International Journal on Emerging Technologies **12**(1): 25-30(2021)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

Performance of RDF Library of Java, C# and Python on Large RDF Models

Mustafa Ali Bamboat[1], Abdul Hafeez Khan[2] and Asif Wagan[3]
[1]Department of Computer Science, Sindh Madressatul Islam University (SMIU), Karachi, (Sindh), Pakistan.
[2]Department of Software Engineering, Sindh Madressatul Islam University (SMIU) Karachi, (Sindh), Pakistan.
[3]Department of Computer Science, Sindh Madressatul Islam University (SMIU), Karachi (Sindh), Pakistan.

(Corresponding author: Mustafa Ali Bamboat)
(Received 03 November 2020, Revised 22 December 2020, Accepted 28 January 2021)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: The semantic web is an extension of the traditional web, in which contents are understandable to the machine and human. RDF is a Semantic Web technology used to create data stores, build vocabularies, and write rules for approachable LinkedData. RDF Framework expresses the Web Data using Uniform Resource Identifiers, which elaborate the resource in triples consisting of subject, predicate, and object. This study examines RDF libraries' performance on three platforms like Java, DotNet, and Python. We analyzed the performance of Apache Jena, DotNetRDF, and RDFlib libraries on the RDF model of LinkedMovie and Medical Subject Headings (MeSH) in aspects measuring matrices such as loading time, file traversal time, query response time, and memory utilization of each dataset. SPARQL is the RDF model's query language; we used six queries, three for each dataset, to analyze each query's response time on the selected RDF libraries.

Keywords: dotNetRDF, Apache Jena, RDFlib, LinkedMovie, MeSH.

I. INTRODUCTION

In the 1990s, we have conventional web pages understandable by the human brain,consisting of links to other web pages or documents [1]; computers process a file and display the contents written in the HTML format. However, the file content is not machine-understandable. The conventional web extended its reach into the semantic web around 1997, seeking to make the material relevant to computer machines and to mutate web documents into web data [1].

The semantic web represents data so that humans and computers can understand it and perform various queries based on the content rather than traditional web links [2]. In 1999, Tim Berners-Lee expressed his vision of the web as a vast database [3], where the relational model cannot manage such massive data, so he came up with the semantic model, which stores data into triples [4]. The semantic web mostly depends on formal ontology. According to Gruber, ontology narrates concepts and relations in a well-organized manner rather than other knowledge representation models, such as glossaries, taxonomies thesaurus, and many more [5]. There are standardizer and recommended technologies by W3C to encode semantic data based on formal ontology, such as Resource Description Framework (RDF), Web Ontology Language (OWL), and Darpa Agent Markup Language (DAML). To facilitate the integration and interoperability of ontology and described formally in logic-based syntaxes [3, 6].

Fundamentally, RDF is the standard framework to express web data, which can identify the web resource (pages) and things by using URI's (Uniform Resource Identifiers) and explain the resources in the form of triples, where a triple consisting of subject, predicate, and object [7]. In 2008, the RDF Data Access Working Group (DAWG) acknowledged a language specifically designed to retrieve and manipulate the RDF model known as SPARQL, an acronym for SPARQL Protocol and RDF Query Language [8]. There are three forms of the query in SPARQL, *SELECT, CONSTRUCT,* and *ASK.*

Various comparisons have published on RDF Framework, and Libraries such as RDF Triple stores [4], Ontology development in dot Net [2, 6], and many comparisons between RDF Database and Relational Database, even Empirical study of RDF libraries performed between two languages, Java and C# or Python and Java. This paper compares the open-source RDF libraries of Apache Jena, dotNetRDF, and RDFLib based on Java, C#, and Python languages, respectively, to compare the query response time among them. The pattern of the paper is as follows. Section II presents related studies. Section III presents the methods, section IV presents the study results, and a conclusion ends the paper in the last section.

II. RELATED WORKS

There are many libraries developed to implement a machine-understandable Web. Therefore, in this section, we explorer the studies conducted on RDF libraries.

Authors measured the loading time, query execution time, query response time, and storage capacity of OWL/RDF Ontologies in a dot NET environment using two libraries, dotNetRDF, and SemWeb, respectively. According to their analysis, SemWeb performance is much better on small and medium datasets, whereas, on the other hand, dotNetRDF shows better performance on small datasets. Both libraries quit harder to implement for beginner programmers due to a lack of Graphic User Interface support [2].

Authors have selected ten ontologies datasets fetched from the internet such as OntoDPM, WikiMovie, Agriculture and Forestry Ontology, Tero, Aksiomitveke, lexvo, Gene Ontology (GO), Gene, and Drug Ontology (DRON) and performed the following measures:

Loading Time: Ontology datasets loaded in the memory at least ten times each to calculate the average loading time [2]. They have observed that dotNetRDF quickly loaded small datasets. In contrast, SemWeb has better loading time on a medium and large dataset. Furthermore, dotNetRDF failed to load Gene and DRON.

Query Execution Time: They have observed that dotNetRDF is two times speedy than Sem Web because of SPARQL query execution time.

Query Response Time: They noticed that dotNetRDF query response time is quicker than SemWeb, except for the Gene (GO), Gene, and DRON dataset, dotNetRDF has thrown OutOfMemoryException.

Overall, both libraries' performance is average on medium and small datasets. dotNetRDF has a speedy response time compared to SemWeb, whereas the loading time of SemWeb is much better than dotNetRDF on medium and large size datasets. It is also noticeable that dotNetRDF failed to load some datasets and thrown OutOfMemoryException at query response time [2].

Authors evaluated and compared the two dotNET based tools for ontology, namely, dotNetRDF and SemWeb, freely available. The dotNetRDF and SemWeb APIs are developed in C#, loads the RDF data in-memory, and support the SPARQL engine for querying the loaded RDF dataset. The authors adopted metrics-based frameworks consisting of six factors, such as a General description of tools, Software architecture and evolution, Interoperability with other ontology development tools and languages, Knowledge representation, Inference services attached to the tool, and Usability, whereas for the performance comparison, authors select ten ontology datasets to analysis the APIs in terms of Loading Time (LT), Query Execution Time (QET), and Query Response Time (QRT). The QET and QRT calculated using a SPARQL query that retrieves all classes and subclasses of datasets. The study illustrates that on medium-size ontologies, SemWeb performs better, and dotNetRDF offers a rapid response compared to SemWeb on small ontologies. However, none of these methods could process OWL ontologies with very minimal internal memory storage [6].

Authors have broadened the comparison of dotNET based RDF libraries SemWeb and dotNetRDF against Open-Source libraries such as Apache Jena, Protégé, and RDF4J. In contrast, Jena and SemWeb are command-line interfaces, and Protégé, RDF4J, and dotNetRDF have both GUI and command-line interfaces. The authors applied a qualitative approach to comparison metrics collected from the literature and a quantitative approach to experiments and analysis of these libraries' performance. They parted comparison into four layers, development, storage media, ontology retrieval, and comparison layers [9]. Same metric measures as discussed in [2] used in this study, i.e., loading time, query execution time, and query response time, conducted on the selected five datasets, OntoDPM, WikiMovie, Tero, Gene, and AFO.

Based on the loading time results, it is noticeable that Jena has shown tremendous performance, followed by Protégé and RFD4J. In contrast, SemWeb performance was much better than dotNetRDF, but it failed to load the Gene dataset.

The authors' used the SPARQL query of 'SELECT' form on the five datasets, and their results showed that Jean and RFD4J had achieved faster query execution time. In contrast, SemWeb and dotNetRDF are slower, and even they failed to execute a SPARQL query on Gene due to memory lacking.

Based on the Query Response Time results in [9], they observed that Jena, RDF4J, and Protégé have speedy response time compared to the rest of the libraries. In contrast, SemWeb and dotNetRDF failed to accomplish the query results.

Overall, it reveals that open-source libraries (Jean, Protégé, and RDF4J) much better than dotNET based libraries, and there is more room for improvement dotNetRDF and SemWeb libraries [9].

Researchers compared six ontology editing tools accessible in Desktop and Online versions, such as OntoStudio3.1, Protégé 5.0, SWOOP, TODE, OwlGrEd, and Odase. The parameters, such as architecture, storage, interoperability, library, and GUI design, were considered in this research. The study concluded that the selected six ontology editing tools have user-friendly interfaces and perform almost identical tasks; in addition, it depends on the level of experience of the user and the size of the ontology to choose the appropriate tool; in the authors' opinion, Protégé is ideal for beginners, followed by SWOOP, while OWLGrED is preferable for UML notation or visualization, and for the large size ontologies OntoStudio is suitable [22].

III. METHODOLOGY

In this paper, we appraised RDF libraries' performance in DotNET, Java, and Python. We measured the loading, traversal, query response times, and memory usage of the whole process.

A. Datasets

The first part of this study was mostly in analyzing the different datasets; we have chosen two datasets of various sizes; details are as under:

Table 1: RDF Datasets of different size.

Dataset	Alias used in this paper	Total Triples	Format	File Size
LinkedMovie	DS-1	3,579,616	N-Triples	428 MB
Medical Subject Headings (MeSH)	DS-2	16,442,022	N-Triples	1.91 GB

Linked Movie Dataset (DS-1): The majority of peoples are interested in movies; they liked to discuss various film contents, such as the birthdates of all actors in the specific film or any relation among actors of any particular movie. Unfortunately, the traditional web cannot answer these questions because it does not perform content-based searching compared to the semantic web. *LinkedMovie* is the RDF dataset, which contains entities of movies, actors, directors, even it provides relationships between all of these entities [12].

It was consisting of 3,579,616 triples in the N-Triples format and occupied a disk size of 428MB.

Furthermore, it furnishes various Linking Open Data (LOD) cloud datasets, such as DBpedia, YAGO, Geonames, and Rotten Tomatoes. In contrast, LOD is the powerful blend of Linked Data and Open Data, which broaden the scope of SPARQL queries related to relationships among all entities.

Medical Subject Heading-MeSH Dataset (DS-2): In 2014, the National Library of Medicine established a group named as Linked Data Infrastructure Working Group to form NLM linked data for the semantic web. One of this group's significant tasks is maintaining NLM datasets quality and providing more robust linking between NLM and datasets on the web. MeSH-RDF consists of vocabulary thesaurus come by medical books, journals, and audiovisuals. The structure of MeSH-RDF is a three-tier; the first tier comprises of 'Descriptors (headings), Qualifiers (subheadings), and Supplementary Concept Records (chemicals, drugs, rare diseases).'The second tier comprises 'Concepts' that are the collection of synonymous Terms, and the third-tier comprises 'Terms.' In other words, MeSH-RDF easy our job to fetch NLM data of any level. This study uses the currently available MeSH-RDF dataset consisting of 16,442,022 triples in the N-Triples format and the occupied disk size of 1.91GB [13].

B. RDF Libraries

This paper evaluated the RDF libraries' performance of three different platforms, Java, DotNET, and Python. This section explained the implementation of these libraries in C#, Java, and Python Languages.

Apache Jena (Java): It is an open-source framework of Java, which provides API to draw out data or modify the RDF graphs [9]. We have found that Jena is very stable and robust to fetch RDF data and provide much betterLinkedData features. It supports various RDF formats for both parser and writer, such as Turtle, RDF/XML, N-Triples, JSON-LD, RDF/JSON, TriG, N-Quads, Trix, and RDF Binary. In Jena, the application code written in Java directly acknowledged the RDF API, whereas Ontology API and SPARQL API are the RDF API subsets. It also facilitates the optional features of Inference API and Store API [14].

We developed a Java program using Jena API in Eclipse IDE to execute each RDF dataset file [10,11]. First, we create a model by using the 'ModelFactory' class, which provide standards kinds of models, then 'RDFDataMgr' class open the RDF file of N-Triples format, which return 'InputStream,' used to build a model bypassing as an argument to the 'read' method of the model. After that, a 'QueryFactory' class method 'create' is used to form the SPARQL query from the given string, which returns an object of the 'Query' class. 'Query' object passed to QueryExecution class along with the dataset object 'model' on which query will execute by using the 'exeSelect()' method of QueryExecution class, it returns a 'ResultSet' of the executed query.

As compared to other libraries, Jena is the most reliable API and easy to use; even naive programmers can easily install it. We found Jena Tutorial quite helpful in achieving our goals in no time. Jena is capable of handling various sizes of datasets, from small to larger.

dotNetRDF (DotNET): It is an open-source .Net API of RDF developed in C# language, and it shows good performance on the smaller to medium size datasets. It also supports third-party stores, such as AllegroGraph, 4store, and Virtuoso [9, 15].

We have created a console application in C# using dotNetRDF API in Visual Studio 2019 for the 4.0 dotNET Framework. It was easier to install the API by the built-in tool of NuGet Package Manager in VS-2019. First, we created an IGraph object, an interface of Graphs used to form Graphs mathematically. As mentioned in the UserGuide [16], dotNetRDF supports the same formats as Jean except for the RDF Binary; in our program, we used NTriplesParser classto load the RDF file of N-Triples format, passing the IGraph object as a parameter, it loads the file in memory. To parse the string's SPARQL query, we used SparqlQueryParser class, which returns an object of SparqlQuery. We call a method of SparqlQuery named 'ExecuteQuery,' which returns an object of SparqlResultSet, which contains our query results.

While programing in this API, we observed that working in it not easier as compare to Jena. It needs a moderate to a high level of programming skills to implement. The User-Guide of dotNetRDF is not comprehensive, and not much help is found over the web than Jena.

RDFLib (Python): An open-source RDF library developed in Python for the semantic web; it consists of various parsers to support almost all RDF formats, such as Turtle, N-Triples, and JSON-LD. RDFLib supports both in-memory and persistent Graph to perform RDF manipulation and SPARQL quires on it [17]. In our study, we used RDFLib 5.0, a stable version of it so far. Python programmers can efficiently work in it.

We have created a console program in Python using RDFlib API in Eclipse IDE. Python emphasizes the indenting programming in readability aspects, which is an annoying behavior for the Java programmer. It provides the kick start documentation, which requires basic knowledge of Python programming; by following the RDFlib documentation, we created a Graph object and used the 'parse' function to load the dataset by explicitly defining the format the RDF file as the second parameter of this function. After that, we called the 'query' function to get the results of the executed quires.

We have examined that RDFLib is the slowest library of Python to manipulate or query the RDF Graphs; besides that, a third-party library named 'RedLands' developed in C language is speedy as compare to it.

C. Queries

To measure the performance of each RDF library, we used three SPARQL queries of each dataset. This section explained these queries and their expected results. Query Q1, Q2, and Q3 are for theLinkedMovie dataset [18, 20], and Query Q4, Q5, and Q6 are for the MeSH dataset [21].

Query-1 (Q1) – "Cast of Pulp Fiction and Number of Movies Acted In." This query fetches all actors who acted in the movie "Pulp Fiction" and counts the number of films they worked in their whole acting career.

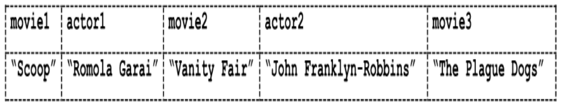
actor	movies
"Harvey Keitel"	61
"Samuel L. Jackson"	61
"Christopher Walken"	58
"Brue Willis"	49
"John Travolta"	42
"Steve Buscemi"	39

Fig. 1. Expected Result of Query-1 (LinkedMovie-Dataset).

The expected results of this query shown in Fig. 1, observed that it returns two columns names of actors and several movies against each actor in which they acted.

Query-2 (Q2) – "Six Degrees of SPARQL." This query is on the famous game of Six Degrees of Kevin Bacon, where six or fewer peoples are not connected socially to each other, somehow linked in a manner without knowing each other personally. This query fetched the

results of five movies and the connecting actors between together.
The expected outcome result of this query, shown in Fig. 2, returns the name of movies and actors related to each other somehow, either as a competitor or as a co-worker of actor Hugh Jackman or Kevin Bacon, or both. For example, RomolaGarai British actress was in the movie Scoop-2006 with Hugh Jackman, whereas the same actress acted with Geoff Bell, who was in the movie "You Should Have Left" with Kevin Bacon.



movie1	actor1	movie2	actor2	movie3
"Scoop"	"Romola Garai"	"Vanity Fair"	"John Franklyn-Robbins"	"The Plague Dogs"

Fig. 2. Expected Result of Query-2 (LinkedMovie-Dataset).

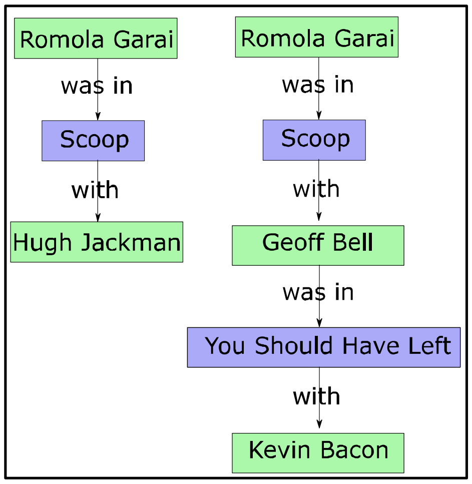
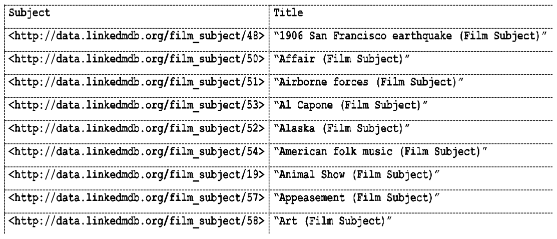


Fig. 3. Graphical View of Six Degree of Kevin Bacon, taken from The Oracle of Bacon [19].

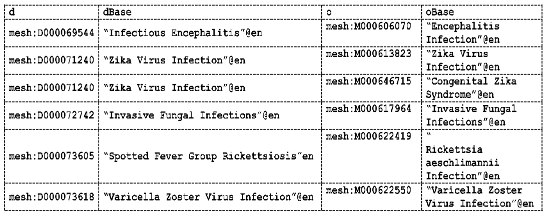
Query-3 (Q3) – "Display identifier and title of all movie topics defined in Linked Movie Database." This query fetched 179 records of all subjects along with their titles as declared in the datasets, a UNION operator used to combine the results of SKOS (Simple Knowledge Organization – a web dataset) and LinkedMovie dataset and return the DISTINCT records based on subject and title, as shown in the Fig. 4.



Subject	Title
<http://data.linkedmdb.org/film_subject/48>	"1906 San Francisco earthquake (Film Subject)"
<http://data.linkedmdb.org/film_subject/50>	"Affair (Film Subject)"
<http://data.linkedmdb.org/film_subject/51>	"Airborne forces (Film Subject)"
<http://data.linkedmdb.org/film_subject/53>	"Al Capone (Film Subject)"
<http://data.linkedmdb.org/film_subject/52>	"Alaska (Film Subject)"
<http://data.linkedmdb.org/film_subject/54>	"American folk music (Film Subject)"
<http://data.linkedmdb.org/film_subject/19>	"Animal Show (Film Subject)"
<http://data.linkedmdb.org/film_subject/57>	"Appeasement (Film Subject)"
<http://data.linkedmdb.org/film_subject/58>	"Art (Film Subject)"

Fig. 4: Expected Result of Query-3 (LinkedMovie-Dataset).

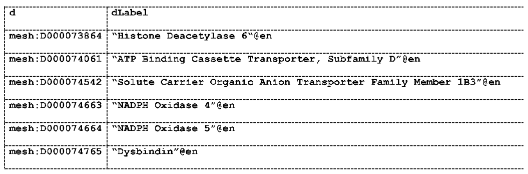
Query-4 (Q4) – "Retrieve all MeSH descriptors or concepts with infection anywhere in its name." This query fetches the 321 records of all descriptors and their Concept and the name of the infection.



d	dBase	o	oBase
mesh:D000069544	"Infectious Encephalitis"@en	mesh:M000606070	"Encephalitis Infection"@en
mesh:D000071240	"Zika Virus Infection"@en	mesh:M000613823	"Zika Virus Infection"@en
mesh:D000071240	"Zika Virus Infection"@en	mesh:M000646715	"Congenital Zika Syndrome"@en
mesh:D000072742	"Invasive Fungal Infections"@en	mesh:M000617964	"Invasive Fungal Infections"@en
mesh:D000073605	"Spotted Fever Group Rickettsiosis"en	mesh:M000622419	" Rickettsia aeschlimannii Infection"@en
mesh:D000073618	"Varicella Zoster Virus Infection"@en	mesh:M000622550	"Varicella Zoster Virus Infection"@en

Fig. 5. Expected Result of Query-4 (MeSH-Dataset).

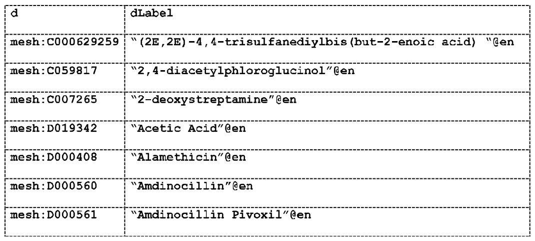
Query-5 (Q5) – "Find all active MeSH descriptors with an allowable qualifier of 'adverse effects.'" This query fetches the 1000 records of all descriptors along with their names of harmful effects.



d	dLabel
mesh:D000073864	"Histone Deacetylase 6"@en
mesh:D000074061	"ATP Binding Cassette Transporter, Subfamily D"@en
mesh:D000074542	"Solute Carrier Organic Anion Transporter Family Member 1B3"@en
mesh:D000074663	"NADPH Oxidase 4"@en
mesh:D000074664	"NADPH Oxidase 5"@en
mesh:D000074765	"Dysbindin"@en

Fig. 6. Expected Result of Query-5 (MeSH-Dataset).

Query-6 (Q6) – "Descriptors and SCRs that have the Pharmacological Action 'Anti-Bacterial Agents'." This query fetches the 422 records of all descriptors along with their names of Anti-Bacterial.



d	dLabel
mesh:C000629259	"(2E,2E)-4,4-trisulfanediylbis(but-2-enoic acid) "@en
mesh:C059817	"2,4-diacetylphloroglucinol"@en
mesh:C007265	"2-deoxystreptamine"@en
mesh:D019342	"Acetic Acid"@en
mesh:D000408	"Alamethicin"@en
mesh:D000560	"Amdinocillin"@en
mesh:D000561	"Amdinocillin Pivoxil"@en

Fig. 7. Expected Result of Query-6 (MeSH-Dataset).

IV. RESULTS AND ANALYSIS

A. Hardware and Software Tools

We have performed this study on the AMD hardware A4-6300B @ 3.70 GHz processor, 8 GB RAM, and Windows 10 operating system.
Three open-source RDF libraries, dotNetRDF, Apache Jena, and RDFLib, are used to compare performance among them. The IDEs utilized are Microsoft Visual Studio 2019 for the dotNetRDF library using C# programming language and Eclipse for Jena and RDFlib libraries using Java and Python programming languages.

B. Performance Metrics

We examined RDF libraries' performance on their loading time, traversal time, query response time, and memory usage of the complete process.
Loading Time: We have evaluated the loading time in seconds of each dataset using dotNetRDF, Jena, and RDFlib APIs. Table 2 reflects these APIs' outcomes, the loading time taken by each RDF library in seconds. We have examined that Jean and RDFlib have loaded datasets successfully, whereas dotNetRDF has failed to load the MeSH dataset.
Fig. 8 depicts the loading time data presented in Table 2; it reflects that RDFlib has taken a colossalloading time compared to Jena and dotNetRDF, whereas dotNetRDF failed to load the MeSH RDF file due to larger size. In contrast, Jena's performance is outstanding from the rest of the libraries.

Table 2: Loading Time of Datasets in RDF Libraries.

Datasets	Apache Jena	dotNetRDF	RDFlib
LinkedMovie	63 sec	189 sec	419 sec
MeSH	249 sec	-	7416 sec

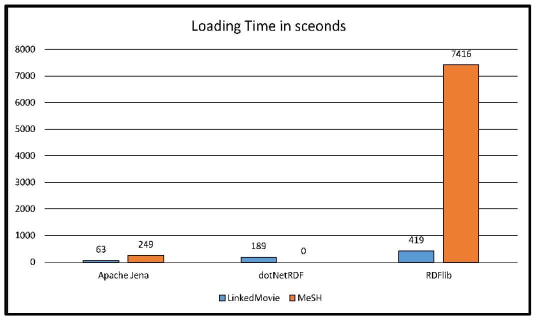


Fig. 8. Loading Times of RDF Datasets in Apache Jena, dotNetRDF and RDFlib.

In contrast, Jena's performance is outstanding from the rest of the libraries.

Traversal Time: We have computed the RDF file's traversal time, results expressed in Table 3, presents each programming language's file traversal time on each dataset, the graphical representation of traversal time values in Fig. 9, we have examined that Python has taken maximum traversal time on both datasets compared to others; on the other hand, C# has shown much better performance as compared to Java.

Table 1: Traversal Time of Datasets.

Datasets	Java	C#	Python
LinkedMovie	58 sec	37 sec	665 sec
MeSH	318 sec	187 sec	790 sec

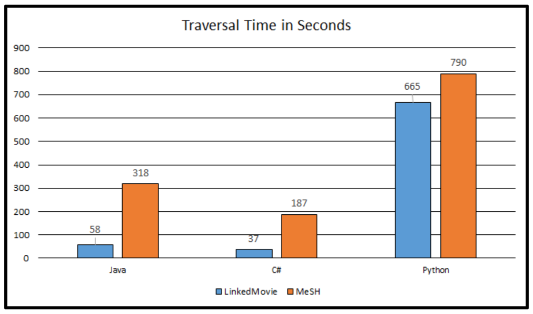


Fig. 9. Traversal Times of RDF Datasets.

Query Response Time:We have executed six SPARQL queries, three for each dataset; in this sub-section, we use aliases against each query such as Q1, Q2, Q3, Q4, Q5, and Q6. SPARQL Queries, as explained in the sub-section of METHODOLOGY, are implying in this study.

Table 4 reflects the response times of each query executed using RDF libraries. We examined that dotNetRDF failed to load MeSH RDF; therefore, Q4 to Q6 failed to execute, and Q2 throws TimeOutException. The default timeout value is 3 minutes (180,000ms); even after increasing its default value up to a maximum limit does not return results of Q2.

Fig. 10 depicts the query response time data, as reflected in Table 4; we observed RDFlib's maximum time in Q2 to return records compared to others. There is a mixed result of Jean and RDFlib. On Q1 and Q2, Jean showed outstanding performance, whereas, on Q3 to Q6, RDFlib results are much better than Jena. dotNetRDF performance cannot be measured except Q1 and Q3, whereas the rest of the query executions are successful.

Table 2: Query Response Time of Datasets in Apache Jena, dotNetRDF and RDFlib.

Query	Apache Jena	dotNetRDF	RDFlib
Q1	2.06 sec	4 sec	4 sec
Q2	26 sec	Timeout Exception	180 sec
Q3	0.32 sec	0.18 sec	0.13 sec
Q4	11 sec	-	7 sec
Q5	3.45 sec	-	0.05 sec
Q6	1.29 sec	-	0.05 sec

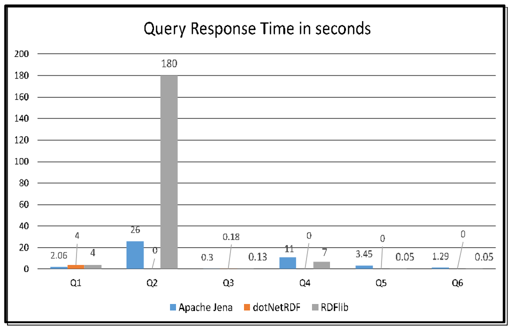


Fig. 10. Query Response Time of SPARQL queries on each RDF Library.

Memory Usage: We have examined the total memory utilization in GBs, of the RDF libraries, from the loading process to all queries' response.

Fig. 11 depicts the total memory utilization of Jena, dotNetRDF, and RDFlib, as dotNetRDF failed to load the MeSH RDF; therefore, we observed that dotNetRDF had utilized 7.29 GB for file traversal time computing only. In contrast, Jena's memory utilization is much better than the rest of the libraries' memory utilization.

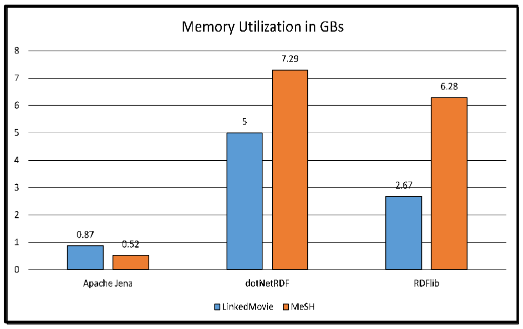


Fig. 11. Memory Utilization of Jena, dotNetRDF, and RDFlib.

V. CONCLUSION

We have conducted this study to evaluate the performance of open-source RDF libraries, Apache Jena, dotNetRDF, and RDFlib, developed in Java, C#, and Python programming languages. Various measure metrics such as loading time, traversal time, query response time, and memory utilization used to compute each library's performance. We have used two RDF datasets of N-Triples format of different sizes, LinkedMovie consists of 3,579,616 triples, and Medical Subject Heading (MeSH) consists of 16,442,022 triples. Linked Movie dataset is the collection of movies, actors, directors, and the relationships among them. In contrast, the MeSH dataset containing vocabulary thesaurus from medical books, journals, and audiovisuals. We have used six SPARQL queries and three queries for each dataset. Apache Jena and RDFlib implemented in Java and Python programming using Eclipse IDE, whereas dotNetRDF was implemented in the C# program using Microsoft Visual Studio 2019.

We have examined that Jena's installation and implementation were quick and easy, followed by dotNetRDF and RDFlib, query Q1, Q2, and Q3 related to LinkedMovie Q4, Q5, and Q6 are related to the MeSH dataset. Jena has taken 63 seconds and 249 seconds to load LinkedMovie and MeSH, respectively;dotNetRDF failed to load MeSH, whereas RDFlib took a huge loading time compared to Jena.

Python took maximum traversal time compared to C# and Java. On the other hand, we have computed Query Response Time by executing six queries on each library, divided into three queries for each dataset. Fig. 10 represented that Jena and RDFlib have executed all six queries. In contrast, dotNetRDF has thrown Time Out Exception at the time of Q2 execution and unable to load MeSH, so the execution of Q4 to Q6 failed on it. In aspects of Memory Usage, Jena has shown outstanding performance, followed by RDFlib and dotNetRDF, as presented in Fig. 11.

This study has witnessed that dotNetRDF is suitable for smaller or medium-sized datasets, whereas the RDFlib is the Python's slowest RDF library. In contrast, the overall performance of Apache Jena is tremendous; it can support various sizes of datasets from smaller to larger. On the other hand, both dotNetRDF and RDFlib took a massive amount of time and maximum memory usage to process larger RDF datasets.

VI. FUTURE SCOPE

The next step of this will be to evaluate other RDF libraries, and we will use larger RDF datasets. It will help us to find the more in-depth performance of the libraries

Conflict of Interest. There is no conflict of interest in this work.

REFERENCES

[1]. Dineshpathak, A. (2021). Semantic Web. Retrieved from https://devopedia.org/semantic-web.

[2]. Mahoro, L. J., & Fonou-Dombeu, J. V. (2019). An Empirical Evaluation of dot NET-Based Tools for OWL/RDF Ontologies Processing. 2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD). doi:10.1109/icabcd.2019.8851017.

[3]. Semantic Web (2021). Retrieved fromhttps://en.wikipedia.org/wiki/Semantic_Web.

[4]. Banane, M., & Belangour, A. (2019). A Comparative Study of RDF Triple Stores. *SSRN Electronic Journal*. doi:10.2139/ssrn.3349399.

[5]. Eine, B., Jurisch, M., & Quint, W. (2017). Ontology-Based Big Data Management. Systems, *5*(3), 45. doi:10.3390/systems5030045.

[6]. Fonou-Dombeu, J. V., & Kadiata, V. K. (2019). Ontology Development in dot NET Platform: An Empirical Assessment. *2019 International Conference on Advances in Big Data, Computing and Data Communication Systems* (icABCD). doi:10.1109/icabcd.2019.8851027.

[7]. Nacional, T., Niinimaki, M., & Heikkurinen, M. (2019). Rdf Databases – Case Study and Performance Evaluation. MATTER: *International Journal of Science and Technology, 5*(3), 01-14. doi:10.20319/mijst.2019.53.0114.

[8]. SPARQL. (2021). Retrieved from https://en.wikipedia.org/wiki/SPARQL.

[9]. Mahoro, L. J., & Fonou-Dombeu, J. V. (2020). A Comparative Analysis of dot NET-Based and Open Source Platforms for Ontologies Development. *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems* (icABCD). doi:10.1109/icabcd49160.2020.9183887

[10]. Linkedmdb - dataset by linked-data. (2017). Retrieved from https://data.world/linked-data/linkedmdb

[11]. MeSH RDF Technical Documentation. (n.d.). Retrieved from https://hhs.github.io/meshrdf/

[12]. Hassanzadeh, O., & Consens, M. P. (1970). [PDF] Linked Movie Database: Semantic Scholar. Retrieved from https://www.semanticscholar.org/paper/Linked-Movie-Data-Base-Hassanzadeh-Consens/0bf1c4ddc8f32b96666bbae0dafdc591254a99a5.

[13]. Bushman, B., Anderson, D., & Fu, G. (2015). Transforming the Medical Subject Headings into Linked Data: Creating the Authorized Version of MeSH in RDF. *Journal of Library Metadata, 15*(3-4), 157-176. doi:10.1080/19386389.2015.1099967.

[14]. Getting started with Apache Jena. (n.d.). Retrieved from https://jena.apache.org/getting_started/

[15]. DotNetRDF. (n.d.). Retrieved from https://www.w3.org/2001/sw/wiki/DotNetRDF

[16]. Dotnetrdf. (n.d.). Dotnetrdf/dotnetrdf. Retrieved from https://github.com/dotnetrdf/dotnetrdf/wiki/UserGuide

[17]. Rdflib 5.0.0. (n.d.). Retrieved from https://rdflib.readthedocs.io/en/stable/

[18]. Linkedmdb - dataset by linked-data. (2017). Retrieved from https://data.world/linked-data/linkedmdb

[19]. The Oracle of Bacon. (n.d.). Retrieved from https://oracleofbacon.org/movielinks.php

[20]. Lecture Slides from Artificial Intelligence Laboratory, "SPARQL QUERY LANGUAGE," slide no. 10, URL: http://ai.fon.bg.ac.rs/wp-content/uploads/2015/04/SPARQL-examples-20141.pdf

[21]. MeSH RDF Technical Documentation. (n.d.). Retrieved from https://hhs.github.io/meshrdf/sample-queries

[22]. Rastogi, N., Verma, P., & Kumar, P. (2017). Analyzing Ontology Editing Tools for Effective Semantic Information Retrieval. *International Journal of Engineering Sciences and Research Technology (IJESRT).* doi: 10.5281/zenodo.571593.

How to cite this article: Bamboat, M. A., Khan, A. H. and Wagan, A. (2021). Performance of RDF Library of Java, C# and Python on Large RDF Models. *International Journal on Emerging Technologies, 12*(1): 25–30.