Abstract — In this paper, we propose to develop an educational e-counseling system which will enable students to find the best available option according to their needs. In this e-counseling system, the student gets the freedom to specify his choice in a simplified query-like natural language, to which a very accurate and specific result is made available to him, irrespective of his physical location. Initially, the Ontology is created by assimilating data from various web resources by assigning hierarchy to the concepts and defining relationships between them. The natural language query goes as an input to the parser which produces the dependencies in the sentence which is then used to construct the dependency tree. The dependency tree goes as an input to mapping code, which employs a particular algorithm to address simple “which” and “what” type queries. The purpose of the system is to make e-counseling a better experience for the students and to help them out in selecting an appropriate career option with as much machine support as possible.

Keywords — e-counseling, Ontology, Semantic Web, Natural Language Query, Parser.

I. INTRODUCTION

A. Academic Counseling system

An academic counseling system assists students in selecting a career option for them according to their field of interest, choice of subjects, and other factors including financial capabilities. In such a system, the interested student approaches the counselor in person, who then helps the student choose among the various available options.

In a general e-counseling system, students enter their choice of course and field of interest online, but all in a very restricted manner, mostly by filling up forms in an interface. This restricts the free expression of queries and hence the students are unable to explore the limitless possibilities which the web offers. With the amount of data that the web hosts, a properly designed query answering system can serve a lot of purposes. Here comes the need to enrich the existing system with a better organized, classified storage of information and a logical query answering system so that the huge amount of resources available can be properly exploited.

B. Semantic Web

With the great success of the World Wide Web and the increasing amount of online resources, the total amount of unorganized data increases manifold and a huge portion of this data is interpreted manually by humans with little or no machine support [1-2]. While the developments around web services provide the underlying architecture, there is another field which promises nothing less than the ‘next generation’ of the web and is gaining momentum: it is a web for machines, the Semantic Web. In a Semantic web, the data is organized in such a fashion that it is machine processable and is not just for direct human consumption.

Semantic Web is a group of methods and technologies to allow machines to understand the meaning - or "semantics" - of information on the World Wide Web. It is the idea of having data on the Web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications. In order to make this vision a reality for the Web, supporting standards, technologies and policies must be designed to enable machines to make more sense of the Web, with the result of making the Web more useful for humans.

C. Ontology

Ontology is a formal explicit specification of a shared conceptualization. Ontologies form the backbone of the Semantic Web; it facilitates machine understanding of linked information resources. The various annotations on the semantic web forms links between resources on the Web and connects them to formal terminologies and these connective structures are called Ontology[2-3].

Formally, Ontology describes the concepts and relationships that are important in a particular domain, providing a vocabulary for that domain as well as a computerized specification of the meaning of terms used in the vocabulary. Ontologies range from taxonomies and classifications, database schemas, to fully axiomatized theories. In recent years, ontologies have been adopted in many business and scientific communities as a way to share, reuse and process domain knowledge. Ontologies are now central to many applications such as scientific knowledge portals, information management and integration systems, electronic commerce, and semantic web services.

Sreeja Golui is with the National Institute of Technology, Durgapur, India, phone: (+91)9734722665; fax: (+91)3432547375; e-mail: sreejagolui1990@gmail.com.
Eshnna V P Ekka is with the National Institute of Technology, Durgapur, India; e-mail: eshnna@gmail.com.
Nidhi Jain is with the National Institute of Technology, Durgapur, India; e-mail: nidhi.jain311089@gmail.com.
Animesh Dutta is with the National Institute of Technology, Durgapur, India; e-mail: animeshnit@gmail.com.
Basic Model of a Semantic Web

The semantic web comprises the standards and tools of XML, XML schema, RDF, RDF Schema and OWL that are organized in the Semantic Web Stack. The OWL Web Ontology Language Overview describes the function and relationship of each of these components of the semantic web.

- XML provides an elemental syntax for content structure within documents, yet associates no semantics with the meaning of the content contained within[13]
- XML Schema is a language for providing and restricting the structure and content of elements contained within XML documents.
- RDF is a simple language for expressing data models, which refer to objects ("resources") and their relationships. An RDF-based model can be represented in a variety of syntaxes, e.g., RDF/XML, N3, Turtle, and RDFa. RDF is a fundamental standard of the Semantic Web [14].
- RDF Schema extends RDF and is a vocabulary for describing properties and classes of RDF-based resources, with semantics for generalized-hierarchies of such properties and classes.
- OWL adds more vocabulary for describing properties and classes: among others, relations between n classes ("e.g. disjointness"), cardinality ("e.g. "exactly one""), equality, richer typing of properties, characteristics of properties ("e.g. symmetry"), and enumerated classes.

II. RELATED WORKS

A lot of work has been previously done in the field of web semantics and Ontology. Semantic web concepts have been used in developing a system for tele-assisting of elderly people [4]. This system, called AINGERU, works anywhere and anytime, is capable of monitoring vital signs and generates alarms when necessary, and allows physicians and concerned relatives to consult data about them through the web. Semantic web technologies has been used to make match making softwares [8]. Match making services are heavily based on string matching. Using formal ontologies and automated reasoning services, the entire system can be improved to meet the expectation of its users.

A lot work is done to improve the existing information systems by incorporating the concepts of semantic web and Ontology, leading to the perspective of Ontology driven information systems [7].

Ontology has immense application in the field of medical science and a lot of new systems have been developed which uses Ontology as its basic architecture. An Ontology based Holonic Diagnostic System (OHDS) [6] has been developed which combines the advantages of holonic paradigm with multi agent system technology and Ontology design, for the research and control of unknown diseases in robust manner. There has been various other works in the field of biological sciences and medical research like the Ontology based negotiation of dental therapy [5]. The entire field of bioinformatics and gene studies is based on Gene Ontology (GO), which involves classifying and structuring the gene set into various controlled vocabularies called GO terms [15]. There are several Gene Ontology tools which are available on the web, like AmiGO, FatiGO, etc [16].

There are a number of other applications of Ontology including e-learning systems [9-10] which includes virtual classrooms, remote courses and distance learning. There are e-government services [11] which integrates various government services across different service providers.

III. PROBLEM OVERVIEW

A general e-counseling system does not allow natural language queries to be made; such a system uses form based syntactic matching with the existing relatively unorganized database. Hence the various queries of the students remain unattended and this greatly reduces the efficiency of the system. The student is left with the task of extracting the required data from the list of answers made available to him after the straight forward syntactic matching. The database, in this case, doesn’t have any machine understandable information and does not use any external logic to answer the queries. This system greatly limits the far reaching possibilities which could be achieved from properly organizing the vast amount of available resources.

IV. SCOPE OF WORK

In a general e-counseling system, form based queries are accepted. Students enter their choice of subject and field of interest in a restricted form based pattern which inhibits the free expression of queries, reducing the efficiency of the counseling system. If the student is left with the task of extracting his required answer from a list of syntactically matched answers made available to him, then a part of the counseling is being done by the student itself with no reasoning support from the machine. And nowadays, with the amount of data available online and the huge number of career options possible, it is rather a rigorous task for the student to manually scan all the options given to him.

The proposed system gives a human-like reasoning capacity to the whole system and hence reduces manual intervention.
The student now can frame his query in a natural language and get the desired result. This increases the degree of interaction between the machine and the user, making the entire system of counseling more user friendly where a wide variety of queries can be entertained instead of simple form based entries. It is the task of the system to understand the queries and then use the linking of the data to give the desired result to the user.

The proposed idea exploits the immense amount of web resources available by first organizing the data into a hierarchy and defining relationships (properties) between various classes of data which can later be used for logical linking and semantic search of the Ontology. Hence the task of the user is further simplified and a part of the reasoning is being done by the computer itself. This also gives the user the freedom to enter a wide variety of queries which will be interpreted by the machine. The matching will be done semantically, where a particular term will represent a concept and any other term with the same meaning will refer to that concept. For example, there won’t be any difference between “Computer Science” and “Computer Science and technology”, the machine will be capable of understanding that both refer to the same concept. Hence, the e-counseling experience won’t be a form filling session but would rather be a natural language query-answer system providing the user with better results.

V. THE SYSTEM ARCHITECTURE

![Diagram of the system architecture]

The system architecture, depicted in Fig 1, shows the various components of the entire system which are described as follows:

**Interface** - This is a web based interface which is instrumental in interacting with the end user by taking the input for further processing. The query is in natural language.

**Parser** - The Parser parses the Natural Language Query and works out the grammatical structure of the sentence; it groups together words which form phrases and resolves the dependencies between the various parts of speech. The output from the parser, which is the analysis of the sentence, is stored in a text file.

**Mapping Code** - This code maps the output of the parser with the Ontology, which forms the knowledge base of the whole system. The result of the parser is obtained in a tree form or in plain text form expressing the structural dependencies between the parts of speech in the query. This is matched with the underlying ontology, with the noun being matched with the concepts (classes) and the dependencies between the words being matched with the various properties defined in the Ontology.

**Ontology** - Ontology serves as the reservoir of all information. Creation of the Ontology would require arrangement and logical linking of data along with its annotations, assigning of various hierarchies of classes and memberships, creating relationships (properties) between individuals and the exceptions in properties etc. The proper organization of all the available data is crucial to the functioning of the entire system.

VI. IMPLEMENTATION AND TOOLS

The Ontology is created in an open source tool called Protegé, the mapping code has been done in C#, and the parsing has been done with the natural language query parser (Stanford parser).

The implementation of the system requires the following steps:

1. Creation of Ontology, which requires the assimilation of data from various resources.
2. Assigning of hierarchy to the various classes, assigning of membership to various individuals, defining relationships (data type as well as object type properties) between individuals, determining exceptions etc.
3. The parser, which will parse the query and will store the output in a text file. The text file will contain structural information about the sentence, namely the dependencies.
4. This text file will be read as an input to the mapping code.
5. The code will map the nouns with the concepts and its annotations, and the dependencies between the nouns with the relations (defined as properties in the Ontology). After a match is found, the individuals participating in the relationship will be returned.
6. The results will be shown to the user in the interface.
The mapping algorithm for simple “what” and “which” type questions:
In this algorithm we assume that each path to the leaf node has the potential to represent a condition or a relation on a concept. All the individuals satisfying these conditions are potential answers to the query and the final answer is obtained by filtering the results according to user requirements, by taking into account various conjunctions between phrases.

**Alg:**

**Input:** The tree constructed from the dependencies in the query

**Output:** An objective answer for simple “which” and “what” queries.

**Data Structure used:**

- **C:** To store the concept.
- **IN:** A two dimensional string array where each row stores the individual strings satisfying the condition in the corresponding path (equal to the row number).
- **SS:** Symbol stack to store the “˄” and “˅” operations.
- **R:** An array of strings to push the relations.
- **IN_FINAL:** Final expression between the IN and SS.

**Step 1:** Identify all the paths to leaf nodes.

**Step 2:** For each path, starting from the left, repeat till the leaf node is reached.

**Step 3:** Find the first node that matches a concept or its annotation in the ontology and save it in C.

**Step 4:** If the next immediate node (NIN) = “AND”, then push ˄ in the symbol stack. If NIN = “OR”, then push ˅ in the symbol stack in the ith position.

**Step 5:** Match the next immediate node (NIN) with a relation or its substring or with its annotations or substring of annotations that already has members of array R as a substring.

**Step 6:** If a match is found in step 5, push it in R.

**Step 7:** If a match is not found, then match the NIN with a relation or its substring or with its annotations or substring of annotations that has individual elements of R (as a substring) in reverse order.

**Step 8:** Repeat step 4 and 5 till leaf node is reached.

**Step 9:** Check in the Ontology whether any member of C shows property R. Save the members in IN (i) where i = the ith path.

**Step 10:** Repeat step 2 to 8.

**Step 11:** Repeat till IN is empty

**Step 12:** Pop the members index wise in IN and apply the operation from the symbol stack (if any) at the corresponding position to form the final expression, IN_FINAL.

**Step 13:** End loop in 10.

**Step 14:** Compute IN_FINAL and return the result.

**Example:** What are the colleges that offer computer and has hostel facility?

The tree constructed from the dependencies would be

![Tree Diagram](image)

The above tree shows one of the paths in red to the leaf node: Root->are->colleges->has->facility->hostel->10K

There are 6 paths to the leaf nodes:

- Path 1. Root->colleges->what
- Path 2. Root->colleges->are
- Path 3. Root->colleges->the->
- Path 4. Root->colleges->offer->that
- Path 5. Root->colleges->offer->computer
- Path 6. Root->colleges->offer->computer->and
- Path 7. Root->are->colleges->has->facility->hostel

For each of these paths, the nodes are scanned in order to come across the first node that matches a concept (or its annotation) in our Ontology. Supposing that our Ontology has a concept called “colleges”, Path 1 encounters a concept at level 2. SS is null in position 1. In Path 2, “colleges” is found at level 3, followed by which there is no “AND” or “OR”, so the condition in step 4 of the algorithm does not hold true for path 2. It might find a match with any substring of a relation (property) with the next word (node) “are”. The loop ends here and the Ontology is searched for individuals or members from the class “Colleges” with a relation “are” on it. Supposing the Ontology doesn’t find any member having the relation “are” on it, it moves on to the next path. The SS for path 2 is also null. Similar scenario is seen in Path 3. In path 4, the procedure repeated with “colleges” being identified as the first concept, followed by a relation “offer” on it. Let us suppose that “offer” matches with a property or a substring of a property. Then the next node, “that”, might not match with any property with substring “offer” or even if it matches with a substring of a property having “offer” as a substring, the path ends and no complete recognizable relation is found in the Ontology. The SS for path 4 is null since condition in line 4 doesn’t hold. For path 5, after coming across “colleges” and after finding a match with a substring of a property with “offer”, it tries to match “computer” with all the relations having substring “offer”. The leaf node is reached and considering that it finds a match with a relation or its annotation (which already has a substring “offer”) and
recognizes “offercomputer” as a valid relation, the loop ends and it checks in the Ontology whether any individual from the class “colleges” participates in this relation. Let us suppose 4 colleges participate in the relation, namely A, B, C, and D. IN[5] will have A, B, C, and D. SS for path 4 is null since it fails step 4. In path 6, after “colleges” is identified, it comes across “AND” in level 5 and satisfies step 4 and it pushes “∧” in the SS at position 6. In path 7, a situation similar to path 5 arises and assuming that it finds “hasfeesbelow10K” as a valid property in the Ontology, individuals from “colleges” satisfying this property is searched. Let us suppose that colleges A, E, C, and F participate in this relation. IN [7] has A, E, C, F. The outer loop ends here since all the paths are traversed.

Now IN is scanned. IN[1], IN[2], IN[3], IN[4] is empty and SS[1], SS[2], SS[3], SS[4] is also empty. IN[5] is retrieved followed by SS[5], which is empty. Then comes IN[6] and followed by this comes SS[6] which is “∧”. So the expression till now is IN[5] ∧. After this IN[6] is retrieved and SS[6] is empty. So final expression is IN[5] ∧ IN[6] which gives us the final result. In this example the final result is \{A,B,C,D\} ∧ \{A,E,C,F\} which is \{A,C\}.

A. Protégé

Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data [12].

Further, Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications [17].

B. Stanford Parser

A natural language parser is a program that works out the grammatical structure of sentences, for instance, which groups of words go together (as "phrases") and which words are the subject or object of a verb. Probabilistic parsers use knowledge of language gained from hand-parsed sentences to try to produce the most likely analysis of new sentences. These statistical parsers still make some mistakes, but commonly work rather well [18] [19].

This package is a Java implementation of probabilistic natural language parsers, both highly optimized PCFG and lexicalized dependency parsers, and a lexicalized PCFG parser. The lexicalized probabilistic parser implements a factored product model, with separate PCFG phrase structure and lexical dependency experts, whose preferences are combined by efficient exact inference, using an A* algorithm. Or the software can be used simply as an accurate unlexicalized stochastic context-free grammar parser. Either of these yields a good performance statistical parsing system. A GUI is provided for viewing the phrase structure tree output of the parser.

REFERENCES

[1] Dr. Dragan Gašević School of Computing and Information Systems Athabasca University, Dr. Dragan Djuric, University of Belgrade FON – School of Business Administration Department of Information Systems, Prof. Vladan Devedžić University of Belgrade School of Business Administration, Model Driven Engineering and Ontology Development, second edition, Springer.


[6] Maja Hadzic1, Mihaela Ulieru2, Elizabeth Chang1, Ontology based Holonic Diagnostic System (OHDS) for the Research and Control of Unknown Diseases, Curtin University of Technology School of Information Systems, Australia.

Sreeja Golui An 8th semester student pursuing B.Tech at National Institute of Technology, Durgapur, India in the field of Information Technology.

Eshnna V P Ekka An 8th semester student pursuing B.Tech at National Institute of Technology, Durgapur, India in the field of Information Technology.

Nidhi Jain An 8th semester student pursuing B.Tech at National Institute of Technology, Durgapur, India in the field of Information Technology.

Animesh Dutta Received B.E. in Computer Science and Engineering from Regional Engineering College, Durgapur, India and M.Tech from National Institute of Technology, Durgapur, India in the year 2002 and 2006 respectively. He is currently with the department of Information Technology of National Institute of Technology, Durgapur, India. He is pursuing his Phd work in the area of Software Engineering and Distributed Computing in the department of Computer Science and Engineering at Jadavpur University, Kolkata, India.