International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Enhancing Question Answering through Augmented Term Extraction on Generated Ontology in Closed Domain

Vikas Bali[1*], Amandeep Verma[2]

Submitted: 10/10/2023 **Revised**: 30/11/2023 **Accepted**: 10/12/2023

Abstract: The ever-increasing use of smartphones and computers has led to a culture of instant gratification, shaping the way information is sought and shared in today's digital age. Natural language understanding (NLU) is the linchpin that allows machines to access, process, and provide answers from the vast amount of human knowledge stored in natural language (NL) text. The ongoing development of NLU technologies continues to drive advances in question answering and a wide range of other natural language processing (NLP) applications. Ontology plays a pivotal role in question answering by enhancing the system's ability to understand, contextualize, and retrieve relevant information from a structured knowledge base or unstructured textual data. In this work, we designed an Ontology-based Question Answering (QA) using a self-created dataset is an interesting and valuable endeavour. The system can leverage augmented term extraction on automatically created ontology to understand the domain context and relationships between concepts and is well integrated into the QA system. The performance of QA prototype is accessed using appropriate evaluation metrics This involve using a portion of self-created dataset as a test set and comparing the system's answers to the ground truth.

Keywords: *Question answering system,; natural language processing, ontology, ontology-based question answering system, term extraction*

1. Introduction

Ever-increasing use of smart phones and exponential growth of computers in communication, supported by high-speed free internet and others, motivated the users to get the answers of respective questions in no time. Now the users are more aware of technology, but consequently losing patience to wait for reaching exact answers. Search engines provide impeccable support to offer answering in almost any domain, but still pose certain limitations [1] including manipulated ranking algorithms, limited answering in natural language question, lacking deduction capabilities and personal privacy issues are among others. Question Answering System (QAS) came as a solution and acts similar to search engines and works better for close and restricted domains [2]. Obviously, such systems are restricted to answer only from the database-encoded information.

Then comes the era of semantic search which drastically promoted the use of ontology-based information to make data more meaningful. This encouraged the new era of Ontology Based Question Answering Systems (OBQAS) that are built on state-of-art technology attempting to answer user queries from heterogeneous and scattered data sources like semantic web. OBQAS takes input in natural languages and output relatively short answers [3] and is more focused on Answer Driven Search. Most of the earlier OBQAS such as [4] and [5], require manual construction of ontology with enormous human efforts and high dependency on relational database like MySQL [6] and other structured database [7] to store QA data and require additional external installations. Though such manually crafted ontologies give good results, but these are still not very popular due to the involvement of highly paid experts and domain specialists. Previous studies show that there are still many chances of improvement in reaching exact results, especially by formulating methods for automatic ontology construction and answering the user-specific questions more precisely.

In this proposed approach, we try to minimize the gap by introducing automatic ontology generation for QA task. This is achieved by identifying terms from text, extracting correct entities from these identified terms and establishing the correct relationships between the extracted entities. The proposed system is also capable of auto translating the NL input question to SPARQL Protocol and Resource Description Framework Query Language (SPARQL) [8] query for answer retrieval. Instead of relational database, we used JSON [9] / XML [10] format as input dataset and output question-answer base.

Problem statement for the currently proposed work is focussed on following points:

1. Understanding the structure and nature of an ontology.
2. Determine the domain that the intended ontology will encompass.

[1] *Department of Computer Science, Panjabi University, Patiala, India*
ORCID ID: 0000-0002-2829-5176
[2] *Department of Computer Science, Panjabi University, Patiala, India*
ORCID ID: 0000-0002-2261-4957
** Corresponding Author Email: vikasbali.pu@yahoo.com*

3. Exploring the practical application of the intended ontology in QA.
4. Defining the specific types of questions that will be addressed using the intended ontology.
5. Determining the necessary packages or tools required to construct an ontology from NL text.
6. Validating the developed ontology through the use of question answering.

The key contributions of this work include:

1. A through study is done to review existing QAS, encompassing both ontology-based and non-ontology-based QAS.
2. The creation of a customised closed domain dataset, which serves as a foundational resource for ontology construction and useful in comprehending OBQAS prototype.
3. Developed a small class library in JAVA [11] [12] using NetBeans IDE [13] for the automatic construction of ontology from input text. This library seamlessly integrates with Stanford's CoreNLP package [14] and comprises a range of classes and methods that provides sustenance to the automatic ontology construction as well as for question answering.
4. The formulation of rules for converting NL questions into their equivalent SPARQL queries.
5. Lastly, the evaluation of the proposed OBQAS using standard performance metrics.

The remaining sections in this paper are organized as: Section-2 contains the background and related work about the state of the art of Question Answering (QA) systems. Section-3 briefly explains about resources and methods showcased in this work along with the tools and packages used. Section-4 explains about experimental results of the proposed approach. Section-5 outlines the conclusions of the proposed work and summarizes the main contributions. Section-6 contains Funding section, disclose Conflicts of Interest, information about data availability and code availability and briefs about data availability and code availability and briefs about Author's contributions. And the last section summarizes the references in the list form.

2. Background and Related Work

Research in QA is an ongoing process since decades. Literature review addresses the QA problem since the beginning of 1960s. As always referenced, the two earlier QAS i.e. BASEBALL [3] and LUNAR [3] that are developed in 1971, are the two well-known successful models in their respective restricted domains. Whereas BASEBALL system answers against the recorded data of Baseball game played over one season by American league, LUNAR is capable of answering questions dealing with rock samples taken during Apollo moon missions. The QUALM system [15], a story understanding system by means of information retrieval, recognize and classify the queries like human beings. AskJeeves [15] is another QA problem in open domain that directs the user to Web pages (similar to search engines) that might contain relevant information about asked question by incorporating advanced NLP and data mining techniques. Another system, with a different approach, is the FAQFinder [15] QA system, is developed for answering factual questions over the web through statistical similarity and semantic similarity on question-answer pairs in FAQ database. START [15] (SynTactic Analysis Using Reversible Transformations) system, is a dynamic open domain QA system and first web based QA that extracts answers from different sources using NL Annotation technique.

Modern QA systems are extended versions of these expert systems and implies NL techniques to process questions and the text & knowledge corpus. Text REtrieval Conference (TREC) [16] [17] retrieve precise answers to questions by searching collection of documents, rather than entire documents. Restricted-domain question answering (RDQA) such as MedQA [18] [19] [20] and HonQA (Health On the Net) [18] [21], are medical domain restricted QA systems that are designed to help biologists to access short definitional type answers. MOSES [16] [3], an OBQAS, exploits the techniques used in NLP, graph theory, text mining, etc. on semantic web to extract answers of queries posted by users. AQUA [16] [22], an OBQAS in closed domain, that combines domain-related documents and database knowledge through academic life based ontology. Another example is JAVELIN [23], an open-domain OBQAS that is extended to focus on restricted domains. JAVELIN is a star architecture where all subtasks such as question analysis, information retrieval, answer extraction and answer computing are observed as nodes that connected to a centre node. Another example of restricted domain ontology-based system, as mentioned in [24], represented an Intelligent QAS whose main aim is to build a QAS for students interested in online QA system without any interference of teacher in answering process.

Two popular ontology based restricted domain QAS are PowerAQUA [25] and AquaLog [26]. PowerAQUA is a Multi-Ontology based QAS that focus on querying multiple Semantic Web resources and return answers from suitable distributed resources on the Semantic Web. AquaLog is ontology based restricted domain portable QAS that takes ontology as an input and return answers drawn from one or more knowledge bases through the use of GATE infrastructure - a linguistic component, to convert NL question to query triples. Currently, numerous OBQAS have emerged over the past two decades. A comprehensive and systematic comparison of all of them is not a practical undertaking. Therefore, a method is needed to compare ontological QAS effectively. In this context, we have

concentrated on specific benchmarking criteria and techniques utilized to distinguish and evaluate some of the most prominent ontology-based QAS, as discussed in [27]. In addition, QAS can also be assessed based on in-depth aspects, as elaborated in [28], such as searching, matching technology (Exact Match, Best Match, etc.), form of answers (Short Answers, Mixed Answers, etc.), types of questions (Simple, Wh-type, Descriptive, hypothetical, etc.), relevancy techniques (Pattern Matching, Syntactic Analysis, Pragmatic Analysis, etc.), knowledge source (Database, Syntactic Web, etc.), models used in information retrieval process (Bag of words, Bag of concepts, Bag of Knowledge, etc.) and reliability (Less, Good, Very Good, etc.).

3. Resources and methods

Ontology generation is initiated with extraction of relevant terms i.e. noun phrases and their corresponding relationships from the textual data. In pursuit of creating a prototype for an OBQAS, especially designed to address straightforward, fact-based natural language questions, we conducted experiments using a Self-Created Closed Domain Dataset (SCCDD). The motivation behind developing SCCDD stemmed from the absence of a dataset encompassing all three essential components necessary for QAS over Ontology: reading comprehension text data, the underlying ontology, and gold standard question-answer pairs based on that ontology.

3.1 SCCDD

The initial segment of the SCCDD centres on the 'Education' domain, wherein the dataset creation process commenced by gathering unstructured text paragraphs from the websites of participating institutions and the Joint Admission Information Brochure [29], which serves as the basis for creating high-quality question-answer pairs. For the remaining three randomly selected distinct domains, namely 'Personality,' 'Entertainment,' and 'Organization,' we utilized the 'SQuAD v1.1' dataset [30], a dataset focused on reading comprehension-based question answering. The domains in SCCDD are briefly explained as follows:

1. Education - We focused on a specific real-world context of college admissions process in various bachelor courses of engineering, collaborating five participating institutions. This is referred as “PU_and_JAC” and is abbreviated as “PU”.
2. Personality - We centred our attention on a real-world figure, the inventor known as ‘Nikolas Tesla.’ This is referenced as ‘Nikola_Tesla’ abbreviated as ‘NT’.
3. Entertainment - Our emphasis was on a science-fiction television program titled ‘Doctor Who.’ This is identified as ‘Doctor_Who’ abbreviated as ‘DW’.
4. Organization - We directed our focus to a real-world organizational entity, ‘Chicago University.’ This is denoted as ‘University_of_Chicago’ abbreviated as ‘UC’.

The text paragraphs sourced from 'SQuAD v1.1' underwent a pre-processing stage involving filtering and cleaning before their integration into the SCCDD. Details about the characteristics of context paragraphs and Wh-type questions within SCCDD are available in

Table ***I***.

Table I Characteristics of Context Paragraphs and Questions

Sr. No.	Characteristic	Size
1	Total input dataset size	13477 words
2	Number of paragraphs	114
3	Number of words in largest paragraph	264 words
4	Number of words in smallest paragraph	41 words
5	Total Number of Questions	548

The categorization of Wh-type questions across the four closed domains within the SCCDD is provided in

Table ***II***, while its corresponding chart representation can be observed in Fig. 1.

Table II Wh-type categorization of NL Questions in four domains

Wh-type	PU	NT	DW	UC	Total Questions
How	46	6	10	24	86
What / In what	104	31	38	67	240
When	16	32	14	5	67
Where	25	18	1	3	47
Which	2	1	1	0	4
Who	33	13	30	28	104
Total	**226**	**101**	**94**	**127**	**548**

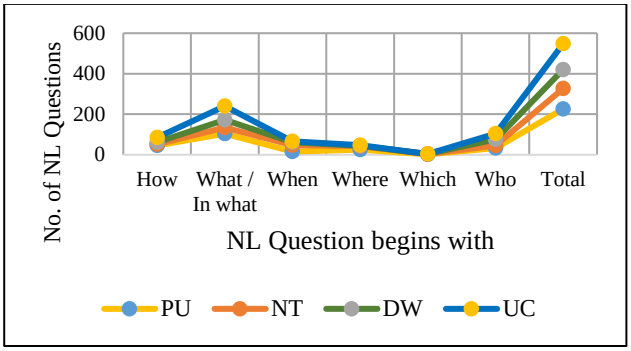

Fig. 1 Wh-type categorization of NL Questions in four domains

3.2 Proposed approach

In this study, we have developed a small class library in JAVA JDK [11] [12] using NetBeans IDE [13] to facilitate automatic ontology learning from English language context paragraphs, with the objective of utilizing it for QA tasks to obtain the desired answers. Our approach leverages the Stanford CoreNLP syntactical parser to produce parse trees from sentences, thus enabling the extraction of relation triples, consisting of subject, predicate, and object, from the input text [31] [13]. During this process, we recognized that the identification and extraction of terms from the input text are pivotal in the overall procedure and should be handled judiciously. In addition, we adopted a rule-based technique [32], in combination with POS taggers, to enhance term extraction within each of the four domains individually. Fig. 2 below depicts the proposed architecture diagram of the QAS prototype keeping the implementation details underneath.

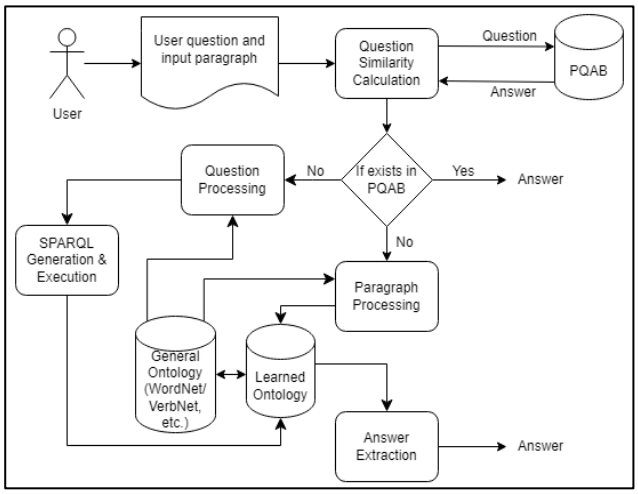

Fig. 2 Proposed Architecture of self-constructing Ontology based QAS

Each step in proposed flowchart is briefly explained as follows:

- User question and input paragraph: This module handles the user's natural language question and domain paragraphs from which the ontology be constructed. It also detects the Expected Answer Type (EAT) or the focus of the input question, a critical step in arriving at the final answer.
- Question Processing: This module is responsible for focus identification of question, question classification, question reformulation and conducting other pre-processing tasks. The generation and

execution of SPARQL from NL questions are also accomplished as a sub-task within this pharse.

- Question Similarity Calculation: This module aids in retrieving corresponding answers from the Processed Question Answer Base (PQAB) when a matching question is found in the database. Sophisticated NLP and ranking technology are pivotal in determining the similarity between the input question and the previously stored question and its answer.
- Paragraph Processing: This module carries out all lexico-syntactic operations and relationship extraction activities. General ontology also supports this module, contributing to the automatic construction and enrichment of the ontology.
- Answer Extraction: This module identifies candidate answers, filters them and rank them. Ultimately, the most reliable answer is presented to the user interface, and the question-answer pair is stored in the PQAB.

3.2.1 Ontology design

The design of ontology plays a pivotal role in addressing the QAS problem. Ontology structure is primarily composed of class, instance, predicate, and property elements. In ontology design, the root class for any ontology is typically ‘owl:thing,’ and under this, we established our initial custom class as ‘DomainEntity.’ Subsequently, subclasses of ‘DomainEntity’ were generated, and they were set as disjoint from one another. It's essential to note that this module is executed only once, specifically when the initial ontology is being created for the first time.

For visualizing the ontology generated by our proposed approach, we utilized an external tool called 'Protégé' [33], which facilitates ontology visualization via the 'ProtégéVOWL' plugin [34] [35]. Additionally, 'Protégé' offers validation capabilities for both ontology structure and data through its internal reasoner, such as the Hermit Reasoner, which can be configured to validate our proposed ontology.

For the purpose of illustration within this paper, we have chosen to focus solely on the 'Education' domain to showcase various facets of the created ontology. Fig. 3 represents a Tab View of the 'Education' domain ontology, including the Class Hierarchy, Object Properties, and Data Properties. In Table ***III*** , you can find an example of the class, its instances and properties within the 'Education' domain ontology, and in Fig. 4, a partial view of the 'Education' domain ontology is presented.

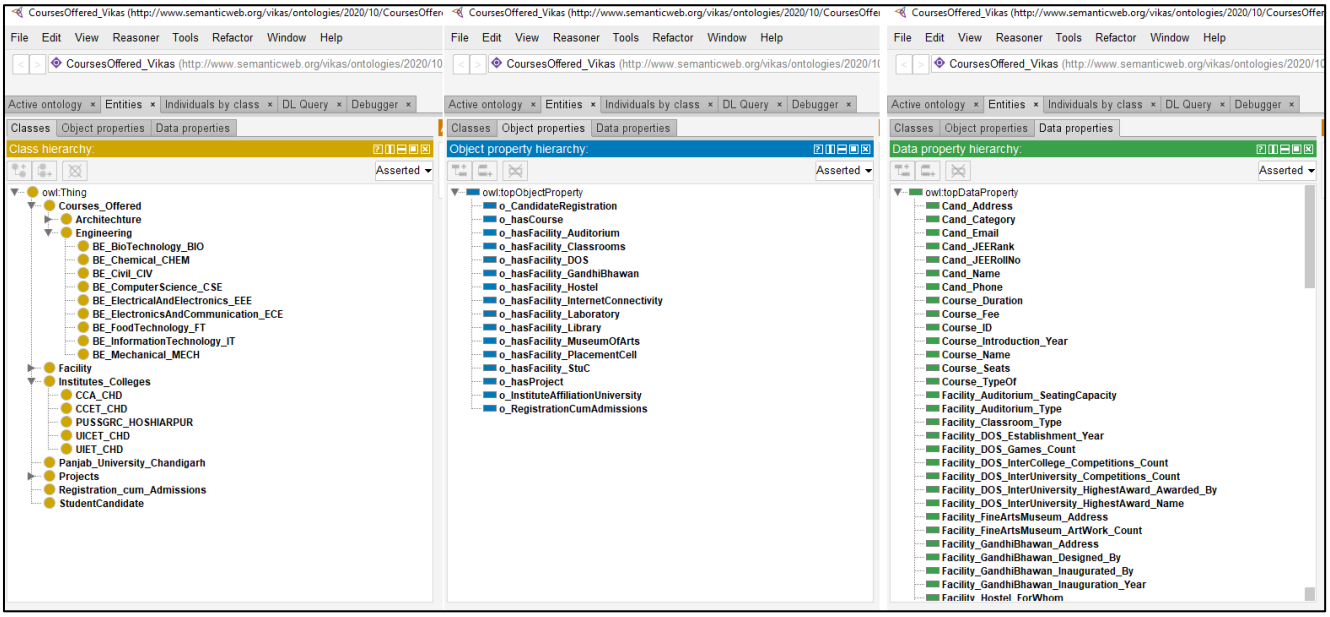

Fig. 3 Tab View of ‘Education’ domain ontology

Table III Example of Class Design of 'Education' domain ontology

Class	Class Instance	Property	Value
Engineering	BE_CSE	- Course_ID	101
		- Course_Name	Computer Science & Engineering
		- Course_Seats	138
		- Course_Fee	106855
Architecture	B_Arch	- Course_ID	501
		- Course_Name	Architecture
		- Course_Seats	44
		- Course_Fee	19434
StudentCandidate	Student	- First_Name	Ajay
		- Last_Name	Singh
		- Rank	12345
		- Eligible	Yes

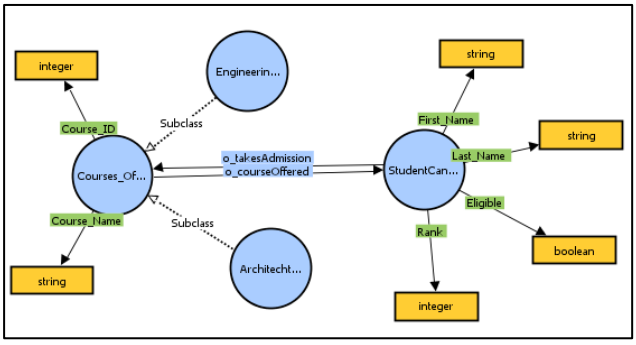

Fig. 4 Partial view of 'Education' domain ontology

3.2.2 Term extraction

In ontology-based systems, term extraction refers to a task of identifying the useful terms from which real world concepts can be mapped. The process of term extraction begins with part-of-speech (POS) tagging and phrase chunking which is done by linguistic techniques. In this study, we employed pre-trained tokenizers provided by the Stanford CoreNLP Group to achieve state-of-the-art results in processing English language text. We conducted a comparative analysis with the outcomes of TextRazor [36] and spaCy-NER [37]. Our input dataset was SCCDD, which lacks token-level annotations in natural language, adding complexity to term and entity extraction tasks. To address the domain variance, we processed each dataset file separately. For each domain within the dataset, we divided the context paragraphs into two parts: a training set (approximately 70%) and a testing set (approximately 30%).

We conducted term extraction in two iterations on the training dataset. In the first iteration (run 1), we utilized the standard pipeline annotators from the Stanford CoreNLP package. However, the results obtained were not highly satisfactory, as they missed many valuable terms. In some cases, only partial terms were extracted or the terms were broken into fragments. This issue arose from the fact that Stanford's POS tagger was not trained on the specific dataset used, namely SCCDD.

For example, the term "sector-25" is parsed into its POS form as follows: "sector/NN" for Noun, "-/HYPH" for a hyphen symbol, and "25/CD" for a cardinal value. In this case, "sector-25" is identified as three distinct terms, although it should ideally be recognized as a single entity categorized as "LOCATION".

In the second iteration (run 2), we utilized rule-based parsing derived from a specialized pipeline annotator in Stanford's toolkit, known as "regexner", facilitated by a rule-based tab-separated text file [32], referred to as "RegexNER.txt." Typically, this file is curated manually by domain experts to achieve superior results when processing English language text. We augment this rule-based dictionary file by adding missing or correcting existing terms. Once the training phase was finalized using the training dataset, we employed the combined process of term extraction and rule-based parsing on the test dataset. To ensure data integrity, we meticulously eliminated duplicate terms from the list of intermediate terms in both the training and test datasets. The findings of our experiments are detailed in Section 4. The process flow model of the proposed term extraction methodology is illustrated in Fig. 5.

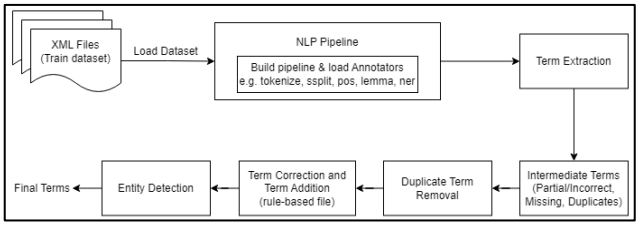

Fig. 5 Process flow model of the proposed methodology

3.2.3 SPARQL

The QA task performed on the leaned ontology hinges on a thorough comprehension of the input question and its

effective execution. To facilitate this, the user's NL question must be transformed into a format compatible with execution on the learned ontology, typically achieved through the use of SPARQL. Prior to formulating SPARQL query, the user entered NL question undergoes pre-processing through linguistic techniques facilitated by Stanford's CoreNLP library. The design encompasses various linguistics, including tokenization, lemmatization, POS tagging, and more [5], followed by sentence simplification, identification of projection words and nominal. A visual representation of the system design for SPARQL query formulation and execution is depicted in Fig. 6 below.

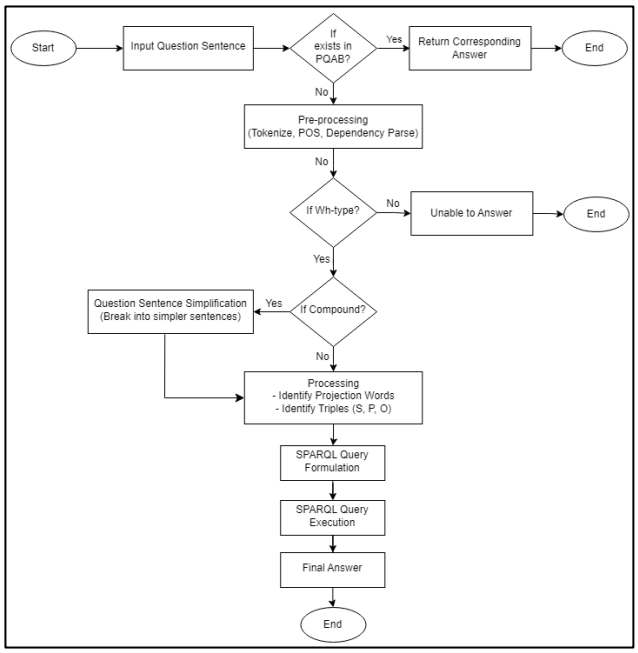

Fig. 6 System Design for Query Analysis and Execution

3.2.3.1. Identification of projection words from NL questions

Identification of projection words and triples within the NL query is a crucial factor in the effective execution of SPARQL construction and execution. We have devised specific rules, leveraging the dependency tree of the NL question and certain POS tag patterns (or syntactic constraints), to streamline the process of SPARQL formulation. Our algorithm receives user questions in the format of a simple sentence comprising various English words. It initiates by examining the presence of coordinating conjunctions like 'and' or 'or' within the Wh-based relational question. Several established syntactic sequences or POS (Part of Speech) patterns as explained in [38], identified by POS taggers, determine whether a sentence is compound or not.

For the purpose of illustration, we have focused solely on questions that involve a relational phrase connecting two nouns within a sentence. Consider an example that includes a relational phrase along with coordinating conjunctions such as 'and' or 'or'. For instance, in the question, *'Which Indian astronaut and researcher was killed in a spacecraft crash?'* The typed dependency parse output is provided in Example 1 below:

root(ROOT-0, killed-7)

det(astronaut-3, Which-1)

amod(astronaut-3, Indian-2)

nsubj:pass(killed-7, astronaut-3)

cc(researcher-5, and-4)

conj:and(astronaut-3, researcher-5)

nsubj:pass(killed-7, researcher-5)

aux:pass(killed-7, was-6)

case(crash-10, in-8)

compound(crash-10, spacecraft-9)

obl:in(killed-7, crash-10)

punct(killed-7, ?-11)

Example 1 Output of Stanford typed dependency parse (in list format)

By employing the Rule (1) outlined in [38] to Example 1 above, the two nouns '*astronaut*' and '*researcher*' are identified as the projection words for the SPARQL query, as demonstrated below:

$$\forall w_x, w_y, w_z . (nsubjpass(w_x, w_y) \wedge conj(w_y, w_z) \Rightarrow Target(w_y) \wedge Target(w_z)) \quad \text{Rule (1)}$$

i.e. nsubjpass(killed, astronaut) ∧ conj(astronaut, researcher)
⇒ Projection word 1 (astronaut) ∧
Projection word 2 (researcher)

Here, 'w_x', 'w_y', 'w_z' denote interdependent words indicated by the dependency function dep(x, y) and dep(y, z). 'nsubj:pass' signifies a 'passive nominal subject' and 'conj:and' represents a 'conjunction word' are the functions utilized in the Stanford Typed Dependency Parser [39] [40].

This rule examines the question sentence, identifies the projection word(s) within the question to serve as SPARQL variable(s) by passing the input sentence to the dependency parser [31] incorporated in the Stanford CoreNLP annotators.

3.2.3.2. Identification of Triples from NL Questions

Triple patterns consist of three components: subject, object, and predicate (SPO). The identification of triples and the formulation of SPARQL queries from NL questions can be a challenging task. In this study, we focused exclusively on simple Wh-based queries that include a relational phrase (forming the predicate part in the query triple) connecting two nouns (forming the subject and object parts in the query triple).

The triple representing a relational statement is determined by extracting the two nominal and the relational phrase that connects them. For instance, in the sentence '*Vikas is enrolled to UIET*', we have '*Vikas*' and '*UIET'* as the two nominals, and '*enrolled*' as the relational phrase. Consequently, the resulting triple consists of '*Vikas*', '*enrolled*', '*UIET*' denoting the subject, predicate, and object in the form {*Vikas, enrolled, UIET*}.

When it's not explicit which nominal a relational phrase (verb) is connecting, two particular scenarios for triple identification arise, and we employ the following rules:

1. When a question begins with '*Who*', for example a sentence '*Who killed Osama?*', we refer to Rule (2) outlined in [38] as follows:

$$\forall w_e, w_h, w_i \,.\, (nsubj(w_e, w_h) \wedge obj(w_e, w_i)) \Rightarrow Triple(?w_h, w_e, w_i)) \quad \text{Rule (2)}$$

$$nsubj(killed, who) \wedge obj(killed, Osama) \Rightarrow Triple\ (?who, killed, Osama)$$

2. In cases where the verb appears as the last word in a sentence and does not occur between the two nouns, as seen in a sentence such as '*In which class does Vikas study?*' we refer to Rule (3) described in [71], which states as follows:

$$\forall w_h, w_i, w_j \,.\, (pobj(w_j, w_h) \wedge nsubj(w_j, w_i)) \Rightarrow Triple(?w_i, w_j, w_h)) \quad \text{Rule (3)}$$

$$pobj(In, class) \wedge nsubj(study, Vikas) \Rightarrow Triple\ (Vikas, study, Class)$$

3.2.3.3. SPARQL query formation and execution

The task of SPARQL query formulation holds great significance in ensuring accurate answers. To facilitate this process, we have incorporated the Apache Jena Library [41], also known as the 'Jena API,' and 'Jena Fuseki' [41], a SPARQL server provided by Apache, into our Java Application. The Jena API allows for the creation of SPARQL queries using either by utilizing the 'SelectBuilder' class from the Jena framework or by using the 'String' class in Java.

For instance, if we wish to retrieve the answer to the question *'Which course has a fee of 70000?'* from the constructed ontology, sample code excerpt (written in Java) for SPARQL execution is shown in Example 2 and a screenshot of the proposed ontology-based QAS prototype demonstrating the retrieved answer is against SPARQL query is depicted in Fig. 7.

```
import java.io.IOException;
import org.apache.jena.ontology.OntModel;
import org.apache.jena.ontology.OntModelSpec;
import org.apache.jena.query.*;
import org.apache.jena.rdf.model.ModelFactory;

public class SPARQLTest_usingSelectBuilder {
     public static void main(String[] args) throws IOException {
        //create a model using reasoner
        OntModel model
                = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM_MICRO_RULE_INF);

        org.apache.jena.arq.querybuilder.SelectBuilder sb
                        = new org.apache.jena.arq.querybuilder.SelectBuilder();

        String SOURCE = "http://localhost:8080/fuseki/SampleOntology_1/data";
        model.read(SOURCE,"RDF/XML");

        sb.addPrefix("owl", "http://www.w3.org/2002/07/owl#");
        sb.addPrefix("xsd", "http://www.w3.org/2001/XMLSchema#");
        sb.addVar("?Course");
        sb.addVar("?Fee");
        sb.addWhere( "?x", "owl:hasFee", "?Fee" );
        sb.addBind("STRAFTER(STR(?x),'#')", "?Course" );
        sb.addFilter("?Fee  = '70000'^^xsd:int");
        sb.addOrderBy("desc(?Course");

        Query query = sb.build() ;
        System.out.println(query);

        System.out.println("--------------------");

        // Execute the query and obtain results
        QueryExecution qe = QueryExecutionFactory.create(query, model);
        org.apache.jena.query.ResultSet results =  qe.execSelect();

        // Output query results
        ResultSetFormatter.out(System.out, results, query);

        qe.close();
    }
}
```

Example 2 SPARQL query formation and execution

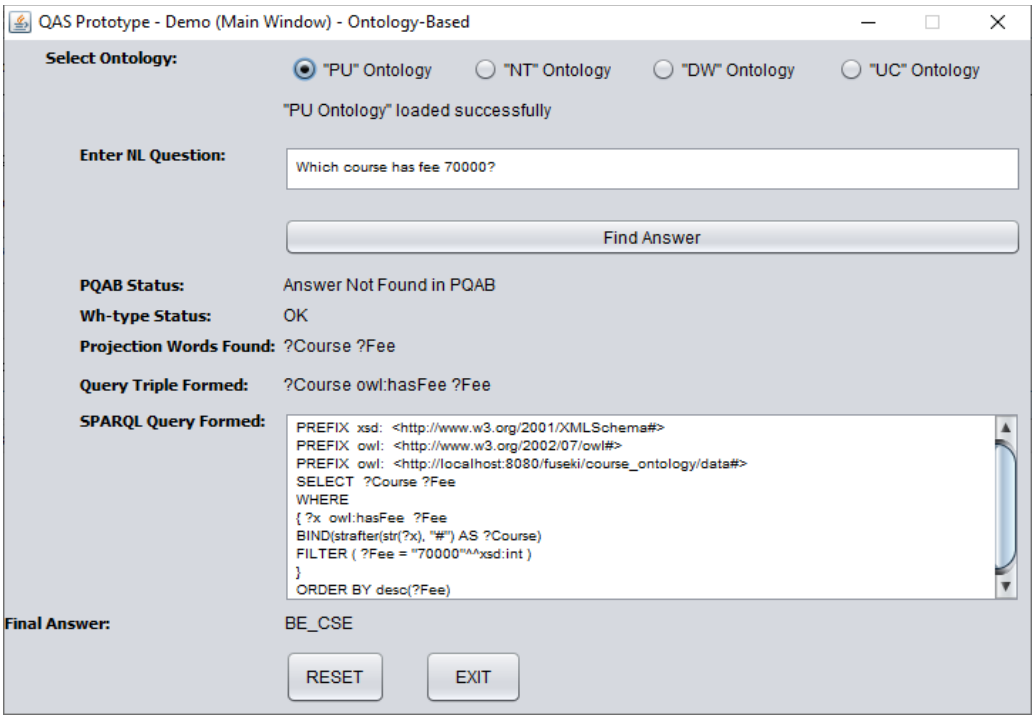

Fig. 7 Screenshot of developed QA prototype

4. Results and discussions

In this work, we performed term extraction on SCCDD to evaluate the efficiency that can be achieved by our proposed Table ***IV*** depicts the confusion matrix for the proposed methodology that is applied on the four independent domains in input dataset. The average Precision, Recall and approach. We used three standard performance metrics [42] [43] to calculate the performance of the proposed term extraction method i.e. Precision, Recall and F-score.

F-score for term extraction task performed by the proposed methodology on a set of 114 context paragraphs is calculated as 83.12%, 78.19% and 80.41%, respectively

Table IV Confusion Matrix for Term Extraction using Proposed Methodology

Sr. No.	No. of Context Paragraphs – Title	TP[1]	FP[2]	FN[3]	Precision	Recall	F-Score
1	27 - PU_and_JAC	291	19	79	93.87	78.65	85.59
2	25 - Nikolas Tesla	147	31	22	82.58	86.98	84.72
3	30 - Doctor Who	179	58	88	75.53	67.04	71.03
4	32 - University of Chicago	338	82	84	80.48	80.09	80.28
	(114 paragraphs)		**Average**		83.12	78.19	80.41

[1] TP='True Positive', [2] FP='False Positive', [3] FN='False Negative'

Table ***V*** depicts the comparative results based on the number of correct terms identified by the proposed methodology, TextRazor and spaCy-NER, alongwith achieved average of Precision, Recall and F-score.

Table V Number of Correctly Identified Terms

Paragraphs Processed →	27 - (PU)	25 - (NT)	30 - (DW)	32 - (UC)	Total Terms	Average Precision	Average Recall	Average F-Score
Proposed Methodology	370	169	267	422	1228	83.12	78.19	80.41
TextRazor	206	112	222	336	876	74.3	89.95	81.17
spaCy-NER	241	115	186	343	885	74.99	85.54	78.44

To assess the performance of proposed QA approach when applied to the SCCDD, we present two confusion matrices. In addition, we used standard performance metrics i.e. Table ***VI*** depicts the confusion matrix for the proposed QA task when applied on the four independent domains within Precision, Recall, F-score, and Accuracy to demonstrate the effectiveness of the proposed QA method.

the input dataset. The average Precision, Recall and F-score for QA task performed by the proposed approach on a set of

548 NL questions are calculated as 85. 82%, 91.02% and 87.62%, respectively. Out of the 548 questions, our proposed approach correctly handled approximately 472 queries, achieving an accuracy of 79.07%.

Table VI Confusion Matrix for QA task: Precision, Recall, F-Score and Accuracy

Sr. No.	# Questions - Title	TP[4]	TN[5]	FP[6]	FN[7]	Precision	Recall	F-score	Accuracy
1	226 - (PU)	175	4	14	33	92.59	84.13	88.16	79.2
2	101 - (NT)	70	2	21	2	76.92	97.22	85.89	75.79
3	94 - (DW)	67	10	19	5	77.91	93.06	84.81	76.24
4	127 - (UC)	104	4	7	12	93.69	89.66	91.63	85.04
	(548 Questions)				Average	85.28	91.02	87.62	79.07

[4] TP='True Positive', [5] TP='True Negative', [6] FP='False Positive', [7] FN='False Negative'

To assess syntactic robustness, we have provided the Confusion Matrix for various Wh-type questions within the SCCDD in
Table ***VII***. It is evident that the system's performance in responding to Wh-type questions that begin with 'Which' is exceptionally strong. This is attributed to the limited frequency of questions commencing with 'Which,' and the system's effective ability to retrieve answers for such queries. However, the system demonstrated excellent performance when addressing questions that initiated with 'Who,' 'How,' and 'Where'.

Table VII Confusion Matrix pertaining to Wh-type Questions within SCCDD

Domain / Title ↓	Wh-type →	How	What / In what	When	Where	Which	Who
PU	TP	44	64	15	22	2	28
	FP	1	11	1	0	0	1
	FN	1	27	0	2	0	3
NT	TP	5	12	24	14	1	11
	FP	1	7	8	2	0	1
	FN	0	4	0	1	0	0
DW	TP	10	27	4	1	1	27
	FP	0	9	9	0	0	3
	FN	0	1	1	0	0	0
UC	TP	23	51	5	1	0	24
	FP	1	5	0	1	0	0
	FN	0	9	0	1	0	2
Total	TP	82	154	48	38	4	90
	FP	3	32	18	3	0	5
	FN	1	41	1	4	0	5
Results	Precision	96.47	82.8	72.73	92.68	100	94.74
	Recall	98.8	79	98	90.5	100	94.7
	F1-score	97.62	80.84	83.48	91.57	100	94.74

Table VIII provides a selection of sample questions that the proposed system addresses. All of these questions share same a common Prefix for their SPARQL queries, which is '*<http://www.semanticweb.org/vikas/ontologies/2020/10/CoursesOffered_Vikas#>*'.

Table VIII Sample questions answered by the proposed approach

Sr. No.	Words in Answer	NL Question / SPARQL Form	Name Entity	Answer
1.		Where is Panjab University currently located?		Chandigarh

Sr. No.	Words in Answer	NL Question / SPARQL Form	Name Entity	Answer
	One word	SELECT ?city WHERE { table:PU_Chd table:PU_Address_City ?city.}	Location (City)	
2.	One word	When was Panjab University established? SELECT ?date WHERE { table:PU_Chd table:PU_Establishment_Year ?date.}	Date	1882
3.	One word	How many university hostels in Panjab University? SELECT ?hostels WHERE { table:PU_Chd table:PU_Hostels_Count ?hostels.}	Number	18
4.	One word	What is the website of Panjab University? SELECT ?website WHERE{ table:PU_Chd table:PU_URL ?website. }	URL	https://puchd.ac.in/
5.	Two words	Who designed the Central Library? SELECT ?person WHERE { table:PU_Library table:Facility_Library_Designed_By ?person.}	Person	Pierre Jeanneret
6.	Three words	What is the start date for Document Verification? SELECT ?Doc_Verification_End_Date WHERE { table:Register table:Reg_Admission_StartDateDocVerification ?Doc_Verification_End_Date.}	Date	28 June 2023
7.	Four words	What group of industries has set up a chair in Telecommunication at UIET? SELECT ?industry_name WHERE { table:UIET table:Institute_Telecommunication_Chair_Owner ? industry_name.}	Organization	Bharti group of industries
8.	Five words	What are Bachelor of Engineering (B.E.) courses in SSBUICET? SELECT ?courseName WHERE { table:UICET table:o_hasCourse ?course. ?course table:Course_Name ?courseName.}	Organization (Customized as Course)	Chemical Engineering and Food Technology
9.	Six words	What is the full form of UIET? SELECT ?fullname WHERE { table:UIET table:Institute_Name ?fullname.}	Organization (Customized as Institute)	University Institute of Engineering and Technology
10.	Seven words	What types of classrooms does CCET have? SELECT ?facility_type WHERE { table:CCET table:o_hasFacility_Classrooms_Type ?facility.	Organization (Customized as Facility)	Virtual Classrooms, Tutorial Rooms and Drawing Halls

Sr. No.	Words in Answer	NL Question / SPARQL Form	Name Entity	Answer
		?facility table:Facility_Classroom_Type ?facility_type.}		
11.	Eight words	What university has UIET signed a MOU with? SELECT ?MOUs_ForeignUniversity_Name WHERE { table:UIET table:Institute_MOUs_ForeignUniversity ?MOUs_ForeignUniversity_Name.}	Organization	University of Western Australia, Nottingham Trent University, UK
12.	Nine words	NIL	NIL	NIL
13.	Ten words	What is SSBUICET currently known as? SELECT ?Institute_Name_Known WHERE { table:UICET table:Institute_Name ?Institute_Name_Known.}	Organization (Customized as Institute)	Dr S.S. Bhatnagar University Institute of Chemical Engineering and Technology
14.	Thirteen words	What are Bachelor of Engineering courses in CCET? SELECT ?courseName WHERE { table:CCET table:o_hasCourse ?course. ?course table:Course_Name ?courseName.}	Organization (Customized as Course)	Civil Engineering, Computer Science and Engineering, Electronics and Communication Engineering and Mechanical Engineering
15.	Nineteen words	What are Bachelor of Engineering courses in UIET? SELECT ?courseName WHERE { table:UIET table:o_hasCourse ?course. ?course table:Course_Name ?courseName.}	Organization (Customized as Course)	Biotechnology Engineering, Computer Science and Engineering, Electronics and Communication Engineering, Electrical and Electronics Engineering, Information Technology and Mechanical Engineering

Table ***IX*** presents comparison results of ontology-based QAS where DBpedia dataset in used for QA purpose. We have considered these QAS because our SCCDD has a portion from SQuAD dataset which is inspired from the DBpedia.

Table IX Comparison results of proposed system vs other ontology-based QAS

QAS	Dataset Used	Total Questions	Precision	Recall	F-measure
Our proposed approach	Self-created dataset	548	0.85	0.91	0.88
UTQA [44]	DBpedia 2015	100	0.82	0.69	0.75
SemGraphQA [44]	DBpedia 2015	100	0.7	0.25	0.37
Xser [45]	DBpedia 2014	50	0.74	0.72	0.73
QAnswer [45]	DBpedia 2014	50	0.46	0.35	0.4
SemGraphQA [45]	DBpedia 2014	50	0.31	0.32	0.31
YodaQA [45]	DBpedia 2014	50	0.28	0.25	0.26
Xser [46]	DBpedia 3.9	50	0.72	0.71	0.72
gAnswer [46]	DBpedia 3.9	50	0.37	0.37	0.37
CASIA [46]	DBpedia 3.9	50	0.32	0.4	0.36
gAnswer [47]	DBpedia 3.8	99	0.4	0.4	0.4
Zhu et al. [48]	DBpedia 3.8	99	0.38	0.42	0.38
RTV [49]	DBpedia 3.8	99	0.32	0.34	0.33
SemSeK [50]	DBpedia 3.7	100	0.44	0.48	0.46
BELA [51]	DBpedia 3.7	100	0.73	0.62	0.67
QAKiS [50]	DBpedia 3.7	100	0.39	0.37	0.38
FREyA [50]	DBpedia 3.6	50	0.63	0.54	0.58
PowerAqua [50]	DBpedia 3.6	50	0.52	0.48	0.5
TBSL [44]	DBpedia 3.6	50	0.41	0.42	0.52

We believe that the accuracy of the proposed system can be further improved by expanding our knowledge base within the constructed ontology and enhancing the linguistic analysis and semantic understanding within the respective domain.

5. Conclusions

In this study, state-of-art ontology based question answering system for English language has been developed. An integrated approach is proposed to that takes care of accepting input in NL text in English language and proposes the novel approach of using external standard packages to speed up the work by reusing the existing functionalities.

Table ***IV*** above.

Table ***VI***, we present the performance metrics for the proposed QA approach, which include a Precision of Table ***IX***, we provide an overview of popular OBQAS, the datasets they utilized, the total number of questions processed, and the performance metrics employed to compare the results of our study. A set of rules with high degree of feasibility is applied to find projection words is helpful in SPARQL query formulation.

In conclusion, it is evident that achieving full automation of the proposed QA system, capable of responding to any input across all domains, is an exceptionally challenging task. Nonetheless, a dedicated endeavour is ongoing to illustrate how an ontology-based automated QA system can be developed through the utilization of domain knowledge and

We processed four independent domains composed of 114 paragraphs of 13477 words and 548 questions in total. The suggested approach for extracting terms is much better than the TextRazor API and spaCy-NER in the field of term extraction. During term extraction task, we have submitted two runs, one run (run 1) is the term extractor using standard pipeline annotators, and the other run (run 2) is term extractor combined with the rule-based parsing obtained from Stanford's special annotator "regexner" to train the NER tagger for custom use-case. This may encourage in system automation and achieve golden standard of NLP performance. Average Precision, Recall and F-score of the proposed method for term extraction on 114 paragraphs in the four given domains is 83.12%, 78.19% and 80.41, respectively as shown in

In

85.28%, a Recall of 91.02%, an F-score of 87.62% and Accuracy of 79.07%. Lastly, in
precise refinement. The proposed approach can be further enhanced by incorporating additional classes, concepts, and individuals into the established ontology.

6. Disclosure instructions

- **Declaration of generative AI and AI-assisted technologies in the writing process**

  During the preparation of this work the author(s) did not use any tool/service to reviewed and edited the content and take(s) full responsibility for the content of the publication. The Authors declares

that there is no use of generative AI and AI-assisted technologies.

- **Funding information**

  This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

- **Conflict of interest**

  The Authors declares that there is no conflict of interest.

- **Data availability**

  The self-created dataset ‘SCCDD’ will be provided on request and ‘SQuAD v1.1’ dataset as referred in this work is freely available to download from [30].

- **Code availability**

  Code may be shared on request.

- **Author contribution**

  First author, Vikas Bali, is fully responsible for the work undertaken in terms of design of work, data acquisition, coding, etc. under the supervision of second author i.e. Dr. Amandeep Verma.

References

[1] L. A. Zadeh, "Chapter 9 From search engines to question answering systems - The problems of world knowledge, relevance, deduction and precisiation," in Fuzzy Logic and the Semantic Web, vol. 1, E. Sanchez, Ed. Elsevier, 2006, pp. 163-210.

[2] L. Zadeh, "From search engines to question-answering systems - The role of fuzzy logic," Progress in Informatics, vol. 1, p. 1, 2005.

[3] C. W. Lee, M. Y. Day, C. L. Sung, Y. H. Lee, T. J. Jiang, C. W. Wu, D. W. Shih, Y. R. Chen and W. L. Hsu, "Boosting Chinese Question Answering with Two Lightweight Methods: ABSPs and SCO-QAT," ACM Transactions on Asian Language Information Processing, vol. 7, p. 12, 2008.

[4] S. Kalaivani and K. Duraiswamy, "Comparison of Question Answering Systems Based on Ontology and Semantic Web in Different Environment," Journal of Computer Science, vol. 8, no. 9, pp. 1407-1413, 2012.

[5] P. M. Athira, M. Sreeja and P. C. Reghuraj, "Architecture of an Ontology-Based Domain-Specific Natural Language Question Answering System," International Journal of Web & Semantic Technology, vol. 4, no. 4, pp. 31-39, 2013.

[6] J. Letkowski, "Doing database design with MySQL," Journal of Technology Research, vol. 6, 2015.

[7] L. Hirschman and R. Gaizauskas, "Natural Language Question Answering: The View from Here," Natural Language Engineering, vol. 7, pp. 275-300, 2001.

[8] "SPARQL Query Language for RDF," W3C, [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/. [Accessed June 2023].

[9] T. Lv, P. Yan and W. He, "Survey on JSON Data Modelling," Journal of Physics: Conference Series, vol. 1069, p. 012101, 2018.

[10] L. Papaleo, "Introduction to XML and its applications," in Handbook of Metadata, Semantics and Ontologies, World Scientific, 2013.

[11] "Java SE at a Glance," Oracle, [Online]. Available: https://www.oracle.com/java/technologies/java-se-glance.html. [Accessed September 2023].

[12] "Java Downloads," Oracle, [Online]. Available: https://www.oracle.com/java/technologies/downloads/#java8-windows. [Accessed September 2023].

[13] "Apache NetBeans," The Apache Software Foundation, [Online]. Available: https://jena.apache.org. [Accessed September 2023].

[14] "Stanford CoreNLP – Natural language software," The Stanford NLP Group, [Online]. Available: https://stanfordnlp.github.io/CoreNLP/. [Accessed September 2023].

[15] R. Barskar, G. Ahmed and N. Barskar, "An Approach for Extracting Exact Answers to Question Answering (QA) System for English Sentences," Procedia Engineering, vol. 30, pp. 1187-1194, 2012.

[16] D. Mollá and J. Vicedo, "Question Answering in Restricted Domains: An Overview," Computational Linguistics, vol. 33(1), pp. 41-61, 2007.

[17] E. M. Voorhees and D. M. Tice, "The TREC-8 Question Answering Track," in Proceedings of the Second International Conference on Language Resources and Evaluation (LREC '00), 2000.

[18] T. Dodiya and S. Jain, "Comparison of Question Answering Systems," Advances in Intelligent Systems and Computing, 1st ed., Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 99-107, 2013.

[19] D. Jin, E. Pan, N. Oufattole, W. H. Weng, H. Fang and P. Szolovits, "What Disease does this Patient Have? A Large-scale Open Domain Question Answering Dataset from Medical Exams," Computation and Language, 2000.

[20] M. Lee, J. Cimino, H. R. Zhu, C. Sable, V. Shanker, J. Ely and H. Yu, "Beyond Information Retrieval - Medical Question Answering," AMIA Annu Symp Proc, p. 469–473, 02 2006.

[21] "HonQA," Health On the Net Foundation, [Online]. Available: https://hon.ch/qa. [Accessed Feb 2017].

[22] S. Dongre and S. Lodhi, "A survey of different semantic and ontology based question answering system," International Journal of Advanced Computational Engineering and Networking, vol. 2, no. 9, pp. 69-74, 2014.

[23] Y. Liu, X. Yi, R. Chen and Y. Song, "A Survey on Frameworks and Methods of Question Answering," in 3rd

International Conference on Information Science and Control Engineering (ICISCE), 2016.
[24] L. Mei, "Intelligent Question Answering System of Research Based Ontology on Excellent Courses," in Fourth International Conference on Computational and Information Sciences, 2012.
[25] A. Bouziane, D. Bouchiha, N. Doumi and M. Malki, "Question Answering Systems: Survey and Trends," vol. 73, pp. 366-375, 2015.
[26] P. Bhatia, R. Madaan, A. Sharma and A. Dixit, "A Comparison Study of Question Answering Systems," Journal of Network Communications and Emerging Technologies (JNCET), vol. 5, no. 2, pp. 192-198, 2015.
[27] V. Bali and A. Verma, "A Study on Components, Benchmark Criteria and Techniques used in Ontology-based Question Answering Systems," International Journal of Intelligent Systems and Applications in Engineering, vol. 10, no. 1s, p. 09–17, 2022.
[28] A. Mishra and S. Jain, "A survey on question answering systems with classification," Journal of King Saud University - Computer and Information Sciences, vol. 28, no. 3, pp. 345-361, 2016.
[29] "Admission and eCounselling Services for Session 2023," Joint Admission Committee Chandigarh, [Online]. Available: https://jacchd.admissions.nic.in/. [Accessed September 2023].
[30] "SQuAD2.0 - The Stanford Question Answering Dataset," The Stanford NLP Group, [Online]. Available: https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/. [Accessed September 2023].
[31] A. Fader, S. Soderland and O. Etzioni, "Identifying relations for Open Information Extraction," Proceedings of the conference on empirical methods in natural language processing (EMNLP), pp. 1535-1545, 2011.
[32] "Software > Stanford RegexNER," Stanford NLP Group, [Online]. Available: https://nlp.stanford.edu/software/regexner.html. [Accessed September 2023].
[33] "Protege," Stanford University, [Online]. Available: https://protege.stanford.edu/. [Accessed June 2023].
[34] "ProtégéVOWL: VOWL Plugin for Protégé," [Online]. Available: http://vowl.visualdataweb.org/protegevowl.html. [Accessed June 2023].
[35] S. Lohmann, S. Negru and D. Bold, "The ProtégéVOWL Plugin: Ontology Visualization for Everyone," Proceedings of ESWC 2014 Satellite Events, vol. 8798, p. 395–400, 2014.
[36] "TextRazor - The Natural Language Processing API," TextRazor Ltd., [Online]. Available: https://www.textrazor.com/. [Accessed September 2023].
[37] "spaCy Named Entity Recognizer (NER)," Text Analysis Online, [Online]. Available: https://textanalysisonline.com/spacy-named-entity-recognition-ner. [Accessed September 2023].
[38] P. Ochieng, "PAROT: Translating natural language to SPARQL," Expert Systems with Applications: X, vol. 5, p. 100024, 2020.
[39] "Stanford Dependencies," Stanford NLP Group, [Online]. Available: https://nlp.stanford.edu/software/stanford-dependencies.html. [Accessed June 2023].
[40] M. C. Marnee and C. D. Manning, "Stanford typed dependencies manual," 2010.
[41] "Apache Jena," The Apache Software Foundation, [Online]. Available: https://jena.apache.org/download/index.cgi. [Accessed September 2023].
[42] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness & correlation," Journal of Machine Learning Technologies, vol. 2(1), p. 37–63, 2011.
[43] H. Dalianis, "Evaluation Metrics and Evaluation," in Clinical Text Mining: Secondary Use of Electronic Patient Records, Cham: Springer International Publishing, 2018, p. 45–53.
[44] D. Diefenbach, V. Lopez, K. Singh and P. Maret, "Core Techniques of Question Answering Systems over Knowledge Bases: a Survey," vol. 55, 2018.
[45] C. Unger, C. Forascu, V. Lopez, . A. -C. Ngomo and E. Cabrio, "Question Answering over Linked Data (QALD-5)," In: Working notes for CLEF 2015 conference, 2015.
[46] C. Unger, C. Forascu, V. Lopez, A. -C. Ngomo and E. Cabrio, "Question answering over linked data (QALD-4)," In: Working notes for CLEF 2014 conference, vol. 1180, p. 1172–1180, 2014.
[47] L. Zou, R. Huang, H. Wang, J. Yu, W. He and D. Zhao, "Natural language question answering over RDF - A graph data driven approach," Proceedings of the ACM SIGMOD International Conference on Management of Data, 2014.
[48] C. Zhu, K. Ren, X. Liu, H. Wang, Y. Tian and Y. Yu, "A Graph Traversal Based Approach to Answer Non-Aggregation Questions over DBpedia," in Semantic Technology, Springer International Publishing, p. 219–234, 2016.
[49] P. Cimiano, V. Lopez, C. Unger, E. Cabrio, A. -C. Ngonga Ngomo and S. Walter, "Multilingual Question Answering over Linked Data (QALD-3): Lab Overview," in Information Access Evaluation. Multilinguality, Multimodality, and Visualization, p. 321–332, 2013.
[50] V. Lopez, C. Unger, P. Cimiano and E. Motta, "Evaluating Question Answering over Linked Data," Journal of Web Semantics, vol. 21, 2013.
[51] S. Walter, C. Unger, P. Cimiano and D. Bar, "Evaluation of a Layered Approach to Question Answering over Linked Data," in The Semantic Web, p. 362–374, 2012.