AUTOMATIC RULE RETRIEVAL FROM WEBSITES USING ONTOLOGY AND TEXT MINING

D. Santhakumar #1 , S. Rajasekar #2
# M.Tech, Department of CSE, CK College of Engineering & Technology, Cuddalore, India
* M.Tech, Department of CSE, CK College of Engineering & Technology, Cuddalore, India

Abstract: A rule-based system like an intelligent service comparing portal may compare product prices, shipping options, refund options etc., Such rule based system requires an automatic knowledge acquisition procedure from the Web that consists of unstructured texts. Knowledge acquisition can be carried out by ontology acquisition and rule acquisition. Obtaining information such as product prices from web content can be done using ontology and text mining. Rule acquisition typically carried out by knowledge experts as well as domain experts, hence very time consuming and tedious. Acquiring rules from unstructured sources such as texts from web are very tedious and hence lacks an automated procedure. To overcome this problem, taking the advantages of ontology this paper propose an approach to generate rules automatically from similar websites by using rule ontology and improving rule extraction using text mining. We use two main steps of rule acquisition, which consists of rule component identification and rule composition with the identified rule components. Here, we identify rule components such as variables and values in Web pages by using RuleToOnto in the first step, and we combine the variables to compose rules in the second step. To improve the rule component identification step from text obtained from web sites we use parser to represent grammatical and syntactic structures, pruning to filter low-frequency words so that fewer but more informative words remain in the final solution vector and dependency analysis to find the relationships between words and apply them to acquiring variable and value relationships.

Index Terms— Ontology, RuleToOnto, Rule Based Systems, Semantic Web.

I. INTRODUCTION

The Semantic Web, which is the key component of Web 2.0 and Web 3.0, is an evolving development of the World Wide Web [7] in which the semantics of information and services on the Web are being defined. This is enabling the Web to understand and satisfy the requests of people and machines to use the Web content. Knowledge is an essential part of most Semantic Web applications and ontology, which is a formal explicit description of concepts or classes in a domain of discourse, is the most important part of the knowledge. However, ontology is not sufficient to represent inferential knowledge. This is because ontology-based reasoning has limitations compared with rule-based reasoning, even though ontology-based reasoning [8] with description logic is a popular issue of the Semantic Web. That is, inferential rules are also the essential part of knowledge of the Semantic Web. SWRL is a proposal for a rule representation standard based on ontology. Many attempts have been made at knowledge acquisition in order to obtain enough knowledge for Semantic Web applications. Ontology learning, which refers to extracting conceptual knowledge from several sources and building an ontology from scratch, enriching, or adapting an existing ontology, is one of the attempts at knowledge acquisition. Most ontology learning approaches acquire knowledge from the Web, because it offers a large amount of valuable information for every possible domain. Rule acquisition is as essential as ontology acquisition, even though rule acquisition is still a bottleneck in the deployment of rule-based systems. This is time consuming and laborious, because it requires knowledge experts as well as domain experts, and there are communication problems between them. However, sometimes rules have already been implied in Web pages, and it is necessary to acquire them from Web pages in the same manner as ontology learning.

II. BACKGROUND

A. Ontology basics

In recent years the development of ontologies—explicit formal specifications of the terms in the domain and relations among them (Gruber 1993)—has been moving from the realm of Artificial-Intelligence laboratories to the desktops of domain experts. Ontologies have become common on the World-Wide Web. The ontologies on the Web range from large taxonomies categorizing Web sites (such as on Yahoo!) to categorizations of products for sale and their features (such as on Amazon.com).

An ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them.

Following are the reasons for using ontology:

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge

Often an ontology of the domain is not a goal in itself. Developing an ontology is akin to defining a set of data and their structure for other programs to use. Problem-solving methods, domain-independent applications, and software
agents use ontologies and knowledge bases built from ontologies as data[10].

B. Rule-based Systems

In computer science, rule-based systems are used as a way to store and manipulate knowledge to interpret information in a useful way [9]. They are often used in artificial intelligence applications and research.

C. Applications

A classic example of a rule-based system is the domain-specific expert system that uses rules to make deductions or choices. For example, an expert system might help a doctor choose the correct diagnosis based on a cluster of symptoms, or select tactical moves to play a game. Rule-based systems can be used to perform lexical analysis to compile or interpret computer programs, or in natural language processing. Rule-based programming attempts to derive execution instructions from a starting set of data and rules. This is a more indirect method than that employed by an imperative programming language, which lists execution steps sequentially.

D. Construction

A typical rule-based system has four basic components:

- A list of rules or rule base, which is a specific type of knowledge base.
- An inference engine or semantic reasoner, which infers information or takes action based on the interaction of input and the rule base. The interpreter executes a production system program by performing the following match-resolve-act cycle
- Match: In this first phase, the left-hand sides of all productions are matched against the contents of working memory. As a result a conflict set is obtained, which consists of instantiations of all satisfied productions. An instantiation of a production is an ordered list of working memory elements that satisfies the left-hand side of the production.
- Conflict-Resolution: In this second phase, one of the production instantiations in the conflict set is chosen for execution. If no productions are satisfied, the interpreter halts.
- Act: In this third phase, the actions of the production selected in the conflict-resolution phase are executed. These actions may change the contents of working memory. At the end of this phase, execution returns to the first phase.
- Temporary working memory.
- A user interface or other connection to the outside world through which input and output signals are received and sent.

III. LITERATURE SURVEY

In this paper [3], they present an unsupervised algorithm, DIRT, for Discovery of Inference Rules from Text. Algorithms for finding similar words assume the Distributional Hypothesis, which states that words that occurred in the same contexts tend to have similar meanings. Instead of applying the Distributional Hypothesis to words, we apply it to paths in dependency trees. Essentially, if two paths tend to link the same sets of words, we hypothesize that their meanings are similar. Since a path represents a binary relationship, we generate an inference rule for each pair of similar paths. [4] TEASE Present a fully unsupervised learning algorithm for Web-based extraction of entailment relations. The algorithm takes as its input a verb lexicon and for each verb searches the Web for related syntactic entailment templates. Algorithms for paraphrase acquisition address two problems: (a) finding matching anchors and (b) identifying template structure. The sentences “Yahoo bought Overture” and “Yahoo acquired Overture” share the anchors {X=Yahoo, Y=Overture}, suggesting that the templates ‘X buy Y’ and ‘X acquire Y’ paraphrase each other.[5] The eXensible Rule Markup Language (XRML) approach is a framework for extracting rules from texts and tables of Web pages. The core of the XRML framework is rule identification, in which a knowledge engineer identifies various rule components such as variables and values from the Web pages with a rule editor. The effectiveness of the rule acquisition procedure of the XRML approach depends on the rule identification step, which also depends on the large amount of manual work done by the knowledge engineer. Park and Lee [6] tried to reduce the burden on the knowledge engineer for manual identification. They defined an ontology that includes information on rule components and structures acquired from other sites in the same domain. By using the ontology, the XRML approach could be extended to automatically identify rule components such as variables, values, and rules from Web pages. However, the procedure of making rules with identified variables and values was far from automatic.

IV. PROPOSED METHOD

Rule acquisition consists of rule component identification and rule composition from the acquired rule components[1]. In order to automatically acquire rules through ontology, we divided the rule acquisition procedure into two main steps in order to apply proper methods to each step. In the rule component identification step, we identify variables and values by using an ontology that describes frequently used variables and values in other rule bases. Subsequently, we compose rules from the identified rule components by using the rule structures of the ontology. The ontology helps to recommend feasible rules with variables.

To improve the rule component identification step from text obtained from web sites we use grammatical and syntactic structures [2] in the rule identification step. For example, we will be able to detect negations in the text and add them to matching variables if we use grammatical analysis. Moreover, we use dependency relationships, we can find the relationships between words and apply them to acquiring variable and value relationships.
In step 1, RuleToOnto is generated from rules which are acquired in another site. In step 2, variables and values are automatically identified from the Web page using RuleToOnto and the first rule draft is generated. In step 3, rules are automatically composed by combining the identified variables and values.

A. Proposed Architecture

We developed A* algorithm for this purpose. However, the generated rules may be incomplete. Therefore, the knowledge engineer needs to refine the second rule draft to make it complete in step 4. Following section describes the different steps of our proposed work:

A. Ontology Creation

RuleToOnto is domain specific knowledge that provides information about rule components and structures. It is possible to directly use the rules of the previous system instead of the proposed ontology. However, it requires a large space and additional processes to utilize information on rules, while RuleToOnto is a generalized compact set of information for rule acquisition. While the rule component identification step needs variables, values, and the relationship between them, the rule composition step requires generalized rule structures.

Therefore, RuleToOnto represents the IF and THEN parts of each rule by connecting rules with variables with the IF and THEN relations, in addition to basic information about variables, values, and connections between variables and values.

The RuleToOnto schema has three object properties HasValue, IF and THEN, and three classes, Variable, Value, and Rule, as shown in Fig. 3, which is an RDF graph generated from the OWL ontology by the RDF validator. In order to utilize ontology inference, we added some axioms. First, something is a Variable precisely if all the values of the hasValue property are instances of the Value class, as shown in the following axiom represented in the Functional-Style syntax:

![RuleToOnto Instances](image)

![Figure 1 Ruletoonto Instances](image)

B. Optimization Of Rule Identification

- Pre-Processing

Preprocessing is the initial step of text processing and it consists of standard routines such as removal of non-alphabetic characters and mark-up tags, case folding, elimination of stopwords, and stemming. Stoplists are used for automatic removal of uninformative words which causes a significant reduction in the number of features that have to be stored. In this work, the list of 571 stopwords of the Smart system was used for this
purpose. Stemming was implemented using the Porter stemmer which is one of the most experienced stemmers for word forms.

•  Syntactic Support
The parser identifies the dependencies in the sentence in two phases. In the first phase, the sentence is parsed using a statistical phrase structure parser. The part-of-speech tags of the tokens, the semantic heads in the sentence, and the dependents of the heads are identified. In the second phase, the dependencies extracted are labeled with grammatical relations using the tree-expression syntax. Stanford Parser is known to be one of the most powerful and efficient parsers having the least error rate. Given a sentence, the parser identifies the dependencies in the sentence in two phases. In the first phase, the sentence is parsed using a statistical phrase structure parser based on a probabilistic context-free grammar (PCFG), which was trained on the Penn Wall Street Journal treebank. The part-of-speech tags of the tokens, the semantic heads in the sentence, and the dependents of the heads (auxiliaries, complement, etc.).

•  Pruning
Pruning is used in order to filter low-frequency features so that fewer but more informative features remain in the final solution vector.

•  Dependency Analysis
To find the relationships between words and apply them to acquiring variable and value relationships Table 1 shows the complete list of dependency types accompanied with their definitions and some examples.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ncomp</td>
<td>Adjectival complement</td>
<td>Terminal, nontan etc.</td>
</tr>
<tr>
<td>adv</td>
<td>Adverbial clause modifier</td>
<td>Quickly, open, able plane</td>
</tr>
<tr>
<td>agent</td>
<td>Agent</td>
<td>Approve, bank, approach, sectors</td>
</tr>
<tr>
<td>noun</td>
<td>Adjective modifier</td>
<td>Scientific, study, principal investigator</td>
</tr>
<tr>
<td>app</td>
<td>Appositive modifier</td>
<td>Monitoring, detection, eigenvalue, separation</td>
</tr>
<tr>
<td>state</td>
<td>Attribute complement</td>
<td>Remote, year, paycheck, paycheck</td>
</tr>
</tbody>
</table>

Table 1

C. Identifying Variables

Rule component identification is to elicit variables and values by comparing parsed words of the given text with the variables and values of RuleToOnto. In the first step, we expanded RuleToOnto by adding synonyms of each term using WordNet. In the comparison between the terms of RuleToOnto and the terms of the Web page, we used semantic matching instead of simple string comparison. In order to find the semantic similarity between two terms, we use the hyponym structure of WordNet. The similarity measure is a reciprocal number of the distance between two terms in the hyponym hierarchy in WordNet, as given by the following equation:

\[ \text{Semantic Similarity} = \frac{1}{\text{path length}} \]

This measure is calculated only when one term is a hyponym of the other term, and the path length is the path length between the two terms in the hyponym hierarchy. We decided that two terms are semantically related when the measure is larger than 0.25.

D. Compose Rules And Refine Rules

The objective of rule composition is to combine identified variable instances into rules. There are several possible variable instances for one variable on a Web page. The first step of rule composition is the preparation step, where we find appropriate rules from RuleToOnto. This is done by comparing the identified variable instances with the variables of the rules in RuleToOnto. The input of this step is the variable instances of rule draft 1 that is an output of the rule component identification step and RuleToOnto.

The next step is rule ordering, which generates RuleOrder from the variable instances and the candidate rules. The third step is variable ordering, which generates TotalOrder with the RuleOrder and VariableOrder that is calculated in
this step. The last step is best-first search that makes rule
draft 2.

1 Initialization:
2 choose candidate rules;
3 rule ordering;
4 variable ordering within each rule;
5 firstVIs = instances(firstVar(TotalOrder));
6 OPEN = firstVIs; CLOSED = {};
7 begin
8 repeat
9 choose a currentVI from OPEN having the lowest f-value;
10 if nextVar(currentVI, TotalOrder) is empty then
11 finished = true; result = path(currentVI);
12 else
13 begin
14 transfer currentVI from OPEN to CLOSED;
15 update TotalOrder with variable ordering;
16 construct nextVIs such that
17 nextVIs = instances(nextVar(currentVI, TotalOrder)) -
    path(currentVI);
18 set OPEN = OPEN ∪ nextVIs;
19 end;
20 until finished;
21 output result;
22 end;

Figure 3 Breadth First Search
The Initialization step includes choosing candidate rules,
rule ordering, and variable ordering. The purpose of getting
firstVIs at line 5 is to make an initial OPEN set for the first
loop of the algorithm. In this step, we extract variable
instances that match the first variable of TotalOrder(RC) and
put them into OPEN. The first step of the loop at line 9 is
choosing the most suitable variable instance for the current
variable from OPEN with the evaluation function f(n). The
loop ends if the current variable is the last in
TotalOrder(RC), because it means the variable instances are
assigned to every variable in TotalOrder(RC). The function
nextVar (currentVI, totalOrder) at line 10 gets the next
variable of currentVI from TotalOrder(RC). It is empty
when we assign all variable instances to the variables in
TotalOrder(RC). If the loop stops, the result is the path
from one of the firstVIs to the last variable instance
currentVI at line 11. If currentVI is not the last variable, we
transfer it from OPEN to CLOSED at line 14, and update
TotalOrder(RC) at line 15, because assigning currentVI can
change TotalOrder. TotalOrder(RC) is determined by
integrating RuleOrder and VariableOrder which are
calculated with Count(). The value of Count() changes while
we assign variable instances to the rule candidates.
Therefore, we update TotalOrder(RC) by recalculating
Count(), VariableOrder, and RuleOrder. Subsequently, we
get the next variable from totalOrder(RC) with the function

V. DISCUSSION
The following figure5 shows the graph view in protégé tool
of RuleToOnto Ontology used by our approach. Figure 6
represents the Rule ordering and variable ordering output.
VI. CONCLUSION

Inferential rules are as essential to the Semantic Web applications as ontology. However, rule acquisition research is relatively unpopular while there are many works on ontology learning. This paper proposed an approach to develop an automatic rule acquisition procedure using an ontology, named RuleToOnto, that includes information about the rule components and their structures.

VII. REFERENCES


