SPARQL for Semantic Information Retrieval from RDF Knowledge Base

Gunnam Swathi¹, S. Mahaboob Hussain², Prathyusha Kanakam³, D. Suryanarayana⁴

¹PG Scholar, CSE, Vishnu Institute of Technology, Bhimavaram, A.P, India
²#3Assistant Professor, CSE, Vishnu Institute of Technology, Bhimavaram, A.P, India
⁴Professor, CSE, Vishnu Institute of Technology, Bhimavaram, A.P, India

Abstract—This paper proposes a SPARQL query language to obtain accurate results to the end users. Semantic search expects to contribute more accurate results than the present-day keyword search. However, progress with semantic search delayed due to the complexity of its query languages. SPARQL is used to extract the data from the RDF knowledge base that gives useful information to the user. In this paper, authors explore an approach of adapting keywords to querying the semantic web. This approach translates the natural language keyword queries into formal queries to perform the semantic search so that end users can retrieve semantic results. Therefore, this paper explicitly explains about the retrieving technique in a semantic approach from the natural language to the various constrained formal SPARQL queries to retrieve the relevant and accurate information.

Keywords — NLP, information retrieval, semantic Web, SPARQL, ontology, RDF, triplets, semantic search.

I. INTRODUCTION

SPARQL is an RDF query language, it is a semantic query language for databases, and it is used to retrieve the data stored in RDF format [1]. SPARQL allows queries to consist of triple patterns, conjunctions, disjunctions, and optional patterns.

Extraction of data as RDF subgraphs, URIs, blank nodes, typed and untyped literals with aggregate functions, subqueries, complex joins property paths exploration of data via the query for unknown relations. The transformation will do from one RDF data vocabulary into another. Construction of new RDF graphs based on RDF query graphs [2].

A query distributed over different SPARQL endpoints implementations for multiple programming languages appears. Already existed tools allowed to connect and semi-automatically to construct an SPARQL query for an SPARQL endpoint. The use of this exist tools is to translate SPARQL queries to other query languages.

RDF is a format for representing data about resources and based on a graph that represents subject and object nodes that are associated and they related to predicate represented by arcs. RDFs are written in XML or in triplet format. RDF schema is a method for defining RDF files. It allows RDF resources grouped into classes, subclass, sub property and domain, range descriptions are specified. SPARQL is a query language for RDF, it provides a standard format for writing queries for target RDF data and set of standard rules are processed for those queries and return the results. SPARQL is not directly based on XML, it will not follow the syntax conventions. Names that begins with a? or $ are the variables. Graph patterns are represented in a list of triple patterns enclosed within braces { }. The variables named after the SELECT keyword are variables that will return the results [3].

II. PRELIMINARIES

This process involves converting the terms of a natural language sentence to an intake of keyword query to semantic search to overcome the following reasons.

1) Vocabulary Gap: Most of the Web users do not have experience about the ontology, so the words in their queries will be little different from those in the ontology [4].

2) Relation: Relations between concepts are required to be formal queries, which are missing in keyword queries. The Web users are not having knowledge that how to discover automatically these missing relations becomes is a big problem [5].

SPARQL (pronounced "sparkle", a recursive acronym for Simple Protocol and RDF Query Language) is an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format. It is a standard by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium (W3C).

SPARQL language is based on RDF Turtle serialization and basic graph pattern matching. A graph pattern matching is a triple pattern (Subject, Property and Object) i.e., an RDF triple that contains variables at any point.

Triple Pattern = Turtle + Variables.

III. WORKING METHODOLOGY

For defining graph patterns, at first triple patterns need to define: A triple pattern is like RDF triple, but there is the option of a variable in place of RDF...
terms (i.e., IRIs, literals or blank nodes) in the subject, predicate or object positions.

For example, consider the statement: "John plays Cricket".

The subject of the statement is: John
The predicate is: plays
The object is: Cricket

Accordingly, the triplets will be represented as,
<Subject: John, Predicate: plays, Object: Cricket>

As per [6], there are different types of graph patterns existed and these group graph patterns examined with examples. The more general graph patterns that they build out of these following,
- Basic graph patterns
- Filter conditions
- Optional graph patterns

A. Basic graph patterns

A basic graph pattern (BGP) is a set of triple patterns and written as a sequence of patterns. BGP is easily understood by the users as the conjunction of its triple patterns. { ?x rdfs:name ?name. ?x rdfs:mbox ?mbox } A graph pattern is a set of graph patterns delimited by braces { }.

For example, consider the following group patterns which are equivalent.
{l }{ ?x rdfs:name ?name . ?rdfs:mbox ?mbox . }
{l }{ ?x rdfs:name ?name . ?x rdfs:mbox ?mbox . }
{l }{ ?x rdfs:name ?name . }
{l }{ ?x rdfs:mbox ?mbox . }

When a group graph pattern consists of only triple patterns or BGP, these patterns interpreted, combinable and the group graph pattern is same they are relevant set of triple patterns.

Null set {} is the empty group graph pattern.

Group graph patterns are the general type of graph patterns; they are involved in other constructs. These constructs are introduced by certain keywords. There is no keyword for the conjunction to form a group graph pattern.

B. Filter Condition

The FILTER construct restricts variable bindings to the filter expression it evaluates to TRUE. FILTER constraints use Boolean conditions to filter out undesired query results [7]. The shortcut for using a semicolon (;) and it can be used to separate two triple patterns that share the same subject.

\[
\text{Select } ?\text{thing} ?\text{Age\_Limit} \text{where } \{ \\
\text{thing a dbo:GovernmentExams} . \\
\text{thing tto: Age\_Limit} ?\text{Age\_Limit} . \\
\text{FILTER } (?\text{Age\_Limit} >21 \text{&& } ?\text{Age\_Limit} < 26) \text{order by } ?\text{Age\_Limit}
\]

C. Optional Graph Pattern

Regular, complete structures are not assumed in all RDF graphs. It is available, but the response cannot be rejected because some part of the query pattern may not match. Optional graph pattern matching provides the facility that is if the optional part does not match; it will not create any bindings but does not eliminate the solution. Optional parts of a graph pattern that are tried to evaluate will be specified by starting with a graph pattern P1 and then it will apply the keyword OPTIONAL to another graph pattern P2 that follows [P1] OPTIONAL [P2].

For example, perceive the following query.
\[
\text{SELECT } ?\text{twelth} ?\text{courses WHERE } \{ \\
?\text{twelth a dbo:Twelth} . \\
?\text{twelth rdf:type bo:Twelth} . \\
\text{OPTIONAL } (?\text{twelth a courses ?courses }) \\
\}
\]

IV. SPARQL QUERIES FOR INFORMATION RETRIEVAL

In the case of queries that will read the data from the database, the SPARQL language determines four different query forms for different purposes.

A. SELECT Query

It is used to extract values from an SPARQL endpoint, the results are returned in a table format.

\[
\text{SELECT ?Courses ?Duration ?Jobs WHERE } \{ \\
?\text{thing a ont:Bachelor_of_Arts} . \\
?\text{thing ont:Courses ?Courses} . \\
?\text{thing tto:Duration ?Duration} . \\
?\text{thing tto:Jobs ?Jobs} . \\
\}
\]

TABLE I

RESULTS OBTAINED FROM SELECT QUERY

<table>
<thead>
<tr>
<th>Courses</th>
<th>Duration</th>
<th>Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master of Arts</td>
<td>2.0</td>
<td>Teacher , Bank Jobs , Defense</td>
</tr>
<tr>
<td>Advertising and Commercial Diploma</td>
<td>2.0</td>
<td>Digital Marketing , Creativity</td>
</tr>
<tr>
<td>Event Management Diploma</td>
<td>2.0</td>
<td>Event Manager, Marketing</td>
</tr>
<tr>
<td>Bachelor of Journalism</td>
<td>1.0</td>
<td>Press sub-editor, Broadcast</td>
</tr>
<tr>
<td>M.B.A</td>
<td>2.0</td>
<td>Business Analyst, Financial</td>
</tr>
<tr>
<td>Bachelor of Law</td>
<td>5.0</td>
<td>Lawyer, Judiciary Editing</td>
</tr>
</tbody>
</table>

From the Table I, the semantic results obtained for the given query is in specific format with courses, duration and jobs and classified according to the values in triplet form as (subject, predicate, object).

B. CONSTRUCT Query

It is used to extract information from the SPARQL endpoint and transform the results into valid RDF.

\[
\text{CONSTRUCT } \{ ?\text{twelth a ont:Twelth} \} \\
\text{WHERE } (?\text{twelth rdf:type ont:Twelth} .)
\]

ISSN: 2231-5381  http://www.ijettjournal.org  Page 352
C. ASK Query

It is used to provide a simple True or False result for a query on an SPARQL query. The ASK query is used to test whether a graph pattern has a solution or not.

\[
\text{ASK} \{ \text{<http://example.org/tuto/ontology#>} \}
\text{tto:duration} \ ?BArch. \\
\text{<http://example.org/tuto/ontology#>}\text{tto:duration} \ ?BTech. \\
\text{FILTER(} \ ?BArch> \ ?BTech \text{)}\}
\]

TABLE IV 
RESULTS OBTAINED FROM DESCRIBE QUERY

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science &amp; Engineering</td>
<td>Jobs</td>
<td>Software Engineer, System Analyst, Assistant Professor, Software Tester, Bank jobs</td>
</tr>
<tr>
<td>Electrical Engineering</td>
<td>Jobs</td>
<td>Development Engineer, Embedded Trainer</td>
</tr>
<tr>
<td>Chemical Engineering</td>
<td>Jobs</td>
<td>Scientist, Professor, Trainee Software, Sales manager</td>
</tr>
<tr>
<td>Electronics Communication Engineering</td>
<td>Jobs</td>
<td>Hardware Networking Engineer, Telecom Engineer Graphic Designer</td>
</tr>
<tr>
<td>Mechanical Engineering</td>
<td>Jobs</td>
<td>Space Instrument Engineer, Machine Design Engineer</td>
</tr>
<tr>
<td>Civil Engineering</td>
<td>Jobs</td>
<td>Site Engineer, Project Engineer</td>
</tr>
</tbody>
</table>

All of these query forms takes a WHERE block to restrict the query, for that case of the DESCRIBE query, the WHERE is optional. There is also a DESCRIBE query form which is not important and SPARQL does not prescribe any semantics for it.

V. RESULTS AND DISCUSSIONS

This paper works on SPARQL queries to overcome the gap between formal language semantic search and the end users to built a semantic search engine [8]. The query imposed on SPARQL endpoint using user interface because it may have different RDF or DBpedia graphs and another more data sets. Then, specify a query in WHERE clause. After the WHERE clause, it has curly braces and within the braces, specify basic triple patterns or graph patterns of the query. Then required results will be the output. SPARQL is based on RDF and basic graph pattern matching. Various types of queries are written in order to retrieve the semantic results.

For example, the query for enquiring for courses and their colleges is interpreted as "country,degree:colleges?colleges". The SPARQL query is applied to obtain the name of the author and the work done along with its published date.

It will be written in SPARQL as follows:

\[
\text{PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>} \\
\text{PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>} \\
\text{PREFIX db: <http://dbpedia.org/ontology>} \\
\text{PREFIX dbp: <http://dbpedia.org/property/>} \\
\text{SELECT ?author ?work ?date} \\
\text{FROM <http://dbpedia.org/>} \\
\text{WHERE} \{ \\
\text{?author rdf:type:db:Writer} . \\
\text{?author db:notableWork ?work .} \\
\text{?work dbp:releaseDate ?date } \\
\text{\} ORDER BY ?date}
\]

The following results will be obtained as shown in the Table I for the above query.

GROUP graph patterns are used to minimize the filter conditions. A FILTER condition is a constraint on solutions for the total group in which filter appears. For instance, consider the query:

\[
\text{Select ?thing ?duration where } \{ \\
\text{?thing a db:Degree} . \\
\text{?thing tto:duration ?duration .} \\
\text{FILTER ( ?duration > 1 \&\& ?duration < 7.0) } \\
\text{order by ?duration}
\]

On existing RDF database by applying the above FILTER query the following results will occur.
TABLE VI
RESULTS OBTAINED AFTER APPLYING THE FILTER QUERY

<table>
<thead>
<tr>
<th>Degree</th>
<th>Duration (in years)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.Tech</td>
<td>4.0</td>
</tr>
<tr>
<td>L.L.B</td>
<td>2.0</td>
</tr>
<tr>
<td>M.B.B.S</td>
<td>5.0</td>
</tr>
<tr>
<td>B.Pharmacy</td>
<td>4.0</td>
</tr>
</tbody>
</table>

The alternative graph patterns that the two graphs are combined by UNION (a binary operator) and they are processed independently of each other and the results are combined using a UNION. One has to be careful whether they have to use the same variables or not in each alternative. The query will be in the form as shown below.

```
SELECT ?duration
WHERE {
    [ttr:DegreeEngineering:duration ?duration.]
    UNION
    [ttr:Merchantnavytto:duration ?duration.]
}
```

TABLE VII
RESULTS OBTAINED FROM ASK QUERY

<table>
<thead>
<tr>
<th>Duration (in years)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
</tr>
<tr>
<td>3.0</td>
</tr>
</tbody>
</table>

Turtle is used to present the data and syntax also used in SPARQL. SELECT and WHERE clauses are used in SQL. But, the fact is, ha the SPARQL and SQL are very different languages. Variables are like in Prolog or Datalog. In this case, variables as $x$ instead of $\exists x$ will be written. The query will be written as SELECT * as in SQL. The query result is a set of bindings the variables will appear in the SELECT clause and these results are exhibited in tabular form.

VI. CONCLUSION

This paper works on the transformed natural language query to the SPARQL query form, i.e., the format which can understand by the machine to retrieve the relevant results from the existed database. The natural language query translated into triplets as Subject, Object, Predicate and these triplets are used to form the SPARQL query to retrieve the accurate results. In this paper authors explains the applications of the variety of SPARQL queries to apply and retrieve the semantic results from the knowledge base of RDF.

Regular Web users may not have specific knowledge about the ontology as the words in their queries may be quite different from those in the ontology [9]. For that reason, SPARQL query is used in the representation of triplets for accurate results. By applying and experimented with these formatted queries most of the results are satisfied and hence the authors retrieved relevant results rather than the links provided by the traditional search engines.

ACKNOWLEDGMENT

This work has been funded and supported by the Department of Science and Technology (DST), Govt. of India, under the Grants No. SRC/CSI/153/2011.

REFERENCES