficiently, we perform a search over the graph OG, find relevant paths and then find instance paths by mapping those over the graph OS. This is done because graph OG is less dense than OS and we can avoid unnecessary search. Hence classification of nodes is an important step as we need to map the user keywords along with context and dependent information as selected nodes SN and selected edges SE on graph OG.

Classification of the selected nodes is done as:

- Add all the classes identified in C to the set SN of selected nodes.
6.3 Search Algorithm

- For adding instances stored in $I$ to $SN$, we need to identify corresponding classes of it and store in $SN$. Classes of an instance will include its class and super classes defined for it.

- Similarly, classes of instances stored in activity$I$ and context$I$ are captured and stored in set $SN$.

- Add all classes of set dependent$C$ to set $SN$

- Add all object properties in $O$ to the set $SE$

- Add all object properties in $D$ to the set $SE$

- For the literals in $L$, add their corresponding datatype property to set $SE$ and its connecting node (domain) to set $SN$

6.3.6 Calculate Paths

Paths are calculated between every node in the set $SN$. These are the paths on the graph $O_G$. This may include are possible paths between any two nodes. We use BFS algorithm to find the paths between nodes. Many scenarios are taken care for the paths obtained here which are discussed further.

For all the paths found, instance paths are calculated over the graph $O_S$. This is illustrated in Figure 6.2

Many of these paths would contain redundant information, either as a path being sub path of another or a path being inverse of the other. These paths are discarded based on relevancy of
6.3 Search Algorithm

the nodes. Let $P_1$ and $P_2$ be two paths. Some factors which are taken care for selecting either of them is:

if $P_1 \subset P_2$ then
    if $\exists [i \in (P_2 - P_1) \& i \in SN] : i \neq \phi$ then
        Discard path $P_1$
    else
        Discard path $P_2$
    end if
else
    Discard path $P_2$
end if

if $P_1$ is inverse of $P_2$ then
    Keep path which has initial few nodes as user query keywords
end if

This is done as because both the paths $P_1$ and $P_2$ convey the same information. But to make the result more semantically understandable, information is given from user query keyword to details about that.

6.3.7 Ranking of Paths

Retrieved paths are scored using a combination of three factors. First one and most important is the similarity count. The next priority factor is actionable information and the last one is dependent information. Each of these is described in detail in the next section.

Similarity Count: The similarity count reflects the similarity of the ontologies concept’s name with the part of the query keywords (some set of keywords). The highest score is given to the relation that is the most similar to the chunk. For ranking the paths, path which contains maximum number of matched keywords is given the highest score. Since we are making use of the WordNet, we have synonyms matches also apart from exact keyword match. In similarity count, more weightage is given to keyword matches as compared to synonym match.

Actionable Information: The actionable information is the one which requires user to perform immediate action. Some of the actionable information is performing activities like irrigation, fertilizing, etc. on time to avoid damage to the crop. Other could be control measures to be performed for controlling of diseases and pests attacking the crop. Since these are the information which need attention, any path containing more of these nodes gets a higher score. :context information derived in algorithm’s previous steps acts as the actionable information.

Dependent Information: Dependent information is the related information which we tend to give the user for better answering of user query. Information which is added as :dependent-
6.4 Description of Algorithm

The above algorithm is explained in detail below taking an example query and running for a particular scenario.

State of farmer crop

Crop sown date: 15th Apr
As per current date: Irrigation on time, Potassium & Phosphoros application on time.
Lack of Nitrogen.
Current crop stage: Flowering
Query posted by farmer: “Boll Shedding”

Mapping Keywords on Ontology

In this step, keywords are mapped to ontology. For the given query, we get keyword matches found from ontology are as below.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boll</td>
<td>Boll, Boll_Infection, Boll_Shedding, Bad_Boll_Opening, ...</td>
</tr>
<tr>
<td>Shedding</td>
<td>Boll_Shedding, Flower_Shedding, Buds_Shedding, ...</td>
</tr>
</tbody>
</table>

But from the above set, not all the matches are selected. For any keyword, if an exact match is found, then other matches corresponding to it are selected only if they match more than one keyword. If no exact match is found, then all possible matches are accounted. Below is the final set of matches

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boll</td>
<td>Boll, Boll_Shedding</td>
</tr>
<tr>
<td>Shedding</td>
<td>Boll_Shedding, Flower_Shedding, Buds_Shedding, ...</td>
</tr>
</tbody>
</table>

From the above matches, sets $<C, I, O, D, L>$ are identified to be as below.

$C = {}$

$I = \{Boll, Boll_Shedding, Flower_Shedding, Buds_Shedding\}$
Infer Farmer Activity from Database

Farmer’s activities performed are analyzed from the database and checked for consistency of activity. This is done as described below.

get dates of activity from database
calculate dates of activity from the ontology till today
if #dates from database == #dates from ontology then
   Activity is fine
else if #dates from database > #dates from ontology then
   Report excess of activity performed
else
   Check for stage where activity is missed
   if stage == Critical stage then
      Lack of activity reported
      related instances added to set activityI
   else
      Farmer is simply advised to perform activity
   end if
end if

Hence as per the current crop state, “Nitrogen Deficiency” is reported and hence related instances are added to activityI set.

\[
\text{activityI} = \{\text{Nitrogen Deficiency, Application of Urea}\}
\]

Infer Information related to Farmer’s Context

For the instances found in previous steps, related context information is identified from the ontology by associating :context property. For classes of instance sets I and activityI, classes obtained from :context property are Control Measure, Disease and Pest. As per current implementation, :context information is related to temporal aspect and gives context of the crop related to its stage and ongoing season. Based on these, we obtain our set contextI.

\[
\text{contextI} = \{\text{Spotted Bollworm, Bacterial Blight, Tirak}\}
\]

Here, Spotted Bollworm is obtained as its season of attack is identified as 01-May to 31-Oct. Bacterial Blight and Tirak are considered as their stage of occurrence is All Stages
6.4 DESCRIPTION OF ALGORITHM

Infer Dependent Information

Dependent information is found for all the classes obtained as well as classes of the instances identified till now and stored in set dependentC. This is got from the :dependentClass property associated with these classes. For classes of instance sets I, activityI and contextI, classes obtained from :dependentClass property are Damage and Symptom. Hence,

\[ \text{dependentC} = \{\text{Damage, Symptom}\} \]

for the above mentioned case.

Classifying Nodes and Edges on Graph

For classification of nodes and edges on graph, set of selected nodes SN and selected edges SE are calculated. SN is set of classes on the generic ontology for which paths are to be calculated. It is union of classes identified earlier as C, classes of I, classes of activityI, classes of contextI and dependentC.

\[ \text{SN} = \text{C} \cup \text{classes of I} \cup \text{classes of activityI} \cup \text{classes of contextI} \cup \text{dependentC} \]

Hence, for the case defined we get SN set as,

\[ \text{SN} = \{\text{Damage, Part, Control Measure, Factor, Pest, Disease, Symptom}\} \]

Calculate Paths

Paths are calculated over the generic graph between each of the nodes in SN. Hence we get set of paths P as,

\[ P = \{\text{Damage is Caused By Pest, Damage is Caused By Factor may Lead To Disease, Disease causes Damage, Symptom is Symptom Of Disease occurs Due To Factor is Controlled By Control Measure, . . .}\} \]

Once we obtain paths from generic ontology, instance paths are calculated for these using SPARQL. Paths with known instances of classes are selected.

Eg. Path \{Damage is Caused By Pest\} would result in multiple instance paths like,

\{Leaf_Curling is Caused By Jassid\},
\{Stained_Lint is Caused By Red_Cotton_Bug\},
\{Boll_Shedding is Caused By Spotted_Bollworm\},

Out of these, only 3rd path is selected as instances of class Damage and Pest are known to be that. Similarly all other paths are calculated.

Some of the instance paths calculated are as below.
6.4 DESCRIPTION OF ALGORITHM

\{Boll\_Shedding is\_Caused\_By Lack\_of\_Irrigation may\_Lead\_To Tirak is\_Controlled\_By Proper\_Irrigation\},
\{Boll\_Shedding is\_Caused\_By Nitrogen\_Deficiency is\_Controlled\_By Application\_of\_Urea\},
\{Nitrogen\_Deficiency causes Boll\_Shedding\},
\{Buds\_Shedding is\_Caused\_By Nitrogen\_Deficiency is\_Controlled\_By Application\_of\_Urea\},
\{Flower\_Shedding is\_Caused\_By Lack\_of\_Irrigation may\_Lead\_To Tirak is\_Controlled\_By Proper\_Irrigation\},
\{Boll\_Shedding is\_Caused\_By Spotted\_Bollworm\},
\{Boll\_Shedding is\_Caused\_By Lack\_of\_Irrigation may\_Lead\_To Tirak has\_Symptom Black\_Boll\_Color\}

**Ranking of Paths**

Ranking of paths is done by taking into account several factors. First is the *Similarity Count* which states that paths which satisfy more user keyword are given more priority. For the above obtained paths, we see that path containing keyword *Boll\_Shedding* gets more priority over paths with keyword *Buds\_Shedding* and *Flower\_Shedding*.

Secondly, *Actionable Information* is considered where paths containing more actionable nodes gets a priority. Hence, for the paths above \{Boll\_Shedding is\_Caused\_By Lack\_of\_Irrigation . . . \} is prioritized over \{Boll\_Shedding is\_Caused\_By Spotted\_Bollworm\}.

And the third factor is *Dependent Information* where nodes which contain dependent class information gets a higher priority. For the above paths, \{Boll\_Shedding . . . Tirak has\_Symptom Black\_Boll\_Color\} gets priority over others.

We assign weightage for each of these factors with keyword match being the highest, followed by actionable information and last is dependent information. Hence, the above paths gets finally ranked as,

\{Boll\_Shedding is\_Caused\_By Nitrogen\_Deficiency is\_Controlled\_By Application\_of\_Urea\},
\{Boll\_Shedding is\_Caused\_By Lack\_of\_Irrigation may\_Lead\_To Tirak is\_Controlled\_By Proper\_Irrigation\},
\{Boll\_Shedding is\_Caused\_By Lack\_of\_Irrigation may\_Lead\_To Tirak has\_Symptom Black\_Boll\_Color\},
\{Boll\_Shedding is\_Caused\_By Spotted\_Bollworm\},
\{Buds\_Shedding is\_Caused\_By Nitrogen\_Deficiency is\_Controlled\_By Application\_of\_Urea\},
\{Flower\_Shedding is\_Caused\_By Lack\_of\_Irrigation may\_Lead\_To Tirak
6.5 Ontology Validation

Ontology validation will involve the practical experiences of the farmer being validated against the information given in the ontology. Information stored about the farmer in database mainly holds the activities performed by him, time of their occurrence, quantity or amount applied, if any, and the cost incurred in performing those activities. Parallel to this we have similar information stored in the ontology which conveys all of these parameters. Knowledge stored in the ontology is from the agro-experts which may or may not be the best for the crop. Hence we validate this with the actual experiences of the farmer.

The practices followed by the farmers are analyzed and compared with the activity information given in the ontology. Differences between the two are pointed out and seen whether maximum number of farmers follow a practice which is in accordance with the one suggested by the ontology or maximum perform a practice which violates the ontology information.

6.5.1 Steps for validation

The steps followed to perform ontology validation are as below.

**Input:** Scope of analysis -

- **Year of farming:** All crops in the year are selected and/or
- **District:** All crops in the district are selected and/or
- **Village:** All crops in the specified village are selected

**Output:** Result of analysis shows how the steps, actually performed by farmer, differs from those suggested in ontology. Steps considered for analysis are -

- **Days after sowing**
- **Average interval** between activities
- **Frequency** of performing an activity
- **Yield**

If values do not lie within some threshold of those in ontology, they are highlighted. The above analysis is done for each of the following activities -

- Irrigation, Hoeing, DAP Application, Urea Application, Potassium Application

1: Users selects the scope for which analysis needs to be performed; year-wise (or) district-wise (or) village-wise
2: All Crops sown during that period or place are selected
6.5 Ontology Validation

3: For each crop, following is calculated from database for each of the activity
   Days after sowing = First date of activity - Date of sowing
   Frequency = No. of times activity is performed
   Average interval = (Interval between all dates of activity)/Frequency
   Yield of crop

4: From the ontology, find
   Days after sowing = min(F(i,s)), where
     i = interval
     s = stage after which activity is performed
     F = it calculates the dates based on all the pairs of i and s given in ontology. Minimum of this date is selected. It signifies the date at which that activity should begin for the first time.
   Frequency = frequency property of activity from ontology
   Interval = interval property of activity from ontology

5: For each crop, values in databases are compared with those in ontology.
6: If values are above/below threshold, such differences are highlighted.
6.5 Ontology Validation

6.5.2 Observations

Figure 6.3: Observations for Ontology Validation
Chapter 7

Results

Search on farmer query performed can be compared on different parameters. Different results can be seen on varying these parameters. These are stated below:

(a) information and action

Information: Returns results of query which are information based and are same for all farmers

Action: Returns results which need context information for better answering

(b) context-aware and context-independent

Context-aware: Returns results specific to farmer taking context into account

Context-independent: Returns all possible information based results

(c) season specific

Season: Results are dependent on the ongoing season, i.e. current date

(d) stage specific

Stage: Results are dependent on current stage of the crop.

(e) activity specific

Activity: Form of context-aware query where results depend upon activity performed by farmer.

Example Query#1: Boll Shedding

Case 1:

Table 7.1: Parameter details - Example Query#1 - Case 1

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Information/Action</th>
<th>Context</th>
<th>Season</th>
<th>Stage</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>No</td>
<td>18-Jun</td>
<td>Flowering</td>
<td>No Irrigation</td>
<td></td>
</tr>
</tbody>
</table>
Results:

Boll_Shedding is_Caused_By American_Bollworm is_Controlled_By Acephate
Boll_Shedding is_Caused_By Tobacco_Caterpillar is_Controlled_By
   Thiodicarb
Boll_Shedding is_Caused_By Whitefly is_Controlled_By Triazophos
Boll_Shedding is_Caused_By Spotted_Bollworm is_Controlled_By Triazophos
Boll_Shedding is_Caused_By Tobacco_Caterpillar is_Controlled_By
   Chlorpyriphos
Boll_Shedding is_Caused_By American_Bollworm is_Controlled_By
   Chlorpyriphos
   ............

Results (As Tree):

|--Boll_Shedding
   |--is_Caused_By
      |--American_Bollworm
         |--is_Controlled_By
            |--Acephate
            |--Chlorpyriphos
            |--Triazophos
            |--Thiodicarb
            |--Quinalphos
            |--Endosulfan
      |--Tobacco_Caterpillar
         |--is_Controlled_By
            |--Thiodicarb
            |--Chlorpyriphos
            |--Quinalphos
            |--Acephate
      |--Whitefly
         |--is_Controlled_By
            |--Triazophos
            |--Ethion
      |--Spotted_Bollworm
         |--is_Controlled_By
            |--Triazophos
            |--Endosulfan
            |--Quinalphos

32
Case 2:

Table 7.2: Parameter details - Example Query#1 - Case 2

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Information/Context</th>
<th>Season</th>
<th>Stage</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Yes</td>
<td>18-Jun</td>
<td>Flowering</td>
<td>No Irrigation</td>
</tr>
</tbody>
</table>

Results:

Boll_Shedding is_Caused_By Lack_of_Irrigation may_Lead_To Tirak
is_Prevented_By Proper_Irrigation

Boll_Shedding is_Caused_By Lack_of_Irrigation may_Lead_To Tirak
has_Symptom Yellow_to_Red_Leaf_Color

Black_Boll_Color is_Symptom_Of Tirak occurs_Due_To Lack_of_Irrigation
causes Boll_Shedding

Bad_Boll_Opening is_Symptom_Of Tirak occurs_Due_To Lack_of_Irrigation
causes Boll_Shedding

Boll_Shedding is_Caused_By Spotted_Bollworm

Case 3:

Table 7.3: Parameter details - Example Query#1 - Case 3

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Information/Context</th>
<th>Season</th>
<th>Stage</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Yes</td>
<td>18-Jun</td>
<td>Flowering</td>
<td>No Irrigation</td>
</tr>
</tbody>
</table>

No Nitrogen
Results:

Boll_Shedding is_Caused_By Lack_of_Irrigation may_Lead_To Tirak
is_Prevented_By Proper_Irrigation
Boll_Shedding is_Caused_By Nitrogen_Deficiency is_Controlled_By
Application_of_Urea
Boll_Shedding is_Caused_By Lack_of_Irrigation may_Lead_To Tirak
has_Symptom Yellow_to_Red_Leaf_Color
Black_Boll_COLOR is_Symptom_Of Tirak occurs_Due_To Lack_of_Irrigation
causes Boll_Shedding
Bad_Boll_Opening is_Symptom_Of Tirak occurs_Due_To Lack_of_Irrigation
causes Boll_Shedding
Boll_Shedding is_Caused_By Spotted_Bollworm

.............

Case 4:

Table 7.4: Parameter details - Example Query#1 - Case 4

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Information/Action</th>
<th>Context</th>
<th>Season</th>
<th>Stage</th>
<th>Activity</th>
</tr>
</thead>
</table>
|            | Action            | Yes     | 01-Jul | Flowering| No Nitrogen
|            |                   |         |        |          | No Potassium No Phosphoros     |

Results:

Boll_Shedding is_Caused_By Nitrogen_Deficiency is_Controlled_By
Application_of_Urea
Tobacco_Caterpillar causes Boll_Shedding
Boll_Shedding is_Caused_By Spotted_Bollworm
Whitefly causes Boll_Shedding
Mealy_Bug causes Boll_Shedding

.............

In each of the above searches, results vary based on the scenario. Results are presented as paths. Also, they can be viewed as tree structure.

In Case 1, since context is not used, it becomes a context-independent search giving all possible results for keyword search performed.
In Case 2, it is a context-aware where farmer’s context information is used. In this case, farmer is lacking only irrigation. And hence he is provided with suggestions for performing irrigation for the symptoms observed by him.

In Case 3, farmer lacks both irrigation and nitrogen and hence he is advised to perform both.

In Case 4, results are varied because of the season. Mainly in July, pest attack is more. Hence, apart from nitrogen deficiency, attack of different pests is reported.

Example Query#2: Varieties of Cotton

Case 1:

Table 7.5: Parameter details - Example Query#2 - Case 1

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Information/Action</th>
<th>Context</th>
<th>Season</th>
<th>Stage</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Information</td>
<td>No</td>
<td>18-Jun</td>
<td>Flowering</td>
<td>No Irrigation No Nitrogen</td>
</tr>
</tbody>
</table>

Results:

Cotton has Variety PAU_626_H
Cotton has Variety LD_694
Cotton has Variety RCH_308
Cotton has Variety LH_1556
Cotton has Variety MRC_6304
Cotton has Variety MRC_6301
Cotton has Variety RCH_314

...............

Case 2:

Table 7.6: Parameter details - Example Query#2 - Case 2

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Information/Action</th>
<th>Context</th>
<th>Season</th>
<th>Stage</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Information</td>
<td>Yes</td>
<td>18-Jun</td>
<td>Flowering</td>
<td>No Irrigation No Nitrogen</td>
</tr>
</tbody>
</table>
Results:

Cotton has Variety PAU_626_H
Cotton has Variety LD_694
Cotton has Variety RCH_308
Cotton has Variety LH_1556
Cotton has Variety MRC_6304
Cotton has Variety MRC_6301
Cotton has Variety RCH_314

.............

This is an informative search where whether or not user selects context, query results are same.

Example Query#3: Round Patches on Leaf

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Information/ Action</th>
<th>Context</th>
<th>Season</th>
<th>Stage</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leaf_Shedding</td>
<td>Action</td>
<td>Yes</td>
<td>18-Jun</td>
<td>Emergence</td>
<td>No Potassium</td>
</tr>
</tbody>
</table>

Results:

Leaf_Shedding is Caused_By Leaf_Blight has Symptom Circular_Leaf_Spot
Yellow_to_Red_Leaf_Color is Caused_By Potassium_Deficiency may Lead_To
Leaf_Blight has Symptom Circular_Leaf_Spot
Circular_Leaf_Spot is Symptom Of Leaf_Blight is Controlled_By Mancozeb
Circular_Leaf_Spot is Symptom Of Leaf_Blight occurs Due To Potassium_Deficiency is Controlled_By Application_of_Potassium

.............

In the above query, use of WordNet is highlighted. Keywords “Round” does not occur in the ontology. Hence its synonym match “Circular” is found and selected as a user query keyword. Taking into account the potassium deficiency, results are returned appropriately.
Chapter 8

Conclusion and Future Work

System is designed which handles farmer’s crop related query where he can post his observations about the crop. The system handles this by searching a knowledge base ontology stored as a graph in database performing a graph based search. The search over ontology graph is aided by filtering information based on farmer’s context as well as weather predictions made. Finally, user is presented with set of paths from the ontology which best answer his query.

Inferred Data - Rules and Observations: The designed system answers the query in a better way by referring to farmer’s contextual information. The current system takes care of this as, each of the activity performed by the farmer is validated against the knowledge in the ontology and the farmer is recommended or warned suitably. It can be further extended by validating the farmer’s current activity with the activity performed for the same variety, at the same location and in the same season last year which gave the best yield. This would assist in confirming whether the farmer is following the practice which is best in his area.

Crop knowledge evolves from time to time by the new observations seen by farmers, impact of specific local conditions, experimentation done in consultation with agriculture research institutes, etc. This knowledge is not captured in the system requires ontology to be extended for conveying such knowledge.

Context - Based Spatial Search: Answering the user query by referring to farmer’s contextual information currently deals with the temporal aspect of the activities performed by him. Suggestions are made to the farmer by verifying the stage of the crop, the time for performing certain activities, time of attack of pests, time of occurrence of a disease, time of performing control measures. Apart from looking into the temporal part, spatial aspect of it can also be considered. This would require finding relatedness between the farmer’s crop and the one’s sown in his nearby villages. Some of the scenarios like a disease affecting a crop would lead farmers in that location to be warned prior the occurrence of the disease as it can spread in their fields too. Also other cases would be that if a farmer has his field near a tubewell, he can make use of it or we could suggest him a way of irrigation through that.
Appendix

System Interface Details

Ontology Keyword Search

Farmer ID: 10
Crop ID: 16608
Enter query: Boll Shedding

Search Results

RESULTS BASED ON QUERY SEARCH

- 0.69 - Boll_Shedding is Caused By American_Bollworm is Controlled By Acephate
- 0.69 - Boll_Shedding is Caused By Tobacco_Caterpillar is Controlled By Thiodicarb
- 0.69 - Boll_Shedding is Caused By Whitefly is Controlled By Triazophos
- 0.69 - Boll_Shedding is Caused By Spotted_Bollworm is Controlled By Triazophos
- 0.69 - Boll_Shedding is Caused By Tobacco_Caterpillar is Controlled By Chlorpyriphos
- 0.69 - Boll_Shedding is Caused By American_Bollworm is Controlled By Chlorpyriphos
- 0.69 - Boll_Shedding is Caused By American_Bollworm is Controlled By Triazophos
- 0.69 - Boll_Shedding is Caused By American_Bollworm is Controlled By Thiodicarb
- 0.69 - Boll_Shedding is Caused By Tobacco_Caterpillar is Controlled By Quinalphos
- 0.69 - Boll_Shedding is Caused By Whitefly is Controlled By Ethion

Figure A1: Results - Example Query#1 - Case 1
Ontology Keyword Search

Enter query: Boll Shedding

Search Results

DETAILS OF ACTIVITIES DONE FOR CROP MRC6301 SOWN ON
DATE 2012-04-15

- Nitrogen is fine. Recommendation: Next date for Nitrogen suggested is 2012-06-23
- Phosphorus is fine. Recommendation: Next date for Phosphorus suggested is 2012-06-30
- Potassium is fine. Recommendation: Next date for Potassium suggested is 2012-06-23
- Lack of Irrigation at the stage of Flowering

RESULTS BASED ON QUERY SEARCH

- 0.886 - Boll Shedding is Caused By Lack of Irrigation may Lead To Tirak is Prevented By Proper Irrigation
- 0.815 - Boll Shedding is Caused By Lack of Irrigation may Lead To Tirak has Symptom Yellow to Red Leaf Color
- 0.815 - Black Boll Color is Symptom Of Tirak occurs Due To Lack of Irrigation causes Boll Shedding
- 0.815 - Bad Boll Opening is Symptom Of Tirak occurs Due To Lack of Irrigation causes Boll Shedding

Figure A2: Results - Example Query#1 - Case 2
Ontology Keyword Search

Farmer ID: 15

Crop ID: 16609

Enter query: Boll Sheding

[ ] Use Context

Submit

Search Results

DETAILS OF ACTIVITIES DONE FOR CROP RCH154 SOWN ON
DATE 2012-04-15

- Phosphorus is fine. Recommendation: Next date for Phosphorus suggested is 2012-06-30
- Potassium is fine. Recommendation: Next date for Potassium suggested is 2012-06-22
- Lack of Irrigation at the stage of Flowering
- Lack of Nitrogen at the stage of Flowering

RESULTS BASED ON QUERY SEARCH

- 0.886 - Boll Shedding is Caused By Lack of Irrigation may Lead To Trak is_Prevented By Proper Irrigation
- 0.833 - Boll Shedding is Caused By Nitrogen Deficiency is Controlled By Application of Urea
- 0.815 - Boll Shedding is Caused By Lack of Irrigation may Lead To Trak has_Symptom Yellow to Red Leaf Color
- 0.815 - Black_Boll_Color is_Symptom Of Trak occurs Due To Lack of Irrigation causes Boll Shedding
- 0.815 - Bad_Boll_Opening is_Symptom Of Trak occurs Due To

Figure A3: Results - Example Query#1 - Case 3
Ontology Keyword Search

Farmer ID: 35
Crop ID: 16606
Enter query: Boll Shedding

Use Context

Submit

Search Results

DETAILS OF ACTIVITIES DONE FOR CROP MRC6301 SOWN ON DATE 2012-04-15

- Irrigation is fine. Recommendation: Next date for irrigation suggested is 2012-07-06
- Lack of Nitrogen at the stage of Flowering
- Lack of Phosphorus at the stage of Flowering
- Lack of Potassium at the stage of Flowering

RESULTS BASED ON QUERY SEARCH

- 0.833 - Boll_Shedding is Caused By Nitrogen_Deficiency is Controlled By Application_of_Urea
- 0.638 - Tobacco_Caterpillar causes Boll_Shedding
- 0.638 - Boll_Shedding is Caused By Spotted_Bollworm
- 0.638 - Whitefly causes Boll_Shedding
- 0.638 - Mealy_Bug causes Boll_Shedding
- 0.546 - Buds_Shedding is Caused By Nitrogen_Deficiency is Controlled By Application_of_Urea
- 0.351 - Jassid causes Leaf_Shedding
- 0.351 - Tobacco_Caterpillar causes Flower_Shedding

Figure A4: Results - Example Query#1 - Case 4
Ontology Keyword Search

Farmer ID: 15
Crop ID: 16609
Enter query: Varieties of Cotton

Submit

Search Results

RESULTS BASED ON QUERY SEARCH

As Paths  As Tree
0.381 - Cotton has Variety PAU_626_H
0.381 - Cotton has Variety LD_694
0.381 - Cotton has Variety RCH_308
0.381 - Cotton has Variety LH_1856
0.381 - Cotton has Variety MRC_6304
0.381 - Cotton has Variety MRC_5301

Figure A5: Results - Example Query#2 - Case 1
Ontology Keyword Search

Farmer ID: 75

Crop ID: 16609

Enter query: Varieties of Cotton

Use Context

Submit

Search Results

DETAILS OF ACTIVITIES DONE FOR CROP RCH134 SOWN ON 
DATE 2012-04-15

- Phosphorus is fine. Recommendation: Next date for Phosphorus suggested is 2012-06-30
- Potassium is fine. Recommendation: Next date for Potassium suggested is 2012-06-22
- Lack of irrigation at the stage of Flowering
- Lack of Nitrogen at the stage of Flowering

RESULTS BASED ON QUERY SEARCH

As Paths  © As Tree
- 0.381 - Cotton has Variety PAU_626_H
- 0.381 - Cotton has Variety LD_694
- 0.381 - Cotton has Variety RCH_308
- 0.381 - Cotton has Variety LH_1556

Figure A6: Results - Example Query#2 - Case 2
Ontology Keyword Search

Farmer ID: 

Crop ID: 16605

Enter query: Round patches on leaf

Use Context

Submit

Search Results

DETAILS OF ACTIVITIES DONE FOR CROP RCH134 SOWN ON
DATE 2012-06-07

- Irrigation is fine. Recommendation: Next date for Irrigation suggested is 2012-06-27
- Nitrogen is fine. Recommendation: Next date for Nitrogen suggested is 2012-07-17
- Phosphorus is fine. Recommendation: Next date for Phosphorus suggested is 2012-08-06
- Potassium not done. Recommendation: Perform Potassium

RESULTS BASED ON QUERY SEARCH

As Paths © As Tree
- 0.435 - Leaf_Shedding_is_Caused_By_Leaf_Blight has_Symptom Circular_Leaf_Spot
- 0.415 - Yellow_to_Red_Leaf_Color_is_Caused_By_Potassium_Deficiency may_Lead_To Leaf_Blight has_Symptom Circular_Leaf_Spot
- 0.336 - Circular_Leaf_Spot is_Symptom_Of Leaf_Blight is_Controlled_By Mancozeb
- 0.317 - Circular_Leaf_Spot is_Symptom_Of Leaf_Blight occur_Due_To

Figure A7: Results - Example Query#3
References


REFERENCES


[16] Thanh Tran, Philipp Cimiano, Sebastian Rudolph, and Rudi Studer. Ontology-Based Interpretation of Keywords for Semantic Search. pages 523–536. 2008. 4
