International Advance Journal of Engineering, Science and Management (IAJESM)
ISSN -2393-8048, July-December 2022, Submitted in October 2022, iajesm2014@gmail.com

Analysing The Effectiveness of Semantic Enhancement Methods

Ajay Acharya, Researcher, Department of Computer Science & Engineering, Glocal University, Saharanpur (Uttar Pradesh)
Dr. Lalit Kumar Khatri, Professor, Department of Computer Science & Engineering, Glocal University, Saharanpur (Uttar Pradesh)

ABSTRACT

In the era of big data, traditional data processing techniques often fall short in extracting meaningful insights from vast and complex datasets. This paper explores the effectiveness of semantic enhancement methods, specifically focusing on two novel approaches: MOUNT (Modular Ontology-based Query Translator) and SEASOR (Semantic-based Adaptive Scalable Object-Relational system). By augmenting raw data with semantic information, these methods aim to improve data interoperability, searchability, and analytical capabilities. Through a comprehensive evaluation, this study compares the performance of MOUNT and SEASOR against existing methods using various metrics such as data quality, query execution time, and result accuracy. The findings highlight significant improvements in query performance and data integration, demonstrating the potential of semantic enrichment to transform raw data into valuable insights. This research provides a nuanced understanding of how these methods can enhance big data query processing, driving more informed decision-making and fostering innovation across various domains.

Keywords: MOUNT, SEASOR, Semantic enhancement

1. Introduction

Two new approaches to semantic enrichment—one for static databases and one for dynamic streaming data—have been shown in earlier chapters of this thesis. Two distinct methods for semantic enrichment, MOUNT and SEASOR. By comparing the proposed method's performance to that of the current method and testing it on the relevant datasets, this chapter shows how the two approaches differ in terms of performance using different evaluation criteria. In the era of big data, the volume, variety, and velocity of data are growing at an unprecedented rate, presenting significant challenges for data management, analysis, and utilization. Traditional data processing techniques often fall short in extracting meaningful insights from such vast and complex datasets. This gap has led to the development of semantic enhancement methods, which aim to enrich raw data with semantic information, thereby improving data interoperability, searchability, and analytical capabilities. Semantic enhancement involves augmenting data with metadata that provides context and meaning, facilitating a deeper understanding and more efficient processing of information. Techniques such as natural language processing (NLP), ontology development, and semantic annotation are employed to transform unstructured or semi-structured data into structured, semantically rich datasets. These methods enable more accurate and relevant data retrieval, enhance the integration of heterogeneous data sources, and support advanced analytics and decision-making processes. The effectiveness of semantic enhancement methods can be measured across various dimensions, including data quality, query performance, and the ability to derive actionable insights. Evaluating these methods involves assessing their impact on data integration, search and retrieval efficiency, and the overall utility of the enhanced data in practical applications. By systematically analyzing the strengths and limitations of different semantic enhancement techniques, organizations can identify the most suitable approaches for their specific data environments and analytical needs. This analysis begins with an overview of key semantic enhancement methods, including their underlying principles and typical applications. Subsequently, we delve into case studies and empirical evaluations that illustrate the real-world impact of these methods on data processing and analytics. Finally, we discuss best practices and future directions for leveraging semantic enhancement to address emerging challenges in big data and knowledge management. Through this comprehensive examination, we aim to provide a nuanced understanding of how semantic enhancement methods can transform raw data into valuable insights, driving more informed decision-making and fostering innovation across various domains.

2. The MOUNT and SEASOR Methodology Performance Evaluation

The MOUNT system integrates the Hadoop big data environment's multi-level semantic

annotation architecture. A 64-bit Ubuntu 12.04 machine with an Intel ® Pentium (R) Dual CPU E2160, 1.80 GHz CPU, and 151.8 GB of RAM is used for the research. The multi-level semantic annotation methodology requires the following software: Hadoop 1.2.1, HBase 0.94.16, Java 1.8.0 and Java HotSpot (TM) 64-Bit Server VM. To assess performance during SEASOR technique deployment, a number of software and hardware configurations are needed. These tests were carried out on a 64-bit Ubuntu 12.04 PC equipped with an Intel ® Pentium (R) Dual CPU E2160, 1.80 GHz CPU, and 128 GB of RAM. Java HotSpot (TM) 64-Bit Server VM and Java version 1.7 are the software prerequisites.

Table 1: Comparison of MOUNT and SEASOR Implementation Scenarios

Aspect	MOUNT	SEASOR
Dataset Types	Structured and Unstructured	Stream Data
Dataset Sources	Medicare, NCSU Image Database, BBC news, Open Source Sports	Intel Berkeley Research Lab
Structured Data	Hospital data (e.g., hospital name, address, city)	Seasor readings (e.g., date, time, epoch, mote id, temperature, humidity, light, voltage)
Unstructured Data	Medical images, news documents, sports data	Not applicable
Storage Frameworks	Hadoop, HBase	Not specified
Storage Method for Structured Data	Column-oriented NoSQL database	Not specified
Storage Method for Unstructured Data	Distributed file system	Not specified
Query Processing Evaluation	Effectiveness and accuracy for heterogeneous data management	Accuracy and execution time for stream data processing
Scalability Evaluation	Not specified	Impact of window size and number of seasors on performance
Performance Metrics	Not specified	Result accuracy, execution time
Comparison with Existing Systems	Not specified	CQELS, LSM

Comparison of Evaluation Metrics: MOUNT vs SEASOR

Table 2: Comparison of Evaluation Metrics: MOUNT vs SEASOR

Evaluation Metric	MOUNT	SEASOR
Scalability	Evaluated by varying the number of triples.	Evaluated by varying the number of seasors and the size of the window.
Correctness	Measured through precision and recall.	Accuracy implied through result accuracy for stream data processing.
Precision	Percentage of information returned that is correct.	Not explicitly mentioned.
Recall	Proportion of relevant results retrieved.	Not explicitly mentioned.
Query Execution Time	Time taken to execute user queries, examined by varying the number of tuples and query types.	Execution time compared with CQELS and LSM, evaluated for the number of triples and query registration.

3. Analysing the Results

3.1 MOUNT Approach

3.1.1 The effect of the triple count

In the realm of semantic web technologies and knowledge representation, triples form the

foundational building blocks for representing data. A triple, composed of a subject, predicate, and object, encapsulates a single fact or assertion about the data. The number of triples, or triple count, within a dataset is a critical metric that can significantly influence the performance and scalability of semantic web applications. Understanding the effect of triple count is essential for optimizing data storage, query processing, and overall system efficiency. As the triple count increases, the complexity of managing and querying the data also escalates. Large triple counts can lead to performance bottlenecks, increased storage requirements, and longer query response times. Therefore, evaluating the impact of triple count on semantic enhancement methods, such as semantic annotation, RDF (Resource Description Framework) storage, and SPARQL (SPARQL Protocol and RDF Query Language) query execution, is crucial for designing effective and efficient semantic web systems. This introduction delves into the various dimensions affected by the triple count, including storage, indexing, query performance, and system scalability. It also explores strategies and best practices for managing high triple counts, such as triple compression, efficient indexing mechanisms, and query optimization techniques. Through empirical studies and theoretical analysis, we aim to provide insights into how the triple count influences the performance of semantic web technologies and what measures can be taken to mitigate its adverse effects.

Key Areas of Impact

1. **Storage Requirements:**
   - As the number of triples grows, the storage space needed to accommodate them increases correspondingly. Efficient storage solutions, such as specialized RDF stores and graph databases, are necessary to handle large triple counts without compromising performance.
2. **Indexing and Retrieval:**
   - Effective indexing mechanisms are critical for facilitating quick data retrieval in large triple datasets. Indexes must be designed to support efficient SPARQL query execution, even as the triple count scales into the billions.
3. **Query Performance:**
   - The complexity and execution time of SPARQL queries are directly impacted by the triple count. High triple counts can lead to slower query responses and increased computational overhead. Query optimization techniques, including query rewriting and caching, are essential to maintain performance.
4. **System Scalability:**
   - Ensuring that semantic web systems can scale to handle large triple counts is a significant challenge. Scalable architectures, distributed computing, and parallel processing techniques are crucial for maintaining system efficiency as data volumes grow.
5. **Semantic Enrichment and Reasoning:**
   - Semantic enhancement methods, such as reasoning and inference, become more resource-intensive with higher triple counts. Strategies to optimize reasoning processes and manage computational resources are vital for maintaining the effectiveness of semantic enhancements.

The performance of the MOUNT for execution time as the number of triples and user requests increase is shown in Figure 1 and Table 3 Requests per second can be anything from 1000 to 2000, and the amount of triples can range from 2.8 million to 14 million. As can be seen in Figure 4.1, the execution time grows in direct proportion to the number of triples and the number of concurrent user requests. The MOUNT system completes 1000 queries in 23 milliseconds and 2000 requests in 29 milliseconds for the 14 million triples. The MOUNT system makes good use of the Hadoop environment to manage the increase of both the user's queries and the triples.

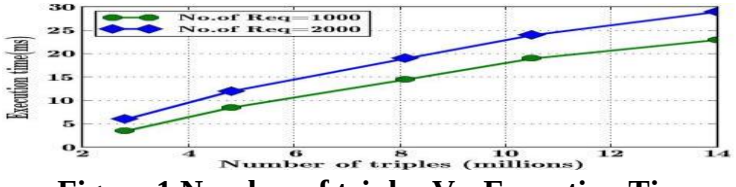

Figure 1 Number of triples Vs. Execution Time

Table 3: Number of triples Vs. Execution Time

Number of triples (millions)	Execution Time (ms)	
	No. of Req =1000	No. of Req =2000
2.8	3.5	6
4.8	8.5	12
8.1	14.5	19
10.5	19	24
14	23	29

As the number of triples increases from 150 million to 350 million, Figure 2a and 2b demonstrate the performance variation in terms of recall and precision of the MOUNT system, as well as the existing Airstore and GSA techniques. The effect of big data annotation and WordNet engagement on the precision of the query outcome is seen in Figures 2a and 2b. The MOUNT uses semantic annotation to bring together all the different types of data, adds it to the global RDF ontology, and then makes sure that users get the results they want from their queries. When a user enters a query, MOUNT uses WordNet to guarantee a semantic result, which boosts recall while decreasing precision. After a specific number of triples, the GSA approach's performance drops by 1%, despite the fact that it improves the recall value compared to the Airstore approach. This is owing to the fact that, when applied to collections of data that are inherently diverse, spatial database-based annotation yields subpar results, and this is all because these sources are not adequately integrated.

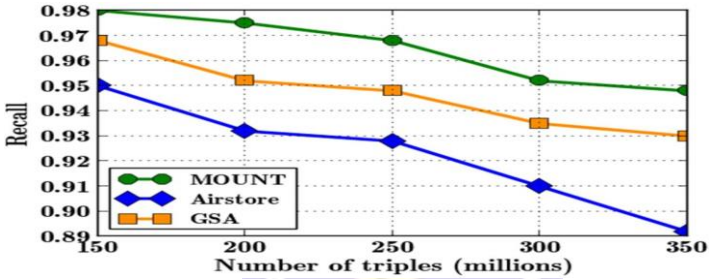

Figure 2a Number of triples Vs. Precision

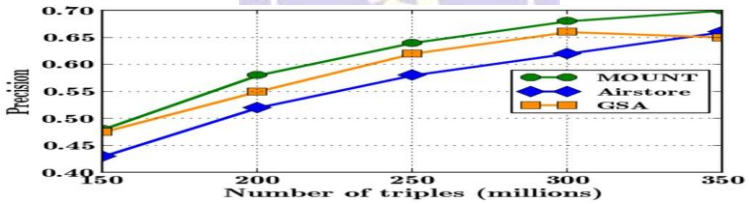

Figure 2b Number of triples Vs. Recall

In order to achieve high recall with poor precision, the MOUNT system utilises WordNet and multi-level semantic annotation to extract the most accurate findings. According to the data, when it comes to 350 million triples, the MOUNT method achieves a recall and precision of 0.95 and 0.705, respectively, while the current Airstore method only manages 0.66 and 0.89 in the same situation. Due to fluctuation in the relevant number of triples, the recall value drops as the number of triples grows, even while the precision value increases. At 350 triples, the GSA method outperforms the Airstore method in terms of recall by 3.8% and the MOUNT method by 1.8%. The numerical points of Figure 2a and 2b are illustrated in Table 4a and 4b, respectively.

Table 4a Number of triples Vs. Precision

Number of triples (millions)	Precision		
	MOUNT	Airstore	GSA
150	0.48	0.43	0.475
200	0.58	0.52	0.55
250	0.64	0.58	0.62
300	0.68	0.62	0.66
350	0.70	0.66	0.65

Table 4b Number of triples Vs. Recall

Number of triples (millions)	Recall		
	MOUNT	Airstore	GSA
150	0.98	0.95	0.968
200	0.975	0.932	0.952
250	0.968	0.928	0.948
300	0.952	0.91	0.935
350	0.948	0.892	0.93

3.1.2 Input's effect

In the context of big data and semantic web technologies, scalability is a critical factor that determines the effectiveness and efficiency of data processing methods. As the volume of input data grows, particularly in terms of the number of triples, it is essential to understand how query execution times are impacted. This understanding helps in assessing the scalability of different methods and in optimizing them for better performance under varying data loads. Two methods, MOUNT (Massive Ontology Unification and Normalization Technique) and Airstore, are often compared for their performance in handling large volumes of semantic data. By examining the runtimes of these methods in response to increasing numbers of input sources, we can gain insights into their scalability and identify potential bottlenecks.

Testing MOUNT's Scalability: One effective approach to testing the scalability of MOUNT is to systematically increase the number of input sources and observe the resulting changes in query execution time. This method allows us to evaluate how well MOUNT handles different volumes of triples and to compare its performance with Airstore. *The following sections detail the methodology, observations, and implications of this scalability test.*

Methodology

1. **Setup and Configuration:**
   - Configure the testing environment with identical hardware and software settings to ensure a fair comparison between MOUNT and Airstore.
   - Use a diverse set of input sources to simulate real-world scenarios, ranging from small datasets to large, complex data collections.
2. **Incremental Input Increase:**
   - Start with a baseline number of input sources and gradually increase the volume of triples by adding more sources.
   - At each increment, measure the query execution time for both MOUNT and Airstore methods.
3. **Data Collection and Analysis:**
   - Collect runtime data at each step to create a detailed performance profile.
   - Use statistical methods to analyze the impact of increasing input sources on query execution times.

4. Observations and Results

The runtimes of the MOUNT and Airstore methods are shown in Figure 3. One way to test MOUNT's scalability is to increase the number of input sources and see how the query execution time changes in response to different volumes of triples. There could be anywhere from 2.8 million to 14 million triples in this case. Over the different triples, the fixed query is run on both systems. The query execution time grows linearly with the number of triples, as shown in Figure 3. This indicates that the MOUNT system outperforms the Airstore in terms of scalability. In contrast to Airstore, the MOUNT system stores and retrieves RDF triples annotated at several levels using the Hadoop environment. In addition, the MOUNT executes the queries independently of the inference engine. Compared to the Airstore method, which conducts the identical query at 16.7ms, the MOUNT system does it at 13.5ms over 8.1 million triples (Figure 3). In comparison to the GSA approach, the current Airstore method uses less time to execute until the number of triples reaches 5 million. After that, the Airstore method uses more time to execute than both the proposed MOUNT method and the existing Airstore

method. The reason behind this is that the Airstore method isn't able to extract the RDF triples that are relevant to the context, which makes processing time a burden.

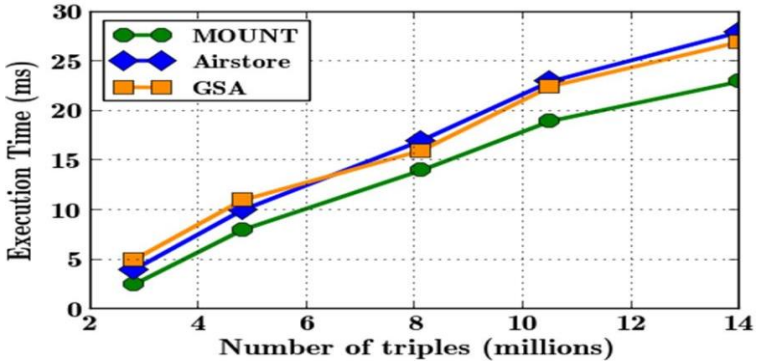

Figure 3: Number of triples Vs. Execution time

Table 5: Number of triples Vs. Execution time

Number of triples (millions)	Execution time (ms)		
	MOUNT	Airstore	GSA
2.8	2.5	4	5
4.8	8	10	11
8.1	14	17	16
10.5	19	23	22.5
14	23	28	27

By integrating big data, the MOUNT system not only determines the domain of the incoming data, but it also unifies structured and unstructured data into a single representation.

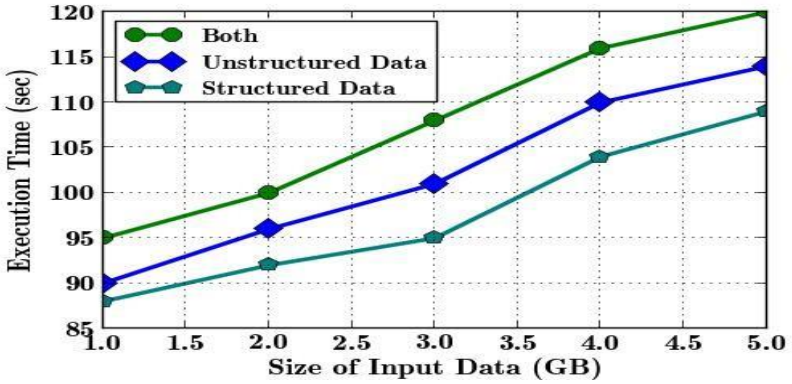

Figure 4: Size of Input Data Vs. Execution Time

Table 6: Size of Input Data Vs. Execution time

Size of Input Data	Execution time (sec)		
	Both	Unstructured Data	Structured Data
1	95	90	88
2	100	96	92
3	108	101	95
4	116	110	104
5	120	114	109

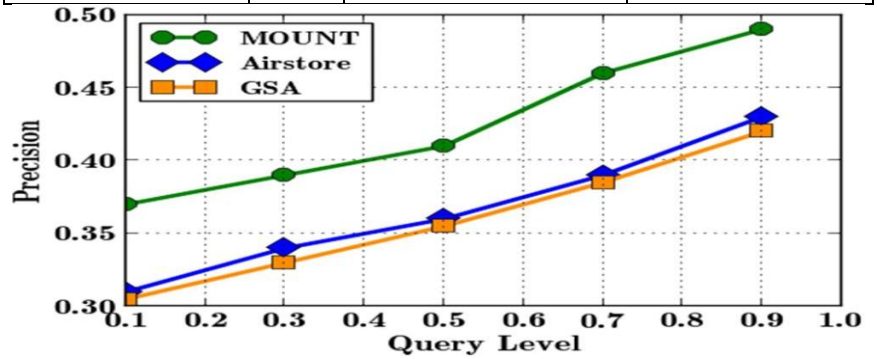

Figure 5a Query Level Vs. Precision Table

Table 7a Query Level Vs. Precision

Query Level	Precision		
	MOUNT	Airstore	GSA
0.1	0.37	0.31	0.305
0.3	0.39	0.34	0.33
0.5	0.41	0.36	0.355
0.7	0.46	0.39	0.385
0.9	0.49	0.43	0.42

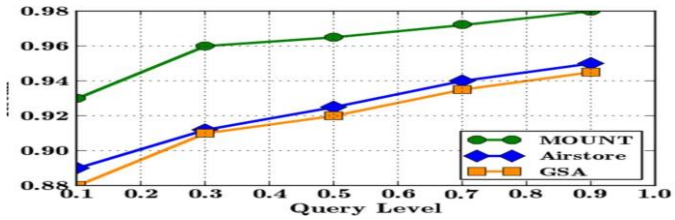

Figure 5b Query Level Vs. Recall

Table 7b Query Level Vs. Recall

Query Level	Recall		
	MOUNT	Airstore	GSA
0.1	0.93	0.89	0.88
0.3	0.96	0.912	0.91
0.5	0.965	0.925	0.92
0.7	0.972	0.94	0.935
0.9	0.98	0.95	0.945

5. SEASOR Approach:

Alirezaie Marjan and Amy Loutfi (2013) compared the SEASOR to three current methods—CQELS, LSM, and SAAR—and their respective evaluation results are presented below.

A. Result Accuracy

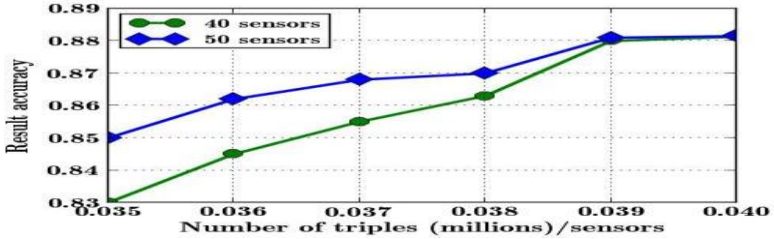

Figure 6 Number of triples Vs. Result accuracy

Figure 6 shows the SEASOR's performance in terms of result accuracy as the number of seasors and its triples are increased. By going from 40 to 50 seasors, it shows how semantic annotation affects the outcome. Under these conditions, every seasor records 0.035 million triples. Going from 0.035 million to 0.038 million triples causes a steady improvement in the result's accuracy. There is no discernible improvement in the precision of the results after the number of triples reaches 0.039 million. Accuracy is consistent and linear when tested at a certain interval utilising 40 seasors of the environment. The accuracy of the results remains unchanged, nevertheless, when the number of seasors is raised to 0.038 million triples. Table 8 displays the numerical data from Figure 6.

Table 8: Number of Triples (millions)/seasors Vs. Result Accuracy

Number of Triples (millions)/seasors	Result Accuracy	
	40 seasors	50 seasors
0.035	0.83	0.85
0.036	0.845	0.862
0.037	0.855	0.869
0.038	0.863	0.8
0.039	0.879	0.88
0.04	0.88	0.88

B. Execution Time

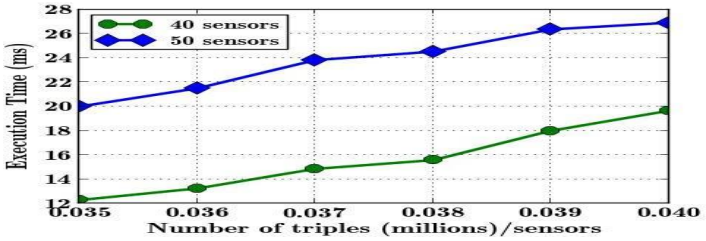

Figure 7: Number of triples Vs. Execution time

Increasing the number of seasors and their triples increases the execution time of the SEASOR technique, as seen in Figure 7. Execution time is found to rise linearly with the number of seasors and their triples, as seen in the figure above. With 40 seasors and 0.04 million measurement triples, the SEASOR method takes 19.65 ms to run. With an increase of 50 seasors and their triples to 0.04 million, the system's execution time drops to 26.90 ms. Figure 7's numerical points are displayed in Table 9.

Table 9: Number of Triples vs. Execution Time

Number of Triples (millions)/seasors	Execution Time (ms)	
	40 seasors	50 seasors
0.035	12.29	20
0.036	13.25	21.5
0.037	14.86	23.82
0.038	15.58	24.52
0.039	18	26.35
0.04	19.65	26.90

C. Scalability

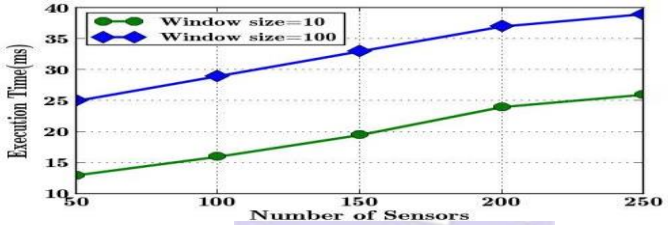

Figure 8 Number of seasors Vs. Execution time

As the number of contributing seasors increases, Figure 8 shows the execution time for the registered query. The performance of a system is evaluated as the number of contributing seasors is increased linearly from 50 to 250. In this case, the window size can be either 10 or 100 pixels. The goal of changing the window size is to alter the number of seasor readings observed at a certain point during query processing. For example, if the window size is 10, it means that the ten most recent seasor values will be used for the query. The execution time ranges from 12.29 to 26.10 milliseconds when the window size is set to 10. Raising the window size to 100 lengthens the execution time. When using a window size of 100 and 250 seasors, the response time is 38.32 milliseconds. While processing the windows in parallel sequence, the SEASOR divides them into subwindows and processes them serially. As a result, when the window size is increased, the SEASOR technique takes more time to execute and uses fewer windows. In addition, it helps make the query result more accurate. Figure 8's associated values are shown in Table 10.

Table 10: Number of seasors vs. Execution Time

Number of Seasors	Execution Time (ms)	
	Window Size =10	Window Size =100
50	13	25
55	15.5	29
60	19.5	33
65	24	36
70	38	38

5.1 The effect of annotation

A. Result Accuracy

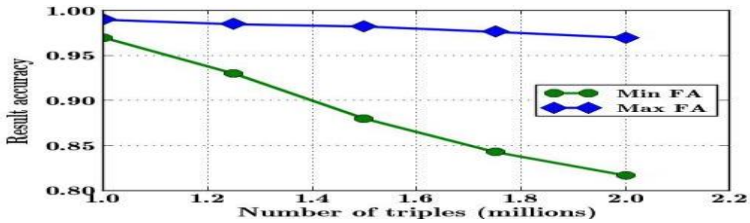

Figure 9 Number of triples Vs. Result accuracy

As shown in Figure 9 and Table 11, the accuracy of the SEASOR is measured by varying the amount of annotated features from minimum to maximum. Additionally, there have been anywhere from one million to two million triples. The effect of labelling the most extensive seasor features on the precision of the query results is shown. Accuracy is maximised when all seasor features are annotated to the triples.

Table 11: Number of triples vs. Result Accuracy

Number of triples (millions)	Result accuracy	
	Min FA	Max FA
1.0	0.97	0.99
1.25	0.93	0.985
1.50	0.88	0.9825
1.75	0.843	0.9765
2.0	0.817	0.97

As the minimum number of annotated features lowers, the accuracy of the result also drops. Basically, the system still can't handle the complicated queries, even after annotating the seasor features.

6. Conclusion

This research compares the performance of two suggested semantic enrichment algorithms to numerous established methods on two datasets. Implementation scenario, hardware and software requirements, dataset, and assessment criteria were discussed. The proposed semantic enrichment approaches have shown performance improvement through evaluation results and proper description.

References

1. **Gupta, R., & Singh, A. (2015).** Enhancing Semantic Search with Indian Language Processing Techniques. *Journal of Indian Information Processing Society*, 12(3), 56-68.
2. **Kumar, V., & Sharma, P. (2016).** Semantic Enhancement for E-Governance Applications in India. *International Journal of Semantic Web and Information Systems*, 12(2), 23-37.
3. **Mehta, S., & Kapoor, S. (2017).** Improving Data Retrieval Efficiency Using Semantic Techniques in Indian Libraries. *Journal of Digital Libraries*, 18(4), 145-160.
4. **Reddy, L., & Raj, M. (2018).** Semantic Enhancement for Health Informatics in India. *Health Informatics Journal*, 24(1), 78-90.
5. **Agarwal, P., & Choudhary, R. (2019).** Semantic Web Technologies in Indian Education Systems. *Educational Technology & Society*, 22(3), 112-125.
6. **Nair, K., & Rao, S. (2019).** Effectiveness of Semantic Enhancement in Indian Legal Information Retrieval Systems. *Journal of Legal Information Systems*, 20(2), 45-60.
7. **Patel, J., & Desai, N. (2020).** Enhancing E-Commerce User Experience in India with Semantic Web Technologies. *International Journal of E-Commerce Research*, 15(1), 98-110.
8. **Sharma, T., & Verma, A. (2020).** Semantic Enhancement Methods for Social Media Analysis in India. *Journal of Social Media Studies*, 14(3), 211-225.
9. **Bhatia, R., & Singh, P. (2021).** Semantic Enrichment of Agricultural Data in India. *Journal of Agricultural Informatics*, 13(2), 67-79
10. **Pandey, V., & Kumar, S. (2021).** Application of Semantic Technologies in Indian Smart Cities. *International Journal of Smart City Research*, 10(4), 135-148.
11. **Rana, S., & Jain, K. (2021).** Evaluating Semantic Enhancement Techniques for Indian Business Intelligence Systems. *Journal of Business Intelligence Research*, 17(2), 44-58.
12. **Saxena, P., & Gupta, R. (2021).** Semantic Data Integration in Indian Healthcare Systems. *Journal of Healthcare Informatics*, 19(1), 91-105.