Efficient Retrieval of Data Using Semantic Search Engine Based on NLP and RDF

Usha Yadav[1,2,*] and Neelam Duhan[2]

[1]*National Institute of Fashion Technology, Jodhpur, India*
[2]*J.C. Bose University of Science & Technology, YMCA, Faridabad, India*
E-mail: Usha.yadav.912@gmail.com
[*]*Corresponding Author*

Received 19 November 2020; Accepted 15 August 2021;
Publication 27 October 2021

Abstract

With the evolution of Web 3.0, the traditional algorithm of searching Web 2.0 would become obsolete and underperform in retrieving the precise and accurate information from the growing semantic web. It is very reasonable to presume that common users might not possess any understanding of the ontology used in the knowledge base or SPARQL query. Therefore, providing easy access of this enormous knowledge base to all level of users is challenging. The ability for all level of users to effortlessly formulate structure query such as SPARQL is very diverse. In this paper, semantic web based search methodology is proposed which converts user query in natural language into SPARQL query, which could be directed to domain ontology based knowledge base. Each query word is further mapped to the relevant concept or relations in ontology. Score is assigned to each mapping to find out the best possible mapping for the query generation. Mapping with highest score are taken into consideration along with interrogative or other function to finally formulate the user query into SPARQL query. If there is no search result retrieved from the knowledge base, then instead of returning null to the user, the query is further directed to the Web 3.0. The top "k" documents are

Journal of Web Engineering, Vol. 20_8, 2285–2318.
doi: 10.13052/jwe1540-9589.2084

considered to further converting them into RDF format using *Text2Onto* tool and the corpus of semantically structured web documents is build. Alongside, semantic crawl agent is used to get <Subject-Predicate-Object> set from the semantic wiki. The *Term Frequency Matrix* and *Co-occurrence Matrix* are applied on the corpus following by singular Value decomposition (SVD) to find the results relevant for the user query. The result evaluations proved that the proposed system is efficient in terms of execution time, precision, recall and f-measures.

Keywords: Domain ontology, semantic search engine, SPARQL, natural language processing, RDF.

1 Introduction

Data over the web is growing exponentially since last decade, so as linked Open data. There are hundreds of SPARQL end points are available, which can provide access to trillions of triples across multiple domains such as movie, sports, commerce, science, technology etc. [1]. The vision of semantic web is to improve its quality and realization of its full potential to the end user. However, providing easy access of this enormous knowledge base to all level of users is challenging. The ability for all level of users to effortlessly formulate structure query such as SPARQL is very diverse.

Specific purpose interfaces using domain based knowledge base are usually appropriate to use for their simplicity and homogeneity nature are known as ***Knowledge Base Specific Interface***. Majority of the web based form search comes under this category. These interfaces are designed to cater to the most specific and relevant user queries, which are known to system in advance. Other drawbacks of using such systems are the effort needed for specific interface development and its rigidity due to change in schema.

Faceted browsing technique offers users with the limited facets based on the available resources they are exploring. As this technique is knowledge base independent, existing SPARQL endpoints require small or no adjustment on top of it. The major limitation of this technique is that it restricts the users with limited set of queries. For example, it is easier to search for objects specific to class *Student*, but it lacks to handle complex queries such as *Student who studies in school in London*, it is because of the fact that restriction *in London* related to the school not the class student. Although, there are few tools such as Visual SPARQL Query builder [2] which provides ease in

generating SPARQL queries, still they are not in reach to common users and usually knowledge developers and engineers are their target users. For using such tools, common users still require understanding of how SPARQL works and the construct which needs to be used in formulating the queries and some knowledge of the core schema.

Although Question Answering system allows users to query his questions directly such as Ginseng [3], NLP-Reduce [4], still they need to be confined to a particular domain using models or patterns. The limitation of such system is that if the user query belongs to cross domain, it could be very difficult to handle without taking feedback from users. To retrieve the most relevant answer, Linear SVC has been applied to classify answers along with the selection technique such as univariate and chi-square for feature space reduction [5].

Nowadays, many domain dependent search systems use domain specific ontologies as their backbone for creating knowledge base. Using ontologies based data repositories over traditional relational database management systems has lots of advantages. Ontologies can structurally store the concepts and maintaining the semantic relationship among them instead of using table and mapping to store the concepts. Using semantic web standard language such as RDF and OWL, more semantic queries could easily be processed. Structured Query Language (SPARQL) is a standard which is used for semantically querying the resource in RDF/OWL format. There are some challenges faced by common user in understanding the SPARQL query and framing their requirements as per their need. Although it is again challenging to understand the user query in natural language and interpreting into the SPARQL query for semantically searching the resources and returning the precise and accurate results to the users.

It is very reasonable to presume that common users might not possess any understanding of the ontology used in the knowledge base or SPARQL. Interpreting the requirement directly into SPARQL is indeed very challenging for the common user [6, 7]. Therefore, it is highly required to provide alternate solutions to overcome the problem of semantically searching resources. Various works have proposed to deal with such issues by creating user friendly interfaces or developing new techniques [6–8]. The leading techniques provides the system [6, 8] which assists users in creating queries like SPARQL without perquisite knowledge of SPARQL syntax.

The increasing number of devices allowing voice search is very common nowadays. User simply put the queries using voice search in natural language, without being informed about its underlying schema or SPARQL syntax.

In contrast to traditional searching which uses keywords to retrieve relevant results, semantic web search engines should handle the ambiguity of queries put using natural language, to return precise search results.

There are few challenges which need to be resolved for better and precise search results. Mapping query words to the proper resources in any domain specific ontologies has some limitations. There could be case of synonyms; homonyms etc. which could map query words to number of resources in the ontology, ending up interpreting very different query.

Another challenge is in understanding the users' perspectives, considering they are unaware about the underlying domain ontology schema. Triples are the subject, predicate and object statement about a resource which are used to construct SPARQL queries along with where clause. The triple graph is used to find the matching graph in the ontology. Sometimes, user queries do not provide complete information for generating the complete graph pattern. Mostly a missing predicate in user query can affect the search results and greatly increase the probability of getting the irrelevant results. For examples, the user put a query, *Kangana Ranaut movie*, the predicate or the relation combining the two concepts are missing. Does the user want to search for movies acted by *Kangana Ranaut* or directed by *Kangana Ranaut*, it is very challenging to deal with missing predicates. Even if users are made familiar with the underlying schema of domain ontology, it would not be much helpful. The schema of ontologies is regularly updated and extended or could be merged with other ontologies.

Motivation: The traditional search retrieval algorithms of Web 2.0 become outdated in retrieving the precise and accurate information from the growing Web 3.0, known as Semantic Web. The more advanced and strategic techniques compliant with the semantic web would be highly needed to find the desired information from evolving semantically linked data. The semantically enriched information is the need of the hour, it is vital to make this information from semantic web accessible to the general user. It is logical to assume that the general user lacks in understanding the ontology structure of the system or structured language to retrieve data such as SPARQL. Interpreting the requirement directly into SPARQL is indeed very challenging for the common user. Also, knowledge base in some domain specific application gradually extended and updated. Therefore, there might be cases in which there is no relevant information present in the KB in accordance with user query, in such scenarios, alternative methodologies need to be adapted to retrieve the relevant information from other sources.

Contributions: Below mentioned are the key contributions included in this work.

(i) Automated formulation of SPARQL queries from the given user query in natural language.
(ii) To reduce ambiguity and improving the top search results, weightage score assigned to each mapping between the user query word and the resource in the ontology.
(iii) Formulation of *Term Frequency matrix* and *Co-occurrence matrix* for the RDF triples to improve the context relevance issue and further decomposing it to generate priority vector.
(iv) *Multiphase Search approach*: In many domain specific applications which deals with handling natural language user queries often resulted into *null search results found issue*, therefore in this approach, another search phase to redirect user query further over the semantic structured web is devised to overcome this issue.

Organization: Rest of the paper is organized as follows: Related work in the area of semantic search engine and converting user query in natural language to SPARQL provided in Section 2. Section 3 provides overview of the proposed system architecture. Sections 4 and 5 explains about the phase 1 and phase 2 of the proposed semantic search engine respectively. Section 6 described all the experiments conducted and its result analysis. Finally, the conclusion is presented in Section 7.

2 Related Work

It is very challenging for the common user to explore RDF based Linked Data or knowledge base in an effective manner. Also, it is not expected from the user to be well versed in writing SPARQL or to know about the underlying ontology structure. Various related research works are presented in this section.

Researchers in [9] proposed Semantic Focused Crawler to overcome the mentioned limitations. It is graph based interface for querying using Subject-Predicate-Object format. The auto-complete feature of query builder assists users in choosing the domain ontology related entities without requiring them to have prior knowledge about ontology or SPARQL. In various scenarios specially, user query may span to more than one domain which bringing a necessity for developing cross domain ontology for better search results

or recommendation. Authors in [10], the online reviews provided by the users are analyzed using various natural language processing for developing domain ontologies. Further, new alignment technique is devised to form cross ontology by aligning the various domain ontologies.

A novel systematic method is proposed to understand natural language question rather than using semantic parsers. Large numbers of low budget binary templates were created automatically. Efficient indexing was used to facilitate better searching over template decomposition. Two level disambiguation strategies was designed and performed namely, 'entity level ambiguity' and 'structure level ambiguity'. Researchers in [11] proposed a framework which assist users in translating question in Natural Language into structured query for a specific domain knowledge based system. The new graph structure, vocabulary and the semantic query graph has been defined to handle the complexity of compounded questions. The sub graph in the knowledge base has been identified based on the query expansion and its semantic graph generation. The sub graph generated is directly converted into structure query language.

Researchers in [12], Proposed OSCAR, the Open Citations RDF Search Application, provides user friendly interface by hiding the complexity of writing SPARQL query by the common user. It provides access to any RDF triple store with easy to use SPARQL endpoint. Authors in [13] identified the fragment of first order logic which capture the underlying structure of the user query by formulating the faceted search interface. The complexity while answering such queries is well studied for RDF / OWL formats. An efficient algorithm for interface generation and updating was also proposed. System was also tested for scalability with relevant results.

Researchers in [14], extended the natural language pattern of user query by assigning a node known as 'Query Focus' for further matching it semantically. To find out the top diversified 'k' matches with the 'query focus', an efficient technique was proposed, thereby querying the knowledge graph by user query in natural language. Many researches allowed user to ask query in natural language with domain restricted vocabulary [15, 16] . Authors in [17] proposed SQUALL structure to take user query as input and it is similar to natural language. The query has been directly mapped to the SPARQL based upon semantic and syntactic analysis without requiring resource mapping process.

Authors in [18] proposed Semantic Supported Information Retrieval System (SIRS), it is ontology based and the input query is processed using Probabilistic Latent Semantic Indexing (PLSI) algorithm. This work also

concentrates on providing the personalized web search using *Multi-Criteria Particle Swarm Optimisation* (MCPSO) for handling personal interest of user. Traditional content based web page recommendation system had not utilized the power of semantic knowledge in discovering the patters and recommendations. Opinion based sentiment analysis approach could be utilized in Collaborative Filtering for providing personal recommendations [19]. Also, researchers in [20] proposed using the semantic knowledge in all phases of data mining. Sequential Pattern mining algorithm, *CloSpan* was used to create frequent sequential patterns over semantic space. Semantically enriched pattern was generated, further in offline mode, it is provided to the web page recommendation process for better results.

To improve the quality of search results, web database need to be enriched and semantic knowledge base should be created. Authors in [21] framework to rank web pages was proposed known as ONTOPARK, which is based on ontology. In this work, the Vector Space Model has been combined with ontology for Information Retrieval. In this framework, RDF knowledge base was created by annotating the RDF files semantically for each query. Researchers in [22] the three layer architecture known as SRD-CP was proposed. In the first layer, document domain was decided by constructing semantic tree pattern based on RDF. The second layer handles the processing of complex queries using constrained application protocol and HTTP protocol. The third layer uses the generated SPT to include the word co-occurrence with the help of association frequency. *Apriori algorithm* was used to find the association between the documents on web.

Human lacks in refining through large web page result retrieved by tradition search engine for a query, instead machines can able to process the required information efficiently. Machines lacks in understanding the underline structure and also the context of the user query, which requires users to brainstorm within large set of results to find the desired one. The semantic power of Google search engine improve on the search strategy to produce relevant search results to the users based on creating rules that define a user's intent and the contextual meaning of search terms. The schema.org [23] is a set of vocabulary based on standard syntax allowing used widely to allow seamless generation and integration of data. The knowledge graph developed by Google [24] gained popularity as it represents the facts structurally using entities, relationships and semantic descriptions. The formal specifications may be used for inference and interpretation over facts [25].

Many researchers presented an approach to rank the web pages based on its context sensitivity. Domain related ontology was used to generate the

ranking factor for web pages [26]. The data properties on a web page were analyzed with respect to the domain ontology, and higher is the number of data properties inside the web document, higher would be its ranking factor value. Author in [27] worked on retrieving the web documents while enabling the effortless integration of data from multiple sources and also on reducing inconsistencies. An efficient web search engine framework was proposed to accurately fulfilling the user requirements by enabling the multiple data sources integration during retrieval of search results web pages. The tabular comparison of some of the related research work is presented in the Table 1.

It is observed that most of the semantic search engines are facing challenges such as handling very specific or limited set of queries, effort required to develop specific interfaces, inability to handle complex, null search results and cross domain queries etc. In order to solve the gap stated in the references, the proposed model focuses on handling user queries based on ontology term mapping and path generation through weightage assignment technique, multilayer search approach and overall improved the efficiency taking advantage of semantic wiki and *co-occurrence matrix*.

3 Multiphase Semantic Search Engine Framework

The architecture is divided into two phases, Retrieval from domain KB and retrieval from Web, each of the phases is explained in Figure 1.

Phase 1: (Result Retrieval from created Domain Knowledge base)

1. In this phase, preprocessing is applied on the query submitted by the user in natural language. The preprocessing involves sentence segmentation, tokenization, and stop words removal, determining each token's part of Speech (POS) and conducting lemmatization. As a result, a set of query terms $QT = \{t1, t2 \ldots, tn\}$, which retains the input order, is retrieved.
2. The token generated are analyzed for interrogatives such as *when, which, who* and for functions such as *how many, max*.
3. The interrogatives and the function terms are kept aside to be used while creating SPARQL structure. Finally, set containing user query terms are generated.
4. Each term in the query is mapped with the most similar resource in the ontology using proposed *Term Ontology Mapping Technique*.
5. A weightage function is proposed to assign weight to each of the mapping done in previous step.
6. SPARQL is constructed based on the finding of above steps.

Table 1 Comparison among related research work

Publication	Query language	Scalability	Domain/Queries for Evaluation	Ontology Based	Technique Used
OTNLS [28] – Ontology-Based Translation of Natural Language Queries to SPARQL	Natural Language sentence	Yes	Static Queries, Event Occurrence queries, Turbine background queries, KPI queries	Yes	Transforming NLP sentence in SPARQL. RDBMS, RDF and triple store are analyzed for efficiency, accuracy and data storage
UPSIR [18] – A personalised user preference and feature based semantic information retrieval system in semantic web search	User query	Yes	11 queries from four Fields namely Food, Education, Health care and News	Yes	Semantic Supported Information Retrieval System (SIRS), using Probabilistic Latent Semantic Indexing (PLSI) and *Multi-Criteria Particle Swarm Optimisation* (MCPSO) for handling personal interest
SQG_NLS [11] – Semantic query graph based SPARQL generation from natural language questions	Natural Language Questions	No	QALD, WebQuestions and Free917	No	Translate NLP questions into structured query for a specific domain KB based on new graph structure to handle the complexity
IWSF [27] – An intelligent web search framework for performing efficient retrieval of data.	Query Terms	Yes	5 queries on Galaxy Search, AOL Search, City Search and Ask	No	retrieving the web documents based on enabling the effortless integration of data from multiple sources
SWQ_RDF [22]– A Semantic Web query Optimization Using Resource Description Framework	RDF query	Yes	Reuters-21578 Text Categorization Collection Data Set from UCI repository	No	Three layer architecture known as SRD-CP was proposed based on HTTP protocol, association frequency, Apriori algorithm

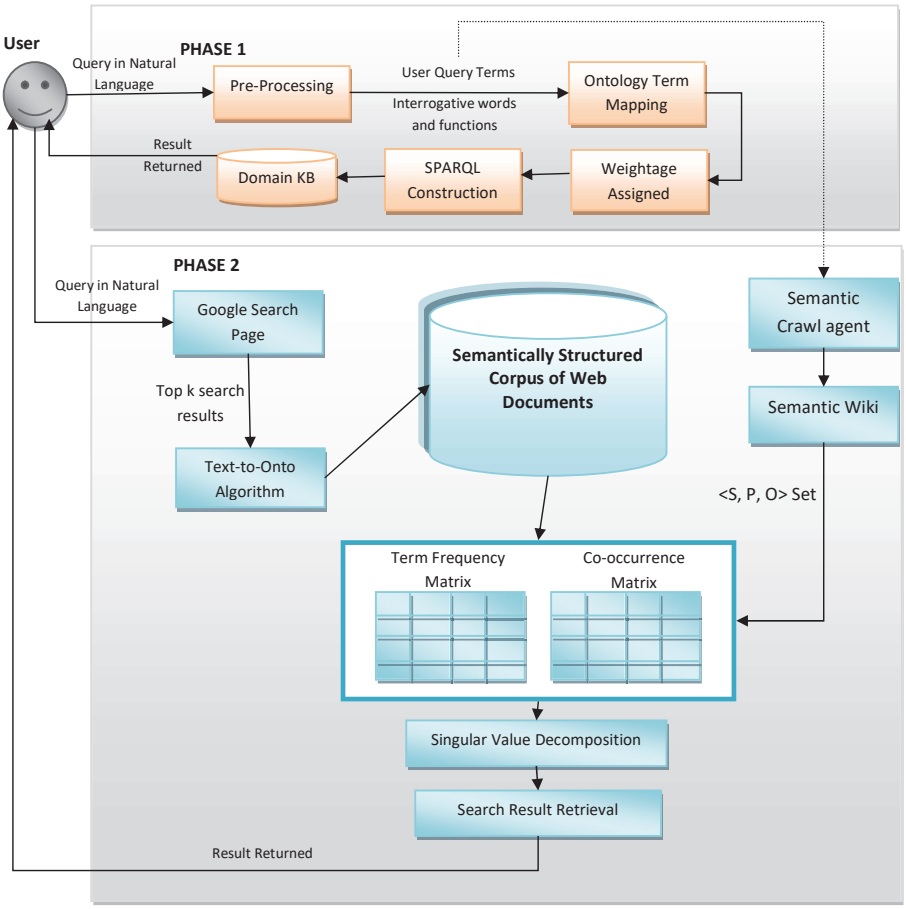

Figure 1 Proposed Framework for Multiphase semantic search engine.

7. Finally the SPARQL query applied on domain KB using Jena ARQ2 engine and results displayed to user.

Phase 2: (Result Retrieval from Web 2.0)

If the query does not return any result due to lack of related information from the domain KB, the query is further fired onto Web 2.0.

1. Query in natural language is redirected to the Google search page.
2. Top *k* search results retrieved are taken into account and converted into semantically structured document.

3. *Text2Onto* algorithm is applied for converting unstructured web document into semantically annotated web documents.
4. The semantically annotated documents are added into the Web document corpus.
5. Semantic crawl agent uses the query term generated in first phase and utilize the *semantic wiki* to generate the <S,P,O> sets related to the user query.
6. Considering each term, of the user query, the *Term Frequency Matrix* is generated and <S,P,O> set is used to generate *Co-occurrence Matrix*.
7. *Singular value decomposition* is used to find out the best possible query vector and corresponding web documents are retrieved.
8. Finally, results are returned to the user using semantic search retrieval.

4 Phase 1: Semantic Search Engine

In this section, the proposed conversion process from user query in natural language to SPARQL is explained in detail. The phase mainly explains the pre-processing, term ontology mapping, weightage assigned and finally SPARQL conversion process.

The user is not well versed in writing structured query such as SPARQL to retrieve the desired result, so to all mass participation, there need to find some way to overcome this problem. So, in the proposed system, user can comfortably enter his query in natural language. **Example Illustration:** Suppose User entered query is *What is the genre of movie directed by Kangana Ranaut?*.

In this phase, pre-processing is applied on the query submitted by the user. The pre-processing involves sentence segmentation, tokenization, and stop words removal, determining part of Speech (POS) of each token and conducting lemmatization. As a result, a set of query terms $QT = \{t1, t2, \ldots, tn\}$, which retains the input order, is retrieved. All these pre-processing tasks can be implemented by NLTK in Python, which is a famous library for NLP [29].

After retrieving set of query terms, function or interrogative terms present in the query are identified for further conversion process. These identified terms are kept separately to be used in final SPARQL conversion process.

Interrogatives: The query terms such as *where, who, which, when* etc are identified as interrogatives. The interrogatives are considered as search target and modification in these terms could affect the final SPARQL query conversions.

Functions: The query terms such as *maximum, how many* are required to be included in final SPARQL such as *ORDER BY, COUNT* etc. This allows aggregating the resources as per the requirement.

Example Illustration: Continuing with the same example, after pre-processing query terms generated are *what Genre movie directed Kangana Ranaut* and the interrogative term is *what*.

4.1 Term Ontology Mapping

On completion of pre-processing, the terms generated from the user is further mapped to the domain resources present in the ontology based on similarity computation. The triple paths mapped with the user's query terms are identified and weightage is assigned to each triple path. Finally, the process is followed by the SPARQL conversion process.

Index is created using the resources present in the domain ontology. The snippet of the indexed domain ontology for the movie domain is shown in Table 2. The index created comprises of the URI of the resources, its type and the annotated values taken from comments, labels, and titles. The URI of the resources indexed may belong to the Classes, Data or Object Properties, Instances and the Literals. The literal values represent the value of type integer, string, numeric etc. The annotation values are mainly intended to assist humans to understand more about the resources, and therefore considered as an important parameter to be included in the Index creation. Therefore, each term in the user query written in natural language are compared with the values of these annotations for finding the best possible match of the query term with the resources mapped.

Table 2 Snippet of movie ontology

URI	Type	Annotation (Comments)
mv:MovieArtist	Class	Movie Artist, Actor, Hero,Heroine,Actress
mv:Manikarnika	Instance	Manikarnika, Jhansi ki Rani
mv:KanganaRanaut	Instance	KanganaRanaut, name of actress, heroine
mv:hasRuntime	DataProperty	Runtime, Length of movie
mv:hasGenre	DataProperty	Genre, type, field
mv:directedBy	ObjectProperty	Director, directed by, direction given by
45	Integer	45, Forty Five
Hindi	Literal	Hindi Language, Regional Language

It has been proven that "*difference should play a less important role on the computation of the overall similarity"* [30]. Therefore each query term is matched with the annotation values using similarity measure based on *commonality and contextuality (SMCC)*. The commonality technique is the normalization of the common substring to the length of the input strings. In this process, two matching strings are compared and the maximum common string found is removed. This is a repeated process which executes till minimum string length is not achieved. Length 2 is chosen as the minimum string length. The length of the generated substrings is scaled to the string length. The contextual similarity between the terms is calculated using '*Word2vec*' [31] which creates the vector of the possible contextual terms related to the given terms and similarity between them is calculated. Consider *'p'* and *'q'* as two strings, the similarity among them is described below in Equation (1).

$$SMCC(p,q) = \alpha * \frac{2 * \sum_i len(commonstring)}{len(p) + len(q)} + \beta * word2vec(p,q) \tag{1}$$

where $\alpha + \beta = 1$, α and β are the control values. If any one of the commonality or the contextuality similarity measure has zero value, then $\alpha = \beta = 1$.

Example Illustration

Suppose the two strings are *director* and *directed*, the lengthy common strings is *direct* which is removed. The leftover string is *or* and *ed*, and in next iteration there is no common string. The length of the total common substring is 6 as *direct* is 6 character long.

The computed similarity value between these strings are $2 * 6/(8 + 8) = 0.75$. And the similarity using word2vec is 0.72. Therefore, the total similarity is $(0.5 * 0.75 + 0.5 * 0.72 = 0.73)$. In another scenario, suppose the two strings are *movie* and *film*, there is no similarity based on commonality but the word2vector calculates 0.71 as similarity value.

The query term is mapped to the URI resource of the domain ontology if the similarity value computed is above threshold value, in this work, it has been taken (>=0.7). The query terms mapped to the resources are further taken into consideration for exploring the entire possible triple paths.

Table 3 Snippet of triple path data structure

Relation	Type	Domain	Range	Triple Path
mv:directedBy	Object Property	mv:Movie	mv:Person	mv:Movie – mv:directedBy – mv:Person
mv:hasGenre	Data Property	mv:Movie	Rdfs:Literal	mv:Movie – mv:hasGenre – Rdfs:Literal
mv:HasRuntime	Data Property	mv:Movie	Rdfs:Literal	mv:Movie – mv:HasRuntime – Rdfs:Literal

4.2 Weightage Assignment to Triples

Each triple consists of <S,P,O> subject, predicate and object. Therefore, a data structure is created to define the triple path, with each row representing different data or object property/relation value present in the domain ontology. The data structure consists of the relation, its type, corresponding domain and range and the triple path. The triple path composed of domain as its subject, property/relation as its predicate and the range as its object. Table 3 shows the snippet of the data structure created.

After mapping user query terms to the corresponding resource URI using indexing and the commonality similarity measure, the probability of each triple path is calculated which provides the relevancy to the user query.

On the basis of generality [32], each triple path is assigned a weight W_r. Higher is the generality of a triple path if there are more number of connections between the edge and node (domain and range) of the path as compared to triple paths that has less connection. The weight represents ratio between the number of instance level triples connected with the property of the path and the total number of triples having the same domain and range irrespective of the property connecting them using Equation (2).

$$W_r = \frac{x(dom, r, range)}{x(dom, *, range)} \tag{2}$$

Where *'x'* is the number of triple paths, *'r'* is the property containing the domain and range and '*' is any property connecting the same domain and range. The interpretation of the generality is the probability of the triple path present in the user query. In construction of user query graph, the arc is identified using the triple path whose weight is greater than the threshold

value. To find the shortest path connecting the nodes, A* algorithm has been used. Finally, it has been converted in the SPARQL query.

4.3 Conversion to SPARQL Query

In this process of conversion, it's very crucial to deal with the interrogative query terms and the *functions terms identified in the Phase 1*. The interrogative terms in the query clearly reflects the user's intension or aim. In English language, the interrogatives are placed in the beginning of the question. On the basis of the interrogatives dependency, they are divided into two categories namely *Dependent Interrogatives* and the *Independent Interrogatives*. The terms such as *which* or *what* are the types of dependent interrogatives preceding the *Class* of resources. In such scenarios, the query is rearranged, the 'class' resources are placed at the end of the query and the dependent interrogative term are removed from it. For example, the user query terms are *what genre movie directed Kangana Ranaut*, in this *what* is removed and thus the query rearranged as *movie directed Kangana Ranaut genre*.

The terms such as *where, who, when or which* that are not preceded by *class* resources. All such interrogative terms are removed from query and *owl:Thing* is added at the end as the mapped resources. For example the user query is, *Who is the manager of Royal club?,* so in this query, *who* is removed and the *owl:Things* is added at the end as mapped resources. Although particular class could be referred, such as *who*, class is *foaf:Person*, but *owl:Things* is the suitable option is the class representing people is not known in any ontology. Therefore, interrogative describes the target of the given user query and it is required while creating the basic structure of SPARQL query.

The basic structure of SPARQL query contains two clauses namely *select* and *where*. The target which needs to be searched is specified by the clause *select* and corresponding to it, a variable was assigned. The triple set which could restrict the target variable is specified in *where* clause. The shortest path derived using the *A* algorithm* in previous section is directly translated into the conditional triples.

After the basic structure of the query has been formed, the function terms identified in pre-processing phase such as *FILTER, LIMIT, OFFSET, ORDER BY* etc is added in the structure of the SPARQL query, if required. All the time related constraints of the query could be processed using the SPARQL time related functions such as days(), months() and year(). If there is requirement

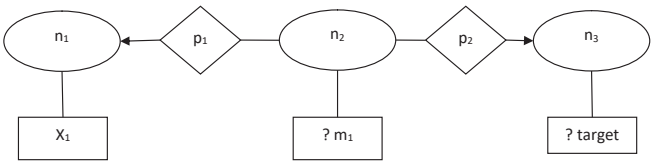

Figure 2 Creation of query graph.

for number of result information, the COUNT function could be used in the 'select' clause. Other function terms such as LIMIT or ORDER BY could be added in the SPARQL structure.

As shown in Figure 2, the conversion into a SPARQL query is represented. The example shows that the select clause uses the target variable belonging to the class *Genre*. Also, the where clause represents the shortest path which has been translated directly. The relationship among the nodes has already been defined as per the unit path triples. For example, the property p_1 is binding the node x_1 with $?m_1$ and has direction from m_1 *to* x_1. This relationship could be described using triple $(?m_1, p_1, x_1)$.

Finally conversion of query graph into its SPARQL query is represented below:

SELECT ?genre
WHERE {
$?m_1$ p_1 x_1
$?m_1$ p_2 ?target
$?m_1$ rdf:type n_2
?target rdf:type n_3
}

Example Illustration

SELECT ?genre
WHERE {
?movie . . . directedby. . . . KanganaRanaut
?movie.. .. hasGenre. . . . .?genre
?movie . . . . rdf:type.. Movie
?target. . . . . rdf:type. . . . . Genre
}

Finally, the query generated in SPARQL is processed by the Jena ARQ2 Engine. There is a Jena model uploaded corresponding to the domain ontology, therefore the Jena ARQ2 Engine could access this model and execute the query. The search results generated due to processing the SPARQL query on the knowledge base are returned to the user. The algorithm for the Phase 1 is shown in Figure 3.

5 Phase 2: Semantic Search Engine

In this phase, if no search results are retrieved from the domain Knowledge base related to user's natural language query in phase 1, the query would be directed to Phase 2 for further processing from the Web 2.0.

5.1 Retrieval from Web 2.0

The user query in natural language is redirected further to the Google search engine page, if no result is returned from the developed domain KB. The top k results are the only relevant results. So, any value of k could be assigned, for this work, the value of k is taken as 10, as suggested that the search result after 10th position gets less than 5% of traffic [33]. The top k search results are crawled and parsed by the proposed system.

5.2 Text2Onto

Text2Onto [34], is an improvised version of Text2Onto, it is developed at the AIFB Institute of University, Karlsruhe, Germany. Combination of various machine learning techniques along with linguistic processing approaches has been used by Text2Onto. GATE framework was used for linguistic processing by the Text2Onto. In Text2Onto, Linguistic processing starts with tokenization followed by sentence splitting. The annotation set created, is given as input to POS tagger which allocates to all tokens its suitable syntactic categories. Lastly, lemmatizing is done using morphological analyzer and stemming using stemmer. Learning process is then begun to recognize the concepts and the relations based on linguistic and machine learning heuristics.

Several measures have been adopted by Text2Onto to analyze the relevance of a particular term with the corpus. Various algorithms have been used by Text2Onto to calculate the measures such as entropy, Term Frequency Inverse Document Frequency (TFIDF) and Relative Term Frequency (RTF). Further, there are several algorithms for utilizing the hyponym of Word Net

ALGORITHM: PHASE 1
Input: User Query in natural language (U_{NL})
Output: Result generated from SPARQL query.
begin
ValidURL = null;
P_{query}= null;
STEP 1: Load user query U_{NL}
STEP 2: For preprocessing, apply NLTK on U_{NL} from python library and generate
Query Terms $Q_T = \{t_1 , t_2 , \ldots\ldots, t_x\}$
Interrogative Terms $I_T = \{i_1 , i_2 , \ldots\ldots, i_y\}$
Function Terms $F_T = \{f_1 , f_2 , \ldots\ldots, f_z\}$
STEP 3: Generate indexes for URI from the domain ontology in the form
<URI , Type , Annotations>
STEP 4: Load $Q_T = \{t_1 , t_2 , \ldots\ldots, t_x\}$ and URI indexes for Term Ontology Mapping
STEP 5: // Calculating Term Ontology Mapping
For each URI (URI_q), from URI set
For each query term *p* from Q_T
For each word *q* from annotations A(q)

$$SMCC(p,q) = \alpha * \frac{2*\sum_i len(commonstring)}{len(p\ \ )+len(q\ \ )} + \beta * word2vec(p,q)$$

Similarity (p,A(q)) = max ($\sum_{i=0}^{n} SMCC(p,q)_{(n)}$)
if (Similarity (p,A(q)) > Threshold (0.7))
ValidURL = ValidURL U URI_q
endif
endfor
endfor
endfor
STEP 6: Load triple path store, S_T from domain ontology represented using data structure
<Relation , Type , Domain, Range, Triple Path>
STEP 7: **// Finding Possible Triple Path**
For each URI_i in valid URI set, ValidURI
For each Triplet path $T_j$from Triple Store, S_T
If (URI_ipresent in T_j)
Possible Triple Path,
TP_p= TP_pUT_j
Endif
Endfor
Endfor
STEP 8: // Assign Weightage
For each path *r* in Possible Triple Path, TP_p
Calculate the weightage, W_r for each path *r*

$$W_r = \frac{x(dom,r,range)}{x(dom,*,range)}$$

If (W_r> Threshold value)
User Query Path,
P_{query}= P_{query}UW_r
Endif
Endfor
STEP 9: Apply A* algorithm to find the shortest path between the identified path's nodes to form Query Graph
STEP 10: Conversion of Query Graph into SPARQL query syntax
STEP 11: SPARQL query processed by Jena ARQ2 Engine and result returned to the user.
end

Figure 3 Algorithm for phase 1 semantic search engine.

structure, employing linguistic heuristics and matching Hearst patterns for learning about relations. Text2Onto uses sources such as databases, dictionaries, free texts, legacy ontologies and semi-structured texts as its input. This learning process outcome is the domain related ontology containing its specific and not related concepts. The non-related concepts are removed to fine tune the domain ontology. Therefore, only the domain related concepts are present in the resultant ontology generated after the learning process. The complete process is iterative and administered by the ontology engineers to better refine and complete the ontology.

5.3 Generating RDF

The semantic web is all about storing the semantically structured data, and the RDF triples are the strong metadata which represents the data on the web. RDF snippets are suggested as the strong indicators for retrieving the useful and required information form the web. The RDF structure indicates the context of the content and helps in filtering the ambiguous web page based on its context. The triple structure of RDF is <Subject-Predicate-Object>, i.e. <S,P,O>.

In this process, Semantic crawl Agent has been used which take each unique user query terms as an input to obtain its relevant RDF structure <S,P,O> from the Semantic Wikis. The semantic crawl agent is considered as intelligent software which is able to retrieve RDF entities from the Semantic Wikis. Semantic wiki consists of the RDF triple structure which represents the content on the pages. It is also the reflection of the real world Semantic Web. RDF triples <S,P,O> are extracted from the Semantic Wiki with the aim of fetching all the contexts of the user query.

The individual entities forming triples, may have high correlation with many closely linked domain, but taking the triples together, means the co-occurrence of the <S,P,O> pattern when searched, they reflects the clear domain indication. Therefore, instead of querying individual entities, their triples co-occurrence are considered for querying which represents the clear domain which the user query belongs to. This eventually removes the ambiguous results and the most related results retrieved by the user. The output generated from this process is the set of <S,P,O> set according to the user query.

Table 4 Term frequency matrix (TF_{QW})

	Web Doc 1	Web Doc 2	Web Doc 3	Web Doc 4	.....	Web Doc y
Query T1	N_{11}	N_{12}	N_{13}	N_{14}	.....	N_{1y}
Query T2	N_{21}	N_{22}	N_{23}	N_{24}	.....	N_{2y}
Query T3	N_{31}	N_{32}	N_{33}	N_{34}	.....	N_{3y}
...	.....	.....	.....	.....	.....	.....
Query Tx	N_{x1}	N_{x2}	N_{x3}	N_{x4}	.....	N_{xy}

5.4 Term Frequency Matrix

The general structure of the *Term Frequency Matrix(TF_{QW})* is shown in Table 4. In *Term Frequency matrix*, each query term generated after pre-processing has been evaluated for its frequency in every web document page of the corpus. Many traditional approaches formulate the *Term Frequency Matrix* using query terms for the purpose of Information Retrieval from the web pages. However, in this work, *Term Frequency matrix* along with the *Co-occurrence matrix* created using RDF triple <S,P,O> has been used which restricts its co-occurrence in number of web pages. The matrix in Table 4 represents the 'x' number of unique user query terms on the one rows and 'y' number of web pages along the columns from the corpus. The cell value 'N_{xy}' represents the frequency of query term 'x' in web document 'y'.

5.5 Co-occurrence Matrix

The general structure of *Co-occurrence Matrix (CO_{RW})* using RDF triple <S,P,O> is depicted in Table 5 which is the novel approach of the proposed work. The matrix represents the frequency of co-occurrence of the triple i.e. the occurrence of the <S-P-O> together in each web page from the corpus. This process removes the terms that are irrelevant to the domain represented by the user query due to restricted co-occurrence of the RDF triples. This strategy of co-occurrence of the Subject-Predicate-Object is formulated due to the triple structure of the RDF schema. The co-occurrence of the RDF triple acts as a strong indicator and thus contributes indirectly in handling context irrelevance and reduces search results ambiguity. The *co-occurrence matrix* in Table 5 shows the 'x' number of RDF triple set generated by the semantic crawl agent represented on the one rows and 'y' number of web pages represented along the columns from the web corpus. The cell value 'N_{xy}' represents the frequency of RDF triple <Sx,Px,Qx> co-occurrence in web document 'y'.

Table 5 Co-occurrence matrix (CO_{RW})

	Web Doc 1	Web Doc 2	Web Doc 3	Web Doc 4	.....	Web Doc y
<S1-P1-O1>	CN_{11}	CN_{12}	CN_{13}	CN_{14}	.....	CN_{1y}
<S2-P2-O2>	CN_{21}	CN_{22}	CN_{23}	CN_{24}	.....	CN_{2y}
<S2-P2-O2>	CN_{31}	CN_{32}	CN_{33}	CN_{34}	.....	CN_{3y}
...	.....	.....	.....	.....	.....	.....
<Sx-Px-Ox>	CN_{x1}	CN_{x2}	CN_{x3}	CN_{x4}	.....	CN_{xy}

Like in traditional schemes, relying on term frequency or on Inverse Document Frequency, the combination of the *co-occurrence matrix* using RDF triple and *term frequency matrix* has been included in this work. Therefore, it resulted into reducing the irrelevancy in such semantic environment and further helps in decreasing the computational complexity. Thereafter, both the matrix, i.e. Term Frequency and the co-occurrence are subjected to Singular Value Decomposition, which could further condense the matrices into Query Term Priority Vector (PV_{QT}). The Query Term Priority Vector represents the terms that are most relevant to the user query.

5.6 Singular Value Decomposition

Singular Value Decomposition has been applied to the *Term Frequency Matrix* along with the *Co-occurrence Matrix* of RDF triple. This technique further reduces both the matrix into Query Term Priority Vector. The generated Query Term Priority Vector has been linked to the most relevant query terms. The algorithm depicted in Figure 4 considers RDF triple generated using semantic crawl agent and the web pages corpus as an input and generate the *Query Term Priority Vector* as an output.

5.7 Search Result Retrieval

The reason behind enriching the top search results to the users through this approach is that it does not take into account blindly the frequency of occurrence of user query term in the web documents. However, the knowledge hidden in the schema level has been used for the solution and it is retrieved from the metadata structure linked to it. The derivation of the *co-occurrence matrix* using RDF structure <S,P,O> act as a strong indicator for the query term as it has been obtained from the Semantic Wiki knowledge base.

Moreover, the focus is on formulating the Query Term Prioritization Vector based on the convergence of co-occurrence RDF triple matrix and the *Term Frequency Matrix* using Singular Value Decomposition. The query term frequencies $\{N_{11},N_{12},N_{13},\ldots,N_{xy}\}$ of the *Term Frequency Matrix* are shown in Table 4. However the frequencies of Subject-Predicate-Object co-occurrence $\{CN_{11},CN_{12},CN_{13},\ldots,CN_{xy}\}$ are depicted in Table 5. It is obvious that the frequency of co-occurrence 'CX' is very less as compared to the query term frequency 'X', i.e. $CN_{xy} < N_{xy}$. Therefore, it removes the less related web pages and thus improves the relevancy while obtaining Query Term Prioritization Vector which considers the incidence between the co-occurrence RDF matrix and the *Term Frequency Matrix*. Moreover, as the *co-occurrence matrix* provides strong indication while retrieving document from the semantic web, it overtakes other traditional approaches such as Page Rank, TF-IDF etc.

The Query Term Priority Vector generated provides the highly relevant query terms. The web document containing <S,P,O> RDF triple belonging to the highly relevant terms are retrieved and shown to the user in the order of highest frequency. Therefore, the all the web documents which contain relevant query term triples and have frequency greater than the threshold are presented to the user. The detailed algorithm for Phase 2 semantic search engine is described in Figure 4.

ALGORITHM: PHASE 2
Input: Preprocessed User Query Terms (U_{QT})
Output: Relevant Web Pages URL.
begin
STEP 1: User query redirected on Web 2.0
STEP 2: Process Top k search results and placed in document corpus D_k
STEP 3: Semantically structured web document corpus, D_{ss}= null
for each document $d_i D_k$
Apply Text2Onto algorithm
$D_{ss} = D_{ss} U\ d_i$
endfor
STEP 4: Extract RDF triplets from Semantic Wiki using U_{QT}
STEP 5: Load each RDF triplet from Triplet Store T_R
STEP 6: for each triplet in T_R
LookUp web document in $D_{ss}$and generate
a) Term Frequency Matrix (TF_{QW}) depicting the occurrence of query terms in $D_{ss.}$
b) Co-occurrence Matrix (CO_{RW}) depicting the occurrence of RDF triplet T_R in $D_{ss.}$
STEP 7: Apply Singular Value decomposition on above matrix.
STEP 8: Formulate the Query Term Prioritization Vector (PV_{QT}) depicting highly related terms.
STEP 9: Web document URLs containing highly relevant terms <S,P,O> triplet are returned to the user.
end

Figure 4 Algorithm for phase 2 of semantic search engine.

6 Experimental Evaluation

In this section, the detailed experimental evaluation has been done to compare the proposed approach of semantic search engine with the existing approaches. The proposed system prototype was implemented in using Java 8, under 4 GHz processor, 8GB RAM and 64-bit Microsoft Windows 2010. Various libraries and plug-ins has been used such as Jena for accessing ontology, *Text2Onto* for converting web documents in structured format, AgentSpeak extension Jason for semantic crawl agent and Lucene library for indexing the resources of the domain ontology. Ontology was created for the *Movie* domain and *Books* domain using our other research work, "EasyOnto", a platform for developing domain ontology. The *Movie* and *Book* domain ontology respectively consists of 35/28 classes, 44/42 data properties, 53/50 object properties and 633/549 individuals, comprising total of 3,933/2845 axioms. The performance metrics parameters are represented in Equations (3)–(7).

$$Precision = \frac{Number\ of\ documents\ retrieved\ and\ relevant}{Number\ of\ documents\ retrieved} \tag{3}$$

$$Recall = \frac{Number\ of\ documents\ retrieved\ and\ relevant}{Number\ of\ documents\ relevant} \tag{4}$$

$$F\text{-}measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5}$$

$$Accuracy = \frac{Precision + Recall}{2} \tag{6}$$

$$FDR\ (False\ Discovery\ Rate) = \frac{FP}{FP + TP} \tag{7}$$

where *FP* is *False Positive* and *TP* is *True Positive*.

We had done interaction with 178 people and 128 people and requested them to provide the natural language query for the *Movie* domain and *Books* domain respectively. All these people were unfamiliar of the term ontology, semantic web and SPARQL. 135 queries and 102 queries in *Movie* and *Book* domain respectively had been collected and few people were reluctant in responding properly. Three domain experts were given task for manually evaluating the system. The experts analyzed the feasible answers that should be generated from the user queries; also, they analyzed the correctness of the SPARQL query generated by the proposed system. To

measure the performance metrics such as precision, recall, f-measure etc, manual evaluation was necessary.

The experiment was done based on various criteria and generated results are analyzed with other existing semantic web or Knowledge base search approaches such as OTNLS [28] , SQG_NLS [11], SWQ_RDF [22], UPSIR [18], IWSF [27].

***Experiment 1 (Performance Metrics)*:** The experiment to evaluate the performance of proposed system with the other related approaches was conducted. The existing approaches such as OTNLS, SQG_RDF, SWQ_RDF, UPSIR and IWSF were taken into consideration and evaluated for the same datasets as of the proposed approach for the same 132 queries. It is clearly depicted in the Figure 5(a–b) that the proposed approach outstands in comparison with other semantic based search techniques. There are several reasons for such high value performance of the proposed approach. The incorporation of the efficient semantic similarity measure and the assigning weightage to the possible query path helps in creation of precise SPARQL query. In scenarios where desired information is not available in Knowledge base, alternative solution to handle use query is presented. Also, the idea of using the *co-occurrence matrix* <Subject, Predicate, Object> depicts the relevancy of user query with the web pages.

SWQ_RDF shows the lowest performance measure with an avergae 74% precision, 80% recall, 76.9% f-measure and 77% accuracy. Proposed approach shows improvement in precision with 16.48%, 7.69%, 18.68%, 12.09 and 6.59% in comparison OTNLS, SQG_NLS, SWQ_RDF, UPSIR, IWSF respectively. On an average, the performance improvement shown by

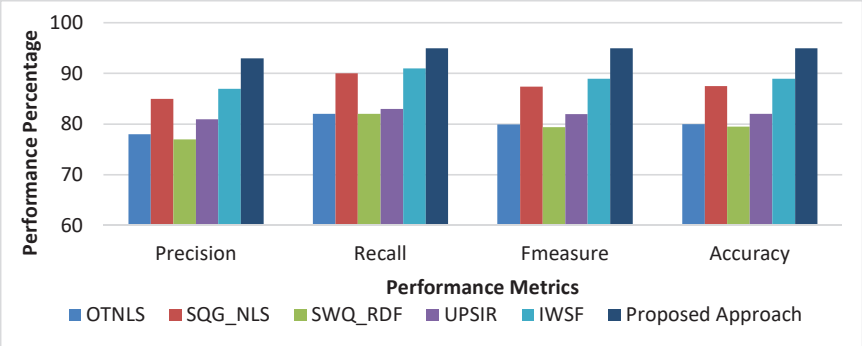

Figure 5(a) *Movie* Domain: Proposed Approach Performance Metrics comparison.

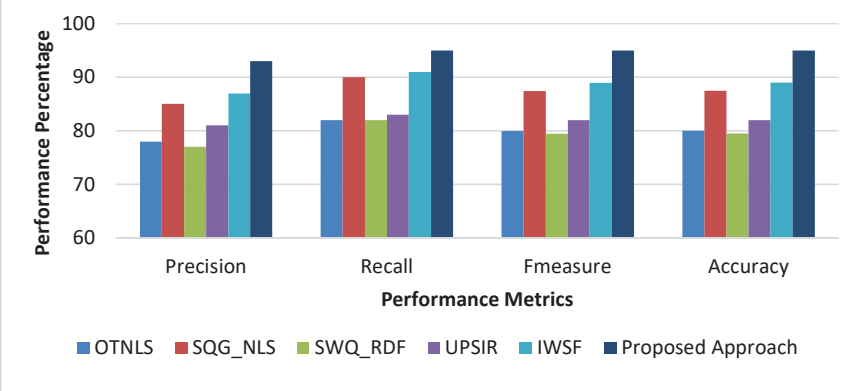

Figure 5(b) *Books* Domain: Proposed Approach Performance Metrics comparison.

proposed method for precision, recall, f-measure and accuracy are 12.31%, 12.29%, 13.76, 13.68% respectively.

***Experiment 2 (False Discovery Rate)*:** False Discovery Rate indicates the number of the false or incorrect search results retrieved by the web search engine for a particular user query. The importance of calculating the FDR is to analyse the percentage of the search results which are excluded by the user for a particular search query. Although, there are various statistical tools for FDR computation, but the technique mentioned in Equation (7) is the easiest method for calculating FDR value in respect to search engines. The quality of search engines is considered as high, if the value of FDR is low. This implies that the search engine is able to retrieve the most appropriate search results for a specific query.

It can be observed from Figure 6(a–b), that the proposed system has lowest value of FDR as compared with other systems. SWQ_RDF shows the highest value of FDR as 0.33. The proposed approach is better than OTNLS, SQG_NLS, SWQ_RDF, UPSIR and IWSF showing an average of 31.58%, 21.05%, 73.68%, 47.37%, 15.79% reduction in FDR value. The lowest value of FDR reflects that the proposed system provides very less inappropriate and rejected recommendations, thereby it is proven to be an appropriate semantic search engine. RDF triple *co-occurrence matrix* and the efficient semantic similarity, ISI are the reasons for such low value of FDR.

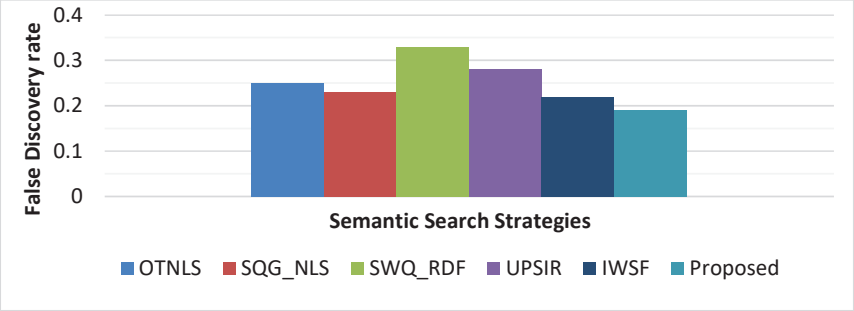

Figure 6(a) *Movie* Domain: False Discovery Rate comparison.

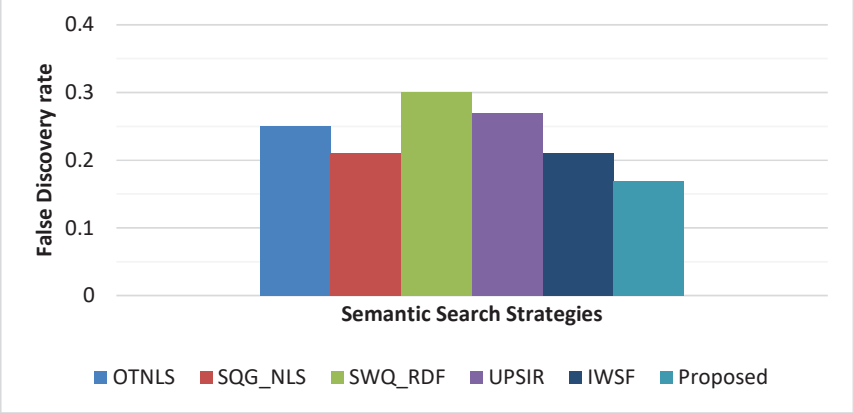

Figure 6(b) *Books* Domain: False Discovery Rate comparison.

***Experiment 3 (SMCC Performance Metrics)*:** The evaluation of the proposed framework effectiveness and the appropriateness of the approaches incorporated had been done and its relative performance in context of the semantic similarity measure chosen is depicted in Figure 7(a–b). The traditional measures of similarity such as Cosine similarity, Jaccard similarity are replaced with the proposed similarity measure SMCC. It is evident from the experiment that the high performance was depicted by the proposed SMCC measure as compared to the other similarity measures.

It is observed that the adaptation of Cosine Similarity yields an average of 85% precision, 86% recall, 85.5% F-measure and 85.5% accuracy. However, using the Jaccard Similarity measures shows improvements in the performance matrix as compare with the Cosine Similarity. The average values for

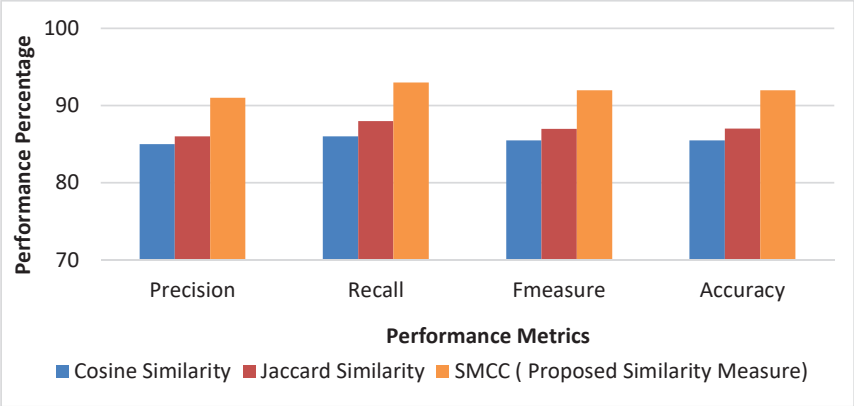

Figure 7(a) *Movie* domain: SMCC performance metrics.

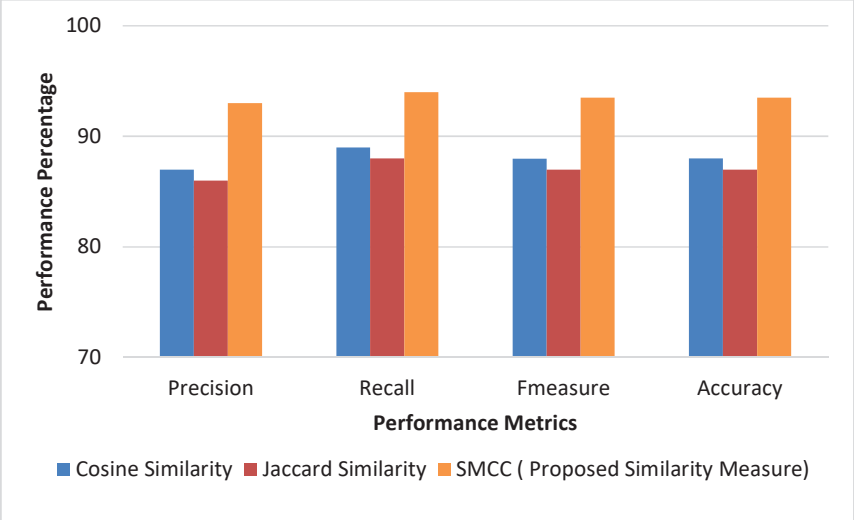

Figure 7(b) *Books* domain: SMCC performance metrics.

precision, recall, F-measure and accuracy are 86%, 88%, 87%, 87% respectively. The proposed SMCC similarity measure outstands in performance as it is hybrid and efficient for a semantic space. There is an average 6% increase in precision 6.5% in recall, 6.2% in F-measure and 6.3% in accuracy as compared to Cosine and Jaccard similarity.

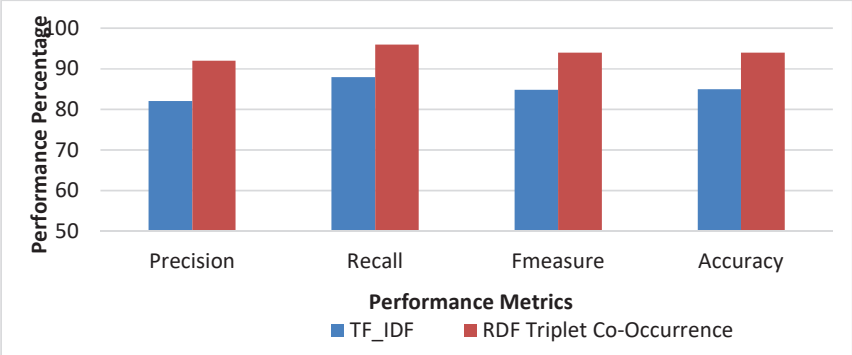

Figure 8(a) *Movie* domain: performance metrics using *Co-occurrence Matrix*.

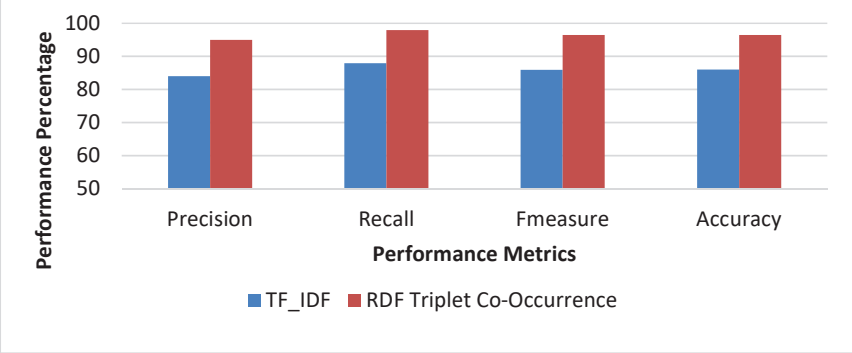

Figure 8(b) *Books* domain: performance metrics using *Co-occurrence Matrix*.

***Experiment 4 (Co-occurrence Matrix Performance Metrics)*:** From Figure 8(a–b), it is proven that the including the RDF triple *Co-occurrence matrix* in this work has greatly contributed in increasing the efficiency as compared to classical TF-IDF. In this experiment, the *co-occurrence matrix* has been replaced by the TF-IDF matrix to compare the statistics, it has been observed using *co-occurrence matrix*, the Precision is increased by an average of 10.9% and recall increased by 8.3%. If TF-IDF has been used, it would reduce the F-measure from 94% to 84.9% and decrease the accuracy by 9.6%. The reason behind these numbers is quite clear as this approach for semantic search depends on the knowledge inferred from the RDF triple. TF-IDF could be appropriate while query the traditional web, as it depends on the number of occurrence of the query term in the web pages, unlike Semantic Web. The *co-occurrence matrix* depends upon the co-occurrence

of RDF triple which represents the knowledge, instead of depending merely on the occurrence of query terms in web documents, therefore it increases the overall performance in comparison to the TF-IDF.

7 Conclusion

To overcome the limitation of web 2.0 and the challenge faced by common user in retrieving the information from the domain specific ontology in this novel work, an efficient semantic search engine has been proposed. The SPARQL query has been automatically generated according to the user query submitted in natural language. Also, assigning the weightage score to each triple path and finding the possible triple path as per query also reduces the ambiguity. The proposed semantic similarity measure (SMCC), increases the effectiveness and the appropriateness of the approach. Further, redirecting the user query to Web 2.0, greatly increases the performance measures. The proposed approach increases the overall precision, recall and f-measure by 9.74%, 9.02%, 9.88%, 9.83% respectively. The formulation of *Term Frequency matrix* and the *Co-occurrence matrix* improves the context relevance and reduces the false discovery by 37.89%. In future work, this technique could be extended for cross-domain ontology and various other knowledge bases reflecting the world such as DBpedia could be explored.

References

[1] The Linked Open Data Cloud, https://lod-cloud.net/

[2] Vargas, H., Buil-Aranda, C., Hogan, A., López, C.: RDF Explorer: A Visual SPARQL Query Builder. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 647–663. Springer (2019)

[3] Bernstein, A., Kaufmann, E., Kaiser, C., Kiefer, C.: Ginseng: A Guided Input Natural Language Search Engine for Querying Ontologies. Jena User Conf. Bristol, UK. (2006)

[4] Kaufmann, E., Bernstein, A., Fischer, L.: NLP-Reduce: A "naïvenaïve" but Domain-independent Natural Language Interface for Querying Ontologies. 4th Eur. Semant. Web Conf. (ESWC). (2007)

[5] Khan, A., Ibrahim, I., Uddin, M.I., Zubair, M., Ahmad, S., Al Firdausi, M.D., Zaindin, M.: Machine Learning Approach for Answer Detection

in Discussion Forums: An Application of Big Data Analytics. Sci. Program. 2020, (2020). https://doi.org/10.1155/2020/4621196

[6] Han, L., Finin, T., Joshi, A.: GoRelations: An intuitive query system for DBpedia. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 334–341. Springer, Berlin, Heidelberg (2012).

[7] Damljanovic, D., Agatonovic, M., Cunningham, H.: Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 106–120. Springer, Berlin, Heidelberg (2010).

[8] Kasneci, G., Suchanek, F.M., Ifrim, G., Ramanath, M., Weikum, G.: NAGA: Searching and ranking knowledge. In: Proceedings - International Conference on Data Engineering. pp. 953–962 (2008).

[9] Styperek, A., Ciesielczyk, M., Szwabe, A.: SPARQL - Compliant semantic search engine with an intuitive user interface. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 201–210. Springer Verlag (2014).

[10] Geng, Q., Deng, S., Jia, D., Jin, J.: Cross-domain ontology construction and alignment from online customer product reviews. Inf. Sci. (Ny). 531, 47–67 (2020). https://doi.org/10.1016/j.ins.2020.03.058

[11] Song, S., Huang, W., Sun, Y.: Semantic query graph based SPARQL generation from natural language questions. Cluster Comput. (2017). https://doi.org/10.1007/s10586-017-1332-3

[12] Heibi, I., Peroni, S., Shotton, D.: Enabling text search on SPARQL endpoints through OSCAR. Data Sci. 2, 205–227 (2019). https://doi.org/10.3233/ds-190016

[13] Arenas, M., Grau, B.C., Kharlamov, E., Marciuska, S., Zheleznyakov, D.: Faceted search over ontology-enhanced RDF data. CIKM 2014 – Proc. 2014 ACM Int. Conf. Inf. Knowl. Manag. 939–948 (2014). https://doi.org/10.1145/2661829.2662027

[14] Wang, X., Yang, L., Zhu, Y., Zhan, H., Jin, Y.: Querying Knowledge Graphs with Natural Languages. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 30–46. Springer (2019).

[15] Wang, C., Xiong, M., Zhou, Q., Yu, Y.: PANTO: A portable natural language interface to ontologies. In: Lecture Notes in Computer Science

(including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 473–487. Springer Verlag (2007).

[16] Yahya, M., Berberich, K., Elbassuoni, S., Ramanath, M., Tresp, V., Weikum, G.: Natural Language Questions for the Web of Data. Association for Computational Linguistics (2012).

[17] Ferré, S.: SQUALL: A controlled natural language as expressive as SPARQL 1.1. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 114–125 (2013).

[18] John, P.M., Arockiasamy, S., Thangiah, P.R.J.: A personalised user preference and feature based semantic information retrieval system in semantic web search. Int. J. Grid Util. Comput. 9, 256–267 (2018). https://doi.org/10.1504/IJGUC.2018.093987

[19] Ramzan, B., Bajwa, I.S., Jamil, N., Amin, R.U., Ramzan, S., Mirza, F., Sarwar, N.: An Intelligent Data Analysis for Recommendation Systems Using Machine Learning. Sci. Program. 2019, (2019). https://doi.org/10.1155/2019/5941096

[20] Ramesh, C., Rao, K.V.C., Govardhan, A.: Ontology based web usage mining model. In: Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2017. pp. 356–362. Institute of Electrical and Electronics Engineers Inc. (2017).

[21] Yasodha, S., Dhenakaran, S.S.: ONTOPARK: Ontology based page ranking framework using resource description framework. J. Comput. Sci. 10, 1776–1781 (2014). https://doi.org/10.3844/jcssp.2014.1776.1781

[22] Chooralil, V.S., Gopinathan, E.: A Semantic Web query Optimization Using Resource Description Framework. In: Procedia Computer Science. pp. 723–732. Elsevier B.V. (2015).

[23] Guha, R. V., Brickley, D., Macbeth, S.: Schemaorg: Evolution of structured data on the web. Commun. ACM. 59, 44–51 (2016). https://doi.org/10.1145/2844544

[24] Introducing the Knowledge Graph: things, not strings, https://blog.google/products/search/introducing-knowledge-graph-things-not/

[25] Ji, S., Pan, S., Cambria, E., Member, S., Marttinen, P., Yu, P.S., Fellow, L.: A Survey on Knowledge Graphs: Representation, Acquisition and Applications. (2021).

[26] Bansal, R., Jyoti, Bhatia, K.K.: Ontology-based ranking in search engine. In: Advances in Intelligent Systems and Computing. pp. 97–109. Springer Verlag (2018).
[27] Ahamed, B.B., Ramkumar, T.: An intelligent web search framework for performing efficient retrieval of data. Comput. Electr. Eng. 56, 289–299 (2016). https://doi.org/10.1016/j.compeleceng.2016.09.033
[28] Sander, M., Waltinger, U., Roshchin, M., Runkler, T.: Ontology-Based Translation of Natural Language Queries to SPARQL. AAAI Fall Symposia (2014).
[29] Natural Language Toolkit – NLTK 3.6.2 documentation, https://www.nltk.org/
[30] Stoilos, G., Stamou, G., Kollias, S.: A string metric for ontology alignment. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 3729 LNCS, 624–637 (2005). https://doi.org/10.1007/11574620_45
[31] Word embedding demo, http://bionlp-www.utu.fi/wv_demo/
[32] Lee, M., Kim, W., Park, S.: Searching and ranking method of relevant resources by user intention on the Semantic Web. Expert Syst. Appl. 39, 4111–4121 (2012). https://doi.org/10.1016/j.eswa.2011.09.127
[33] No. 1 Position in Google Gets 33% of Search Traffic [Study], https://www.searchenginewatch.com/2013/06/20/no-1-position-in-google-gets-33-of-search-traffic-study
[34] Cimiano, P., Völker, J.: Text2Onto A framework for ontology learning and data-driven change discovery. In: Lecture Notes in Computer Science. pp. 227–238. Springer Verlag (2005).

Biographies

Usha Yadav is presently working as an Assistant Professor in National Institute of Fashion Technology, Jodhpur, India and has more than 7 years of working experience. She is also pursuing Ph.D. from J. C. Bose University

of Science and Technology, YMCA, Faridabad, India. She received her B.E. in Information Technology in 2009 and M.Tech. in Computer Engineering in 2011. She has published more than 11 research papers in reputed journals and conferences indexed with SCIE, SCOPUS etc. Her areas of interest are semantic web, information retrieval, AR VR, Artificial Intelligence and Internet of Things.

Neelam Duhan has an academic work experience of 17 years and currently working as an Associate Professor in Computer Engineering Department at J. C. Bose University of Science and Technology, YMCA, Faridabad. She received her B.Tech. in Computer Science and Engineering, M.Tech. in Computer Engineering and Ph.D. in Computer Engineering in 2002, 2005 and 2011 respectively. She has successfully guided three Ph.Ds and is currently guiding four Ph.D. scholars in the areas of machine learning, semantic web and social networks. She has guided more than 30 M.Tech dissertations. She has published more than 75 research papers in reputed journals and conferences. Her areas of interest are databases, data analytics, information retrieval and web mining.