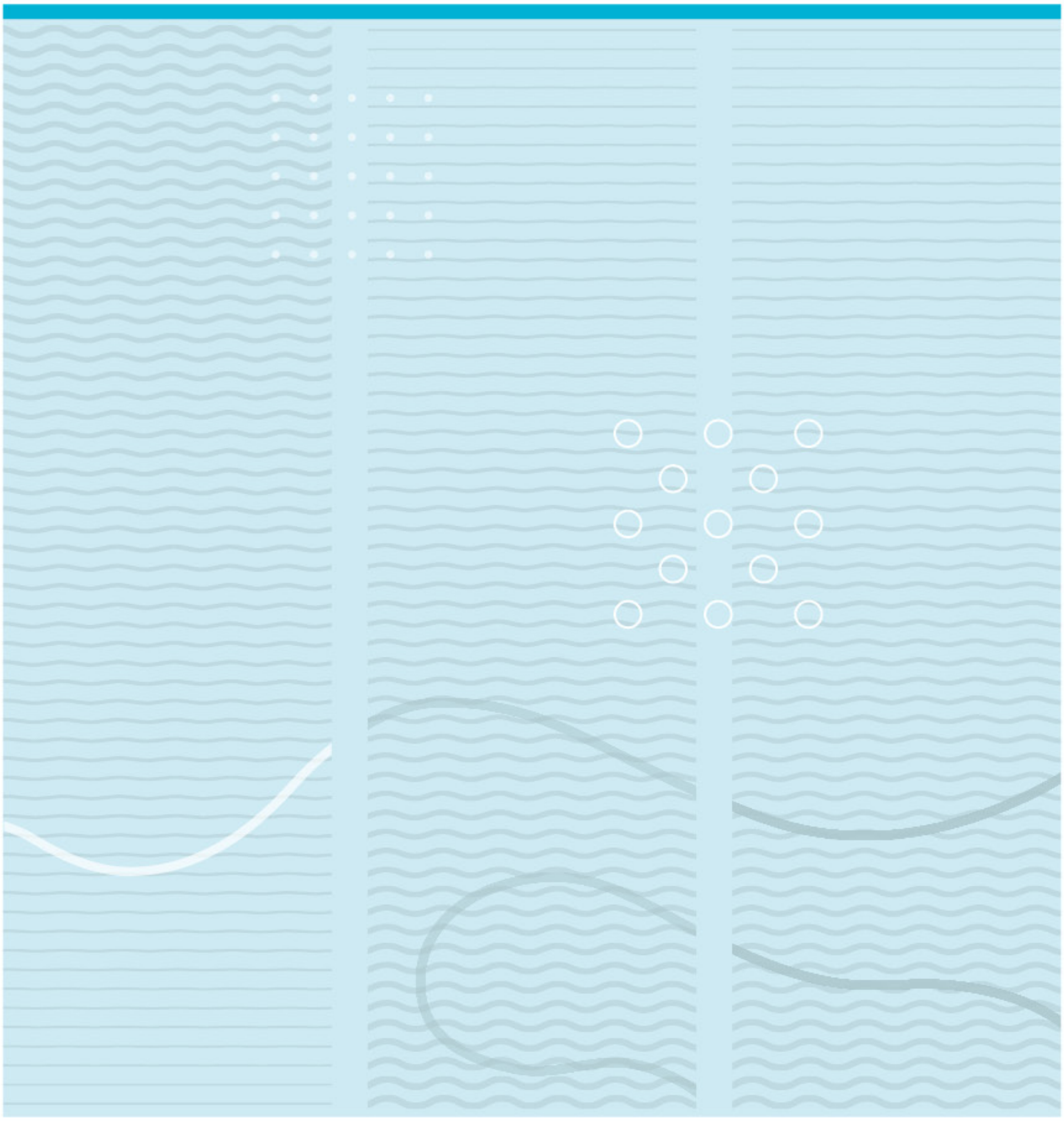
University of South-Eastern Norway
Department of
Science and Industry Systems
Master’s Thesis
Study programme: CS5000-1 22H
Spring/Autumn 2023

S M Habibul Mursaleen Chowdhury

Semantic Search on Equipment



University of South-Eastern Norway
Department of Science and Industry Systems
PO Box 235
NO-3603 Kongsberg, Norway

http://www.usn.no

This thesis is worth 60 study points

University of South-Eastern Norway

Semantic Search on Equipment

Master's Thesis in Computer Science

S M Habibul Mursaleen Chowdhury

Student number: 250804

Email address: 250804@usn.no

Supervisor(s): Fahim Ahmed Salim

Department of Science and Industry Systems

University of South-Eastern Norway

Campus Kongsberg

26th May 2023

Abstract

The outcomes of this research have significant implications for enterprises seeking effective semantic search solutions considering information beyond what is explicitly shown in the raw documents from different inventories and formats and responding to users' inquiries with relevant answers helping them to reduce their workload, time, and efforts. This master's thesis investigates semantic search on heterogeneous documents in an enterprise-level context by exploring two distinct approaches: RDF ontology with RML and entity extraction with vector embedding. The thesis aims to evaluate the effectiveness of these approaches individually and identify opportunities for their combined application as future research scope.

The first experiment employs entity extraction techniques and vector embedding with the support of Pinecone DB. By transforming documents into high-dimensional vectors, it captures semantic similarities, enabling similarity-based search. The experiment specifically targets CSV, Excel, and datasets, offering a focused investigation into the semantic search within specific formats. The second experiment focuses on RDF ontology with RML, utilizing graph-based modeling and SPARQL querying. It demonstrates the ability to capture complex semantic relationships, hierarchies, and ontological concepts, providing a powerful framework for semantic search. The experiment handles structured (CSV, Excel) semi-structured, and unstructured (JSON, XML, DOCX, PDF) documents, enabling effective retrieval of information from diverse file formats.

Through a thorough comparison of the results, the thesis reaches an optimal solution in terms of semantic search on heterogeneous documents at both large and small enterprise-level. Although the RDF ontology approach exhibits superior semantic representation, advanced querying capabilities, and greater semantic expressiveness, enabling more accurate and contextual search results, this thesis proposes future research on merging RDF ontology and vector embedding to leverage their respective strengths such as knowledge representation and semantic similarity, facilitating data integration, and efficient search. This combined approach can hold promise for providing a more comprehensive and powerful semantic search solution.

Finally, this master's thesis contributes to the advancement of semantic search on heterogeneous documents at an enterprise level, offering valuable insights and paving the way for further research and development in this field.

Contents

List of Figures

List of Tables

RDF: Resource Description Framework

RML: RDF Mapping Language

RDFS: RDF Schema

HTTP: Hypertext Transfer Protocol

HTTPS: Hypertext Transfer Protocol Secure

DB: Database

CSV: Comma-Separated Values

SPARQL: SPARQL Protocol and RDF Query Language

JSON: JavaScript Object Notation

XML: Extensible Markup Language

OWL: Web Ontology Language

OWL 2: Web Ontology Language 2

DOCX: Microsoft Word Document

PDF: Portable Document Format

IR: Information Retrieval

NLP: Natural Language Processing

NER: Named Entity Recognition

TF-IDF: Term Frequency-Inverse Document Frequency

LSI: Latent Semantic Indexing

CI: Concept Indexing

KB: Knowledge Base

URI: Uniform Resource Identifier

IRI: Internationalized Resource Identifier

HTML	Hypertext Markup Language
WWW	World Wide Web
W3C	World Wide Web Consortium
FOAF	Friend of a Friend (machine-readable ontology describing persons, their activities and their relations to other people and objects)
PREFIX	Namespace Prefix
DL	Description Logic
CRUD	Create, Read, Update, Delete (common operations in database management)
ML	Machine Learning
BERT	Bidirectional Encoder Representations from Transformers (a popular natural language processing model)

1 Introduction

In today's digital age, the amount of information available to us is growing exponentially. The fourth industrial revolution also brings a huge amount of data availability and the explosion in various trends. (Holter, 2015) We observe an exponential rise in the volume of data available from a widening number of heterogeneous sources, both structured, semi-structured, and unstructured, using various technologies, thanks to the achievements and ongoing improvements in the sector, for example, healthcare, inventory systems especially in energy sectors and so on (Salim, Muller, Ali, & Falk, 2019). As a result, effectively searching and retrieving specific information from heterogeneous documents has become a daunting challenge. This issue is particularly relevant when it comes to searching for equipment-related information (see section 1.1.1), where accurate and efficient retrieval is crucial for various domains such as engineering, manufacturing, and research. We must manage enormous amounts of data from several sources with all the difficulties that entail benefiting from the currently available sources with multiple structured and unstructured documents. One of the suggested answers to this issue is semantic search technology, which response to users' inquiries with relevant answers helping them to reduce their workload, time, and effort. (Holter, 2015; Kumar, Singh, & De, 2017) This is the technique that will be examined in this thesis.

With the advent of Industry 4.0, the efficient management and interpretation of data have become indispensable for industries to realize their maximum potential. However, many project-driven organizations confront the challenge of dealing with dispersed and disconnected systems that generate voluminous document-centric data, making it difficult to explore and extract valuable insights. There is a growing need to transition from unstructured and non-intelligent raw data to intelligent data models that can support semantic reasoning in order to surmount this obstacle. In order to resolve this issue, organizations from both the public and private sectors are pursuing methods to accelerate the digital transformation of the semantic modeling industry. Information modeling, which entails documenting and representing the interactions between various pieces of information in a knowledge graph, has emerged as a technique to enable stakeholders to perform calculations, evaluate status, and facilitate decision-making processes. However, several factors hinder the utilization of available data in complex systems engineering environments, including the absence of a standardized language among engineers and interdisciplinary teams, inefficient knowledge exchange, difficulties in locating system information, and ineffective stakeholder communication.

The significance of this research lies in its potential to revolutionize the search and retrieval of equipment-related information. By integrating semantic understanding with heterogeneous document processing, the proposed system has the capacity to greatly improve the accuracy, efficiency, and comprehensiveness of equipment search. This advancement would have practical applications in industries heavily reliant on equipment-related data, such as manufacturing, engineering design, and scientific research.

1.1 Motivation and Problem Statement

Most organizations working on complex engineering projects generate a large amount of document-centered data scattered around many different often disconnected systems where it is not easily explorable during the project's life cycle. (Salim et al., 2019)

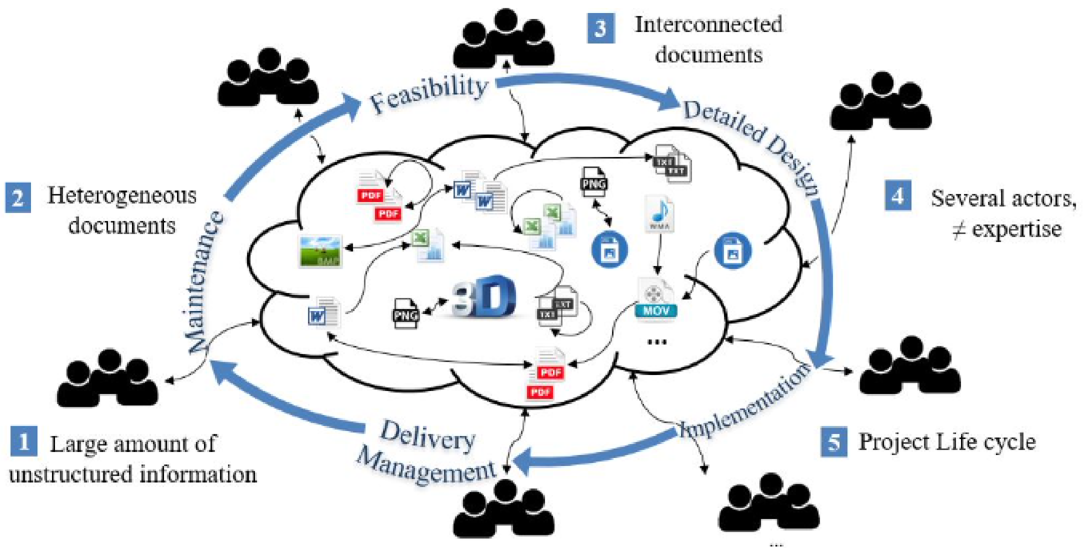
1.1.1 Industry Scenario and Inventory Equipment

Different multinational firms offer full project life cycle services to the energy sector. Organizations struggle to sort through the information jumbled up around many systems, some of which may or may not be interconnected.(Salim et al., 2019) Within the extensive landscape that encompasses contemporary energy production, there are numerous vendors who depend on varied inventories full of highly specialized gear designed to meet particular needs across different aspects of their work involving generation, transmission, and distribution operations. Among this varied arsenal are power generation devices like turbines or engines which harness an extensive array of different forms of sustainable energies to be transformed into vital electrical current through essential mediums like transformers plus switch-gear which enable crucial processes for efficient voltage protection within complex distribution networks while lines capable allowing transportation infrastructure spanning long distances play critical roles powering end-users from remote locations where supply can be limited by geography or other sources, however, sub-stations remain equally important components holding core apparatuses for voltage transformation and regulation for effective operations. Additionally, the focus on greener options like solar panels, wind turbines, and biomass generators has introduced specialized equipment as methods pure to these technologies while monitoring systems have become indispensable providing essential real-time data and keeping track of vital energy consumption levels along with equipment performance too. To store excess power generated there are energy storage mechanisms that also safeguard workers and the environment simultaneously too so that the combination of these varied types of gear empowers energy providers to satisfy evergrowing alternative sustainable and safe demands for supplying all kinds of energetic solutions.

As the inventories are not stored in a central repository, the data would require exploration through multiple systems because of different vendors, contractors, separate subsidiaries, and so on. When an unexpected event happens, an event log is created. These project-specific logs are kept in Excel sheets on which no analyses are made (Salim et al., 2019). Moreover, organizations that usually have their data dispersed around many different systems can not routinely track their equipment and tools to track the location, status, or maintenance history due to a proper structure to store or process that information. (see more in section 1.1.2)

1.1.2 Problem Definition

Managing inventory systems for energy sectors such as oil and gas or renewable energy requires various types of machinery including generators, turbines, compressors, transformers pumps, etc from different vendors with each providing multiple models/versions and companies often having diverse ranges of equipment from different vendors.



3 Interconnected documents
Feasibility
Detailed Design
2 Heterogeneous documents
4 Several actors, ≠ expertise
Maintenance
Implementation
Delivery Management
1 Large amount of unstructured information
5 Project Life cycle
3D
...

Figure 1: Example of a chaotic heterogeneous document in large energy industry

With numerous models available there's usually a large range of diverse machinery within most companies that require effective management by utilizing tools like Microsoft Excel. (Salim et al., 2019) Excel spreadsheets are commonly used to maintain inventories, record equipment details, track maintenance schedules, and manage other relevant information. However, relying solely on Excel data sets may not be the best solution. One common problem is the lack of uniformity between Excel sheets generated by disparate manufacturers. These sheets may have inconsistent formatting, column names, and data structures, making it difficult to extract and compare information across different files. Additionally, relevant data about equipment, including model numbers, specifications, maintenance schedules, and supplier details, is often fragmented across multiple files. As a result, searching for specific product details or comparing different models becomes cumbersome and time-consuming. This leads to major obstacles in comparing different models or searching for specific equipment details.

This figure 1 (Charbel, 2018) illustrates a large-scale and multi-disciplinary project involving a large amount of unstructured information dispersed across various heterogeneous documents having different formats, content, and structure.(Charbel, 2018) Different entities involved in the project generate interconnected documents that are critical to achieving project progress. These documents contain diverse information with varying formats such as PDFs, DOCXs, Excel sheets XMLs images amongst others. However, due to inadequate knowledge-sharing mechanisms, a lack of common search methods searching for system-related information poses a stumbling block for many organizations seeking to track their equipment and tools efficiently. Semantic Search presents an opportunity to tackle this challenge by enabling actors within the project space to conduct specific searches across all forms of documents comprehended both in format and databases. Semantic Search goes beyond basic keyword searches thus able to discern the context and meaning behind user query. This ability facilitates pinpointing relevant details within a single document even when clouded by irrelevant content, resulting in accurate tracking of dependencies within a given dataset.

1.2 Thesis Goals and Main Challenges

Semantic search from heterogeneous documents poses a significant challenge due to the lack of standardization, ambiguity, and the presence of diverse data sources. The absence of a standardized format across documents makes it challenging to extract and compare information effectively. Additionally, the ambiguity inherent in natural language further complicates the search process, as different manufacturers may use different terminology or abbreviations for similar concepts. Moreover, the inclusion of diverse data sources, such as Excel sheets and CSV files from multiple vendors, adds complexity to the retrieval process. Despite these challenges, implementing semantic search for industry-level information retrieval is worth pursuing. By leveraging advanced natural language processing and machine learning techniques, semantic search can provide a powerful solution for extracting meaningful insights and facilitating efficient retrieval of relevant information. It enables users to overcome the barriers of heterogeneous documents, enabling effective decision-making, improved operational efficiency, and better inventory management across the energy sector, including oil and gas and renewable energy industries. Figure 2 will give an idea of the proposed retrieval system.

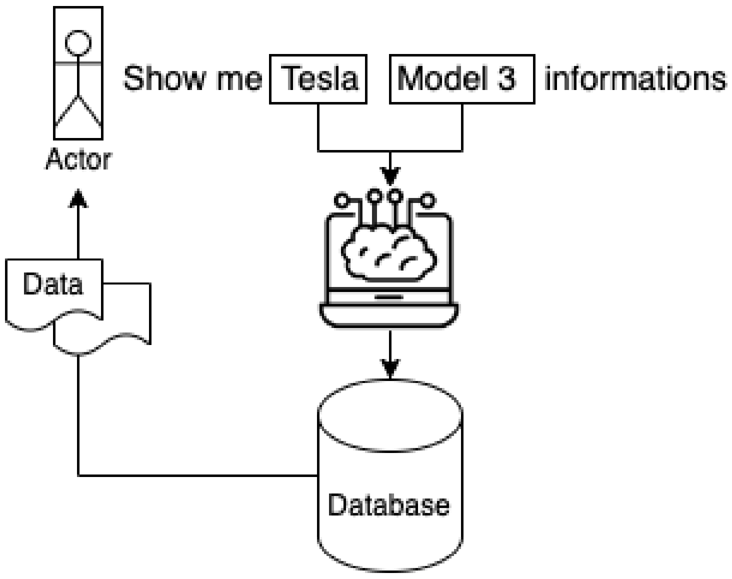


Figure 2: Example of a proposed retrieval system

1.2.1 Project Goals

In this thesis, we mainly address the challenge of allowing enhanced search in a specific context, in a novel way, for relevant information from the heterogeneous document corpus representing the multidisciplinary inventory of enterprises. Here is the goal of the thesis -

- Consider information beyond what is explicitly shown in the raw documents from different inventories and formats.
- Respond to information inquiries with relevant answers helping them to reduce their workload, time, and efforts.

1.2.2 Main Challenges

These goals can result in some challenges, for example:

- **Challenge 1** - Collect and Represent the collective knowledge embedded in a heterogeneous document in different formats corpus through a robust data model;

- **Challenge 2** - Provide search results based on the proposed data model.

Meeting this challenge, enterprises can routinely track their equipment and tools around different systems to track the location, status, or maintenance history. Tackling these challenges is crucial and a step forward in reducing the gap between new technologies imposed in the era of Industry 4.0.

1.3 Thesis Plan and Research Questions

By analyzing unstructured sources leveraging a combination of semantic technologies alongside information retrieval methods like vector embedding or graph databases (as well as incorporating SPARQL), we will discuss on these more in sections 2.3.11 and 2.4.4, this study conducted three separate case studies centered around varied approaches towards enhancing existing semantic search capabilities. More specifically, researchers sought out ideal ways in which document retrieval could be further improved via assessing its accuracy and speed when weighed against a given query utilizing both aforementioned techniques; likewise examining how effective either database technology can be employed when it comes to semantically enabling analysis of the relationships between different entities. Finally, this investigation aimed to identify potential drawbacks and suggest ways in which to overcome them - offering final recommendations for further research while examining broader applications of such techniques.

1.3.1 Research Questions

- How can domain applications exploit semantic technologies and information retrieval techniques to utilize diverse data in different formats to enhance faster information retrieval and decision-making? How RDF can make enterprise-level searches more contextual.

1.3.2 Research Objective

- Examining the efficacy of semantic technologies and information retrieval techniques for data extraction and analysis from unstructured sources.
- Examining how domain applications can effectively utilize a vast array of available data in various formats.
- Using vector embedding, graph databases, and SPARQL to conduct three case studies examining distinct approaches to semantic search using vector embedding.

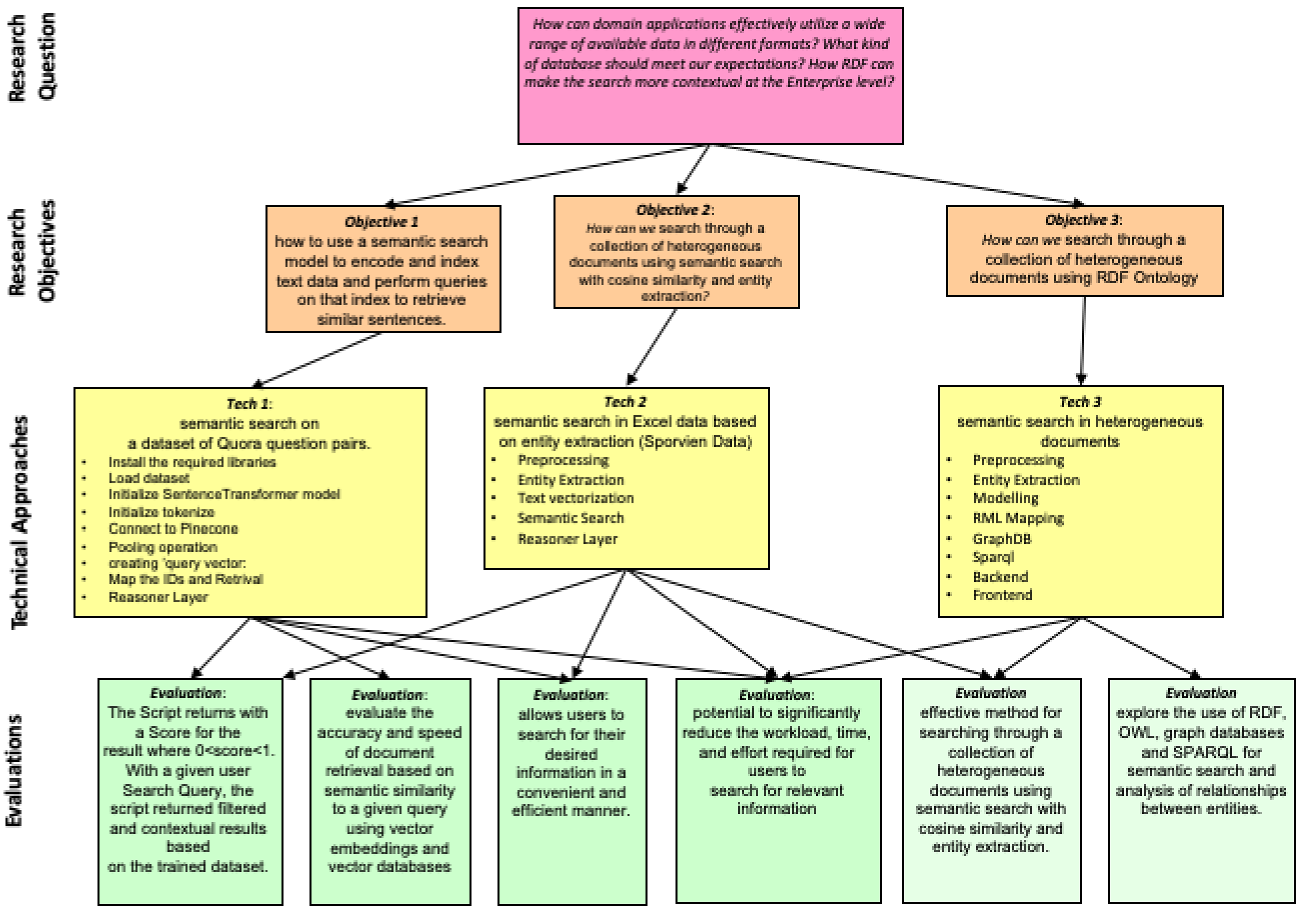

Research Question
Research Objectives
Technical Approaches
Evaluations
How can domain applications effectively utilize a wide range of available data in different formats? What kind of database should meet our expectations? How RDF can make the search more contextual at the Enterprise level?
Objective 1 how to use a semantic search model to encode and index text data and perform queries on that index to retrieve similar sentences.
Objective 2: How can we search through a collection of heterogeneous documents using semantic search with cosine similarity and entity extraction?
Objective 3: How can we search through a collection of heterogeneous documents using RDF Ontology
Tech 1: semantic search on a dataset of Quora question pairs.
• Install the required libraries • Load dataset • Initialize SentenceTransformer model • Initialize tokenize • Connect to Pinecone • Pooling operation • creating 'query vector: • Map the IDs and Retrival • Reasoner Layer
Tech 2 semantic search in Excel data based on entity extraction (Sporvien Data)
• Preprocessing • Entity Extraction • Text vectorization • Semantic Search • Reasoner Layer
Tech 3 semantic search in heterogeneous documents
• Preprocessing • Entity Extraction • Modelling • RML Mapping • GraphDB • Sparql • Backend • Frontend
Evaluation: The Script returns with a Score for the result where 0<score<1. With a given user Search Query, the script returned filtered and contextual results based on the trained dataset.
Evaluation: evaluate the accuracy and speed of document retrieval based on semantic similarity to a given query using vector embeddings and vector databases
Evaluation: allows users to search for their desired information in a convenient and efficient manner.
Evaluation: potential to significantly reduce the workload, time, and effort required for users to search for relevant information
Evaluation effective method for searching through a collection of heterogeneous documents using semantic search with cosine similarity and entity extraction.
Evaluation explore the use of RDF, OWL, graph databases and SPARQL for semantic search and analysis of relationships between entities.

Figure 3: Overview of Case studies

- Evaluating the precision and efficiency of document retrieval using vector embedding and vector databases based on semantic similarity.
- Examine the integration of RDF or OWL to improve semantic search functionality.
- Exploration of the utilization of graph databases and SPARQL for semantic search and analysis of entity relationships.
- Potential applications of semantic technologies in disciplines such as data analysis and document management are evaluated.
- Identify the limitations and obstacles posed by semantic search and suggest solutions.
- Future research recommendations in the area of semantic technologies and information retrieval techniques for extracting and analyzing data from unstructured sources.

1.3.3 Hypothesis and Expectations

1.3.3.1 Hypothesis

The utilization of semantic technologies and information retrieval techniques, such as vector embedding, graph databases, and SPARQL, will lead to the retrieval and analysis of data from heterogeneous sources that are more precise and efficient.

1.3.3.2 Expectations

From the case studies and experiments, our expectations are the following -

- The case studies will demonstrate that semantic search using vector embedding and vector databases can provide accurate and efficient document retrieval based on semantic similarity to a given query.
- The use of graph databases and SPARQL for semantic search and analysis of relationships between entities will reveal hidden patterns and insights that may not be apparent through traditional search methods.
- The potential applications of these semantic technologies and information retrieval techniques in fields such as data analysis and document management will be evident, highlighting the importance of these technologies in today's data-driven world.
- Limitations and challenges of these approaches will be identified, but possible solutions and ways to overcome them will also be proposed.
- The recommendations provided for future research in the area of semantic technologies and information retrieval techniques will help guide further development and adoption of these technologies.

1.4 Assumptions and Limitations

1.4.1 Assumptions

- We surmise that the documents under scrutiny possess a structured or semi-structured design with consistent data types and formatting.

- The files carry trustworthy information that is kept up-to-date on a regular basis. Consequently, it's assumed by the search engine that users possess sufficient familiarity with both information and their required search criteria.

- Regardless of whether it is about file format or structure, there's an expectation from semantic search technology to accurately comprehend file content while extracting relevant details out of them. Henceforth, it presupposes a scenario where all searched files come in a readable format for this particular software while accounting for simultaneous transformations to make sense when needed during searching operations.

- Permissions are also presumed to be provided as required to access those searched locations.

1.4.2 Limitations

The lack of access to data is the study's biggest drawback. A thorough semantic search system or knowledge map cannot be built without the required data being available. The semantic search engine and knowledge mapping could be affected by this. The study is constrained to the time frame in which the data was gathered and might not take into consideration any modifications or advancements in the business's operations or activities that may have occurred after that window of time.

The consistency and caliber of the data present in the files, the algorithms applied, and the layout of the user interface may all have an impact on the effectiveness of the semantic search system and knowledge map. The system might have limitations due to the complexity and variety of file formats, which might make it challenging to correctly understand the content and retrieve pertinent information. The amount of computer power available may place a cap on the search mechanism, especially when looking through big amounts of data or several files at once.

1.5 Thesis Contributions

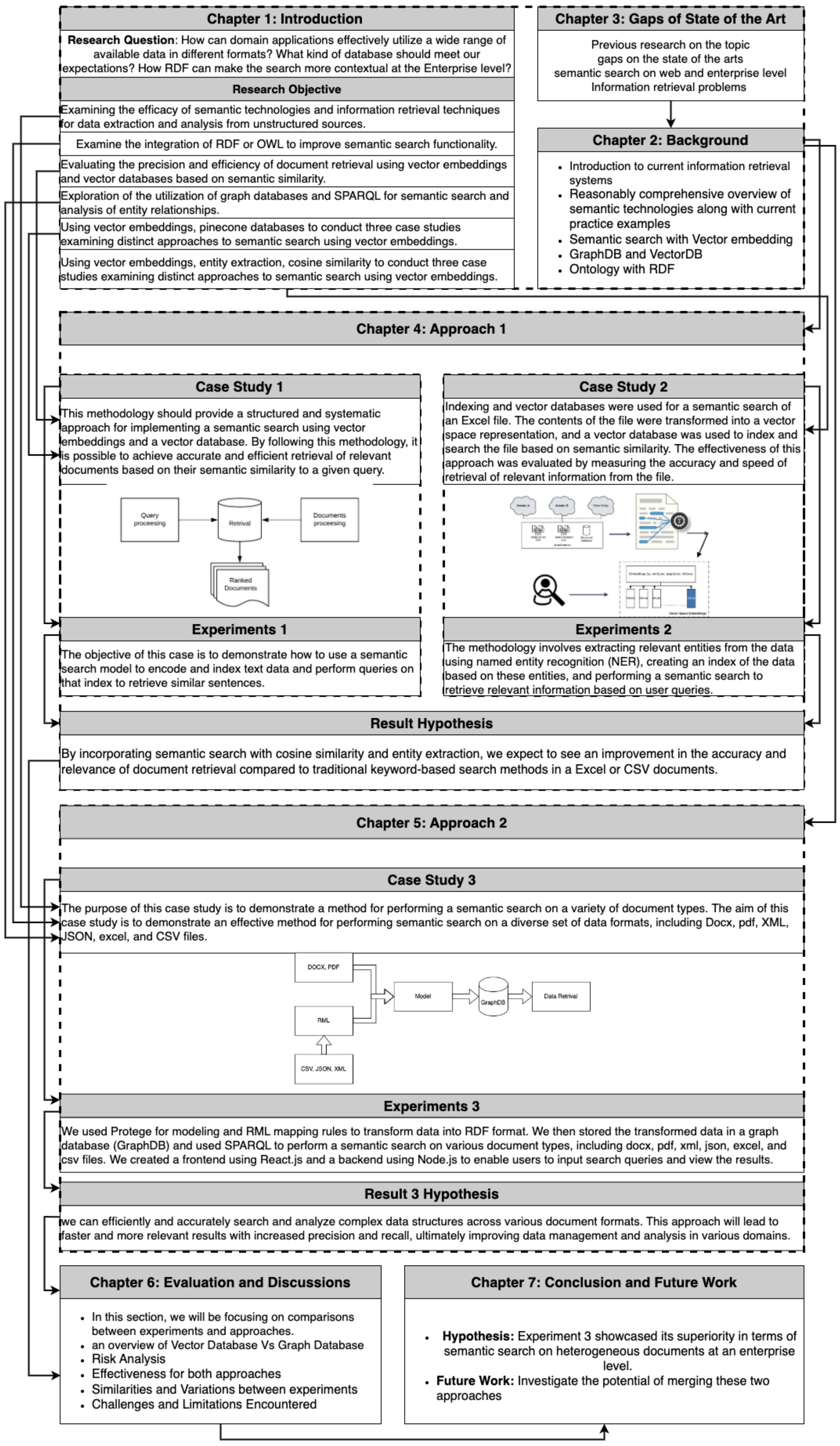
The proposed Master's thesis will concentrate on investigating the application of semantic search technologies to enhance information retrieval and management for global businesses providing project life cycle services in the energy sector. The thesis tries to address the problems businesses encounter when organizing and sorting through massive amounts of data from various systems and vendors. The thesis will begin by introducing the firm and outlining the difficulties in handling data in this sector. A review of the most recent state-of-the-art semantic search technologies, including knowledge graphs, natural language processing, and machine learning methods, will come after. The thesis will next investigate the application of the technologies outlined in the Background Section (section 2) in the context of the energy industry and make a recommendation for the best design for a more effective search system that can respond to users' queries with pertinent information.

The thesis will employ a case study methodology to do this, concentrating on a multinational corporation and its data management procedures. Data collection, analysis, and evaluation of current information retrieval and management systems, as well as a determination of the merits of the suggested remedy, will all be part of the study. The study will also assess how much less work, time, and effort will be required to complete the heterogeneous document that serves as the interdisciplinary inventory for a particular industry.

By showing how semantic search technology may be used in the energy industry and outlining the best course of action for improved information management and retrieval, the thesis will make a significant contribution to the area. Insights into the possible advantages of applying semantic search technology in other sectors with comparable data management issues will also be provided by the study.

1.6 Thesis Outline

The research is divided into eight sections, each serving a specific purpose. In Section 1, a theoretical introduction is provided, covering motivation, goals, problem statements, research objectives, and questions, as well as assumptions and limitations. Section 2 offers a comprehensive overview of the technology utilized in various case studies and experiments, including semantic technologies and vectors. Furthermore, Section 3 explores recent relevant research on a semantic search conducted in recent years. Moving on to Section 4, two case studies are presented, utilizing structured datasets and employing vector embedding. Additionally, two experiments are discussed, and a subsection is dedicated to presenting the results. In Section 5, a case study and experiment are conducted on heterogeneous structured and unstructured documents, utilizing an RDF ontology. Section 6 engages in a discussion comparing and contrasting the two approaches used in the research, analyzing their effectiveness, similarities, and differences. Finally, section 8 concludes by presenting the best possible solution derived from the approaches and includes a future work section to highlight potential avenues for further research. Figure 4 will give an overview outline and visual map of this whole thesis.



Chapter 1: Introduction

Research Question: How can domain applications effectively utilize a wide range of available data in different formats? What kind of database should meet our expectations? How RDF can make the search more contextual at the Enterprise level?

Research Objective

Examining the efficacy of semantic technologies and information retrieval techniques for data extraction and analysis from unstructured sources.

Examine the integration of RDF or OWL to improve semantic search functionality.

Evaluating the precision and efficiency of document retrieval using vector embeddings and vector databases based on semantic similarity.

Exploration of the utilization of graph databases and SPARQL for semantic search and analysis of entity relationships.

Using vector embeddings, pinecone databases to conduct three case studies examining distinct approaches to semantic search using vector embeddings.

Using vector embeddings, entity extraction, cosine similarity to conduct three case studies examining distinct approaches to semantic search using vector embeddings.

Chapter 3: Gaps of State of the Art

Previous research on the topic
gaps on the state of the arts
semantic search on web and enterprise level
Information retrieval problems

Chapter 2: Background

- Introduction to current information retrieval systems
- Reasonably comprehensive overview of semantic technologies along with current practice examples
- Semantic search with Vector embedding
- GraphDB and VectorDB
- Ontology with RDF

Chapter 4: Approach 1

Case Study 1

This methodology should provide a structured and systematic approach for implementing a semantic search using vector embeddings and a vector database. By following this methodology, it is possible to achieve accurate and efficient retrieval of relevant documents based on their semantic similarity to a given query.

Case Study 2

Indexing and vector databases were used for a semantic search of an Excel file. The contents of the file were transformed into a vector space representation, and a vector database was used to index and search the file based on semantic similarity. The effectiveness of this approach was evaluated by measuring the accuracy and speed of retrieval of relevant information from the file.

Experiments 1

The objective of this case is to demonstrate how to use a semantic search model to encode and index text data and perform queries on that index to retrieve similar sentences.

Experiments 2

The methodology involves extracting relevant entities from the data using named entity recognition (NER), creating an index of the data based on these entities, and performing a semantic search to retrieve relevant information based on user queries.

Result Hypothesis

By incorporating semantic search with cosine similarity and entity extraction, we expect to see an improvement in the accuracy and relevance of document retrieval compared to traditional keyword-based search methods in a Excel or CSV documents.

Chapter 5: Approach 2

Case Study 3

The purpose of this case study is to demonstrate a method for performing a semantic search on a variety of document types. The aim of this case study is to demonstrate an effective method for performing semantic search on a diverse set of data formats, including Docx, pdf, XML, JSON, excel, and CSV files.

Experiments 3

We used Protege for modeling and RML mapping rules to transform data into RDF format. We then stored the transformed data in a graph database (GraphDB) and used SPARQL to perform a semantic search on various document types, including docx, pdf, xml, json, excel, and csv files. We created a frontend using React.js and a backend using Node.js to enable users to input search queries and view the results.

Result 3 Hypothesis

we can efficiently and accurately search and analyze complex data structures across various document formats. This approach will lead to faster and more relevant results with increased precision and recall, ultimately improving data management and analysis in various domains.

Chapter 6: Evaluation and Discussions

- In this section, we will be focusing on comparisons between experiments and approaches.
- an overview of Vector Database Vs Graph Database
- Risk Analysis
- Effectiveness for both approaches
- Similarities and Variations between experiments
- Challenges and Limitations Encountered

Chapter 7: Conclusion and Future Work

- **Hypothesis:** Experiment 3 showcased its superiority in terms of semantic search on heterogeneous documents at an enterprise level.
- **Future Work:** Investigate the potential of merging these two approaches

Figure 4: Visual Map of this Masters Thesis

2 Background: Understanding Semantic Search - Techniques, Applications, and Challenges in industries

Information Retrieval (IR) is the field of study concerned with the process of retrieving relevant information from a large set of unstructured data. In recent years, there has been a growing interest in the application of IR techniques to industry-specific domains such as e-commerce, healthcare, and finance. (Holter, 2015) To achieve more accurate and efficient retrieval, semantic technologies such as RDF, OWL, SPARQL, and triple stores have been developed to represent and query complex data structures. Natural Language Processing (NLP) and Named Entity Recognition (NER) enable computers to understand and process human language, allowing for more natural and efficient communication with users.(Klinger, Gampe, Tolle, & Peter, 2018)

As the amount of data continues to grow exponentially, researchers have turned to graph databases and vector databases such as GraphDB (Kivikangas & Ishizuka, 2020) and Pinecone to store and retrieve complex relationships between data points. These databases are designed to handle large volumes of interconnected data and support semantic search capabilities for efficient retrieval of relevant information. Researchers (Rekabsaz, 2017; Hagelien, 2019; V. Jain & Singh, 2021; Tang, Wang, Feng, & Jiang, 2021; Mäkelä, 2021) have been experimenting with a variety of approaches to improve the accuracy and efficiency of IR systems. For example, they have explored the use of RML (Ancona, Franceschini, Ferrando, & Mascardi, 2021) (RDF Mapping Language) to transform data from various sources into a common RDF format, allowing for easier integration and querying. The combination of semantic technologies, graph databases, and NLP techniques has enabled researchers to develop more sophisticated and effective IR systems, leading to a wide range of applications from personalized recommendations to medical (Buriro, Siddiqui, Arain, Shaikh, & Babar, 2015; Boulos, 2004) diagnosis and treatment.

The background study comprises five main subsections that lay the foundation for the research. The first subsection, "Current Information Retrieval (IR) System" (see section 2.1) explores the existing landscape of IR systems, examining the techniques and methodologies employed in modern information retrieval. The second subsection, "Industrial Aspect of IRs" (see section 2.2) delves into the practical implications and applications of IR systems in real-world industries, emphasizing their significance and impact. Moving on, the study explores "The Semantic Technologies" (see section 2.3) investigating approaches such as ontology development, knowledge representation, and semantic web technologies that enhance the understanding and retrieval of information. The fourth subsection, "Semantic Search with Vector embedding" (see section 2.4) focuses on the utilization of vector embedding to capture semantic relationships and improve search effectiveness. Lastly, the study presents 3 Case Studies (section 2.5) that showcase real-world examples and outcomes, demonstrating the practical implementation of IR and semantic search concepts. Together, these subsections provide a comprehensive overview of the background and set the stage for the subsequent research.

2.1 Current Information Retrieval (IR) System

One of the key features of the IR system is ranking the recovered documents. The ranking shows how well the documents that were retrieved matched the user's query. (Obiniyi, Oyelade, & Junaidu, 2014; Abass & Arowolo, 2018; Kumar et al., 2017) Predicting which document is relevant and which is not is one of the main concerns with IR systems. The topic of what is relevant and what does not depend mostly on the IR model under consideration because different IR models evaluate relevancy in their own peculiar ways. Over the years, many various models have been presented forward. A few of the relevant ones here are discussed, starting from the three classical and primitive models:

- Boolean model
- Vector model
- Probabilistic model

First, the fixed, well-structured, and controllable size of the document corpus is an implicit assumption in the traditional IR models. Second, compared to modern IR models, the users in conventional IR models are considered to be trained and cooperative, leading to a considerably more controlled overall environment. Thirdly, the initial objectives of traditional IR models were not particularly sophisticated: they simply searched out documents that were related to the query, ignoring context, user-specific data, and other complex representations. (Wrigley, Elbedweihy, Reinhard, Bernstein, & Ciravegna, 2013)

2.1.1 Boolean Model

One of the simplest and most basic IR models, the Boolean model applies the exact matching principle to match documents to user queries. The AND, OR, and NOT Boolean logic operators are a

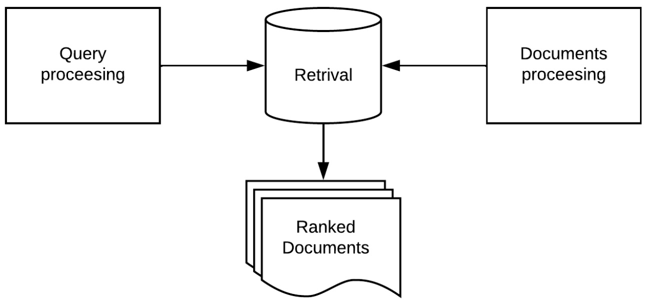


Figure 5: Representation of High-level Information Retrieval

sophisticated combination that must be utilized by users of the Boolean IR system to express their queries. Boolean algebra is used to combine and compare the terms or keywords in user queries with the documents, hence the name "Boolean". The documents are matched with all the terms in the queries exactly as they appear in the queries, and the results are shown without differentiation or enumeration. The user query ‘X AND Y’ means to fetch all the documents which contain both query terms X and Y. Matching only involves finding the presence or absence of keywords in the document and relevancies are judged on a scale of relevant or irrelevant, there is no concept of partially relevant. This method’s fundamental flaw is that accurate matching of this kind frequently yields either too few or too many documents.(Wrigley et al., 2013) The two common IR issues of polysemy and synonymy also affect the Boolean model. Words with similar meanings, such as "vehicle" and "automobile," are said to be synonyms. Words with many meanings, such as "bank," are said to be polysemous (could mean bank of a river or financial center). Numerous irrelevant outputs may be produced as a result of these issues and the inherent limitations of the Boolean model. The user of the Boolean model must also be familiar with query syntax. A user who neglected to enclose a term in quotation marks might receive a large number of irrelevant results.(Wrigley et al., 2013; Tang et al., 2021)

2.1.2 Vector Model

Because the use of binary weights in Boolean-based models is too constrained, the vector-based model has recognized this flaw and suggested a solution that makes partial matching objectively achievable. The vector model employs non-binary weights to index terms in the queries and documents rather than binary weighting (relevant or irrelevant, 1 or 0). As a result, in terms of space, both documents and queries are represented as vectors. In doing so, it considers documents that are only loosely related to the search criteria. As a result, determines how similar the pages are to the search criteria. The level of resemblance between the documents is ranked in decreasing order. As a result, documents from vector-based models are returned in an ordered list that is sorted by how pertinent they are to the user query. Compared to the results returned by the Boolean model, the results are far more useful. (Mäkelä, 2021; Tang et al., 2021) In essence, the vector-based approach converts the textual information in the query and document into vectors before using vector algebra techniques to identify the crucial characteristics and relationships between the documents and queries.

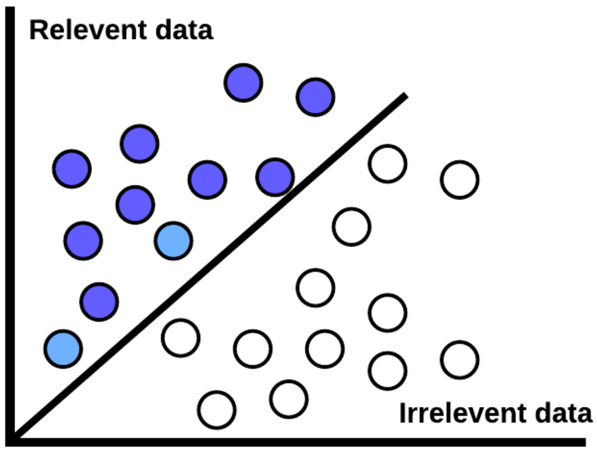


Figure 6: Vector model

models built on vectors expanded the scope of IR models in comparison to Boolean-based models. The vector-based models have a number of benefits, including term-weighting schemes that give the retrieval process an edge, partial matching that enables documents that are partially relevant to the query to appear in the relevant document set, and cosine ranking that sorts them based on similarity and gives the user an easily manageable list.(Tang et al., 2021)

2.1.2.1 TF and IDF:

Term frequency, tf, the number of occurrences of the query term in the document, is widely used as a weighting measure for document ranking. In the perspective of the vector model, the term frequency factor tf corresponds to the intra-cluster similarity and provides a measure of how well the terms describe the document contents. While the inverse document frequency factor, idf, corresponds to the inter-cluster dissimilarity. The motivation for the usage of idf factor is that terms which appear in many documents are not very useful for distinguishing relevant documents from non-relevant ones. (Wrigley et al., 2013; Kumar et al., 2017) See section 4.7 for related experience.

2.1.3 Probabilistic Model

The probabilistic model is based on the probability ranking principle, i.e., it tends to rank the documents in the order of their probability of relevance to the query, given that sufficient evidence is available. Given a user query, we define an ideal setting as a set of results that contains only relevant results and no others. Given the properties of the ideal result set, we won't have problems retrieving documents comprising the ideal set. Querying can be thought of as the process of specifying the properties of the ideal result set (which can also be thought of as a clustering problem). We don't truly understand the characteristics of the ideal result set, which is the only drawback of this newly developed IR model. Only the index terms, whose semantics may be applied in a few different ways to characterize those attributes, are known to us. Since the qualities are unknown, a first guess as to what they might be must be made. These initial hypotheses will be utilized to extract the initial collection of documents and will serve as the preliminary probabilistic representation of what the ideal result set will be. (Wrigley et al., 2013) The system gradually tends to enhance the description of the ideal result set as the probabilistic model runs recursively, requiring the user to review the recovered documents and select which ones are useful and which ones are not. It is anticipated that by repeatedly going through this iterative cycle, a description of the ideal result set can develop that is more accurately representative of the initial description.

In section 2.3, we are going to discuss what kind of methods or strategies are being followed to retrieve information in large-scale and multi-disciplinary projects in an Industrial scenario.

2.2 Industrial Aspect of Information Retrieval

Many large-scale and multi-disciplinary projects involve a large amount of unstructured information dispersed across various heterogeneous documents having different formats, content, and structure.(Charbel, 2018) Although these documents are generated from different entities but are interconnected for the progress of a project. A large amount of unstructured information comes down to text and multimedia content describing several topics. These heterogeneous documents involve diverse information which does not follow a common structure and are stored in different formats (e.g., Pdf, Docx, Excel, XML, IMG and so on). Due to a lack of common search mechanisms among interdisciplinary contractors, ineffective knowledge sharing, and finding system information, many large scaled organizations can not routinely track their equipment and tools and monitor event logs efficiently.(Salim

et al., 2019) In this context, at any point in time of the project, an actor may need to search for a piece of particular information. For his task to be done, he needs to search for all the information regarding that topic from the whole collection of documents in the form of EXCEL sheets, WORD documents, PDFs, XMLs, Databases, and so on, finding relevant parts from a bunch of other non-relevant information within a single document, and tracking document dependencies to search for the related information. A literature review of the information extraction techniques from unstructured and semi-structured scientific resources concluded that there is a dire need for a scheme that can comprehend diverse documents from various pieces of equipment.
The most multinational firm offers full project life cycle services to the energy sector. Organizations struggle to sort through the information jumbled up around many systems, some of which may or may not be interconnected.(Salim et al., 2019) As the inventories are not stored in a central repository, the data would require exploration through multiple systems because of different vendors, contractors, separate subsidiaries, and so on. When an unexpected event happens, an event log is created. These project-specific logs are kept in Excel sheets on which no analyses are made (Salim et al., 2019). Moreover, organizations that usually have their data dispersed around many different systems can not routinely track their equipment and tools to track the location, status, or maintenance history due to a proper structure to store or process that information.

2.3 The Semantic Technologies

Semantic search in a nutshell means “search with meaning” (R. Jain, Duhan, & Sharma, 2021). This contrasts with traditional search, where the primary method for responding to search queries is the lexical matching of query and document terms. Semantic search is a vast field with a variety of features that can be applied to both text and knowledge bases, including query comprehension, answer retrieval and result-in presentation. (R. Jain et al., 2021).
Utilizing keyword queries, relevant data can be retrieved from databases. Exact keyword matching is used in the Exact-Match retrieval paradigm, however, keyword context is not taken into account. Consequently, it has a polysemic and synonymy problem as discussed. The Best-Match retrieval model was created to address these issues. It was decided to use the tf-idf vector representation technique for retrieving exact matches. For best-match retrieval, the latent semantic indexing (LSI) and concept indexing (CI) technique was created. Ontology-based retrieval, on the other hand, has been discovered to be superior to LSI and CI in the literature.(Kumar et al., 2017) There are two basic processes to matching knowledge base, as well as other potential obstacles. First, schema-level matching has

to be done. Ontology matching is the term used to describe this in the context of Semantic Technologies. Without some kind of preliminary mapping, it may be hard to extract and compare the data between the datasets when the schema of one knowledge base and the schema of another knowledge base disagree. Matching at the instance level comes next. Entity matching, also known as link discovery in linked data, is the process of identifying which instances appear in two or more distinct datasets that correspond to the same actual objects. Due to its value for data warehouses, data mining, and duplicate detection, entity matching is also a mature field that is currently the subject of extensive research. (Holter, 2015) The Machine Learning-community is also on the quest for semantics, and increasingly mature tools for capturing the semantics of natural languages are available. The use of word semantic tools such as word2vec has shown promising results on improving ontology matching tools. Recently, numerous studies have investigated the embedding and training of neural network models using an increasing variety of different sources. These sources include RDF knowledge graphs. However, the use of such models for the embedding of OWL 2 ontologies has received very little attention up to this point. This thesis investigates the usefulness of recent neural network models in the process of matching large ontologies. (Ristoski, Rosati, Di Noia, De Leone, & Paulheim, 2018) Its main contribution is OWL2Vec a generic framework for the creation of semantic embedding for OWL 2 ontologies.

2.3.1 Knowledge Base

Knowledge bases are designed to be machine-understandable types of knowledge, in contrast to documents and unstructured language, which can only be understood by humans (section 5.2). Knowledge bases (KBs) are databases that store information on different types of entities, their characteristics, and their connections to other entities. They are the main data components for many semantic-aware applications, such as semantic search (Kumar et al., 2017; R. Jain et al., 2021; Shekhar & Saravanaguru, 2021), question answering, academic, and product search (Shekhar & Saravanaguru, 2021). Commercial search engines also recognize the value of knowledge bases; Google's Knowledge Vault, Bing's Satori, Facebook's, and Linkedin's knowledge graph are just a few examples of efforts made by the industry to create broad or domain-specific knowledge bases. The phrase "Linked Data" refers to a method for achieving the semantic web's objective and is actually the technology underpinning for creating knowledge bases. (V. Jain & Singh, 2021).
Knowledge bases are often stored as RDF triples (Shekhar & Saravanaguru, 2021). More will be discussed in the RDF subsection. Manually curated knowledge bases are of two types: some are

launched as a community project and are directly built with the help of volunteer annotators (e.g., Wikidata and Freebase); others, like YAGO and DBpedia, are built by automatically extracting facts from (semi-structured data in) Wikipedia.

2.3.2 Linked Data

It's very easy to make typed linkages between data from many sources utilizing the Web and Linked Data. These can range from heterogeneous systems within one organization that historically have not interacted well at the data level to databases that are managed by two organizations in different geographic locations. Technically speaking, "Linked Data" refers to data that has been published on the Web in a way that makes it machine-readable, explicit about what it means, linked to other external data sets, and reachable from other external data sets. While HTML (HyperText Markup Language) documents that are connected by untyped hyperlinks make up the main building blocks of the hypertext Web, Linked Data relies on documents that contain data in RDF (Resource Description Framework) format. (V. Jain & Singh, 2021). However, Linked Data uses RDF to create typed assertions that link arbitrary items in the world, as opposed to merely connecting these documents. RDF, a technology that is essential to the Web of Data, is added to URIs and HTTP. While HTML offers a way to structure and link Web content, RDF offers a general, graph-based data model for structuring and linking data that describes things in the real world.

For instance, an RDF triple could indicate that two individuals, A and B, who are both identified by URIs, are connected by the fact that A is familiar with B. Similar to this, an RDF triple may identify person C as the author of a scientific publication D in a bibliographic database. The data in one data source can be linked to the data in another, resulting in the creation of a Web of Data, when two resources are linked in this way. As a result, RDF triples that connect objects in various data sets can be compared to the hypertext links that hold the Web of documents together. (V. Jain & Singh, 2021) A foundation for developing vocabularies that may be used to describe items in the real world and their relationships is provided by the RDF Vocabulary Definition Language (RDFS) and the Web Ontology Language (OWL). These Vocabularies are collections of classes and properties that are described in RDF using terminology from the RDFS and OWL. They offer various levels of expressivity in modeling the relevant domains. RDF triples that link classes and properties in one vocabulary to those in another can be used to connect these vocabularies, providing mappings between related vocabularies (see Figure 7).

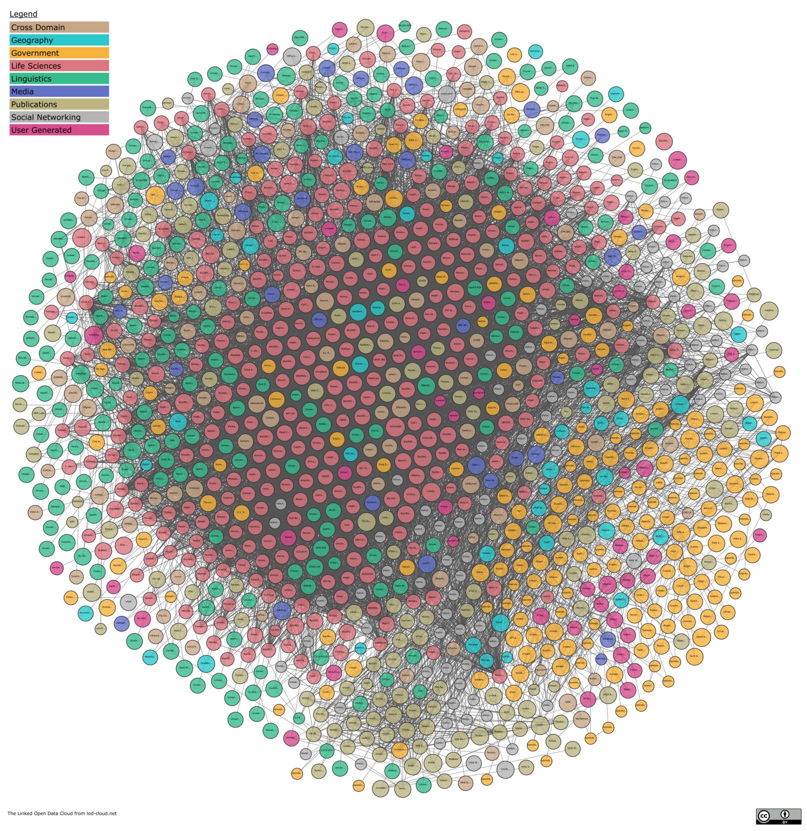


Figure 7: Representation of The Linked Open Data Cloud 2022

2.3.3 Resource Description Framework (RDF)

One of the three core Semantic Web technologies—the other two being SPARQL and OWL—is RDF (Resource Description Framework). The Semantic Web's data paradigm, RDF, in particular, offers built-in flexibility. (Tang et al., 2021; Zende & Baban, 2015) RDF is used to represent all data on the Semantic Web, including schema that describes RDF data. RDF differs from relational databases' tabular data model. It's also not like the XML world's trees. RDF is a graph instead. (section 5.2) Each RDF triple is in the form of **<subject, predicate, object>**, where the subject is a URI (link to another entity), and the object is a URI or a literal value. The predicate specifies the relation between the subject and object and is represented by a URI.

RDF can be visualized as a collection of nodes (the dots) joined by edges (the lines), each of which has a label. Resources and edges are URIs in RDF. Universal Resource Identifier is what URI stands for. The crucial component of that is universal. Instead of providing ad hoc IDs for objects within a single database, the Semantic Web generates universal identities for things that are consistent across databases (think primary keys). We are able to connect everything because of this. The set of URIs representing the edges that make up RDF graphs is known as the "RDF Vocabulary." The graph's edges connect the objects and give them context. A data exchange format called RDF organizes data

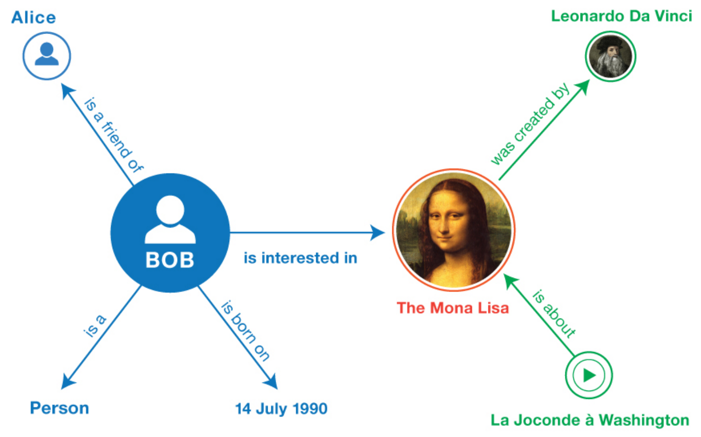
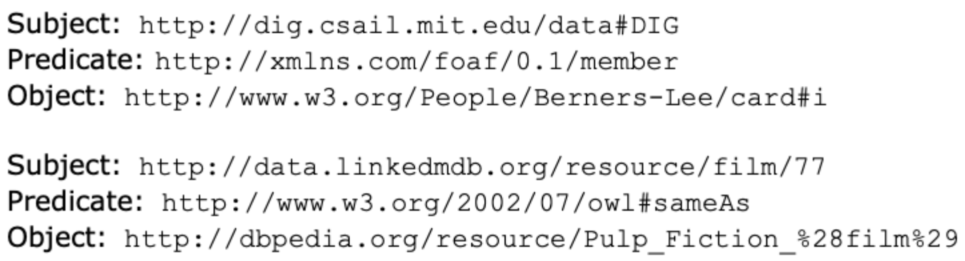


Figure 8: Representation of RDF graph

using triples (subject-predicate-object). (Ristoski et al., 2018; Zende & Baban, 2015) The object can be either a literal or a URI, and the subject and predicate are typically (URIs). Both the subject and the object can be represented as blank nodes, allowing for the construction of more complex structures and the storage of data if some information is unknown. RDF datasets are frequently referred to as graphs of data because URIs are used to link the data. (Ristoski et al., 2018; Zende & Baban, 2015) Therefore, RDF graphs are just a collection of triples. For this reason, an RDF database is frequently referred to as a triple store.



Subject: `http://dig.csail.mit.edu/data#DIG`
Predicate: `http://xmlns.com/foaf/0.1/member`
Object: `http://www.w3.org/People/Berners-Lee/card#i`

Subject: `http://data.linkedmdb.org/resource/film/77`
Predicate: `http://www.w3.org/2002/07/owl#sameAs`
Object: `http://dbpedia.org/resource/Pulp_Fiction_%28film%29`

Figure 9: Example of RDF Links

RDF links take the form of RDF triples, where the subject of the triple is a URI reference in the namespace of one data set, while the object of the triple is a URI reference in the other. Figure 9 shows two examples RDF links. The first link states that a resource identified by the URI http://www.w3.org/People/Berners-Lee/card#i is a member of another resource called http://dig.csail.mit.edu/data#DIG. When the subject URI is dereferenced over the HTTP protocol, the dig.csail.mit.edu server answers with an RDF description of the identified resource. When the object URI is dereferenced the W3C server provides an RDF graph describing Tim Berners-Lee. Dereferencing the predicate URI http://xmlns.com/foaf/0.1/member yields a definition of the link type member, described in RDF using the RDF Vocabulary Definition Language (RDFS), introduced below. The second RDF link connects the description of the film Pulp Fiction in the Linked Movie Database with the description of the film provided by DBpedia, by stating that the URI http://data.linkedmdb.org/resource/film/77 and the URI http://dbpedia.org/resource/Pulp_Fiction_%28film refer to the same real-world entity - the film Pulp Fiction.

2.3.4 RDF Triplestore

The RDF triplestore is a kind of graph database that organizes data as a network of objects and makes use of inference to extract new knowledge from pre-existing relations. (Ristoski et al., 2018) Due to its adaptable and dynamic character, it is possible to link various types of data, index them for semantic search, and enhance them through text analysis to create large knowledge graphs. (Holter, 2015) Triplestores store data as a network of objects with materialized links between them since they are graph databases. As a result, RDF triplestores are the best option for handling densely related data. Compared to a relational database, triplestores are more affordable and adaptable. Ontologies are optional schema models that are supported by RDF triplestore engines. Data can be formally described using ontologies. (Zaman et al., 2018; Noah et al., 2013) Both relationship characteristics and object classes, as well as their hierarchical arrangement, are specified. The Universal Resources Identifier (URI) is the central idea of both the Linked Data paradigm and the RDF triplestore format (URI). The URI is a type of unique ID that is used on the Web as a single global identifying system. (Holter, 2015; Kumar et al., 2017; Ristoski et al., 2018; Zende & Baban, 2015; Zaman et al., 2018)
Here is an example of an RDF triple:

```
<http://example.com/john> <http://xmlns.com/foaf/0.1/name> "John Smith" .
<http://example.com/john> <http://xmlns.com/foaf/0.1/age> "30" .
<http://example.com/john> <http://xmlns.com/foaf/0.1/gender> "male" .
<http://example.com/jane> <http://xmlns.com/foaf/0.1/name> "Jane Doe" .
<http://example.com/jane> <http://xmlns.com/foaf/0.1/age> "25" .
<http://example.com/jane> <http://xmlns.com/foaf/0.1/gender> "female" .
```

In this example, the triplestore contains six RDF triples, representing information about two people, John Smith and Jane Doe. Each person is identified by a unique URI, and their name, age, and gender are represented using the FOAF vocabulary.
Triplestores typically provide a range of functionality for managing and querying RDF data, including search, indexing, and querying using SPARQL (SPARQL Protocol and RDF Query Language). By storing and managing RDF data in a triplestore, it becomes possible to represent and analyze complex information in a structured way, making it easier to build applications that can work with the data.

An RDF triplestore is designed to store and manage large volumes of such triples, making it possible to represent complex information in a structured way.

2.3.4.1 Enterprise Deployments of RDF Triplestore

RDF triplestores can hold enormous amounts of data, which enhances an organization's ability to conduct analytics and knowledge discovery. Information becomes knowledge when relationships are inferred from the original data using a semantic graph database. This makes it possible for businesses to find hidden connections among all of their data. (Kivikangas & Ishizuka, 2020) Enterprises may more easily scale up their knowledge into smarter solutions and have the upper hand in the competition if they have more information than their rivals. RDF triplestores are already widely used by large enterprises across a variety of industries to manage both organized and unstructured data. Additionally, triplestores support a variety of text analytics approaches, including content enrichment and information extraction from unstructured data.

2.3.5 SPARQL Protocol and RDF Query Language

SPARQL (SPARQL Protocol and RDF Query Language) is a query language designed for querying RDF (Resource Description Framework) data, which is used to represent information in a structured way on the web. (V. Jain & Singh, 2021; Bizer, Heath, & Berners-Lee, 2009) SPARQL allows users to query RDF data in a flexible and powerful way, making it possible to search and analyze large datasets of structured information.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?name ?email
WHERE {
   ?person foaf:name ?name ;
               foaf:mbox ?email .
}
```

This SPARQL query retrieves the name and email address of all persons in the RDF data that have a name and an email address represented by the FOAF vocabulary (an RDF vocabulary for describing people and their relationships).
The query has three parts:

- PREFIX statements are used to define prefixes for commonly used namespaces. In this example, the FOAF and RDF namespaces are defined using prefixes.

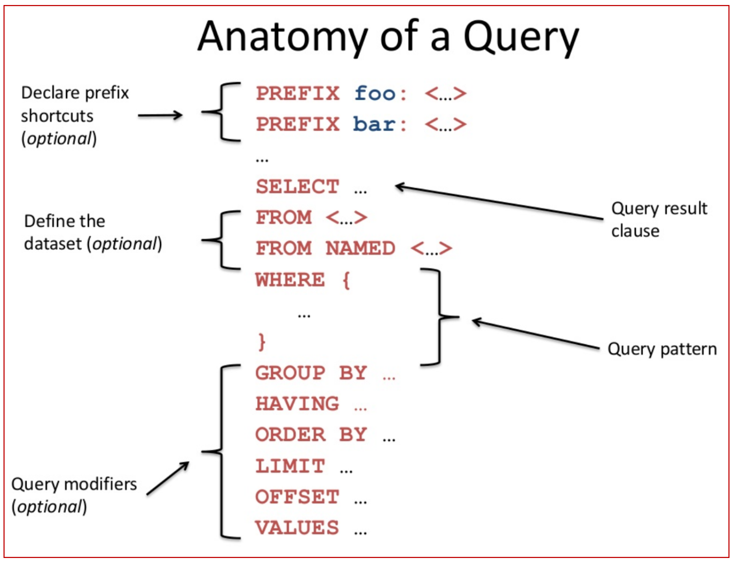


Figure 10: Ananlomy of SPARQL Query

- The SELECT statement specifies the variables that should be returned in the query result. In this example, we want to retrieve the name and email address of each person.
- The WHERE statement is used to define the pattern that the RDF data must match. In this example, we are looking for all persons who have a name and an email address represented by the FOAF vocabulary.

The "?" character is used to indicate variables in the query. The variables specified in the SELECT statement must also be included in the WHERE statement as part of the pattern. When executed against an RDF dataset, the SPARQL query would return a list of names and email addresses of all persons that meet the pattern specified in the WHERE statement.
SPARQL is a powerful tool for querying RDF data and provides a flexible way to extract and analyze information from large datasets.

2.3.6 Resource Description Framework Schema - RDFS

The significance of vocabulary in exchanging RDF data across systems was described in the RDF section, however, RDFS aids in developing its own vocabulary for RDF data. The most fundamental schema language used frequently in the Semantic Web technology stack is RDFS. It is lightweight and quite simple to use. In actuality, the majority of the most widely used RDF vocabularies are written in RDFS. As mentioned, RDF is a graph database. On the other hand, RDFS is by definition object-oriented. In other words, the core purpose of RDFS is to describe classes of objects. RDF is used to represent RDFS.

Example:

```
csipeople: David
rdf:type foaf: Person ;
rdfs: label David Tester ;
```

These descriptions are provided for an RDF resource that represents a person. The triple is the csi: people rdf: type foaf: Person. The resource is described as being of type foaf: Person, where foaf: Person is an RDFS class that has been previously specified.

To define own Person class -

```
csi: MyNewPersonClass
rdfs: label My Person Class.
```

This defined properties for such classes in order to give them richness.

```
csi: admires
rdf: type rdfs: Property ;
rdfs: label Admires ;
rdfs: domain csi: MyNewPersonClass ;
rdfs: range csi: MyNewPersonClass.
```

MyNewPersonClass, a property that it can utilize with the csi, has now been defined.

```
csipeople: David
rdf: type foaf: Person ;
rdf: type csi: MyNewPersonClass ;
rdfs: label David Tester ;

csi:people: Lee
rdf: type csi: MyNewPersonClass ;
csi: admires csipeople: David.
```

2.3.7 The Web Ontology Language (OWL)

Many scholars began to concentrate on ontologies as a technique to communicate definitions and rules in a domain of knowledge as a result of the rising interest in the Semantic Web witnessed in the early 2000s.(Holter, 2015) It's been known for a while that the Web would gain from content that was machine-processable and comprehensible. Studying the nature of existence, beings, and their relationships is known as ontology. Formal ontologies offer a context or meaning that is precisely understood to both humans and machines. A common understanding of information is ensured through ontologies. Ontologies are used in real life to link and characterize diverse and complex data. An ontology is a depiction of the connections between words in a lexicon and the meanings that individuals have for those terms. A family of languages known as OWL/OWL 2 is created to represent knowledge about concepts and their relationships. RDF graph made up of a collection of assertions is an OWL ontology. A subject, a property, and an object are the three components that make up a statement, often known as a triple. It is represented graphically as a directed graph, with the subjects and objects acting as nodes and the qualities as edges. (Holter, 2015; Wrigley et al., 2013; Shekhar & Saravanaguru, 2021)

The Semantic Web's ontology ("schema") language is OWL. Along with RDF and SPARQL, it is one of the fundamental Semantic Web standards that is needed to understand. OWL's ability to communicate exceedingly complex and nuanced notions about the data is one of its distinctive qualities. OWL is built on good mathematical logic and goes beyond basic syntactic conventions. This may appear to be a rather esoteric, academic feature given that the majority of organizations don't have staff logicians, but using OWL, it is possible to know for sure what a set of data means (in a mathematical sense) and, as a result, to know when some data provided in response to a request actually satisfies the request. In other words, OWL is clear and expressive. OWL enables the use of data models to support a wide variety of reasoning activities. The term "reasoning" refers to automated reasoning that clarifies implicit data. Information retrieval applications can now be more sophisticated thanks to their strong capability to support reasoning on data and metadata, which actually lowers the complexity of the queries required to get data. Figure 11 shows a graph with two triples. The ellipse to the left with Joe is the subject of these triples, the arrows called likes and hasName are properties and the second ellipse with Kim and the rectangle "Joe" are objects. These triples are statements that

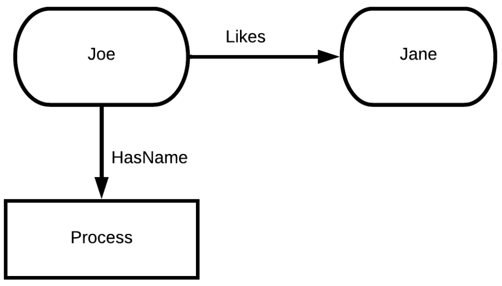


Figure 11: A graph of 2 triples consisting of two common subjects, two properties and two objects

tell something about the subject, Joe. They can be formulated as Joe - likes - Jane and Joe - hasName - "Joe". OWL is able to perform reasoning on triples because it is based on Description Logic (DL). DL is a fragment of first-order predicate logic and a formalism for representing knowledge. OWL was designed in DL in order to achieve a beneficial tradeoff between expressivity and scalability, while at the same time maintaining the possibility to reason over the logic. In DL the different logic axioms and assertions are put in boxes, called T-box and A-box.

2.3.7.1 T-Box:

T-box stands for Terminological knowledge and holds the definitions and "the semantics" in an ontology since it tells how classes and properties are related to one another. The terminology box is independent of the actual data in the ontology.

2.3.7.2 A-Box:

A-box stands for Assertional knowledge and contains the actual data, the facts about concrete individuals such as a, b, and c. The assertion box also contains sets of class membership assertions, C(a), and property assertions R(a, b). When thinking about classes and properties in OWL, it helps to think in terms of sets. (Løvdahl, 2018)

2.3.7.3 OWL Tools:

Today, a wide variety of OWL-based software tools are accessible for ontology creation.

- Stanford University's Protégé, a free, open-source ontology editor
- TopBraid Composer from TopQuadrant
- Any text editor

2.3.8 Differences Between Resource Description Framework Schema vs Web Ontology Language

We can describe our RDF data in great detail using major Semantic Web technologies like OWL and RDFS. By standardizing on a flexible, triple-based syntax and then offering a relatively narrower vocabulary (such as rdf: type or rdfs:subClassOf) that may be used to say things about concepts in a specific area(s) of interest, RDFS enables to express the relationships between things. OWL is comparable, but it's better and bigger. OWL makes it possible to express a data model in much more detail; it demonstrates how to use database queries and automatic reasoners effectively; and it offers helpful annotations for applying data models to actual situations.

2.3.8.1 Vocabulary:

The fact that OWL offers a considerably broader vocabulary is perhaps the most significant distinction between RDFS and OWL.

2.3.8.2 Logical Consistency:

Another important distinction is that, in contrast to RDFS, OWL specifies when and how a user may use a certain vocabulary. To put it another way, RDFS lacks real constraint mechanisms, but OWL does. OWL allows users to choose how expressive they want to be given the computational realities at play, in contrast to RDFS. Actually, OWL allows for restricting the sorts of data modeling options to those that support faster search queries, conceptual reasoning, or straightforward implementation with rules engines.

2.3.8.3 Annotations, the meta-meta-data:

Reusing items is straightforward using OWL. Data models can be linked together to form an interconnected network of ontologies using OWL annotations such as owl:versionInfo, owl:backwardsCompatibleWith, and owl:deprecatedProperty. Owl: Importance is to use an object repeatedly. OWL is likely to fulfil all specifications for meta-metadata modelling, in contrast to RDFS.

2.3.9 RDF Mapping Language (RML)

The RDF Mapping Language (RML) is a language that allows the mapping of data from different sources, such as CSV files, XML files, and databases, to RDF (Resource Description Framework) format. RML provides a flexible and powerful way to transform and integrate data from heterogeneous sources into a unified RDF representation.

RML uses a declarative mapping approach, where the mapping rules are expressed in a separate mapping file. These mapping rules define how the data from the source is mapped to the RDF format.

The RML mapping language consists of three main components: the mapping rules, the source data, and the target RDF model.

```
@prefix rml: <http://semweb.mmlab.be/ns/rml#>.
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/>.

<#Mapping>
   a rr:TriplesMap;
   rr:logicalTable [
       rr:tableName "employees";
   ];
   rr:subjectMap [
       rr:template "http://example.com/employees/{employee_id}";
       rr:class ex:Employee;
   ];
   rr:predicateObjectMap [
       rr:predicate ex:name;
       rr:objectMap [
           rr:column "name";
       ];
   ];
   rr:predicateObjectMap [
       rr:predicate ex:department;
       rr:objectMap [
           rr:column "department";
       ];
   ].
```

In this example, the mapping rules specify that the source data comes from a table called "employees". The subject of the RDF triples is defined by a template that includes the employee ID. The class of the subject is defined as "ex:Employee". The mapping rules also specify that the "name" and "department" columns from the source data should be mapped to the "ex:name" and "ex:department" predicates in the RDF model, respectively.

When the RML mapping file is executed with the source data, it produces an RDF graph that represents the employees' data in a standard and interoperable format.

RML is a powerful and flexible mapping language that enables the integration of data from various sources into a unified RDF representation. RML's declarative mapping approach allows for easy maintenance and modification of the mapping rules, making it a useful tool for data integration and interoperability. See (section 5.2) for the experiment.

2.3.10 Internationalized Resource Identifier - IRI

In ontology, IRI stands for Internationalized Resource Identifier. It is a unique identifier for resources, such as classes, individuals, and properties, in the context of the Semantic Web. (Ryen, 2021)
IRIs are similar to Uniform Resource Identifiers (URIs) and Uniform Resource Locators (URLs) in that they provide a way to identify a resource on the web. However, IRIs have additional features that make them more suitable for use in ontologies, such as support for international characters and the ability to distinguish between different types of resources. IRIs are used extensively in the Web Ontology Language (OWL), which is a language used for representing ontologies on the web. In OWL, IRIs are used to identify classes, individuals, properties, and other elements of an ontology. By using IRIs, ontologies can be shared and reused across different applications and domains. An IRI identifies a resource, which can be anything from a web page, a person, a concept, or an object. See (section 5.2) for the experiment.

An IRI is a string of characters that consists of two parts: a scheme and a scheme-specific part. The scheme identifies the type of resource being identified and is followed by a colon (':'). The scheme-specific part contains the actual identifier for the resource.
The structure of an IRI can be illustrated by the following example:

```
http://example.com/ontology#Person
```

In this example, the scheme is "HTTP", which is the protocol used to access the resource. The scheme-specific part consists of the domain name "example.com", followed by the path to the resource "/ontology" and the fragment identifier "#Person". The fragment identifier refers to a specific element within the resource, in this case, a class named "Person" within an ontology. IRIs can also be used to identify resources that are not on the web, such as local files or database records. In these cases, the scheme is often a custom scheme that is specific to the application or domain. This is where we will conduct our research (See (section 5.2) for the experiment.). IRIs are designed to be human-readable and can include non-ASCII characters, such as accented letters and symbols. For example:

```
http://example.com/ontology#Person
```

In this example, the IRI identifies a class named "Étudiant" in an ontology. The use of non-ASCII characters in the IRI allows for more expressive and natural language identifiers in ontologies.

IRIs are a powerful tool for identifying resources on the web and beyond. They provide a flexible and extensible means of identifying resources that can be used across different domains and applications.

2.3.11 Graph Database

GraphDB is a semantic graph database that is designed to store and manage semantic data based on the RDF (Resource Description Framework) data model. GraphDB is built on top of the OWLIM semantic repository and supports a wide range of ontology languages, including RDF, RDFS, OWL, and SPARQL. (Samuelsen, 2016) In terms of information retrieval and ontology, GraphDB is used to store and manage ontologies and their associated data. An ontology is a formal specification of a conceptualization, which provides a shared vocabulary for describing a domain of interest. Ontologies can be used to represent and organize knowledge in a structured and formal way, which makes it easier to search, retrieve, and reason about information (section 5). Here's an example of how GraphDB can be used to store and manage an ontology:

2.3.11.1 Create a new repository

To create a new repository in GraphDB, we can use the GraphDB Workbench or the REST API. The repository can be configured with the desired settings and parameters, such as the repository ID, storage location, and index options.

2.3.11.2 Import the ontology

Once the repository is created, we can import the ontology into the repository using the GraphDB Workbench or the REST API. The ontology can be in any supported ontology language, such as RDF or OWL.

2.3.11.3 Query the ontology

After the ontology is imported, we can use SPARQL queries to retrieve information from the ontology. For example, we can use SPARQL to retrieve all the instances of a particular class or to find the relationships between different classes.
Here's an example of a SPARQL query that retrieves all the instances of a particular class in an ontology stored in GraphDB:

```
SELECT ?instance
WHERE {
    ?instance rdf:type <http://example.org/MyClass>.
}
```

This query retrieves all the instances of the class "MyClass" in the ontology, which are represented as RDF resources. GraphDB uses the RDF data model to represent and store data in the form of subject-predicate-object triples. Each triple represents a statement about a resource, where the subject is the resource, the predicate is the relationship between the resource and the object, and the object is either another resource or a literal value. The triples can be stored in GraphDB using different storage backends, such as a file system or a database. In addition to storing and managing ontologies, GraphDB can also be used to perform inference and reasoning on the data in the ontology. This allows us to derive new knowledge and relationships from the existing data, which can be used for advanced search and retrieval tasks.

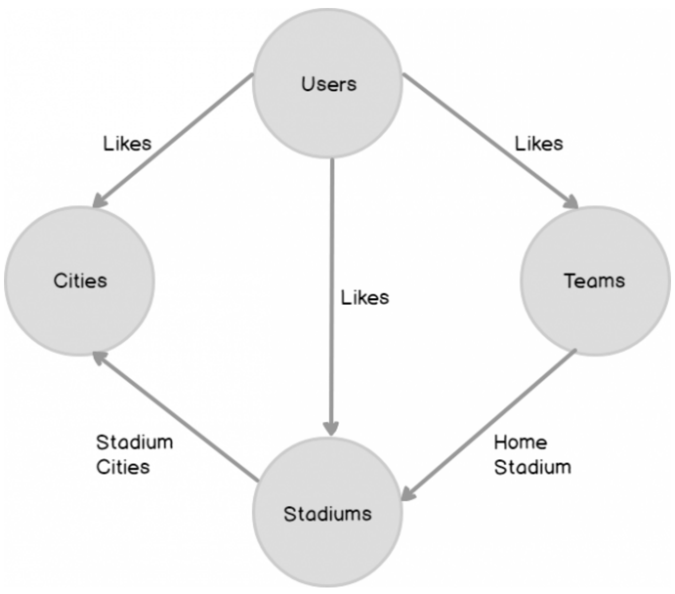


Figure 12: Diagram of Graph Database Structure

GraphDB provides a powerful and flexible platform for managing and querying semantic data based on ontologies, which makes it a valuable tool for information retrieval and knowledge management. We used GraphDB for our Experiement 3. See (section 5.2) for the experiment.

2.4 Semantic Search with Vector embedding

The amount of complex data is growing exponentially. These are unstructured data categories, along with documents, plain text, images, and videos. While storing and analyzing complex data would be advantageous for many firms, complex data can be challenging for conventional databases designed with structured data in mind. It's possible that using keywords and metadata to classify complicated data is insufficient to fully capture all of its unique qualities. (Ganguly, Mitra, Roy, & Jones, 2015; Rekabsaz, 2017)

Semantic search with vector embedding is a technique used in natural language processing (NLP) to improve search results by understanding the meaning behind the text rather than just matching keywords. It involves converting text data into numerical vectors, which can then be used to calculate the similarity between different pieces of text based on their semantic meaning. See (section 4.6) for the experiment.

2.4.1 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a field of computer science and artificial intelligence that focuses on enabling computers to understand and generate natural language text. (Klinger et al., 2018) NLP is a powerful tool for ontology information retrieval, as it can help to automate the process of extracting information from text and mapping it to ontological concepts. In the context of ontology information retrieval, NLP can be used to extract information from unstructured text sources, such as documents, web pages, and social media. This information can then be mapped to relevant ontological concepts, such as classes, properties, and instances, in order to build a structured representation of the knowledge contained in the text.
See (section 4.6 and 4.4) for the experiment.
There are several techniques used in NLP that can be applied to ontology information retrieval but in our case, we tried out NER and Natural methods with Python and JavaScript. our experiment 2 (experiment 2 section), we used the NER method and.

2.4.1.1 Named Entity Recognition (NER)

Named Entity Recognition (NER) is a subtask of Natural Language Processing (NLP) that involves identifying and classifying named entities in text. (Klinger et al., 2018) A named entity is a word or phrase that refers to a specific real-world object, such as a person, place, organization, date, time, or product. NER is used in a wide range of applications, including information extraction, question answering, and text classification. The goal of NER is to identify all the named entities in a text and to classify them into predefined categories, such as Person, Location, Organization, Date, Time, and Product. NER can be performed using machine learning algorithms. For example, for the following sentence: "John Smith works at Google in New York."

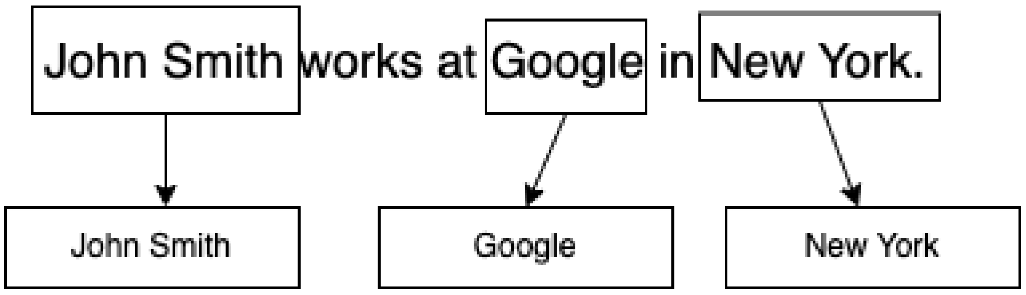


Figure 13: Representation of NLP with NER

- **Tokenization**: The text is divided into tokens, which are individual words or phrases. In the example, the sentence is divided into tokens: "John", "Smith", "works", "at", "Google", "in", "New", and "York".
- **Part-of-speech (POS) tagging:** Each token is assigned a part-of-speech tag: "John" (proper noun), "Smith" (proper noun), "works" (verb), "at" (preposition), "Google" (proper noun), "in" (preposition), "New" (proper noun), and "York" (proper noun).
- **Chunking:** The tokens are grouped into chunks based on their part-of-speech tags: "John Smith" (person), "works" (O), "at Google" (organization), "in" (O), "New York" (location).
- **Named entity classification:** The chunks are classified into named entities: "John Smith" (Person), "Google" (Organization), "New York" (Location).

For our research, we used the Spacy Python library and Natural JavaScript library which includes a named entity recognition (NER). Spacy's NER component is based on machine learning models that are trained on large annotated datasets, and it supports a wide range of named entity types, including people, organizations, locations, products, dates, and more. See (section 4.2) for the experiment.

SpaCy: To use Spacy's NER component, first a pre-trained model is needed to load that includes the NER component, such as the **"en_core_web_sm"** model for English language processing using Python. (section 4.4)

```
import spacy
nlp = spacy.load("en_core_web_sm")
text = "John Smith works at Google in New York"
doc = nlp(text)
for ent in doc.ents:
    print(ent.text, ent.label_)
```

This shows that Spacy’s NER component correctly identified and classified the named entities in the text as "John Smith" (a person), "Google" (an organization), and "New York" (a geopolitical entity).

Natural: Natural is a general-purpose NLP library for JavaScript that provides tools for tokenization, stemming, classification, and more using JavaScript.

```
const natural = require('natural');
const { Analyzer, WordTokenizer } = natural;

// Load the NER model
const Analyzer = new Analyzer();
Analyzer.load('./ner_model.json');

// Process a sample text
const text = 'John Smith works at Google in New York.';
const tokenizer = new WordTokenizer();
const tokens = tokenizer.tokenize(text);
const entities = Analyzer.getEntities(tokens);

// Print the named entities and their labels
console.log(entities);
```

This shows that the Natural Library’s NER module has correctly identified and classified the named entities in the text as "John Smith" (a person), "Google" (an organization), and "New York" (a location). See (section 4.7) for the experiment.

2.4.2 Vector Embedding

Vector embedding are a type of machine-learning technique used to convert text data into numerical vectors. The process involves training a model on a large corpus of text data, such as news articles or social media posts, to learn the relationships between words and phrases in the text. (Caputo, Basile, & Semeraro, 2017; Pappu, Blanco, Mehdad, Stent, & Thadani, 2019) The resulting vectors capture the semantic meaning of the text, allowing for more accurate comparisons between different pieces of text. (section 4)
By turning complex data into vector embedding, machine learning (ML) approaches may provide a much more useful representation. Models like Word2Vec, GLoVE, and BERT turn words, phrases, or paragraphs into vector embedding for text data. Complex data objects are described as numerical values in hundreds or thousands of different dimensions by vector embedding. It is the sole purpose

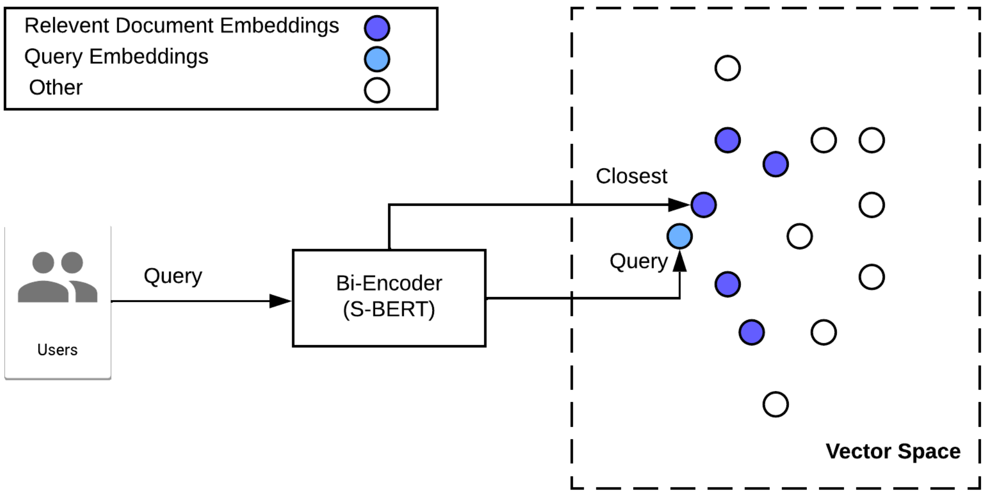


Figure 14: Representation of Vector Space with Semantic Search

of vector databases to manage the peculiar structure of vector embedding. By comparing values and identifying those that are most comparable to one another, they index vectors for simple search and retrieval. However, they are challenging to put into practice. Apps can do better searches while also achieving performance and financial objectives by utilizing a well-built vector database. To make it simpler to implement, there are a number of options. These solutions deal with security, availability, and performance and span from plugins and open-source initiatives to completely managed services. A vector database, which has features like CRUD operations, metadata filtering, and horizontal scaling, indexes and saves vector embedding for quick retrieval and similarity searches. Generally, there are two techniques to search text and documents. Lexical search looks for patterns and exact word or string matches, but the semantic search takes the meaning of the search query or question and sets it into context. Vector databases store and index vector embedding from Natural Language Processing models to understand the meaning and context of text strings, phrases, and complete texts for more accurate and applicable search results. By using natural language searches to obtain relevant results without having to be aware of the nuances of how the data is organized, users can find what they need more quickly and with a better user experience.

2.4.3 Machine Learning Embedding

Machine Learning (ML) will be used for a more helpful representation of complex data by transforming RDF data into vector embedding. For text data, models such as Word2Vec, GLoVE, and BERT transform words, sentences, or paragraphs into vector embedding. Vector embedding describe complex data objects as numeric values in hundreds or thousands of different dimensions. (Rekabsaz, 2017)

2.4.3.1 word2vec:

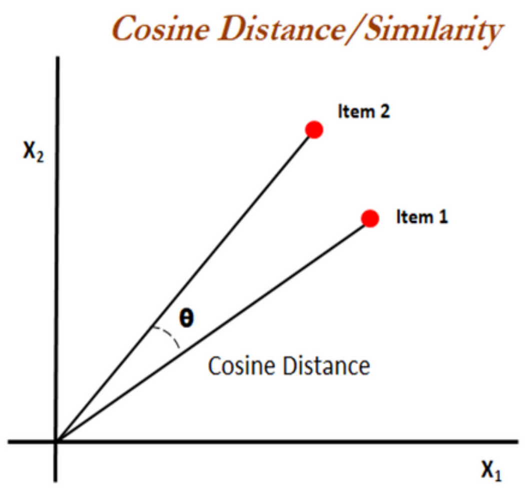
Word2vec is a popular natural language processing technique that is used to convert text data into numerical vectors, making it easier to analyze and process using machine learning algorithms. It comprises vectors corresponding to the words of the natural language rather than ontological instances, in contrast to the graph embedding. This necessitates an additional step of word2vec vectorizing the instances from the knowledge network. Word2vec is a neural network-based approach that uses unsupervised learning to learn the semantic meaning of words based on their context. (Holter, 2015) The basic idea behind word2vec is to create a vector representation of each word in a corpus by training a neural network to predict the probability of a word given its context. The context of a word is defined as the set of words that appear in its neighborhood within a given window size.
First, we preprocess the text by tokenizing it into individual words and removing stop words and punctuation. We then create a vocabulary of unique words in the corpus. Next, we train a word2vec model on the corpus by feeding it a sequence of word-context pairs. For example, given the sentence "The quick brown fox jumps over the lazy dog", the word2vec model would be trained on pairs such as ("quick", "The"), ("quick", "brown"), ("quick", "fox"), and so on. Once the model is trained, it creates a vector representation for each word in the vocabulary. These vectors are usually high-dimensional, typically between 100 and 300 dimensions. Each dimension in the vector represents a different feature of the word's meaning, such as its association with certain topics or its similarity to other words. We can then use these word vectors to perform various natural languages processing tasks, such as text classification or semantic retrieval. For example, we can calculate the cosine similarity between two-word vectors to measure their semantic similarity. The closer the cosine similarity is to 1, the more similar the words are in meaning.

2.4.3.2 Cosine Similarity

Cosine similarity is a measure used to calculate the similarity between two vectors, especially in natural language processing and information retrieval. (Rusinol, Aldavert, Toledo, & Llados, 2017; Abass & Arowolo, 2018) It measures the cosine of the angle between two non-zero vectors in n-dimensional space. In the context of enterprise information retrieval with semantic search, cosine similarity can be used to compare the similarity between a user query and a set of documents. For example, suppose a user enters a query "electric cars." The system would search for all documents related to electric cars, and then compare the similarity between the query vector and each document vector using cosine similarity. The document vectors can be generated by a variety of methods, such as TF-IDF or word embedding. Cosine similarity is calculated using the dot product of the two vectors divided by the product of their magnitudes. The formula for cosine similarity is as follows:

cosine similarity = (A . B) / (||A|| * ||B||)

where A and B are the two vectors being compared, and ||A|| and ||B|| are their magnitudes



$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 15: Representation of Cosine similarity

2.4.4 Vector Database

A vector database is a type of database designed for storing and searching high-dimensional numerical vectors, often used in machine learning and natural language processing (NLP) applications. These databases are optimized for efficient storage and retrieval of large amounts of vector data and often include features for performing vector operations and calculating vector similarity. See (section 4) for the experiment.

One example of a vector database is Pinecone which has been used in our experiment 1. See (section 4.5) for the experiment.

2.4.4.1 Pinecone

Pinecone is a vector database designed for storing and searching high-dimensional vectors. It uses vector embedding and semantic search to provide accurate and efficient search results. Pinecone supports a variety of popular embedding models, including Word2Vec, GloVe, and BERT. These models can be used to create vectors for words, phrases, sentences, and entire documents, allowing for a wide range of NLP applications. The database is optimized for high-dimensional vectors, allowing for fast and efficient storage and retrieval. The vectors can also be easily updated or deleted as needed. Pinecone's real power comes from its semantic search capabilities. When performing a search, Pinecone uses the vector representations of the query and the documents in the database to calculate the similarity between them. This allows for more accurate and relevant search results, even for complex or ambiguous queries. See (section 4.5) for the experiment.

2.5 Case Studies

The purpose of this case study is to demonstrate a method for performing a semantic search on a variety of document types using different combinations of RDF, GraphDB, VectorDB and embedding, Entity Extractions, and indexing. We aim to help users search for relevant information faster, with high accuracy and relevance.
In this study, we will be exploring the application of RDF, entity extraction, vector embedding, graph databases, and SPARQL in the context of three different research experiments. These techniques are commonly used in information retrieval and natural language processing applications to extract and analyze data from large volumes of unstructured text.

2.5.1 Case Study 1: Semantic Search Through Structured Datasets

In case study 1, we used Hugging Face's Quota dataset to train a model for semantic search and then used Pinecone as a vector database to store the embedding and perform an efficient search. This approach allows for quick and accurate retrieval of relevant documents based on their semantic similarity to a given query. See (section 4.4) for the experiment.

Some enterprises have their information in datasets for example JSON to retrieve and followups (see section 1.1.1) using their APIs. JSON (JavaScript Object Notation) is a widely used data interchange format that is easy for humans to read and write and can be easily parsed and processed by machines.

It is a popular choice for representing structured data. With this case study experiment (section 4.1), we will be able to retrieve information from structured datasets. The Hugging Face's Quota datasets and semantic search can be valuable tools for enterprise information retrieval. Incorporating Hugging Face's Quota datasets and leveraging NLP capabilities in semantic search systems can enhance enterprise information retrieval by improving query understanding, enabling fine-grained entity extraction, providing contextual search experiences, supporting multilingual search, and facilitating continuous learning and adaptation. These advancements contribute to more accurate, efficient, and relevant search results, empowering enterprises to efficiently access and utilize their vast information resources.

2.5.2 Case Study 2: Semantic Search Through Excel/CSV Document

From section 1.1.2, we can see that most energy sectors prefer their equipment data stored in an Excel or CSV document. In the case study experiment (section 4.1), we used indexing and vector databases for a semantic search of an Excel/CSV file. By transforming the contents of the file into a vector space representation, we were able to use a vector database to index and search the file based on semantic similarity. This approach has many potential applications in fields such as data analysis and document management. See (section 4.6) for the experiment.

Semantic search refers to the ability to understand the meaning and context of user queries and match them with relevant information, rather than relying solely on keyword matching. By leveraging vector embedding and semantic search techniques on CSV datasets, enterprise information retrieval can be significantly enhanced. The use of embedding enables a more nuanced understanding of text semantics, leading to improved relevance and accuracy in search results. This approach empowers enterprises to efficiently explore and retrieve information from their CSV datasets, enabling better decision-making, knowledge discovery, and insights extraction.

2.5.3 Case Study 3: Semantic Search Through Heterogeneous Structured and Un-Structured Documents using RDF Ontology

In the case study experiment (section 4.1), we used Protege for modeling and RML mapping rules to transform data into RDF format and then used a graph database (GraphDB) and SPARQL to perform a semantic search. This approach is particularly useful for complex data structures and allows for sophisticated querying and analysis of relationships between entities. See (section 5) for the experiment.

The combination of RDF ontology, graph database, Protege modeling, and NLP techniques enhances information retrieval in multiple ways. It enables precise search by leveraging the structured semantics defined in the ontology. It allows for efficient indexing, retrieval, and ranking of documents based on their semantic relevance. It supports exploratory search and discovery of implicit relationships and connections in the knowledge graph. Additionally, it facilitates integration with other enterprise systems and applications, enabling seamless access to relevant information which perfectly aligns with our thesis aim.

These case studies serve as practical demonstrations of the application of semantic technologies and information retrieval techniques in different contexts. They explore the use of vector embedding, graph databases, and SPARQL to extract and analyze data from unstructured sources. By experimenting with different approaches, researchers aim to develop more sophisticated and effective systems for handling unstructured data. The research methodology employed in these studies involves careful data collection and analysis. Researchers selected appropriate datasets for each case study, ensuring their relevance to the research objectives. Evaluation metrics were employed to measure the effectiveness of each approach, providing quantitative insights into their performance. The results obtained were then interpreted to draw meaningful conclusions regarding the effectiveness of semantic technologies and information retrieval techniques for extracting and analyzing data from unstructured sources.

3 State of the Art: Literature Review on Semantic Search Techniques for Heterogeneous Documents

Semantic search plays a crucial role in extracting meaningful information from heterogeneous document collections. This literature review aims to provide an overview and analysis of recent research papers in the field of semantic search, RDF ontology, information retrieval, and related areas. The review focuses on approaches utilizing vector embedding, RDF ontology, Protege modeling, RML ruling, GraphDB, knowledge mapping, and entity extraction to enhance semantic search on heterogeneous documents.

A user-centered data-driven methodology is put forth by the authors to enhance information discovery, collaboration, and traceability in complex systems engineering contexts. Diverse stakeholders who produce and interact with heterogeneous data are present in these environments, which presents difficult problems. To enhance the entire information management process, the authors advise combining data analytics methods, visualization, and user-centered design concepts. They stress the value of including end users in all stages of the design process and provide methods to make it easier for people to explore and find pertinent information. The authors want to aid in decision-making processes and give users pertinent information by analyzing and interpreting data. They emphasize the need for user-friendly, dynamic interfaces for efficient data navigation. The authors of the paper describe methods to enhance collaboration, and information sharing, and maintain traceability throughout the system development lifecycle. The paper emphasizes the significance of clear communication and traceability in complex systems engineering. Overall, the study offers a thorough strategy for tackling the difficulties of information management in complex systems engineering settings. The authors stress the significance of a user-centered strategy that makes use of data analytics methods to glean insights from a variety of data sources. The authors seek to increase the effectiveness and efficiency of information management procedures, which will ultimately improve the results of system development, by incorporating user feedback throughout the design process and offering user-friendly interfaces. (Salim et al., 2019)

V. Jirkovsky and M. Obitko have proposed a solution to address the challenge of integrating heterogeneous data sources in the industrial automation domain. Their approach involves the use of a shared ontology to reduce data source heterogeneity and enable the integration of diverse data, such as sensor measurements and data from MES/ERP systems. This solution was demonstrated in a passive house testing use case, highlighting its capability to handle various data sources stored in databases or files, as well as various streams of data. The ability to handle diverse data sources and capture relationships among them is crucial for effective decision-making, not only in the industrial automation domain but also in other enterprises. As demonstrated in a state-of-the-art review, the variety aspect of Big Data is a critical factor, but a challenging one to address. To remain competitive, enterprises must be able to handle large volumes of data at high speed and integrate various data sources. The heterogeneity of data has always been a significant issue to address, but it has become even more crucial in today's world, where the trend of processing increasingly vast amounts of data continues. (Jirkovsky & Obitko, 2018)

The authors, Smith, J. et al, have tackled the issue of data retrieval from diverse sources within industrial settings. In diverse settings, data is frequently stored in varied formats and structures, thereby posing challenges to efficient access and retrieval of information. The proposed approach put forth by the authors is founded on the analysis of a heterogeneous information network (HIN). The article presents the notion of Heterogeneous Information Networks (HINs) and elucidates their capacity to depict the interconnections and interdependencies among diverse categories of data in industrial settings. The significance of comprehending these associations is underscored by the authors in order to enhance data retrieval and accessibility. The proposed methodology employs network analysis methodologies to extract significant patterns and correlations from the Heterogeneous Information Networks (HINs). Through the utilization of these patterns, the authors illustrate the efficacy of retrieving pertinent information. The authors additionally deliberate on the utilization of machine learning algorithms to augment the retrieval process through the integration of user preferences and feedback. The authors performed experiments on actual industrial datasets to authenticate their methodology. The authors conduct a comparative analysis of their proposed approach against established retrieval methodologies, demonstrating favorable outcomes in both retrieval precision and expediency. (Shi, Li, Zhang, Sun, & Yu, 2015)

Wang, Liu and colleagues acknowledge that industrial settings frequently entail disparate data sources, encompassing diverse formats, structures, and languages. The process of extracting pertinent data from said sources can prove to be intricate and require a significant amount of time. The objective of the present study is to suggest efficacious methodologies for augmenting information retrieval within this particular context. The taxonomy proposed by the authors classifies various data sources that are

frequently encountered in industrial environments. The implementation of this particular taxonomy facilitates comprehension of the multifarious and intricate nature of the various origins of data. This paper examines diverse methodologies for the integration and preprocessing of heterogeneous data sources. The objective of these methodologies is to achieve data harmonization and minimize discrepancies, thereby enhancing the efficiency of data retrieval. The authors suggest the utilization of semantic representation techniques for industrial data to enhance the effectiveness of information retrieval. The authors deliberate on techniques pertaining to the extraction of pertinent features and the construction of semantic indices that effectively encapsulate the fundamental relationships and connotations inherent in the given dataset. The study additionally investigates methodologies for managing user inquiries across disparate data repositories. The aforementioned address the aspects of query translation, optimization, and result ranking, with the aim of enhancing the precision and efficacy of information retrieval. The efficacy of the proposed techniques is demonstrated through the presentation of case studies and experimental evaluations by the authors. The authors assess the efficacy of the information retrieval system by employing authentic industrial datasets. (Wang, Liu, & Zhang, 2019)
Munir and colleagues acknowledge that the storage of information in industrial settings is frequently characterized by disparate formats and structures across multiple systems, posing challenges to the efficient retrieval and integration of data. The intricacy and heterogeneity of data in said environments may surpass the capabilities of conventional keyword-based methodologies. The paper presents a proposed framework for semantic information retrieval that utilizes semantic technologies and techniques as a means of addressing the aforementioned issue. The objective of the framework is to enhance the accuracy and completeness of retrieving information through the utilization of semantic annotations, ontologies, and reasoning mechanisms. The framework's fundamental constituents are expounded by the authors, encompassing data acquisition, semantic annotation, ontology development, query formulation, and results presentation. The authors engage in a discourse regarding the interplay of these constituents, which collectively facilitate proficient acquisition of information from disparate and diverse data repositories. Additionally, the article presents a case study in which the suggested framework is implemented in an industrial setting. The authors illustrate the efficacy of the framework in retrieving pertinent information from diverse data sources by means of semantic annotations and ontologies-based integration. (Munir, Odeh, McClatchey, Khan, & Habib, 2014)

Asfand-e-yar et al emphasize that within numerous industrial and organizational contexts, information is frequently stored within distinct databases, each possessing its own unique schema and structure. The presence of heterogeneity presents difficulties in the retrieval and analysis of data across multiple databases. The article advocates for the utilization of ontology, a structured and systematic representation of knowledge, as a means to overcome the disconnect between databases and to promote semantic integration. The framework proposed by the authors employs ontology as a means to facilitate the mapping and integration of disparate databases. The ontology functions as a universal lexicon that documents the collective notions and associations within the given field. Through the process of schema and data instance mapping from disparate databases onto the ontology, the authors have successfully established a unified semantic framework that facilitates data querying and access. This paper examines the various stages of the semantic integration procedure, encompassing the creation of ontology, schema mapping, and data mapping. The significance of establishing unambiguous connections and correspondences between the entities within the ontology and the related components in the databases is underscored. The authors put forth an algorithm designed for automated schema matching, which facilitates the identification of similarities and correspondences among database schemas. Moreover, the article provides a case study as a means of demonstrating the efficacy of the proposed framework. The authors have successfully integrated databases from a practical domain and have employed ontology to showcase the efficient retrieval, querying, and analysis of data from these diverse sources in a cohesive manner. (Asfand-e yar & Ali, 2016)

Zaman et al acknowledge that scientific knowledge is dispersed across multiple sources and frequently exists in diverse formats, posing difficulties in the efficient retrieval and integration of pertinent information. The authors suggest a solution to this issue by presenting an ontological framework that utilizes the semantic representation of scientific knowledge to aid in the extraction and integration of information. The framework comprises three primary constituents, namely the Ontology Generation Module, the Semantic Annotation Module, and the Information Extraction Module. The module tasked with constructing a domain-specific ontology is the Ontology Generation Module, which accomplishes this by extracting pertinent concepts and relationships from scientific documents. The system employs natural language processing methodologies and domain-specific heuristics to detect significant entities and relationships. The Semantic Annotation Module applies semantic metadata to scientific documents utilizing the ontology that has been produced. The process involves linking pertinent concepts and connections to distinct sections of the documents, thereby augmenting their semantic depiction. The module for Information Extraction employs annotated documents and an ontology to extract information in a structured manner. The utilization of methods such as entity recognition, relation extraction, and document indexing facilitates efficient retrieval and integration of information. The objective of the proposed framework is to address the obstacles associated with extracting information from a wide range of scientific sources by utilizing the semantic representation facilitated by an ontology. The framework is designed to enhance the extraction of significant information and enable seamless retrieval and integration across diverse scientific sources through the utilization of advanced natural language processing techniques and domain-specific knowledge. (Zaman et al., 2018)

Abbas et al commence by introducing the significance of information retrieval in contemporary society, which is increasingly reliant on data. The capacity to access and retrieve pertinent information efficiently is critical. The statement underscores the difficulties presented by the copious quantity of information accessible in various forms and originating from heterogeneous origins, thereby rendering the process of retrieval intricate. The core components of an information retrieval system are deliberated by the authors, which comprise document collection, query processing, indexing, and relevance ranking. Various information retrieval (IR) models are delineated, including the Boolean model, vector space model, probabilistic models, and language models. The paper presents an analysis of the strengths and limitations of various models and evaluates their appropriateness for different retrieval tasks. Additionally, the article examines the methodologies employed in the field of information retrieval, including but not limited to term weighting, query expansion, relevance feedback, and clustering. The objective of these methodologies is to augment the efficacy of information retrieval through the enhancement of document and query representation, along with the integration of user feedback to refine the outcomes. The discourse examines the assessment metrics utilized to evaluate the efficacy of information retrieval systems, encompassing precision, recall, F-measure, and average precision. The significance of user-centered evaluation and the influence of user satisfaction in assessing the efficacy of retrieval systems is emphasized by the authors. The latter section of the manuscript explores diverse implementations of information retrieval in distinct fields, such as web exploration, digital repositories, corporate exploration, and individualized suggestion mechanisms. The authors delineate the distinct obstacles and methodologies linked to each application domain, thereby demonstrating the extensive spectrum of practical situations where information retrieval assumes a crucial function. (Abass & Arowolo, 2018)

Anjomshoaa et al acknowledge the significance of enterprise resource planning (ERP) systems in facilitating business process management. However, these systems are frequently confronted with obstacles related to semantic heterogeneity, data integration, and interoperability. The authors initiate the discourse by delving into the fundamental notions of Semantic Web and ERP systems. The objective of the Semantic Web is to augment the functionality of the internet by imbuing web content with semantic and structural attributes, thereby facilitating the comprehension and processing of data by automated systems. Enterprise Resource Planning (ERP) systems are designed to integrate multiple business functions and processes within an organization. The advantages of integrating Semantic Web technology into Enterprise Resource Planning (ERP) systems are emphasized by the authors. The authors contend that the utilization of semantic technologies, such as ontologies and knowledge representation, can enable Enterprise Resource Planning (ERP) systems to surmount the obstacles of interoperability and data integration. The utilization of semantic technologies has the potential to establish a unified lexicon and mutual comprehension of data throughout diverse modules and constituents of the Enterprise Resource Planning (ERP) system. This manuscript showcases a case analysis that illustrates the utilization of Semantic Web technology within an Enterprise Resource Planning (ERP) framework. The authors explicate the development of an integration framework based on ontology, which effectively integrates disparate data sources and augments data retrieval and decision-making procedures within the ERP system. The authors demonstrate the application of ontologies as a means of representing and enhancing data with semantic information, thereby facilitating sophisticated querying, reasoning, and integration of data functionalities. Moreover, the authors analyze the difficulties and prospective avenues of employing Semantic Web technology within Enterprise Resource Planning (ERP) systems. The aforementioned concerns pertain to matters of scalability, performance, and impediments to adoption. The study concludes by underscoring the capacity of Semantic Web technology to enhance data accessibility, interoperability, and decision-making procedures in Enterprise Resource Planning (ERP) systems. (Anjomshoa, Karim, Shayeganfar, & Tjoa, 2020)

The authors, Dinesh A. Zende and Chavan Ganesh Baban aim to tackle the obstacles related to conducting effective semantic searches on extensive RDF (Resource Description Framework) datasets. The authors express recognition of the insufficiency of conventional keyword-based search techniques in retrieving pertinent information as the quantity of RDF data expands. The authors suggest a semantic search methodology that exploits the organized structure of RDF data and integrates semantic associations to enhance the precision and effectiveness of search outcomes. The authors put forth a proposal for an indexing mechanism that arranges the RDF data into a compressed index structure, thereby enabling expedited search operations. The authors deliberate on diverse methodologies, including vertical partitioning and dictionary encoding, to enhance the effectiveness of storage and retrieval operations. Furthermore, the authors propose a query processing algorithm that effectively assesses semantic search queries on the indexed RDF data. The authors have proposed a technique for measuring semantic similarity that effectively captures the semantic relationships between entities in RDF, thereby improving search accuracy. The utilization of ontological knowledge and the RDF graph's structure is employed to compute similarity scores, thereby facilitating the retrieval of more pertinent outcomes. The present study tackles the matter of scalability through the proposition of a scalable methodology that employs parallel processing methods to manage voluminous RDF datasets. The authors examine the feasibility of deploying the proposed indexing and query processing mechanisms in a distributed computing environment with the aim of enhancing performance. The empirical assessment of the suggested methodology showcases its efficacy with regard to precision in search results and promptness in the query response. The findings suggest that the utilization of semantic search methodology exhibits superior performance compared to conventional keyword-based techniques in retrieving outcomes that are more pertinent. (Zende & Baban, 2015)

The study conducted by Xiaolong Tang and colleagues aims to tackle the difficulties associated with retrieving data from extensive RDF (Resource Description Framework) datasets through the utilization of ontology-based semantic search methodologies. The Resource Description Framework (RDF) is a widely adopted conceptual framework utilized for the purpose of representing knowledge on the Semantic Web. The authors commence their discourse by examining the constraints of conventional keyword-centric search methodologies for Resource Description Framework (RDF) data, which frequently prove inadequate in encapsulating the intricate semantics and interconnections among entities in the data. The authors suggest a framework for semantic search that utilizes ontology-based methods to improve the precision and efficiency of the search process, thereby addressing the aforementioned limitations. The framework comprises three primary constituents, namely query expansion, query rewriting, and result ranking. The objective of the query expansion module is to enhance the original user query by integrating associated terminologies and notions from the ontology. This procedure facilitates the capture of the wider context and semantic associations inherent in the RDF data. The component was responsible for query rewriting is tasked with converting the expanded query into a more precise format, which involves aligning it with the ontology structure to enhance the accuracy of matching. The result ranking module employs a scoring mechanism to arrange the retrieved outcomes in order of their pertinence to the user's inquiry. The authors expound upon the technical aspects of their framework, which encompass the development of an ontology index and the utilization of optimization techniques to address the challenges of accommodating voluminous RDF data. The researchers performed empirical investigations utilizing authentic datasets to assess the efficacy and efficiency of their methodology. The findings of their research indicate that the framework for semantic search based on ontology exhibits superior performance in retrieval accuracy and semantic relevance compared to conventional keyword-based methods. The conclusion drawn by the authors is that the utilization of ontology structure and semantics has the potential to greatly improve the retrieval capabilities of RDF data on a large scale. (Tang et al., 2021)

The authors, Shahrul Azman Noah, and his colleagues investigate the creation and execution of a semantic digital library that is driven by an ontology. The authors' primary objective is to leverage semantic technologies and ontologies to augment the organization, retrieval, and utilization of digital resources in a library context. The article underscores the obstacles encountered by conventional digital libraries, including the challenge of efficiently managing and retrieving diverse resources from multiple origins. The authors suggest utilizing an ontology-driven methodology that utilizes semantic web technologies to enhance the depiction of resources and enhance their detectability, as a means of tackling these obstacles. The authors explicate the fundamental constituents of their proposed system, comprising a module for ontology development, a module for indexing and retrieval, and a module for the user interface. The significance of ontologies in capturing the semantics of digital resources is emphasized, as it enables search and retrieval functionalities that are more intelligent and context-aware. The present study showcases a case analysis wherein the suggested system is executed in an actual digital library environment. The article outlines the methodology employed in the creation of ontologies, the incorporation of these ontologies into the digital library infrastructure, and the assessment of the system's efficacy in facilitating the identification and retrieval of resources. The findings of the case study suggest that the utilization of ontology-driven semantic digital libraries provides enhanced search and retrieval functionalities in contrast to conventional digital libraries. The utilization of ontologies facilitates the system to furnish search outcomes that are more exact and pertinent, in addition to endorsing sophisticated functionalities like semantic exploration and suggestion. (Noah et al., 2013)

The authors, Petri Kivikangas and Mitsuru Ishizuka investigate the potential of leveraging the Universal Networking Language (UNL) ontology and a graph database to optimize the efficacy of semantic queries. The authors have drawn attention to the constraints of conventional search techniques that rely on keywords to retrieve pertinent information. They have suggested a semantic query approach as a viable alternative. The authors present the UNL ontology, a linguistic construct that has been specifically devised to encode knowledge in a manner that is both agnostic to natural language and amenable to machine interpretation. The UNL ontology serves as a tool for the transformation of natural language inquiries into a structured configuration that can be efficiently computed by machines. The authors have utilized a graph database to facilitate the implementation of semantic querying, thereby enabling the effective storage and retrieval of interconnected data. The UNL ontology is modeled utilizing the graph database structure, which facilitates the depiction and interconnection of diverse concepts and relationships. The manuscript outlines a comprehensive framework that integrates the UNL ontology and graph database to facilitate semantic querying. The process of converting natural language queries into UNL expressions is expounded upon by the authors, who also provide an illustration of how these expressions are subsequently mapped to the structure of the graph database. The authors additionally deliberate on the advantages of employing a graph database, including the capability to navigate connections and obtain pertinent data contingent on the query context. The method under consideration is assessed via experimental analysis, wherein a comparison is made between the semantic querying system and conventional keyword-based techniques. The findings indicate that the utilization of the UNL ontology and graph database can enhance the precision of queries and the retrieval of semantically associated data. (Kivikangas & Ishizuka, 2020)

A framework is put out by Yassine Mrabet, Nacéra Bennacer, Nathalie Pernelle, and Mouhamadou Thiam to address the issue of doing semantic searches on diverse semi-structured texts. They emphasize the popularity of these papers in their article across a range of industries, including e-commerce, digital libraries, and scientific journals. Due to the rich and varied content of these papers, which standard keyword-based techniques find difficult to successfully capture, it can be difficult to retrieve pertinent information from them. The authors provide a three-step approach for facilitating semantic search on heterogeneous semi-structured documents in order to overcome this difficulty. They use methods like named entity identification and entity linking in the document preprocessing step to identify entities inside the documents and link them to external knowledge bases like DBpedia or Wikidata. By adding semantic annotations to the documents during preprocessing, the groundwork is laid for a more insightful search. The Resource Description Framework (RDF) data model is then used to convert the enriched pages into a semantic index. As it captures the semantic links between entities present in the texts, this structured representation enables effective information organization and retrieval. The authors make sure that the index includes both the links between entities and individual entities by utilizing RDF. The framework's final phase entails running semantic search queries on the indexed texts. The authors suggest a query expansion strategy to improve the search process that makes use of the semantic annotations gathered during the preprocessing stage. The extended searches produce more precise and pertinent search results because they take into account the semantic relationships between entities. The authors run tests on a dataset made up of various semi-structured papers to validate their system. They contrast the effectiveness of their strategy with more established keyword-based search methods. The outcomes show that the suggested framework greatly increases the recall and precision of the search results, proving its viability in overcoming the difficulties presented by diverse semi-structured texts. (Mrabet, Bennacer, Pernelle, & Thiam, 2018)

The topic of semantic reconciliation in multi-data source management systems is addressed by Gilles Nachouki, Mirna Nachouki, and Marie-Pierre Chastang in their study, which emphasizes the presence of heterogeneous data sources in various areas, notably in industrial settings. It is challenging to successfully integrate and extract information from these data sources due to the frequent variations in their formats, schemas, and vocabularies. The authors suggest a peer multi-data source management system that seeks to resolve semantic discrepancies across participating data sources as a solution to this problem. The system makes use of a distributed design, where each peer represents a different data source and interacts with other peers to exchange and reconcile data. In order to achieve semantic reconciliation, the authors provide the idea of semantic bridges, which serve as intermediary elements that make it easier to map and translate data between various data sources. For the purpose of locating and resolving semantic discrepancies in the data, ontologies and semantic matching techniques are used. The suggested system also has a query processing component that lets users run queries across many data sources. In order to properly analyze the queries and return insightful answers, the system takes into account the semantic bridges and reconciled data. The authors carried out experiments in a simulated setting to assess the system's performance, examining several factors such as query response time and scalability to show the efficiency and viability of their strategy. Overall, their system offers a promising response to the problem of semantic reconciliation in multi-data source management systems, which is crucial for successfully integrating and making use of numerous data sources in various fields. (Nachouki, Nachouki, & Chastang, 2015)

Udayan Khurana and Sainyam Galhotra examine the idea of semantic search and how it relates to structured data. In order to increase the precision and relevance of search results, the authors discuss the shortcomings of standard keyword-based search techniques and suggest a semantic search methodology. The paper starts out by discussing the difficulties of searching structured data, where conventional keyword-based approaches frequently fall short of capturing the context and semantic links between things. The authors stress the requirement for a more sophisticated and semantically based approach to improve search capabilities across structured data. Ontologies and knowledge graphs are two examples of semantic technologies that are used in the suggested approach. The authors explain how these technologies can be used to represent the connections and semantics of data elements, facilitating more accurate and contextually aware search. They also go through several methods for creating ontologies and knowledge graphs, such as automatic extraction from pre-existing data sources and manual development. The integration of semantic search algorithms with relational databases and structured query languages (SQL) is further explored in this work. The difficulties and potential in developing query processing algorithms that include semantic search capabilities are discussed by the authors. They offer knowledge about query speed optimization while utilizing the semantic data that is encoded in the structured data. The significance of entity recognition and disambiguation in semantic search over structured data is also covered by the authors. To increase search accuracy, they highlight methods for locating and clearing up entity ambiguities as well as addressing entity synonyms and polysemy. The authors use examples and case studies to demonstrate the application and advantages of semantic search over structured data throughout the entire study. They stress that by utilizing semantic search approaches, users can obtain more accurate and pertinent search results, facilitating improved data exploration and decision-making across a variety of fields.(Galhotra & Khurana, 2016)

Dragan Gasevic, Sasa Nesic, Fabio Crestani, and Mehdi Jazayeri examine the difficulties and strategies for effective search and navigation in semantically integrated document collections. The challenge of information retrieval and navigation in collections of documents from many sources with various structures and vocabularies is the authors' main concern. To enable efficient search and navigation in such disparate document collections, semantic integration is frequently needed. In the study, a framework for semantic integration that annotates documents with metadata and ontology ideas is proposed. As a result, retrieval and navigation are more accurate and may be done at a higher level of semantic understanding. The authors explore various querying methods, such as ontology-based search and keyword-based search. They stress how crucial it is to use semantic annotations and concepts to enhance the recall and precision of search results. The paper examines different navigation techniques, including concept-based navigation and faceted navigation. While concept-based navigation allows for navigating through the semantic relationships between concepts, faceted navigation allows users to explore documents based on several dimensions. The authors use a case study with a real-world document collection to evaluate their suggested approach. The effectiveness of various search and navigational methods is compared, and the effects of semantic integration on retrieval and navigational performance are examined. The evaluation's findings demonstrate how better search results and a better user experience when moving through document collections thanks to semantic integration. The constraints of the suggested approach, including scalability and ontology creation, are also highlighted in the paper. The requirement for scalable algorithms for semantic integration, methods for dealing with dynamic document collections, and the incorporation of user feedback for tailored search and navigation are just a few of the future research directions covered by the authors. (Nešić, Crestani, Jazayeri, & Gašević, 2010)

In the context of the French healthcare system, Thibaut Pressat-Lafouilhère et al. gives an assessment of Doc'EDS, a semantic search tool built for querying health documents from a clinical data warehouse. The authors begin by pointing out the difficulties experienced by medical professionals in locating pertinent information in the abundance of medical literature available. To make it easier to access relevant medical information, they stress the significance of efficient information retrieval technologies. In order to overcome these issues, the study introduces Doc'EDS, a semantic search engine. To improve the search experience for healthcare professionals, it makes use of semantic technology and natural language processing methods. Through comprehension of the context and meaning of the user's query, the tool seeks to deliver exact and pertinent search results. The authors conducted a study with a panel of medical experts to gauge Doc'EDS' effectiveness. They contrasted the tool's efficiency in locating pertinent data from the clinical data warehouse with that of other traditional search techniques. Precision, recall, and user satisfaction were among the metrics used in the review. The evaluation's findings demonstrated that Doc'EDS performed better in terms of precision and recall than the conventional search techniques. The participants were also more pleased with the tool's capacity to locate pertinent medical records. The authors talk about the ramifications of these findings and emphasize how Doc'EDS and other semantic search technologies can help with information retrieval in healthcare settings. (Pressat-Laffouilh'ere et al., 2018)

By utilizing explicit knowledge through data recycling, Alex Kohn, François Bry, and Alexander Manta explore the idea of semantic search and its application to unstructured data. The authors begin by outlining the difficulties involved in looking for and obtaining pertinent information from unstructured data, especially textual materials. They draw attention to the fact that conventional keyword-based techniques frequently fail to effectively capture the semantic meaning and contextual nuances of the data. The research introduces a semantic search technique that uses data recycling to harness explicit knowledge in order to overcome these restrictions. FACTS (Fact-based Content Exploration and Search), the suggested methodology, focuses on the extraction of explicitly represented knowledge units, or "facts," from unstructured data. Following that, a structured knowledge base is used to store these data, enabling effective semantic search operations. The authors describe the fact extraction procedure, which mixes domain-specific heuristics with natural language processing methods to produce insightful conclusions. The extracted facts enable more accurate and context-aware search results by displaying the unstructured data's underlying semantics. The semantic search capabilities built into the FACTS framework enable users to submit natural language inquiries that are converted into structured searches based on the knowledge base. By determining the facts' semantic relevance to the query, the system retrieves pertinent information, producing more precise and contextually aware results. The paper also highlights the significance of data recycling, where the knowledge base is continuously improved and enriched through iterative user input. Over time, this feedback system improves the search results relevancy and accuracy. The authors run experiments with real-world datasets to verify the FACTS framework's effectiveness. The outcomes show that the semantic search methodology outperforms more conventional keyword-based techniques. The research concludes by presenting a ground-breaking method for semantic search on unstructured data by utilizing explicit knowledge from data recycling. The FACTS framework presents a promising approach for attaining more precise and context-aware information retrieval from unstructured sources, bringing significant contributions to the disciplines of knowledge management and information retrieval. (Kohn, Bry, & Manta, 2016)

A retrieval framework and implementation particularly created for electronic documents with similar layouts are presented by Hyunji Chung in a research article. The difficulty of information retrieval from documents with comparable visual structures, such as forms or templates used in industrial settings, is covered in this work. The author starts out by emphasizing the value of effective information retrieval from structured documents in a variety of fields, including administration, finance, and manufacturing. However, because they rely heavily on text-based indexing and retrieval techniques, conventional retrieval systems frequently have trouble handling documents with comparable layouts. Chung suggests a retrieval strategy that makes use of the visual arrangement information of in documents to get over this issue. Document segmentation, feature extraction, and similarity matching are the framework's three key processes. The division of a document into useful segments, such as form fields or tables, is known as document segmentation. To effectively detect and extract these pieces, Chung combines visual and textual analysis approaches. The framework then concentrates on feature extraction, computing pertinent visual features for each section. The author explains how to identify the different qualities of document segments by using a variety of factors, such as geometric traits, structural relationships, and textual content. Similarity matching, the last phase, entails contrasting the features derived from a query document with those of a target document database. Chung suggests an effective indexing framework based on the traits identified, allowing quick retrieval of publications with comparable layouts. Using a real-world dataset of electronic documents, the author ran experiments to gauge the usefulness of the suggested approach. The findings show that the framework outperforms traditional approaches that just use text-based techniques in terms of retrieval accuracy and efficiency. Hyunji Chung's research concludes by presenting a retrieval architecture and implementation that are specially designed for electronic documents with comparable layouts. The suggested system enables more efficient information retrieval from structured documents in industrial settings by adding visual layout information and combining segmentation, feature extraction, and similarity-matching algorithms. (Chung, 2019)

A methodology for utilizing semantic technologies to improve the integration process in Enterprise Resource Planning (ERP) systems is put forth by Naglaa M. Badr, Emad Elabd, and Hatem M. Abdelkader. The authors begin by outlining the difficulties encountered with integrating ERP systems, which frequently involve difficult data mappings and semantic discrepancies across various systems. They stress the requirement for a solution that takes on these issues and enables seamless data integration. To enable a more clever and effective integration process, the suggested framework makes use of semantic technologies like ontologies and semantic web services. It focuses on improving data transformation, integration, and mapping between different ERP systems. There are many parts to the framework. The shared domain knowledge and the semantic links between various items in ERP systems are first represented by an ontology. Using ontology as a common language will make it easier to map and integrate data. The authors then put forth a semantic mapping component that makes use of the ontology to swiftly find and correct semantic discrepancies between data pieces in various ERP systems. As a result, data mapping requires less human work and integration is more accurate. The framework also has a semantic transformation element that uses semantic rules and reasoning methods to structure the combined data in a unified and consistent manner. With the help of this stage, the integrated data is maintained semantically coherent and is ready for use by ERP systems. The authors ran trials with actual ERP systems to gauge the efficiency of the suggested structure. The outcomes show that the framework effectively handles the difficulties associated with ERP integration and enhances the speed and accuracy of data integration procedures. The research concludes by presenting a semantic-based framework that improves ERP system integration through the use of ontologies, semantic mapping, and transformation methodologies. The suggested architecture provides a potentially effective approach to the problems associated with data integration and makes it easier to collaborate and communicate between various ERP systems. (Badr, Elabd, & Abdelkader, 2016)

An strategy to building an Energy Knowledge Graph (EKG) is presented by Duan Popadi, Enrique Iglesias, and Ahmad Sakor with the goal of improving knowledge sharing and information retrieval in the energy sector. The authors begin by underlining the difficulties in managing and gaining access to enormous amounts of data on energy from many sources. They underline the necessity of structuring this data in order to facilitate effective retrieval and analysis. In response, they offer the EKG as a remedy, which makes use of semantic technologies and the foundations of knowledge graph development. The approach for creating the EKG is described in the study, and it entails the extraction, fusion, and modeling of energy-related data from numerous sources. The authors outline the steps involved in extracting data, which include finding and gathering pertinent information from a variety of sources, including databases, reports, and academic articles. They talk about how crucial data fusion is and how the EKG makes it possible to combine many data types, formats, and vocabularies into a single graph representation. Additionally, the authors emphasize how semantic technologies, particularly ontologies, can be used to enhance the EKG and enable meaningful links between energy concepts and entities. They discuss the creation and use of domain-specific ontologies, which give the knowledge network a common language and permit semantic searching and reasoning. The benefits and possible uses of the EKG are also covered in the paper. It emphasizes how diverse use cases, such as data exploration, decision-making, and knowledge sharing among energy experts, can be supported by the structured representation of energy information. The authors also stress the EKG's potential to aid in the creation of sophisticated analytics and machine learning models in the energy sector. The study concludes by outlining a method for creating an Energy Knowledge Graph (EKG) that tackles the problems associated with information retrieval and knowledge exchange in the energy industry. The authors give evidence for the significance of semantic technologies and structured data representation in facilitating effective data integration, querying, and analysis. By offering a consistent and comprehensive knowledge base, the proposed EKG has the potential to improve energy-related research, decision-making, and collaboration.(Popadić, Iglesias, Sakor, Janev, & Vidal, 2019)

The work of Shanshan Jiang, Thomas F. Hagelien, Marit Natvig, and Jingyue Li focuses on the creation of a semantic search system for open government data that is ontology-based. The authors discuss the difficulties in locating pertinent data in varied and dispersed government datasets. The importance of open government data is emphasized in the article, as well as the necessity for efficient search tools that make it possible for consumers to access and make use of this priceless resource. To improve the retrieval process, the authors suggest a method that includes ontology modeling, semantic annotation, and a search engine. The authors create an ontology that is specially made to capture the linkages and domain knowledge found in government datasets. The data is represented in an organized manner via ontology, improving organization and searchability. The method of applying the created ontology to semantically annotate government data is discussed in the study. This entails adding relevant metadata to the data in order to improve its searchability and interoperability. The authors present a query expansion method that makes use of the semantic connections represented in the ontology. Through the inclusion of relevant terms and concepts in user searches, this strategy improves search accuracy. The study offers a ranking system that compares queries and datasets based on both textual relevance and semantic similarity. The algorithm seeks to deliver more precise and pertinent search results given the current context. Using actual government datasets, the authors test their ontology-based semantic search system. They demonstrate the system's potential to enhance information retrieval from open government data by evaluating its performance in terms of precision, recall, and user happiness. (Jiang, Hagelien, Natvig, & Li, 2015)

A generalized language model for information retrieval utilizing word embedding is suggested by Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J.F. Jones. The authors use word embedding, which identify semantic similarities between words, to overcome the difficulties associated with information retrieval from heterogeneous documents and sources. They put out a cutting-edge framework that incorporates word embedding into a broad language model. The suggested technique converts both query and document terms into continuous vector representations using a pre-trained word embedding model. To calculate document relevance scores, these embedding are then merged with conventional term frequency-inverse document frequency (TF-IDF) weights. The authors present a query extension strategy that leverages word embedding to expand the initial query with extra pertinent terms in order to further improve retrieval performance. This expansion tries to better the retrieval precision and capture the semantic context of the query. The research provides a thorough experimental evaluation on a number of benchmark datasets to contrast the suggested model with current retrieval methods. The outcomes show that the generalized language model based on word embedding performs better than conventional approaches, leading to higher retrieval effectiveness. The authors offer an innovative method for information retrieval that mixes word embedding with conventional retrieval models in their conclusion. The addition of word embedding makes it possible to identify semantic connections between terms, improving retrieval performance. The experimental results support the effectiveness of the suggested approach and demonstrate its potential for resolving industrial issues with data access and information retrieval from diverse sources. (Ganguly et al., 2015)

Utilizing modified word embedding, Navid Rekabsaz focuses on enhancing information retrieval methodologies. The author discusses the difficulty of efficiently locating pertinent information in huge document collections. The significance of word embedding in expressing the meaning and semantic links between words is covered in the first section of the study. Traditional word embedding, like Word2Vec and GloVe, are trained on generic corpora and might be unable to fully capture the particular domain knowledge and context of industrial documents. The author suggests a method to modify word embedding for the industrial sector in order to get over this restriction. Pre-trained word embedding are adjusted during the adaptation process utilizing a reduced domain-specific corpus or a constrained amount of labeled data. The word embedding are better suited for information retrieval tasks in this context as a result of this adaptation, which helps them conform to the language used in industrial texts. The author follows up with an experimental analysis of the suggested methodology using a dataset made up of various industrial papers. In information retrieval tasks like document rating and query expansion, the performance of conventional word embedding and modified word embedding are compared in the evaluation. The outcomes show that the modified word embedding perform substantially better than the conventional ones, resulting in increased retrieval efficiency. The paper also examines many aspects that may affect the performance of adapted word embedding, such as the size of the adaptation corpus, the selection of pertinent texts for adaptation, and the choice of adaptation procedures, in addition to the experimental evaluation. Overall, the article emphasizes how customized word embedding have the potential to improve information retrieval in commercial situations. It includes empirical proof of the efficiency of word embedding and offers insightful information about the process of modifying them to suit domain-specific requirements. The results of this study can aid in the creation of information retrieval systems that are more precise and effective while managing heterogeneous industrial documents.(Rekabsaz, 2017)

In order to increase retrieval performance, Pablo Castells, Miriam Fernández, and David Vallet alter the conventional vector-space model (VSM) of information retrieval to include ontologies. By incorporating ontological knowledge, the authors tackle the problem of efficiently extracting information from enormous document collections. They suggest a technique to improve the accuracy and recall of information retrieval systems by combining the VSM with ontologies. The method for incorporating ontologies into the VSM framework is described by the authors. They add ontological ideas and connections to the term-document matrix, enabling retrieval of more semantically rich information. The authors present a concept extension technique that expands the initial query with related concepts from the ontology in order to address the issue of query-document mismatch. This strategy seeks to close the linguistic gap between the language of the questions and the language of the documents. The ontological structure's significance for information retrieval is emphasized throughout the paper. The authors provide methods to make use of the ontology's hierarchical linkages and semantic interconnections in order to boost retrieval efficiency. Standard assessment metrics like precision, recall, and F-measure are used to assess the proposed VSM adaption. Results from experiments show that the ontology-based approach is more effective than conventional VSM-based retrieval techniques. (Castells, Fernández, & Vallet, 2002)

Semantic search for heterogeneous documents has been studied extensively but at the enterprise level, it is still surfing the surface. But certain gaps still remain unexplored requiring further investigation.

1. Limited Focus on Real-World Enterprise Scenarios: While many studies have focused on examining semantic search in academic or controlled environments, inadequate attention has been paid to investigating its application within an enterprise context where complex challenges exist necessitating exploration of unique requirements, constraints as well as characteristics peculiar to such settings.

2. Insufficient Evaluation Methodologies: Studies exploring innovative techniques or algorithms for semantic search often fail to employ evaluation methodologies with adequate scope resulting in incomplete assessments which fail to uncover key aspects like user satisfaction levels as well as business impacts generated within an enterprise setting. Furthermore, comprehensive evaluation frameworks need to be developed which encompass a range of metrics beyond traditional info-retrieval ones.

3. **Integration of Heterogeneous Data Sources:** Research on semantic search to date focuses on specific data types or limited combinations thereof thus ignoring the critical challenges of effectively integrating and searching across diverse heterogeneous data sources typically found in an enterprise setting.

Moreover, In enterprise settings where massive datasets are often dealt with through distributed architectures while meeting real-time demands – adequate attention is not given to scalability or performance issues pertaining to semantics search algorithms or systems despite their significance. Future research must address these challenges for efficient retrieval and processing of diverse documents irrespective of their volume. The focus must be shifted towards extracting and representing semantics knowledge effectively from diverse documents instead of just studying algorithms in-depth through multiple research papers on Semantic Search – New techniques must be developed so that the contextual information along with semantic relationships prevalent within different sources can contribute significantly towards enhancing the overall performance Semantic Search offers specifically for enterprises.

We can acknowledge the existing contributions of the literature while highlighting the identified gaps. We can emphasize the importance of further investigation should prioritize uncovering novel insights needed for creating a robust yet secure solution tailored precisely towards meeting present-day challenges within enterprise-level document retrieval in complex, heterogeneous environments.

4 Approach 1: Semantic Search Through Structured Datasets Using Vector Embedding

4.1 Purpose and Objectives

The purpose of these 2 case studies (section 4.4 and section 4.6) is to present a methodology for conducting a semantic search on a diverse collection of documents using cosine similarity (section 1.4.3.2) and entity extraction (section 1.4.1.1). Within the context of enterprise information retrieval, entity extraction plays a critical role, contributing to improved search accuracy, query expansion, faceted search, relationship understanding, and knowledge graph enrichment. These enhancements ultimately enhance the user's search experience and enable more effective retrieval of relevant information from the dataset and CSV/excel documents which are structured data. By employing cosine similarity, a semantic search is conducted on a given query, returning the most pertinent documents along with their associated context. The primary aim is to facilitate faster and more accurate information retrieval for users. Furthermore, the research objective is to investigate how semantic search can augment sentence retrieval accuracy in comparison to traditional keyword-based search and to explore the potential applications of these technologies across various industries. Additionally, these case studies delve into the utilization of Vector Space or embedding techniques in the information retrieval model, which could have a significant impact and establish connections with enterprise-level closed industry data. Thus, the key focuses of these case study are as follows:

- Help searching in structured datasets and documents
- Using entity extraction and Text vectorization
- Raking with Accuracy and relevance
- Get relevant information faster
- Results with context

4.2 Experiment Questions

- How can we effectively utilize semantic search techniques, such as cosine similarity and entity extraction, to search through a collection of structured documents?

- How can a semantic search model be employed to encode and index text data, enabling efficient retrieval of similar information?
- How accurate and speedy is document retrieval based on semantic similarity when employing vector embedding and vector databases?
- What are the potential applications and implications of semantic search with vector embedding in various industries, particularly in enterprise-level closed industry data retrieval?

4.3 Hypothesis

Based on the utilization of vector embedding and a vector database in semantic search, we hypothesize that this methodology will enable precise and efficient retrieval of pertinent documents by leveraging their semantic similarity to a given query. By employing vector representations to capture the underlying meaning and relationships within the documents, we anticipate that this approach will outperform traditional keyword-based search methods in terms of accuracy, relevance, and retrieval efficiency. Furthermore, we expect that incorporating vector embedding into the search process will enhance the ranking of informative words and phrases, while minimizing the impact of common and less informative terms. Ultimately, this hypothesis suggests that the proposed methodology will offer an effective and systematic solution for conducting semantic search on a diverse range of documents.

4.4 Case Study 1: Quora Datasets

This case has experimented with a Python script that showcases how to perform a semantic search on a dataset of Quora question pairs. The script uses HuggingFace's datasets and transformers libraries to preprocess the data, and sentence transformers to encode each question pair into a vector. The vectors are then uploaded to Pinecone, a managed vector database that allows for efficient similarity searches. The script also demonstrates how to perform various types of queries on the uploaded data.

4.4.1 Data Source

The data source used in this case is the Quora question duplicates dataset, which contains pairs of questions that are not syntactically the same but share the same meaning. The full dataset contains more than 404K pairs of questions, with about 20% of them labeled as duplicates. In this case,

we load a subset of 10,000 pairs from the training split of the dataset using HuggingFace's datasets library.
The dataset can be accessed through HuggingFace's datasets library using the following code:

```
from datasets import load_dataset
dataset = load_dataset('quora')
```

Once loaded, we can access the questions and their corresponding labels using the keys. Each sample in the dataset is a dictionary containing the following fields:

- **id** - a unique identifier for the sample
- **questions** - a dictionary containing two keys, text, and id, each corresponding to a question in the pair
- **is_duplicate** - a binary value indicating whether the pair of questions are duplicates or not.

4.4.2 Approach

The section of the system that often operates in a low-latency arrangement is the search pipeline. Its objective is to find pertinent results for a particular query. It consists of a method for extracting pertinent information from a query, an encoder that turns that information into embedding, a search engine that makes use of indices created during the encoding process, and lastly a post-filtering system that will choose the best outcomes. The system's initial step is to take a query as input and extract the pertinent information so that it can be encoded as query embedding. When embedding are discovered, we may compare them using, for instance, cosine similarity to determine which is nearest in the vector space. See (section 4.5) for the experiment.
The methodology for this case study involves several steps, starting with data collection from the Hugging Face Quota dataset. Preprocessing of the text data was done using the Tokenizer from Hugging Face to break it down into sentences and tokenize the words in each sentence. The Sentence Transformers library was then used to convert the text data into vector embedding. Next, a vector database was set up using Pinecone to store the embedding. The user's search query is then compared to the embedding in the Pinecone database using cosine similarity. The closest embedding to the query are retrieved and the corresponding sentences are returned as the search results. To evaluate the effectiveness of this approach, precision and recall metrics were calculated. Precision measures the proportion of retrieved sentences that are relevant to the query, while recall measures

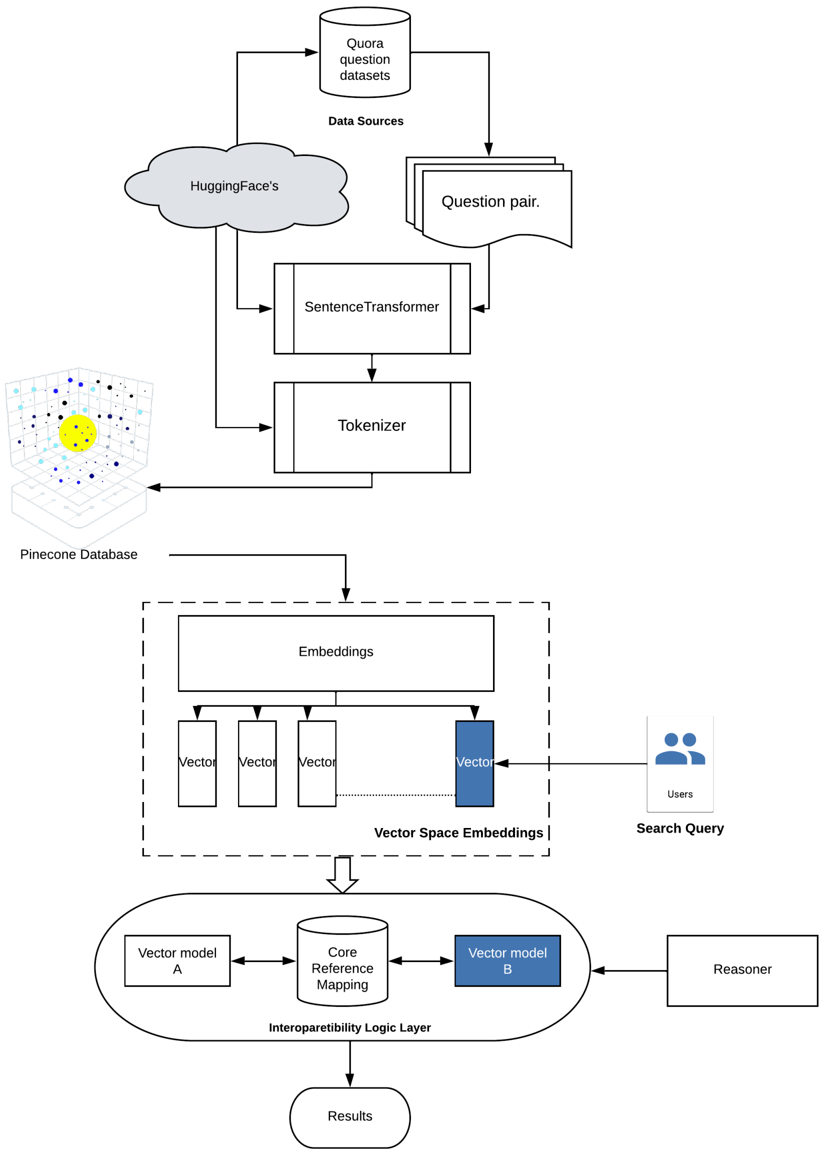


Figure 16: Methodology of the Case Study 1

the proportion of relevant sentences that were retrieved. These metrics were used to determine the accuracy and efficiency of the semantic search approach.

4.5 Experiment 1: Quora Datasets

4.5.1 Experiment Design

A cross-encoder architecture has been planned to be used for the model assessment; this design takes as input a pair of responses, one of which is the correct answer and the other is the system's prediction. The model makes use of a pre-trained semantic text similarity (STS) model to determine how similar the two texts are to one another. The model also picks up on which terms in a sentence are most important to understanding it, doing away with the requirement for preprocessing. When used, the model metric produces a score that ranges from zero (for two solutions that have completely different semantics) to one (for two answers with the same meaning). (Wrigley et al., 2013)

4.5.2 Experiment Implementation

- **Step 1 - Load dataset:** Then, it loads the Quora question pairs dataset using HuggingFace's datasets library and prints out a sample question pair. The dataset contains more than 404k pairs, so encoding all of them at once in-memory is not efficient, so the script upserts them in batches to Pinecone.

- **Step 2 - Initialize SentenceTransformer model:** The script then initializes an instance of the SentenceTransformer class and downloads a pre-trained model from the HuggingFace model hub. For the case study, a Pre-trained Encoder Transformer model (BERT) was used.

- **Step 3 - Initialize tokenizer:** Initialize a tokenizer from HuggingFace transformers and break the text into words.

- **Step 4 - Connect to Pinecone:** To upsert to Pinecone an index was created to upsert to via the Pinecone Python client with initialized connection to Pinecone API key.

- **Step 5 - Pooling operation:** The script then processes the data with some pooling operation in batches, creates vectors and metadata for each question pair, and upserts them to Pinecone. The metadata contains information about each question pair, such as whether or not they are duplicates and their character length. Each sample as a tuple (id, vectors, metadata), which each contain:

  - **id** - a str ID

  - **vectors** - the sentence vector (in list format)
  - **metadata** - a dictionary in the format

- **Step 6 - creating 'query vector:** Once index and data ready, First a 'query vector' is created. This is a sentence (or in this case question) encoded using the same model that will be encoded the quora dataset with.

- **Step 8 - Map the IDs and Retrival:** The script demonstrates several types of queries that can be performed on the data. First, it creates a query vector for a sample question and uses Pinecone's query method to return the five most similar questions. It then maps the returned IDs to their original text, prints out the similarity score, and shows the five most similar questions.

4.5.3 Reasoner Layer

The vectors in the vector space are normally grouped by the topic they are representing. From this vector, we can check similarities for example cosine similarity to find closest in the vector space. By comparing data and identifying those that are most similar to one another, Vector DB indexes vectors for quick search and retrieval. See (section 4.8) for the results.

4.6 Case Study 2: Excel/CSV Documents

In this case study we present a methodology to present for performing semantic searches in Excel data by employing natural language processing (NLP). Our approach entails identifying key entities within the data through named entity recognition (NER), creating an index using these entities, and leveraging it during semantic searches triggered by user queries. We executed our proposed methodology using the powerful Spacy NLP library and validated its effectiveness against a NORSK dataset library and tested it on a NORSK dataset.

4.6.1 Data Source and Structure

Our dataset consists of an Excel file featuring failure logs obtained from a Norwegian corporation. It adopts a structured tabular format where each row denotes unique instances of events or activities related to the failure records of this company. These rows are further subdivided into columns providing detailed information regarding distinct facets of each incident e.g., error details, occurrence date(s), associated repair order(s), specific work carried out on them etcetera; the number of rows and columns we possess are significantly large in volume while some unlabeled components constitute it too. Consequently, parsing through all these dimensions warrants extensive exploration.

- The data has 754116 rows x 22 column
- The main column which we will search in is Feilbeskrivelse
- We will extract the Feilbeskrivelse entity from the excel file
- Data will be cleaned based on the needs, Some columns we might need to drop to improve the performance of the model.
- We need to remove spaces and punctuation for better performance

4.6.2 Approach

To implement this approach, the Excel/CSV document is first read into memory and the relevant columns are extracted. The data is then preprocessed using Spacy NER, removing stop words, and lemmatizing the text. The resulting preprocessed data is then transformed using TfidfVectorizer to obtain the vectors. The vectors are then indexed in a vector database for efficient search. See (section 4.7) for the experiment.

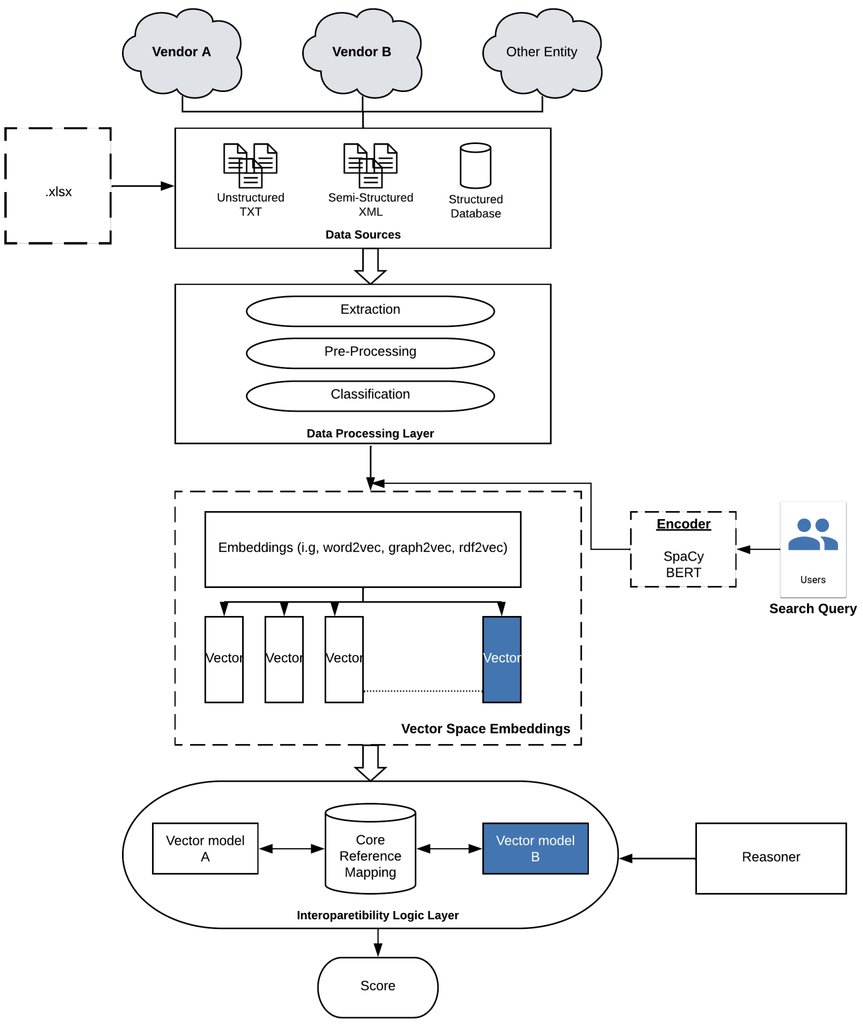


Figure 17: Methodology of the Case Study 2

When a user submits a search query, the query is first preprocessed using Spacy NER, removing stop words, and lemmatizing the text. The preprocessed query is then transformed using TfidfVectorizer to create a vector representation of the query. The cosine similarity function is then used to perform a semantic search on the indexed vectors in the vector database, returning the most relevant documents with their context.

4.7 Experiment 2: Excel/CSV Documents

4.7.1 Experiment Design

The increasing amount of data stored in Excel spreadsheets has led to the need for more sophisticated search and retrieval methods. Semantic search, which involves analyzing the meaning of text to retrieve relevant information, has emerged as a promising approach for searching large datasets. In this paper, we present a methodology for performing semantic search in Excel data based on entity extraction using NLP techniques.

The methodology consists of three main steps: entity extraction, index creation, and semantic search.

4.7.2 Experiment Implementation

4.7.2.1 Preprocessing

Preprocessing text data is a crucial step in natural language processing (NLP), which involves cleaning and transforming raw text data into a format that can be easily analyzed and understood by machines. In this step, the dataset is preprocessed by removing any irrelevant information and cleaning the text. This can involve removing stop words, and punctuation, and converting all text to lowercase. The goal of preprocessing is to make the text easier to analyze and extract entities from. Preprocessing can be done using various techniques, depending on the specific requirements of the dataset.

```
def preprocess(text):
    doc = nlp(text)
    tokens = [token.lemma_ for token in doc if not token.is_stop]
    return " ".join(tokens)
```

4.7.2.2 Entity Extraction

Entity extraction is the process of identifying and extracting important named entities (people, places, organizations, etc.) from text data. The "extract_entities" function defined in the code takes a text input and creates a Spacy "doc" object by running the text through the NLP pipeline. The NLP pipeline consists of a series of components, including a tokenizer, part-of-speech tagger, and named entity recognizer, which work together to parse the text and identify its underlying linguistic structure. Spacy library is used to perform entity extraction on the text data. Once the doc object has been created, the function iterates through each named entity identified in the text by the named entity recognizer component. For each named entity, the function extracts its text representation (i.e., the string of characters corresponding to the entity) and adds it to a list of entities.

```
def extract_entities(text):
    doc = nlp(text)
    entities = []
    for ent in doc.ents:
        entities.append(ent.text)
    return entities
```

4.7.2.3 Text vectorization

Text vectorization is the process of converting text data into a numerical representation that can be used by machine learning models.

- In the case, the TfidfVectorizer class from the sci-kit-learn library is used to perform text vectorization.
- The "tfidf" object created using the TfidfVectorizer class learns the vocabulary of the text data and creates a document-term matrix where each row represents a document and each column represents a unique word in the vocabulary. The values in the matrix represent the importance of each word in the corresponding document, typically calculated using the term frequency-inverse document frequency (TF-IDF) weighting scheme.
- To vectorize the text data, the "fit_transform" method of the "tfidf" object is called on the preprocessed text data. This method fits the TfidfVectorizer object to the text data and returns a sparse matrix representation of the data in numerical form.
- The resulting sparse matrix can be used to perform semantic search using cosine similarity, as shown in the code. The "semantic_search" function takes a query as input and transforms it using the fitted TfidfVectorizer object.

4.7.2.4 Semantic Search

In this step, the index is used to perform a semantic search. This involves querying the index to find the cells or rows that contain the relevant information and ranking the results based on relevance.

The function takes a query and a preprocessed data matrix (created using text vectorization) as input and returns a list of document indices sorted by their cosine similarity to the query.

- To perform the semantic search, the query is first preprocessed using the same preprocessing function used to preprocess the data matrix. The TfidfVectorizer object, which was previously fitted to the preprocessed data matrix, is then used to transform the preprocessed query into a vector representation.
- The cosine similarity between the query vector and each document vector in the preprocessed data matrix is then calculated using the cosine_similarity function from scikit-learn. The resulting cosine similarities are sorted in descending order, and the corresponding document indices are returned as the search results.
- The semantic search function is a useful tool for searching through large text datasets and retrieving documents that are semantically similar to a user query. By using text vectorization and cosine similarity, the function can capture the semantic similarity between documents and queries, even if the documents and queries use different words or phrasing to express the same concepts.

```
def semantic_search(query, data):
    query_vector = tfidf.transform([preprocess(query)])
    cosine_similarities = cosine_similarity(query_vector, data).flatten()
    related_docs_indices = cosine_similarities.argsort()[::-1]
    return [(index, cosine_similarities[index]) for index in related_docs_indices]
```

4.7.3 Reasoner Layer

The vectors in the vector space are normally grouped by the topic they are representing. From this vector, we can check similarities for example cosine similarity to find closest in the vector space. By comparing data and identifying those that are most similar to one another, Vector DB indexes vectors for quick search and retrieval. Cosine similarity is then calculated between the query vector and the document-term matrix, and the resulting cosine similarities are used to rank the documents in order of relevance to the query. See (section 4.8) for the results.

```

query = "Fortell meg om batterika?"
results = semantic_search(query, tfidf_data)
```

```
print("Top results for query: {}".format(query))
for result in results[:5]:
    print("Document index: {}, cosine similarity: {}".format(result[0], result[1]))
    print(df.iloc[result[0]]["Feilbeskrivelse"])
    print()
```

4.8 Results

Semantic search plays a vital role in enabling efficient retrieval of relevant information based on the meaning and context of user queries. In these case studies, we explore the value of semantic search in enabling the effective retrieval of pertinent information based on the meaning and context of user queries. We investigate the efficacy of two potent resources: Pinecone, a vector database for semantic search, and Hugging Face's Quota dataset. Our main goal is to develop a model that can quickly and precisely retrieve items that are semantically comparable to a given query.
In the second case study, we explain how to use entity extraction, cosine similarity, and semantic search to efficiently search through a collection of CSV and Excel documents. We strengthen the link between the user's search query and the search results by utilizing text vectorization algorithms, which increases accuracy and relevancy. See (section 6) for the discussion.

4.8.1 Experiment Summary

In Experiment 1, We leveraged Hugging Face's Quota dataset to accomplish our objective, which provides a diverse range of text documents for training and evaluation purposes. We utilized this dataset to train a semantic search model that could understand the context and meaning of natural language queries. Pinecone, a high-performance vector database, was chosen as the storage solution for embedding derived from the trained model.

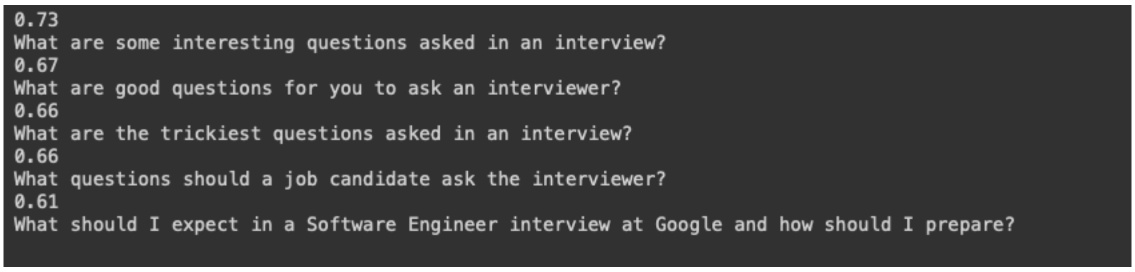
Experiment 2 for performing a semantic search in Excel data based on entity extraction utilizes natural language processing (NLP) techniques to enhance the search capabilities. The process begins by extracting relevant columns focusing on the "Feilbeskrivelse" in this case from the Excel/CSV document, with a focus on the description of errors or relevant textual information. The data is then preprocessed to identify and extract important entities and transform the preprocessed text into vectors. These vectors are indexed in a vector database for efficient search operations. When a user submits a search query, it undergoes the same preprocessing steps as the document data. The preprocessed query is transformed into a query vector using the TfidfVectorizer, and cosine similarity is calculated between the query vector and the indexed document vectors. The documents with the highest cosine similarity scores are retrieved as the most relevant search results, providing users with accurate and context-rich information from the Excel dataset.

4.8.2 Example Query Search

From the given experiment 1, the model worked as it was supposed to. The Script returns with a Score for the result where 0<score<1. With a

query = "What questions are asked in Google Interviews?"

the script returns with the following results -



```
0.73
What are some interesting questions asked in an interview?
0.67
What are good questions for you to ask an interviewer?
0.66
What are the trickiest questions asked in an interview?
0.66
What questions should a job candidate ask the interviewer?
0.61
What should I expect in a Software Engineer interview at Google and how should I prepare?
```

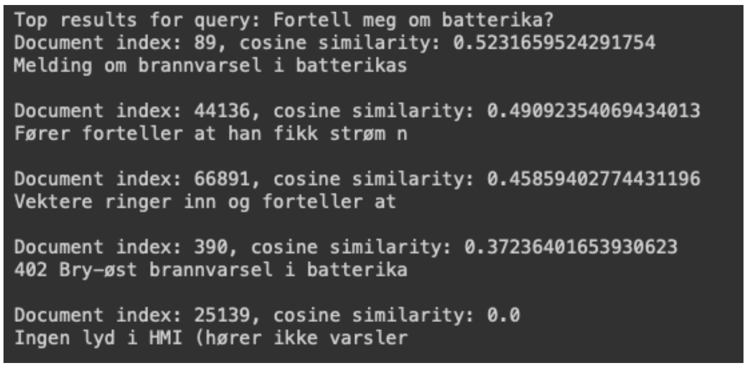
Figure 18: Results of the Case Study 1

In experiment 1, Since the dataset contains pairs of questions that are not syntactically the same but share the same meaning, with a given user Search Query, the script returned filtered and contextual results based on the trained dataset. As you can see in figure 18, the retrieved results are related to the word "Interviews". This model goes through the datasets and finds the closest match for the interview on the pairs of questions. Although the questions are different they all represent the same context as the query question which is related to interviews.

With experiment 2, from the given case, the model worked as it was supposed to. The Script returns with a Score for the result where 0<score<1. With a

query = "Fortell meg om batterika?"

the script returns with the following results -



```
Top results for query: Fortell meg om batterika?
Document index: 89, cosine similarity: 0.5231659524291754
Melding om brannvarsel i batterikas

Document index: 44136, cosine similarity: 0.49092354069434013
Fører forteller at han fikk strøm n

Document index: 66891, cosine similarity: 0.45859402774431196
Vektere ringer inn og forteller at

Document index: 390, cosine similarity: 0.37236401653930623
402 Bry-øst brannvarsel i batterika

Document index: 25139, cosine similarity: 0.0
Ingen lyd i HMI (hører ikke varsler
```

Figure 19: Results of the Case Study 2

By considering information related to the raw documents from different formats, this case study allows users to search for their desired information in a convenient and efficient manner. The extraction of entities using Spacy NER also adds value by providing context to the search results. In our case, a different language (Norwegian) or a custom model might be used for both English and Norsk. As you can see in figure 19, we tried to retrieve information from the CSV file from the Feilbeskrivelse column. Once, the user enters the search query, the model retrieves the document's indexes and finds the closest match for the entity which in this case is "batterika". The model presents the result according to the relevance by the closest or most relevant information being at the top. Although the information is different they all represent the same context as the query question which is related to batterika.

4.8.3 Accuracy and Relevance

The effectiveness of accuracy and relevance in the semantic search was the main focus of the evaluation of method 1 and experiment 2. Both investigations sought to quickly and accurately retrieve pertinent documents that closely matched the semantic intent of the query, with promising results in terms of accuracy. By accuracy, we mean the extent to which the search results are relevant and precisely match the intended meaning or user's query. It measures the correctness and precision of the retrieved information in relation to the user's information needs. In experiment 1, the use of embedding stored in Pinecone enabled quick and effective search, resulting in noticeably shorter retrieval times than with conventional techniques. Experiment 2 on the other hand made sure that the context of the query was fully understood by taking into account the semantic links between things. The comparison and representation of texts in a numerical vector space were made easier by the use of vectorization techniques like TF-IDF and cosine similarity. This allowed for quick similarity computations and the retrieval of data that were semantically related. As we can see the results did not retrieve any information which is not semantically connected representing the accuracy of the experiments. In summary, both case studies showed that accuracy and relevance were highly valued in semantic search, producing positive results. The approaches used cutting-edge methods to find materials that closely matched the semantic purpose of the searches while also improving the search experience with effective and focused results.

4.8.4 Comparison with Our Vector Embedding Approach with Other Relevant Approaches

Several experiments and studies have been explored by researchers on semantic search techniques and methodologies. A comparison of our case study utilizing Hugging Face's Quota dataset and Pinecone with these related experiments sheds light on the unique contributions and strengths of our approach. Compared to traditional keyword-based search methods, our semantic search approach offers a significant advantage. While keyword-based search relies on exact matches or pattern matching, our approach leverages the semantic understanding of queries and documents. This enables retrieval of relevant information even when the exact terms used in the query are not present in the document. In our case study, we utilized Pinecone as a vector database to store embedding and facilitate efficient search. This approach differs from traditional databases, such as relational databases, that rely on structured queries. Vector databases offer advantages in terms of fast indexing and retrieval based on similarity metrics. Comparatively, our integration of Pinecone provides a scalable and efficient solution for storing and retrieving document embedding, leading to improved search performance.

5 Approach 2: Semantic Search Through Heterogeneous Structured, Semi-Structured and Un-Structured Documents using RDF Ontology

5.1 Case Study 3: Heterogeneous Documents using RDF Ontology

The purpose of this case study is to demonstrate a method for performing a semantic search on a variety of document types. By leveraging semantic technologies and sophisticated search algorithms, we expect to provide users with highly relevant and accurate results, even when dealing with complex data structures, and provide a robust, scalable, and efficient solution for searching through large amounts of data in a variety of formats. We used Protege for modeling and RML mapping rules to transform data into RDF format. We then stored the transformed data in a graph database (GraphDB) and used SPARQL to perform a semantic search on various document types, including docx, pdf, XML, JSON, excel, and CSV files. We created a front-end using React.js and a backend using Node.js to enable users to input search queries and view the results.

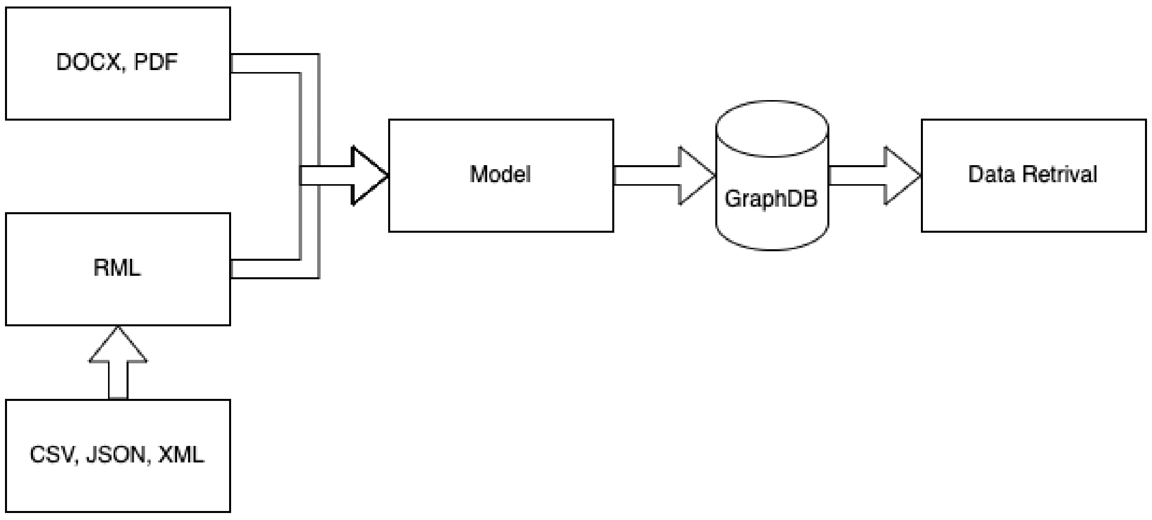


Figure 20: Overview for Methodology of the Case Study 3

5.1.1 Aims and Objective

The aim of this case study is to demonstrate an effective method for performing a semantic search on a diverse set of data formats, including Docx, pdf, XML, JSON, Excel, and CSV files. The approach involves using Protege for modeling and RML mapping rules to transform the data into RDF format, which is then loaded into a graph database (GraphDB). SPARQL is used to perform the semantic search and querying, allowing for sophisticated analysis of relationships between entities.

The objective of this case study is to provide a solution for efficient and accurate semantic search on complex data structures with diverse data formats. This approach allows for flexible querying and analysis of relationships between entities, making it a valuable tool for tasks such as data integration, knowledge discovery, and information retrieval. The use of a web-based interface with React.js and Node.js allows for easy access and usability, making it accessible to a wide range of users with varying technical expertise.

5.1.2 Experiment Questions

How can we search through a collection of heterogeneous documents using RDF Ontology?

5.1.3 Hypothesis

By using Protege for modeling, RML mapping rules for data transformation into RDF format, and GraphDB with SPARQL for semantic search, coupled with a React.js frontend and Node.js backend, we can efficiently and accurately search and analyze complex data structures across various document formats. This approach will lead to faster and more relevant results with increased precision and recall, ultimately improving data management and analysis in various domains.

5.1.4 Data Source and Structure

5.1.4.1 Data Source

The Electric Vehicle Population Data is a dataset that contains information on Battery Electric Vehicles (BEVs) and Plug-in Hybrid Electric Vehicles (PHEVs) that are registered through the Washington State Department of Licensing (DOL). This dataset is available on data.gov, but it is important to note that it is a non-federal dataset and is covered by different terms of use than other datasets on the website. The data is provided in multiple formats including XML, JSON, and Excel files. This allows users to choose the most appropriate format for their needs. The dataset is intended for public access and use, and can be accessed through the Resources section on data.gov. The Electric Vehicle Population Data provides valuable information for researchers, policy makers, and other stakeholders interested in understanding the adoption and usage of electric vehicles in Washington State. The availability of

multiple data formats ensures that the data can be easily integrated into various data analysis and visualization tools.
For unstructured data-sets like Docx and PDF, a simple Electric vehicle specifications file was used.

5.1.4.2 Dataset Structure

The dataset on Electric Vehicle Population Data contains 16 columns of information for each vehicle.

- The first column is the VIN, or Vehicle Identification Number, which uniquely identifies each vehicle.
- The next three columns represent the location of the vehicle, with the county, city, and state of registration listed. The postal code column provides further granularity in identifying the location of the vehicle.
- The model year, make, and model of the vehicle are listed in the next three columns, providing details about the vehicle's make and model.
- The Electric Vehicle Type column provides information on whether the vehicle is a Battery Electric Vehicle (BEV) or Plug-in Hybrid Electric Vehicle (PHEV).
- The Clean Alternative Fuel Vehicle (CAFV) Eligibility column indicates whether the vehicle meets the eligibility criteria for California's Clean Alternative Fuel Vehicle and Advanced Low-Emission Vehicle Technology Program.
- The Electric Range column provides information on the maximum range of the vehicle on a single charge.
- The Base MSRP column lists the manufacturer's suggested retail price for the vehicle. The Legislative District column identifies the legislative district in which the vehicle is registered.
- The DOL Vehicle ID column contains a unique identifier for the vehicle assigned by the Washington State Department of Licensing (DOL).
- The Vehicle Location column provides information on where the vehicle is currently located.
- The Electric Utility column identifies the electric utility company that supplies power to the vehicle.

- Finally, the 2020 Census Tract column provides information on the census tract in which the vehicle is registered.
- Horse Power provides information regarding engine power it can produce.
- fuelCapacity provides information regarding its capacity for fuel consumption.

5.1.5 Approach

For this case study, we have divided the experiment into 2 phases where 1 is associated with Modelling and RDF conversion and importing them to GraphDB and in the second phase, we work on Front-end and backend parts. As shown in figure 21, the first step in the approach is to create a data model using Protege, which includes four classes: Address, Vehicles, Technical Specifications, and Company. These classes are defined with data properties and object properties as appropriate for the specific use case. Next, the data is mapped to the model using RML rules in RocketRML, which takes the RML rules as input and generates RDF formatted data. This process ensures that the data is transformed into a format that can be easily queried and analyzed in a graph database. The generated RDF data is then loaded into a graph database, such as GraphDB, and indexed for efficient querying. Once the data is loaded, SPARQL queries can be performed to retrieve relevant information from the dataset. The final step is to develop a user interface to allow users to perform semantic searches using the SPARQL queries. This can be done using a variety of web-based technologies, such as React.js and Node.js, which provide a flexible and user-friendly interface. See (section 5.2) for the experiment.

5.1.5.1 Pscudo Code

- Create a data model using Protege that defines the entities, attributes, and relationships within the data.
- Define RML mapping rules that specify how to transform the data from its original format (XML, JSON, excel, or CSV) into RDF format.
- Create a model for Docx, pdf using RDF format.
- Use an RML processor (such as the RocketRML) to apply the mapping rules and convert the data into RDF format.
- Load the RDF data into a graph database (such as GraphDB).

- Write SPARQL queries to perform semantic search and analysis on the data
- Use a web-based interface (built with React.js and Node.js) to allow users to interact with the data and perform searches and analysis.

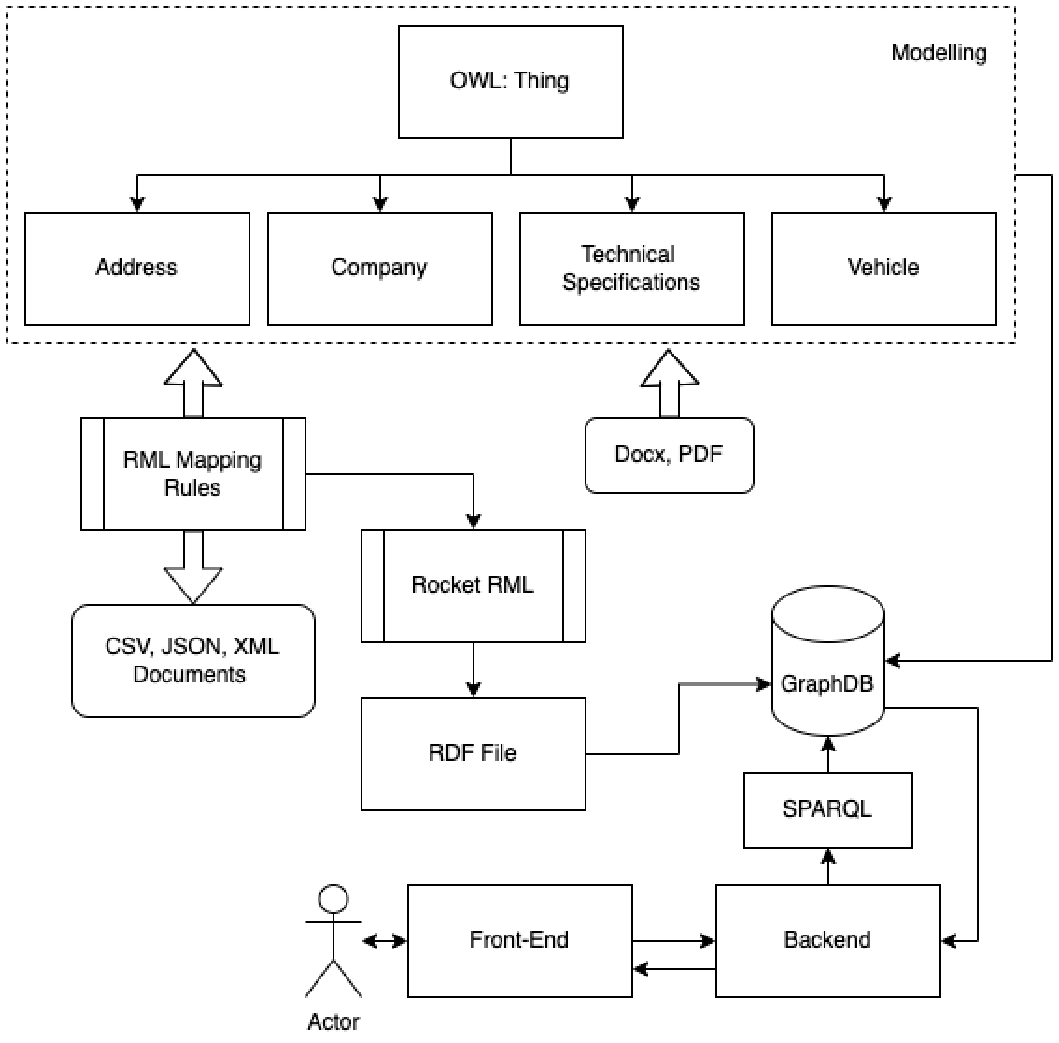


Figure 21: Methodology of the Case Study 3

5.2 Experiment

5.2.1 Protege Modeling

In Protege, modeling refers to the process of creating a conceptual representation of a domain or knowledge using ontologies. An ontology in Protege is a formal specification that defines the concepts, relationships, and properties within our case. Modeling in Protege involves defining classes, properties, individuals, and their relationships to capture the semantics of a case study. This process allows to create a structured representation of knowledge that can be used for various purposes, including semantic search using RDF. By modeling a domain in Protege, we can define the vocabulary and semantics that enable machines to understand and reason about the information within the knowledge base. It provides a way to represent and organize information in a meaningful and structured manner. When it comes to semantic search using RDF, modeling in Protege plays a crucial role. According to our data, we have vehicle information in stored in different files which contain Address information, Company, technical specifications and vehicle information.

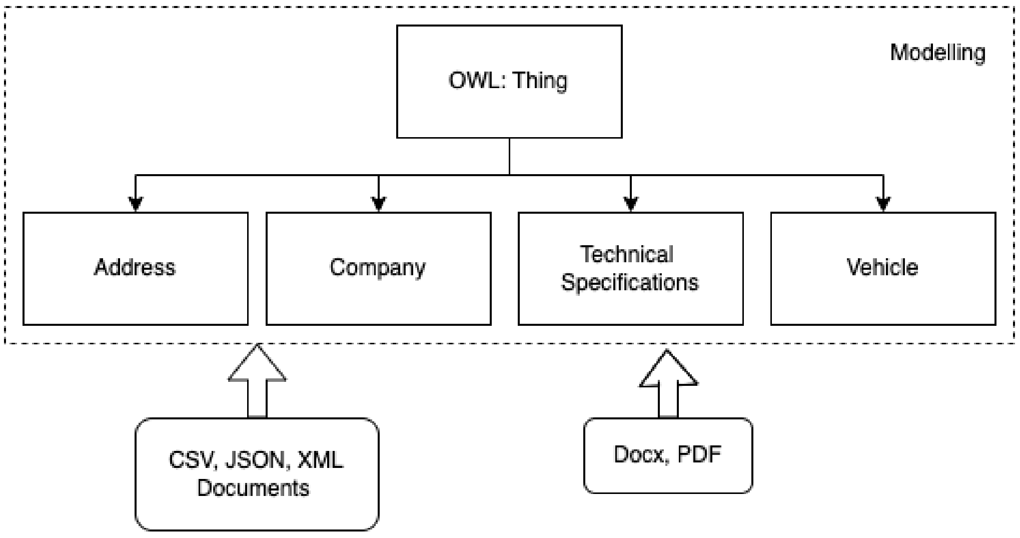


Figure 22: Protege Modeling for case study 3

Conceptual Clarity: Modeling in Protege allows us to define classes and hierarchies that represent the entities or concepts within our case study. This conceptual clarity facilitates the formulation of precise search queries by enabling us to specify the types of entities or relationships we are interested in. (section 5.1.1.1)

Property Definitions: Protege allows us to define properties, both object properties and data properties, which capture relationships and attributes of entities. These properties provide the basis for constructing search queries that retrieve relevant information based on specific criteria or constraints. (section 5.1.1.2 and section 5.1.1.3)

Semantic Relationships: Modeling in Protege enables the establishment of semantic relationships between entities. These relationships, defined using object properties, allow us to express meaningful connections and associations between different elements of your domain. Semantic search can leverage these relationships to infer additional information and provide more contextually relevant search results. (section 5.1.1.4)

Knowledge Inference: Protege supports reasoning capabilities based on the defined ontologies. Through reasoning, the system can infer new information and relationships that are not explicitly stated in the knowledge base. This inference can enhance the search process by expanding the scope of relevant results and facilitating more comprehensive retrieval.

5.2.1.1 Class Create

In Protege, class creation refers to the process of defining and creating classes within an ontology. Classes represent categories or types of entities within a domain, allowing us to organize and classify knowledge based on common characteristics and relationships. In our case, we have created 4 classes in protege which are Address, Company, Technical Specifications, and Vehicle. These classes help us create a more meaningful and structured representation of the case, fostering better understanding, management, and utilization of the knowledge base. This will create an individual unique IRI (section 2.3.10) for each class.

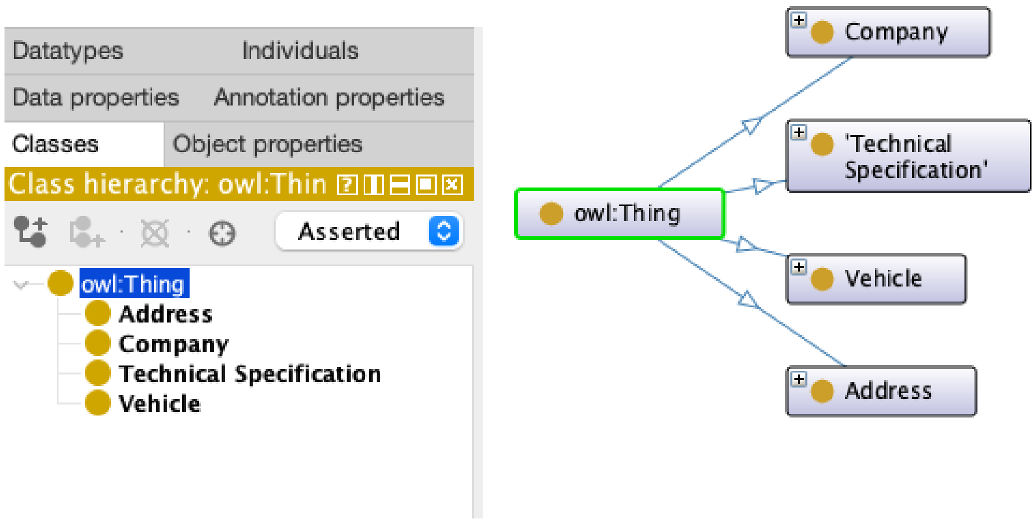


Figure 23: Protege Modeling - Relationships between Classes

5.2.1.2 Data Property

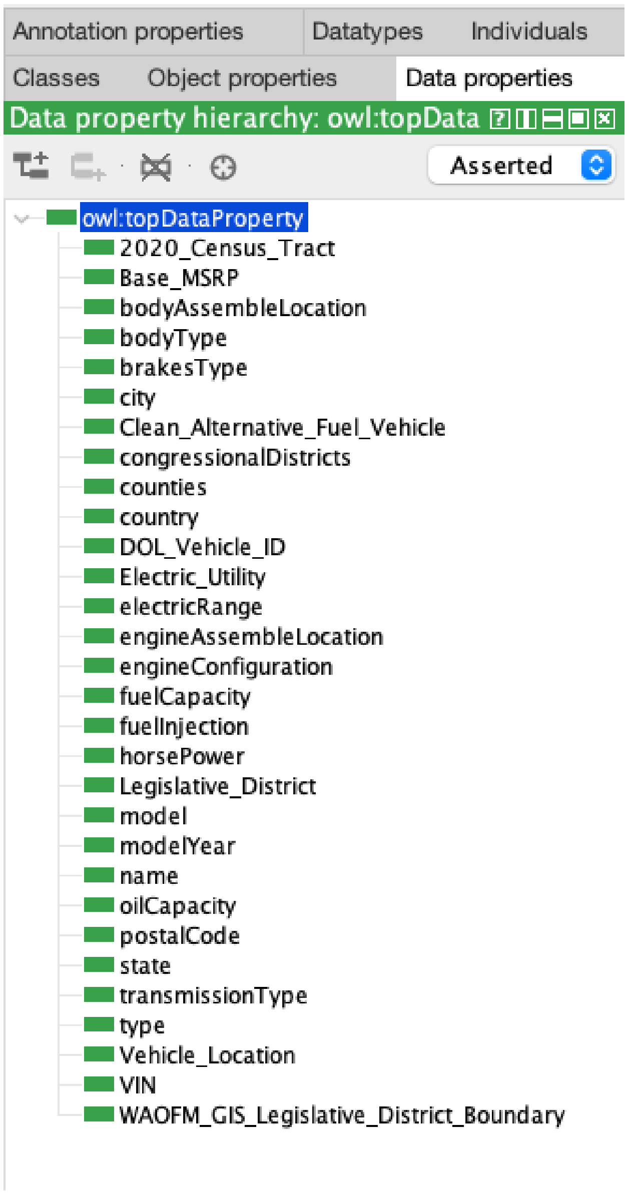
In the context of ontologies and knowledge representation, a data property is used to define attributes or properties that describe specific characteristics or data values associated with individuals or instances. Data properties are essential for capturing and representing measurable or textual information about entities within our experiment. Data properties are instrumental in semantic search through heterogeneous structured and unstructured documents using an RDF ontology. They enable structured data integration, attribute-based search, semantic enrichment, the establishment of semantic relationships, faceted search, and ontology-driven search throughout our documents. By leveraging data properties, the search process becomes more precise, context-aware, and effective in retrieving relevant information from diverse document sources which helps our experiment. This will create an individual unique IRI (section 2.3.10) for each Data Property.

5.2.1.3 Object Property

Object properties play a crucial role in semantic search through heterogeneous structured and unstructured documents using an RDF ontology. They establish meaningful relationships between entities and concepts within the ontology, enabling the representation of semantic connections and associations. By linking documents to related entities through object properties, the search process becomes more context-aware and comprehensive. Object properties facilitate entity extraction and linking, allowing for the identification and connection of relevant entities mentioned in the documents. They contribute to the hierarchical structure of ontology, organizing entities into meaningful hierarchies and taxonomies. Object properties enhance querying capabilities by enabling more expressive search queries that traverse relationships between documents and other entities. They also support reasoning and inference, enabling logical deductions and the discovery of implicit relationships. Concept-based search is facilitated through object properties by associating documents with specific concepts or categories, resulting in more precise and relevant search results. In this experiment, we have 3 object properties.

- hasMake
- hasRegister
- hasTechnicalSpecification

In our case, we have the Vehicle class and Address class. Vehicle hasRegister relation Address. Similarly, the Vehicle hasTechnicalSpecification relation with TechnicalSpecification. This creates a rela-



Data property hierarchy: owl:topData

owl:topDataProperty
- 2020_Census_Tract
- Base_MSRP
- bodyAssembleLocation
- bodyType
- brakesType
- city
- Clean_Alternative_Fuel_Vehicle
- congressionalDistricts
- counties
- country
- DOL_Vehicle_ID
- Electric_Utility
- electricRange
- engineAssembleLocation
- engineConfiguration
- fuelCapacity
- fuelInjection
- horsePower
- Legislative_District
- model
- modelYear
- name
- oilCapacity
- postalCode
- state
- transmissionType
- type
- Vehicle_Location
- VIN
- WAOFM_GIS_Legislative_District_Boundary

Figure 24: Protege Modeling - Data Property

tionship between Vehicle and Address as well as Technical Specification. This will create an individual unique IRI (section 2.3.10) for each Object Property.

5.2.1.4 Relationships Between Class Properties

After creating classes, data properties, and object properties, we are defining the relationship between classes. Connecting data properties, and object properties to classes. Through this, we can find out which properties belong to which classes. See Figure 26 to 29.

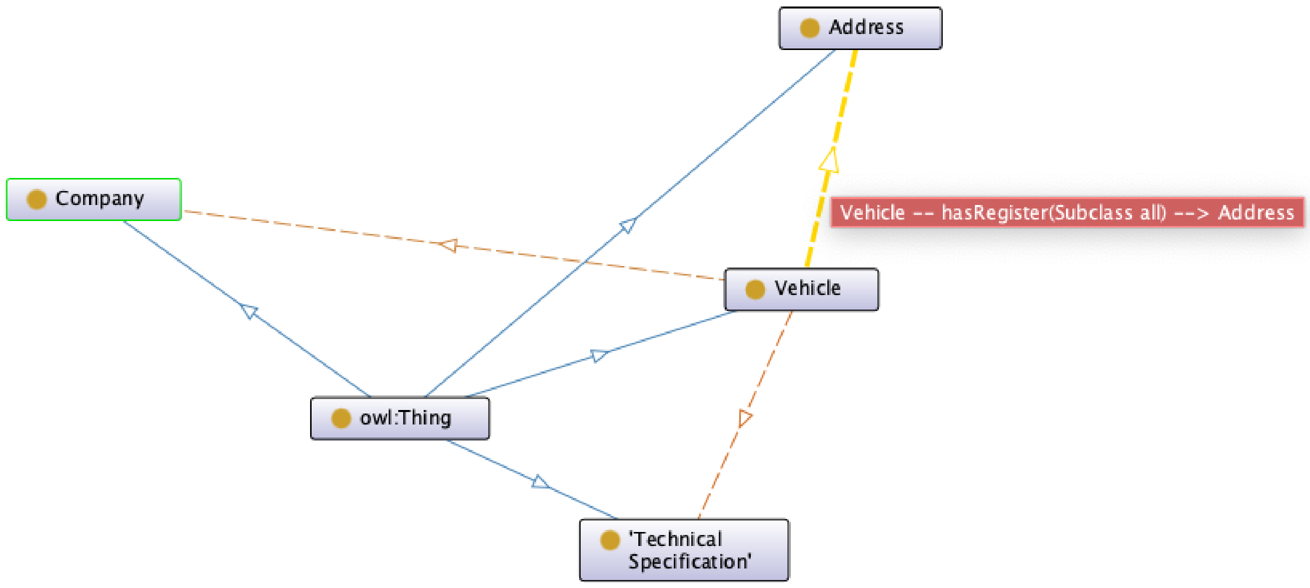
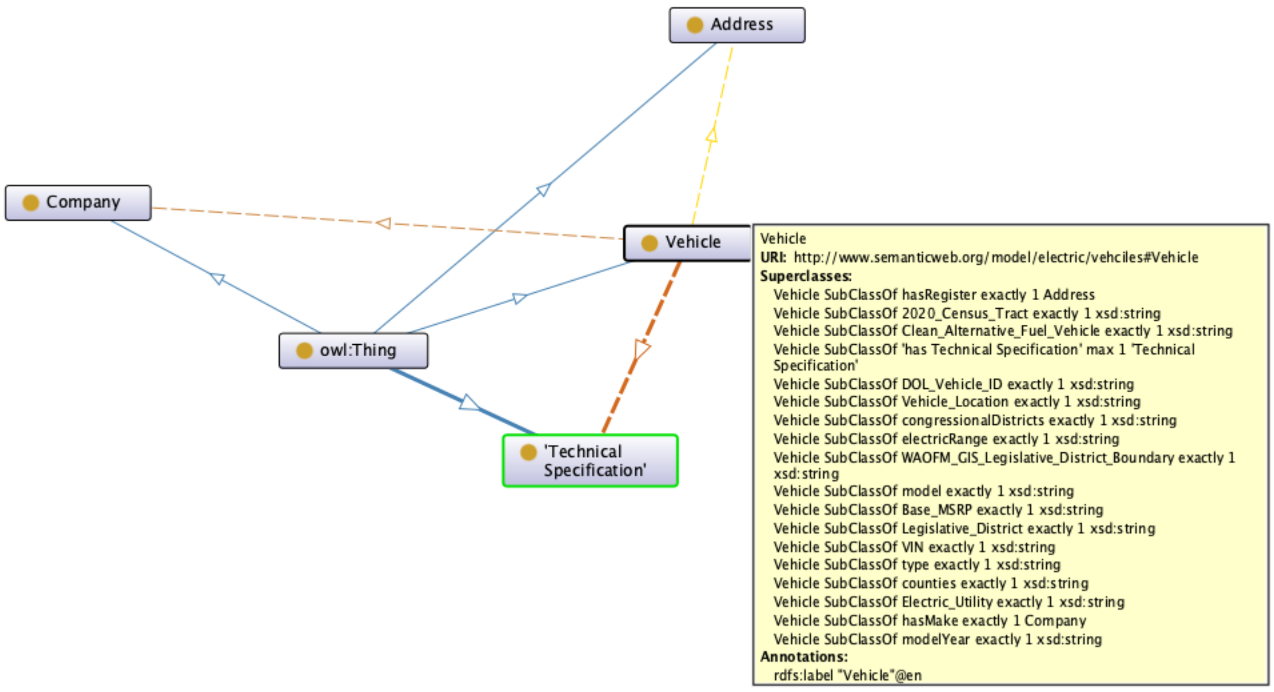

Address
Company
Vehicle -- hasRegister(Subclass all) --> Address
Vehicle
owl:Thing
'Technical Specification'

Figure 25: Protege Modeling - Data Property

5.2.2 RML Mapping Rules

RML (RDF Mapping Language) mapping rules are necessary for semantic search through heterogeneous structured and unstructured documents using an RDF ontology with Protege modeling. RML provides a standardized way to map and transform data from different sources into RDF format, allowing for seamless integration and interoperability (section 2.3.9). By defining RML mapping rules, we can specify how the data from diverse sources, such as JSON, CSV files, EXCEL, or XML, should be transformed and represented as RDF triples. These mapping rules provide instructions on how to extract, transform, and map the relevant data elements from the source documents to the corresponding entities, properties, and relationships defined in the RDF ontology. RML mapping rules facilitate the conversion of heterogeneous documents into a unified RDF representation. This enables the application of semantic search techniques, leveraging the ontology's defined classes, properties, and relationships. Using Protege modeling in conjunction with RML mapping rules further enhances the search capabilities. Protege allows for the creation and management of the RDF ontology, where classes, properties, and relationships are defined. The ontology provides the semantic framework for understanding and searching the transformed RDF data. By combining RML mapping rules, which handle the transformation of heterogeneous documents into RDF, with Protege modeling, which defines the ontology structure, semantic search through diverse documents becomes feasible. The unified RDF representation enables efficient querying, linking, and reasoning based on the defined



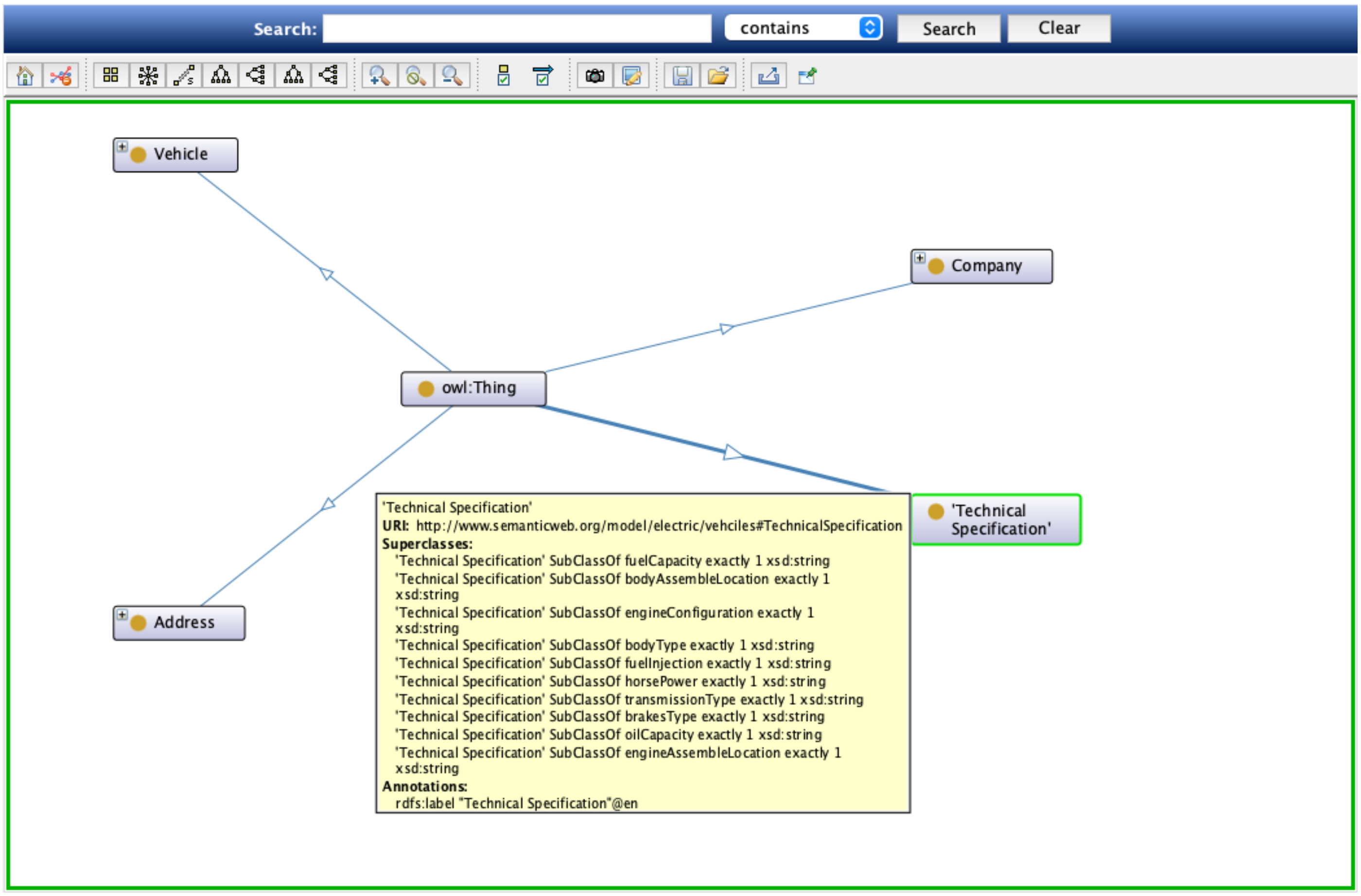
```
Vehicle
URI:  http://www.semanticweb.org/model/electric/vehciles#Vehicle
Superclasses:
  Vehicle SubClassOf hasRegister exactly 1 Address
  Vehicle SubClassOf 2020_Census_Tract exactly 1 xsd:string
  Vehicle SubClassOf Clean_Alternative_Fuel_Vehicle exactly 1 xsd:string
  Vehicle SubClassOf 'has Technical Specification' max 1 'Technical
  Specification'
  Vehicle SubClassOf DOL_Vehicle_ID exactly 1 xsd:string
  Vehicle SubClassOf Vehicle_Location exactly 1 xsd:string
  Vehicle SubClassOf congressionalDistricts exactly 1 xsd:string
  Vehicle SubClassOf electricRange exactly 1 xsd:string
  Vehicle SubClassOf WAOFM_GIS_Legislative_District_Boundary exactly 1
  xsd:string
  Vehicle SubClassOf model exactly 1 xsd:string
  Vehicle SubClassOf Base_MSRP exactly 1 xsd:string
  Vehicle SubClassOf Legislative_District exactly 1 xsd:string
  Vehicle SubClassOf VIN exactly 1 xsd:string
  Vehicle SubClassOf type exactly 1 xsd:string
  Vehicle SubClassOf counties exactly 1 xsd:string
  Vehicle SubClassOf Electric_Utility exactly 1 xsd:string
  Vehicle SubClassOf hasMake exactly 1 Company
  Vehicle SubClassOf modelYear exactly 1 xsd:string
Annotations:
  rdfs:label "Vehicle"@en
```

Figure 26: Class Properties

ontology, resulting in more accurate and comprehensive search results. In this experiment, we used EXCEL, CSV, JSON and XML to RDF conversion using RML Mapping Rules. We have used the RocketRML package to convert the mapped rules to RDF triple.

5.2.2.1 CSV to RDF

The project is a RDF Mapping Language (RML) mapping file written in Turtle syntax. RML is used to define how to map data from non-RDF sources (such as CSV, XML, and JSON files) to RDF. It contains three mappings for a CSV file demo-data/customCSV.csv, which is the source of the data. The first mapping is for the Vehicle class, and it maps the CSV columns to properties of the class. The rr:subjectMap defines the subject of the triples generated by the mapping. In this case, it uses a template to generate IRIs for the Vehicle instances, using the VIN column of the CSV file. The rr:class property sets the class of the subject to vehciles:Vehicle. The rr:predicateObjectMap statements map the columns of the CSV file to properties of the Vehicle class. For example, the first rr:predicateObjectMap maps the Model Year column to the vehciles:modelYear property, using the rr:datatype property to specify the XSD data type of the object. The second mapping is for the Company class, which represents the make of the vehicle. It maps the Make column of the CSV file to the vehciles:name property of the Company class. It also uses a template to generate IRIs for the Com-

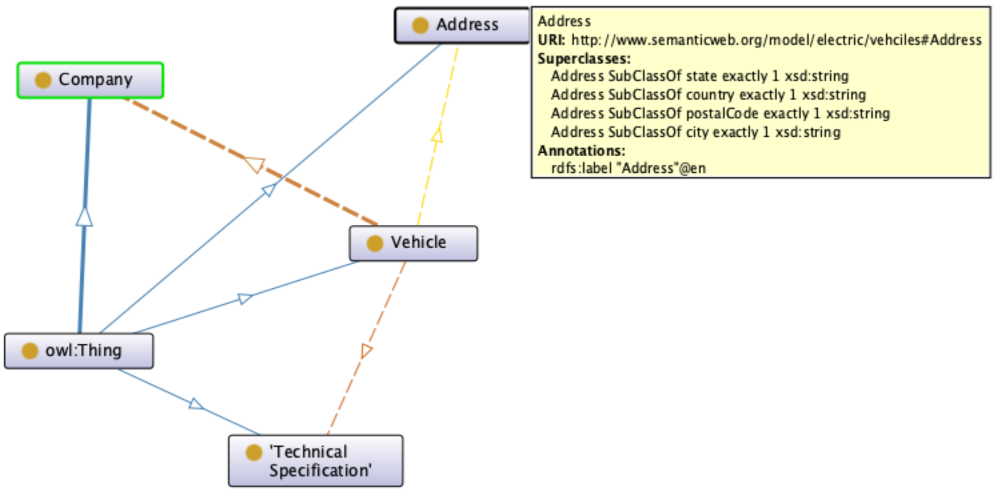


```
'Technical Specification'
URI: http://www.semanticweb.org/model/electric/vehciles#TechnicalSpecification
Superclasses:
  'Technical Specification' SubClassOf fuelCapacity exactly 1 xsd:string
  'Technical Specification' SubClassOf bodyAssembleLocation exactly 1
  xsd:string
  'Technical Specification' SubClassOf engineConfiguration exactly 1
  xsd:string
  'Technical Specification' SubClassOf bodyType exactly 1 xsd:string
  'Technical Specification' SubClassOf fuelInjection exactly 1 xsd:string
  'Technical Specification' SubClassOf horsePower exactly 1 xsd:string
  'Technical Specification' SubClassOf transmissionType exactly 1 xsd:string
  'Technical Specification' SubClassOf brakesType exactly 1 xsd:string
  'Technical Specification' SubClassOf oilCapacity exactly 1 xsd:string
  'Technical Specification' SubClassOf engineAssembleLocation exactly 1
  xsd:string
Annotations:
  rdfs:label "Technical Specification"@en
```

Figure 27: Class Properties

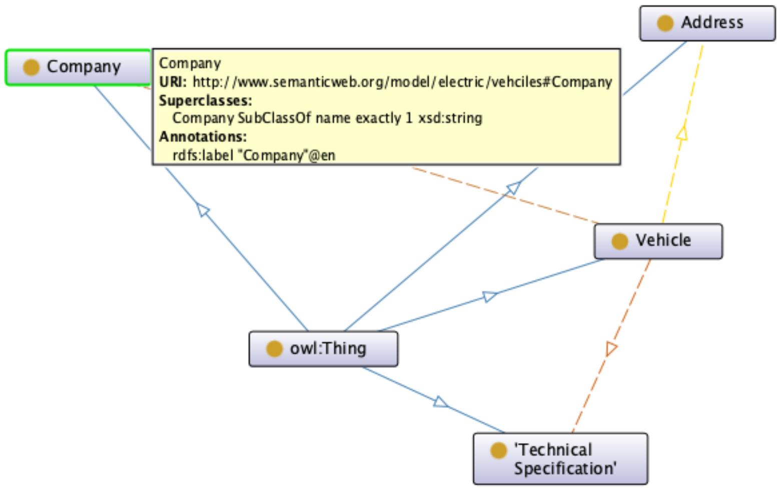
pany instances. The third mapping is for the Address class, which represents the register address of the vehicle. It maps the City column of the CSV file to the rr:template property of the rr:subjectMap, and uses a template to generate IRIs for the Address instances.

```
rml:logicalSource [
    rml:source "demo−data/customCSV.csv" ;
    rml:referenceFormulation ql:CSV
  ];
  rr:subjectMap [
    rr:template "http://www.semanticweb.org/model/electric/vehciles#Vechicle_{VIN (1−10)}";
    rr:class vehciles:Vehicle
  ];
  rr:predicateObjectMap [
    rr:predicate vehciles:modelYear;
    rr:objectMap [
      rml:reference "Model Year";
      rr:datatype xsd:string
      ]
```



Address
URI: http://www.semanticweb.org/model/electric/vehciles#Address
Superclasses:
Address SubClassOf state exactly 1 xsd:string
Address SubClassOf country exactly 1 xsd:string
Address SubClassOf postalCode exactly 1 xsd:string
Address SubClassOf city exactly 1 xsd:string
Annotations:
rdfs:label "Address"@en

Figure 28: Class Properties



Company
URI: http://www.semanticweb.org/model/electric/vehciles#Company
Superclasses:
Company SubClassOf name exactly 1 xsd:string
Annotations:
rdfs:label "Company"@en

Figure 29: Class Properties

```
];
rr:predicateObjectMap [
  rr:predicate vehciles:model;
  rr:objectMap [
    rml:reference "Model";
    rr:datatype xsd:string
    ]
];
rr:predicateObjectMap [
  rr:predicate vehciles:hasMake;
  rr:objectMap [
    rr:template "http://www.semanticweb.org/model/electric/vehciles#Company_{Make}"
    ]
```

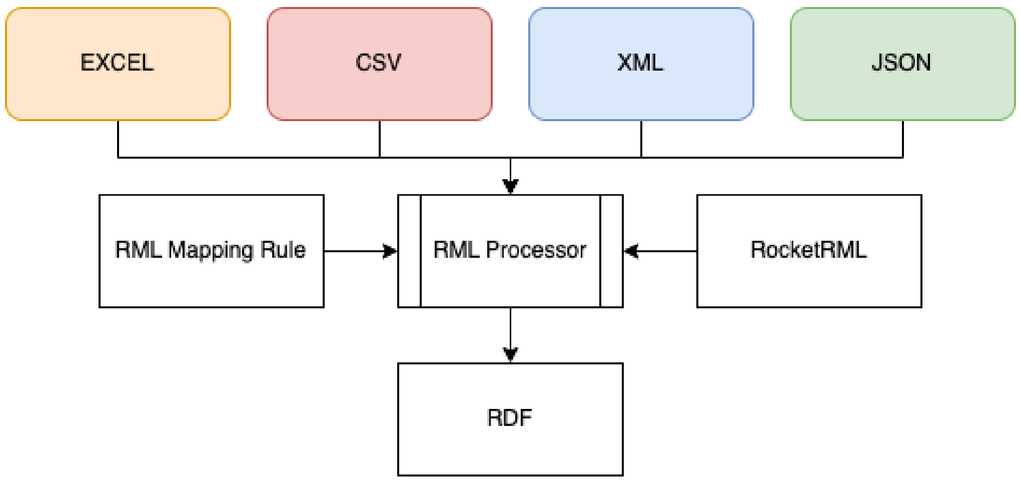


Figure 30: RML Mapping Rules

```
];
rr:predicateObjectMap [
    rr:predicate vehciles:type;
    rr:objectMap [
        rml:reference "Electric Vehicle Type";
        rr:datatype xsd:string
        ]
];
```

5.2.2.2 JSON to RDF

This is a RDF Mapping Language (RML) mapping file written in Turtle syntax for mapping data from a custom JSON file to RDF triples. The file defines two mapping blocks. The first block

```
<\#VehiclesMapping>
```

maps the vehicle data, while the second block

```
<#VehicleRegisterAddressMapping>
```

maps the address information of the vehicles. The first mapping block defines a logical source, which specifies the location of the JSON file, the format of the data (JSONPath in this case), and the iterator used to access the data. It then maps the data to triples, with each subject being a vehicle identified by a IRI constructed using the rr:template property. The class of the vehicle is specified with the rr:class property. The properties of the vehicle are mapped using rr:predicateObjectMap blocks. Each block defines a predicate (property) and an object (value) for the given predicate. The object is obtained from the JSON data using rml:reference property. The second mapping block,

```
<\#VehicleRegisterAddressMapping>
```

, maps the address information of the vehicles using a similar structure. The IRI of the address is constructed using rr:template, and the class is specified with rr:class. The address properties are mapped using rr:predicateObjectMap blocks.

```
rml:logicalSource [
    rml:source "demo-data/customJSON.json";
    rml:referenceFormulation ql:JSONPath ;
    rml:iterator "$.data[*]"
  ];
  rr:subjectMap
  [
    rr:template "http://www.semanticweb.org/model/electric/vehciles#Vechicle_{0}";
    rr:class vehciles:Vehicle;
  ];
  rr:predicateObjectMap [
    rr:predicate vehciles:VIN;
    rr:objectMap [
      rml:reference "0"
    ]
  ];
  rr:predicateObjectMap [
    rr:predicate vehciles:hasRegister;
    rr:objectMap [
      rr:template "http://www.semanticweb.org/model/electric/vehciles#Address_{5}"
      ]
  ];
```

5.2.2.3 XML to RDF

This is a RDF mapping document written in the R2RML language. The document defines three triples maps, each with their own logical source and subject map. The purpose of these triples maps is to map data from an XML file to an RDF graph. The first triples map is called VehicleMapping and is used to map data about vehicles to the graph. It defines a logical source which points to an XML file and an iterator which specifies the path to the elements that will be mapped. It also defines a subject map which specifies the template for the IRIs of the vehicles in the graph and their class. The predicate-object maps define the mappings for the different properties of the vehicle, such as VIN, model year, make, model, etc. The second triples map is called VehicleRegisterAddressMapping and is used to map data about the addresses of vehicle registrations to the graph. It defines a logical source which points to the same XML file as the first triples map and an iterator which specifies the path to the elements that will be mapped. It also defines a subject map which specifies the template for the IRIs of the addresses in the graph and their class. The predicate-object maps define the mappings for the different properties of the address, such as city, state, and postal code. The third triples map is called CompanyMapping and is used to map data about vehicle manufacturers to the graph. It defines a logical source which points to the same XML file as the first two triples maps and an iterator which specifies the path to the elements that will be mapped. It also defines a subject map which specifies the template for the IRIs of the companies in the graph and their class. The predicate-object maps define the mappings for the different properties of the company, such as name.

```
rml:logicalSource [
       rml:source "demo–data/customXML.xml" ;
    rml:iterator "/root/row";
    rml:referenceFormulation ql:XPath;
  ];
  rr:subjectMap [
    rr:template "http://www.semanticweb.org/model/electric/vehciles#Vechicle_{VIN}";
    rr:class vehciles:Vehicle
  ];
  rr:predicateObjectMap [
    rr:predicate vehciles:VIN;
    rr:objectMap [
      rml:reference "VIN"
    ]
  ];
```

5.2.3 Unstructured File to RDF

5.2.3.1 Docx to RDF

The code defines two named individuals of type Vehicle, identified by the IRIs

`http://www.semanticweb.org/model/electric/vehciles#Vechicle\_WAUUPBFF3G`

and

```
http://www.semanticweb.org/model/electric/vehciles#Vechicle\_KNDCD3LD5J
```

. These individuals represent vehicles and have a relation hasTechnicalSpecification to instances of TechnicalSpecification, which represent the technical details of the vehicles.
The technical specification instances have various properties such as bodyAssembleLocation, bodyType, brakesType, engineConfiguration, fuelCapacity, horsePower, and oilCapacity, which describe the technical specifications of the vehicles.

5.2.3.2 PDF to RDF

This ontology is written in the RDF (Resource Description Framework) syntax using the Turtle format. The ontology describes a vehicle and its technical specifications. The first prefix, ":," is used to refer to the base namespace

```
http://www.semanticweb.org/model/electric/vehciles#
```

.
The next section defines an individual named ":Vechicle_WAUUPBFF3G" of type "owl:NamedIndividual" and ":Vehicle." It also has a property ":hasTechnicalSpecification" that refers to another individual named ":Vechicle_WAUUPBFF3G_TechnicalSpecification."
The following section defines the individual ":Vechicle_WAUUPBFF3G_TechnicalSpecification" of type "owl:NamedIndividual" and ":TechnicalSpecification." It has several properties that describe the technical specifications of the vehicle, such as its body type, brakes type, engine configuration, and transmission type.

```
@prefix : <http://www.semanticweb.org/model/electric/vehciles#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://www.semanticweb.org/model/electric/vehciles> .

### http://www.semanticweb.org/model/electric/vehciles#Vechicle_WAUUPBFF3G
:Vechicle_WAUUPBFF3G rdf:type owl:NamedIndividual ,
                                       :Vehicle ;
                           :hasTechnicalSpecification :Vechicle_WAUUPBFF3G_TechnicalSpecification .
```

```
### http://www.semanticweb.org/model/electric/vehciles#Vechicle_WAUUPBFF3G_TechnicalSpecification
:Vechicle_WAUUPBFF3G_TechnicalSpecification rdf:type owl:NamedIndividual ,
                                                    :TechnicalSpecification ;
                                           :bodyAssembleLocation "Hermosillo, Mexico"^^xsd:string ;
                                           :bodyType "Unitized steel body"^^xsd:string ;
                                           :brakesType "Power−assisted, standard four−channel, four−wheel
                                               discs with ABS and AdvanceTrac electronic stability control"^^xsd
                                               :string ;
                                           :engineAssembleLocation "Chihuahua, Mexico"^^xsd:string ;
                                           :engineConfiguration "Aluminum block and head"^^xsd:string ;
                                           :fuelCapacity "16.5 gallons"^^xsd:string ;
                                           :fuelInjection "Sequential multiport electronic"^^xsd:string ;
                                           :horsePower "175 @ 6,000 rpm"^^xsd:string ;
                                           :oilCapacity "5.7 quarts with filter"^^xsd:string ;
                                           :transmissionType "6F35 six−speed SelectShift automatic"^^xsd:string
                                               .
```

5.2.4 GraphDB

GraphDB provides a web-based interface called the Workbench that allows us to manage repositories, import data, and query the data stored in the repository. Access the Workbench by entering the URL of our GraphDB instance in a web browser which is http://localhost:7200.

5.2.4.1 Create Repository

In the GraphDB Workbench, we created "Create new repository" button providing a "customRepo" name for the repository. We can also configure additional settings such as the repository type (e.g., OWLIM, RDF4J Native, RDF4J Memory) and storage options but we kept it as default for now.

5.2.4.2 Upload Ontology Data

After creating the repository, we navigate to the "Import" tab in the Workbench. This is where we can upload our ontology data files. We uploaded the data by clicking "Upload RDF files" button to select the files we want to upload. GraphDB supports various RDF file formats such as .n3, .ttl, .rdf, .xml. Our structured file (csv, excel, json, xml) generated .n3 file using RocketRML package but the un-structured files (docx, pdf) are in .ttl format. Once we have selected the files, by clicking "Open" to start the upload process. GraphDB will import the RDF data from the files and add it to the repository.

5.2.4.3 Import RDF Data

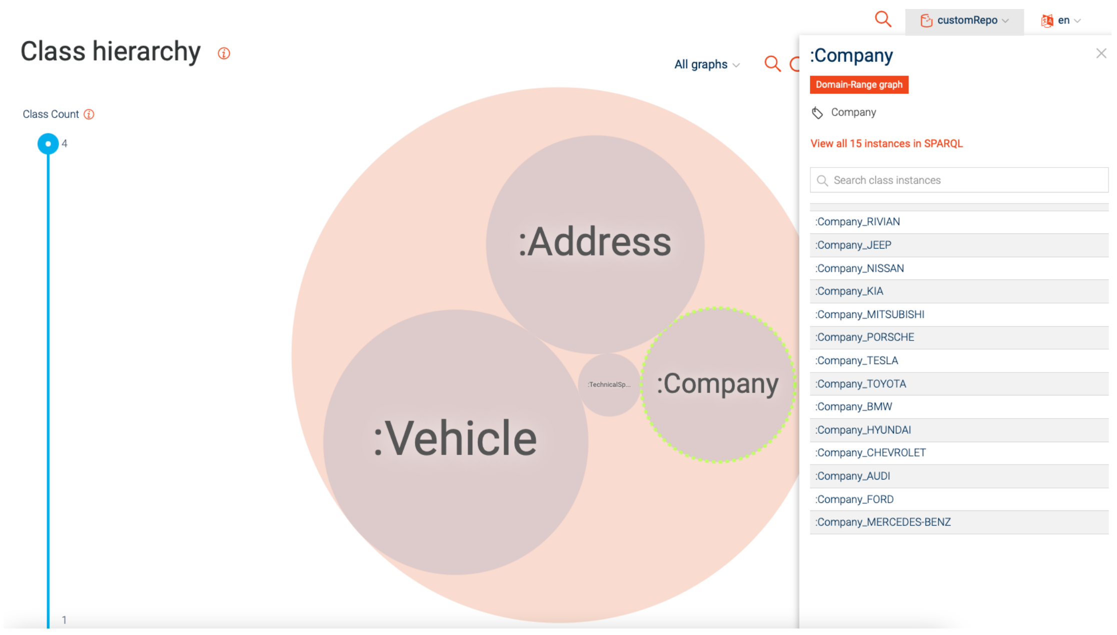
After uploading the RDF data, we need to import them to the GraphDB. For that we need to click on "import" for each RDF files. In the import settings, we have Base IRI and Target Graphs. In Target Graph we select Name graph from which we will know from where the data is coming from. For example -

- http://www.semanticweb.org/model/electric/vehciles/CSV where the data is coming from CSV file
- http://www.semanticweb.org/model/electric/vehciles/JSON where the data is coming from JSON file
- http://www.semanticweb.org/model/electric/vehciles/XML where the data is coming from XML file
- http://www.semanticweb.org/model/electric/vehciles/DOCX where the data is coming from DOCX file
- http://www.semanticweb.org/model/electric/vehciles/PDF where the data is coming from PDF file
- http://www.semanticweb.org/model/electric/vehciles/EXCEL where the data is coming from EXCEL file

Once the upload is complete, "Explore" tab in the Workbench tab allows us to query and visualize the data stored in the repository. We can execute SPARQL queries to verify that the imported data is correct and explore the relationships and properties defined in your ontology.

5.2.4.4 Class Hierarchy

The Class hierarchy view shows the hierarchy of RDF classes by the number of instances. The biggest circles are the parent classes, and the smaller nested ones are their subclasses. We can hover over a given class to see its subclasses or zoom in a nested circle (RDF class) for further exploration.
As we can see in the figure 31 and 32, for the CustomRepo, we have 4 nested classes named :Address, :Company, :Vehicle, :TechnicalSpacifications where :Vehicle class is the biggest and then :Address



:Company

Domain-Range graph

Company

View all 15 instances in SPARQL

Search class instances

:Company_RIVIAN
:Company_JEEP
:Company_NISSAN
:Company_KIA
:Company_MITSUBISHI
:Company_PORSCHE
:Company_TESLA
:Company_TOYOTA
:Company_BMW
:Company_HYUNDAI
:Company_CHEVROLET
:Company_AUDI
:Company_FORD
:Company_MERCEDES-BENZ

Figure 31: :Company class with instances

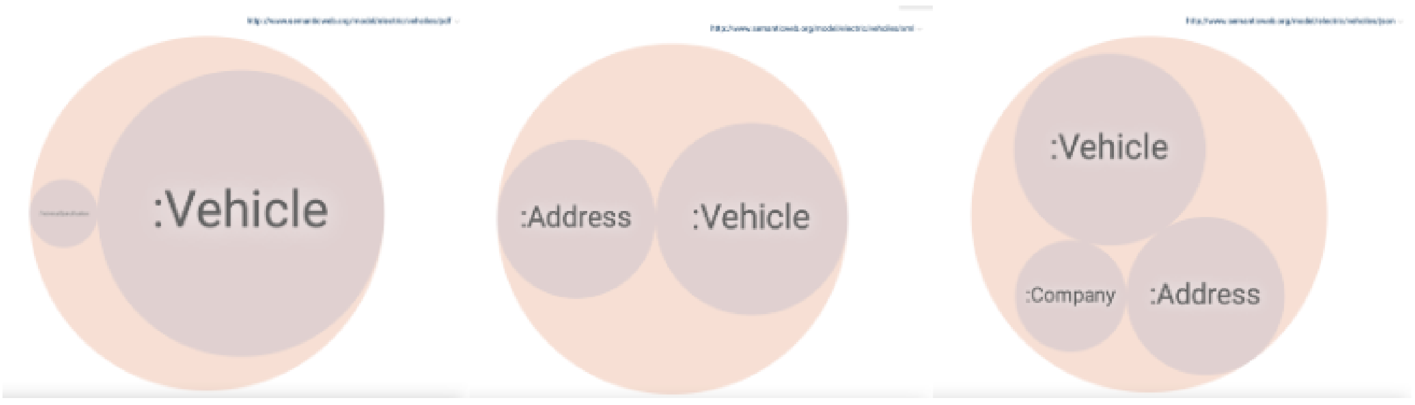
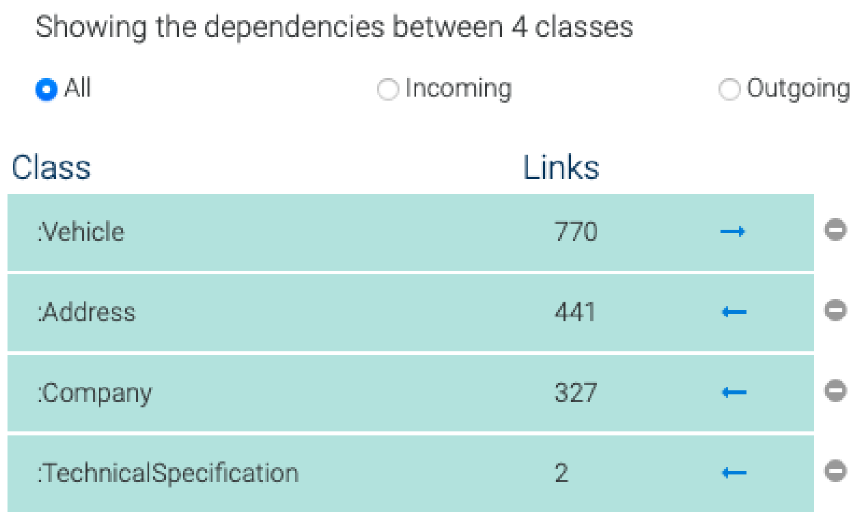


Figure 32: Class Hierarchy

class. There are 4 figures and each figure represents All graphs, CSV and JSON graph, pdf and docx graphs, and XML graphs. We can also see the instances of the graph.

5.2.4.5 Class Relationship

The Class relationships view shows the relationships between RDF classes, where a relationship is represented by links between the individual instances of two classes. Each link is an RDF statement where the subject is an instance of one class, the object is an instance of another class, and the link is the predicate. Depending on the number of links between the instances of two classes, the bundle can be thicker or thinner, and it receives the color of the class with more incoming links. The links can be in both directions.



Showing the dependencies between 4 classes

[✓ All] Incoming Outgoing

Class	Links	
:Vehicle	770	→
:Address	441	←
:Company	327	←
:TechnicalSpecification	2	←

Figure 33: Class Relationship

5.2.4.6 Visual Graph Relationships

The Visual graph view provides a visual representation of parts of the RDF graph. The visualisation starts from a single resource and the resources connected to it or from a graph query result. We created a visual graph using the following query.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
construct {
?node rdfs:subClassOF ?restr.
?node ?prop ?range.
}
where {
?node rdfs:subClassOf ?restr .
optional {
?restr owl:onProperty ?prop .
}
optional {
?restr ( owl:onClass | owl:onDataRange
| owl:someValuesFrom | owl:allValuesFrom |owl:hasValue ) ?range.
}
}
```

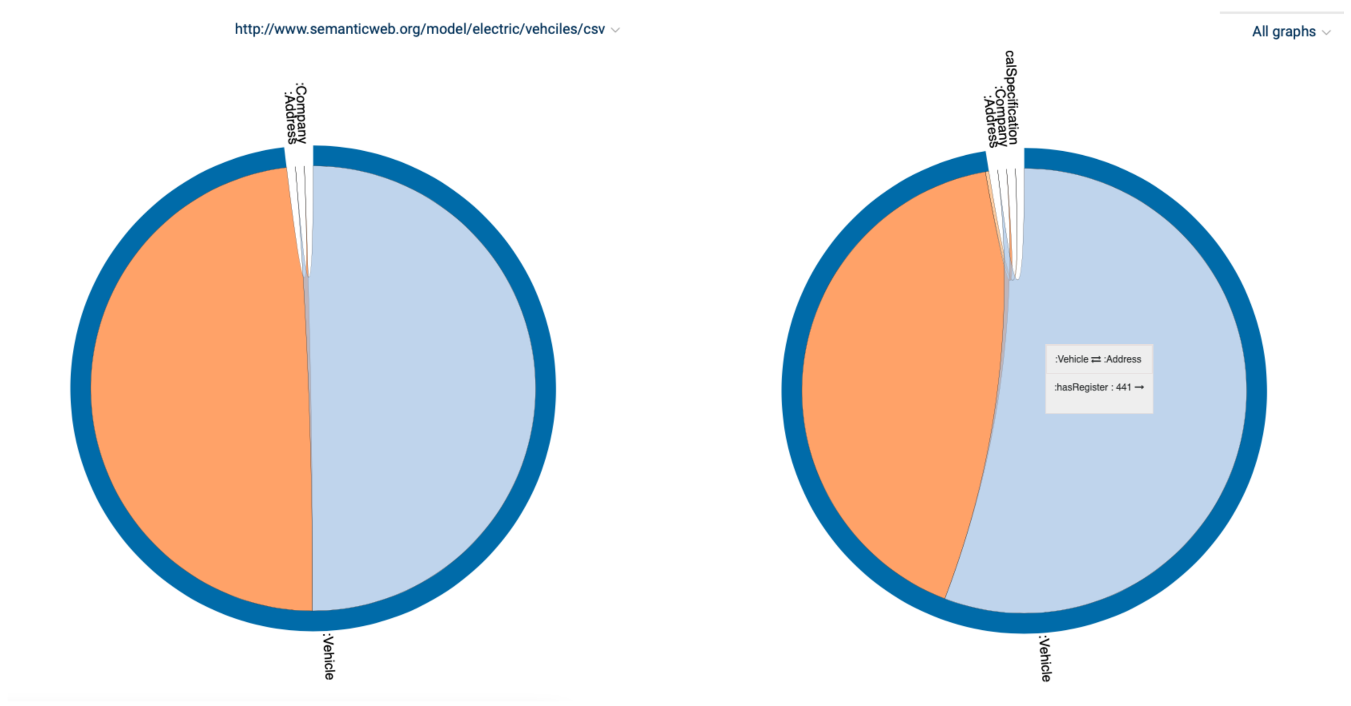


Figure 34: Class Relationship

5.2.5 SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) is a query language used for querying and manipulating RDF (Resource Description Framework) data. It allows users to retrieve, modify, and manipulate data stored in RDF graphs. In the context of GraphDB, the following sections are used in SPARQL queries to achieve the result from the experiment:

5.2.5.1 SELECT

The SELECT clause is used to specify the variables that should be included in the query result. These variables represent the information that you want to retrieve from the graph.When querying RDF data, there may be instances where the same triple pattern matches multiple times in the graph, leading to duplicate results in the query output. The DISTINCT keyword allows you to filter out these duplicate results and obtain a concise and unique set of solutions. For example:

```
select distinct ?g ?make ?2020_Census_Tract ?Base_MSRP ?
Clean_Alternative_Fuel_Vehicle ?congressionalDistricts ?
DOL_Vehicle_ID ?Electric_Utility ?electricRange ?model ?
modelYear ?type ?Vehicle_Location ?VIN ?
WAOFM_GIS_Legislative_District_Boundary ?bodyAssembleLocation ?
bodyType ?brakesType ?engineAssembleLocation ?engineConfiguration ?
fuelCapacity ?fuelInjection ?horsePower ?oilCapacity ?transmissionType
```

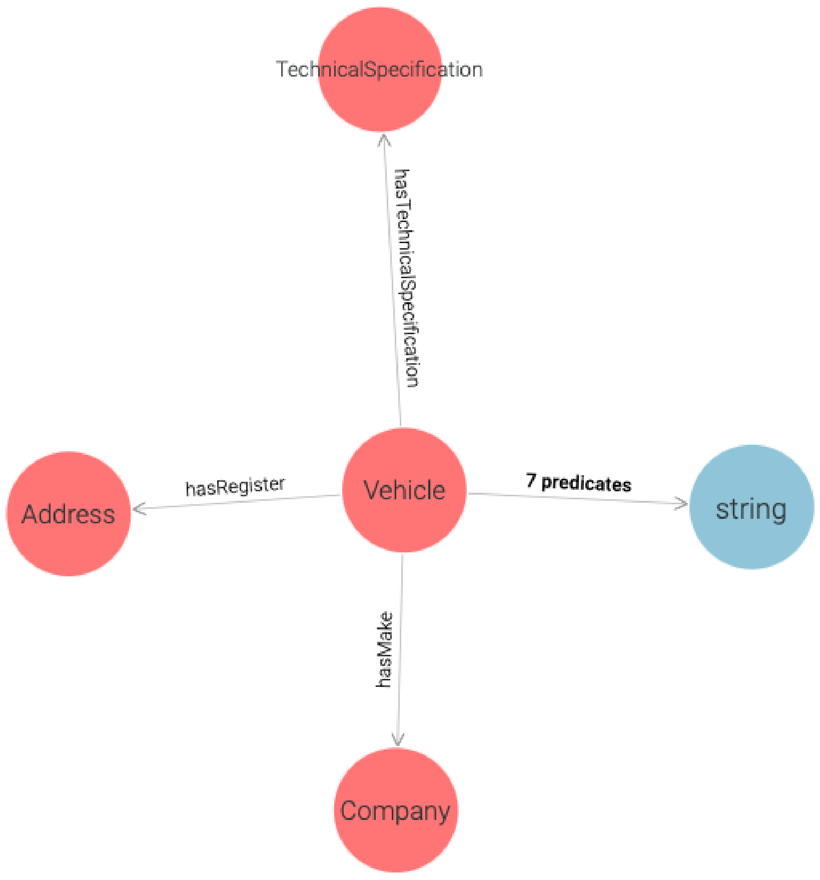


Figure 35: Visual Graph

5.2.5.2 WHERE

The WHERE clause is used to define the pattern or conditions that the queried data should match. It specifies the triple patterns to be matched against the RDF graph. Our "WHERE" part has 2 sections - Vehicle information and Technical Specification information which we marge with UNION. From the first query we can search for the Vehicle information and the second part we can search for With Technical Specification. For example

```
where {
    {
        graph ?g{
            ?iri <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.semanticweb.org/model/
                electric/vehciles#Vehicle>.
```

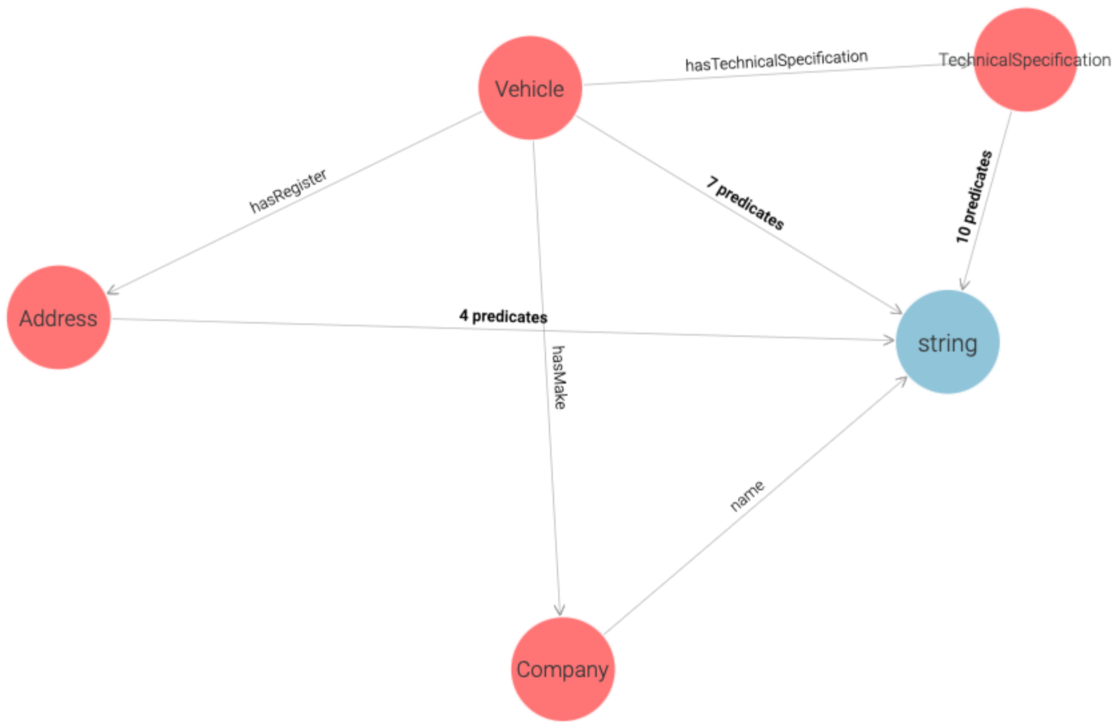


Figure 36: Visual Graph Relationship

```
            optional{
                ?iri <http://www.semanticweb.org/model/electric/vehciles#hasRegister> ?hasRegister.
                ?hasRegister <http://www.semanticweb.org/model/electric/vehciles#city> ?registerCity.
                ?hasRegister <http://www.semanticweb.org/model/electric/vehciles#state> ?registerState.
            }
        }
    }
    union
    {
        graph ?g{
            ?iri <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.semanticweb.org/model/
                electric/vehciles#Vehicle>.
            ?iri <http://www.semanticweb.org/model/electric/vehciles#hasTechnicalSpecification> ?
                hasTechnicalSpecification.
            optional{
                ?hasTechnicalSpecification <http://www.semanticweb.org/model/electric/vehciles#
                    bodyAssembleLocation> ?bodyAssembleLocation.
                }
            }
        }
}
```

5.2.5.3 GRAPH

The GRAPH keyword is used to specify a specific named graph or the default graph in the RDF dataset. It allows you to query data from a specific graph or combine data from multiple graphs. With "?g" variable, it will store the graph data in the g variable. For example:

5.2.5.4 OPTIONAL

The OPTIONAL keyword is used to specify optional patterns in the query. It allows you to retrieve additional data that is not mandatory for a match. If the information is not in the graph then it will not show the data. For some retrieval, we use optional because some of the data are not available in all JSON/XML/CSV files.

5.2.5.5 IRIs

In IRI, it will retrieve all the data which type is Vehicle. After getting this, we use IRI to get the object property and save it in the variable. For example, with IRI, we can get hasRegister. After saving in hasRegister, we can use hasRegister IRI to retrieve the city and state. We used the naming convention according to our model.

```
?iri <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.semanticweb.org/model/electric/vehciles#
    Vehicle>.
            optional{
                ?iri <http://www.semanticweb.org/model/electric/vehciles#hasRegister> ?hasRegister.
                ?hasRegister <http://www.semanticweb.org/model/electric/vehciles#city> ?registerCity.
                ?hasRegister <http://www.semanticweb.org/model/electric/vehciles#state> ?registerState.
```

5.2.6 Backend

For this execution, we used JavaScript language and Node.js for the backend and React.js for Frontend. Backend code sets up a Node.js server with Express to handle semantic search requests on a

GraphDB repository. It receives a search sentence, executes a SPARQL query against the repository, filters the results based on the search sentence, and returns the filtered results as a response.

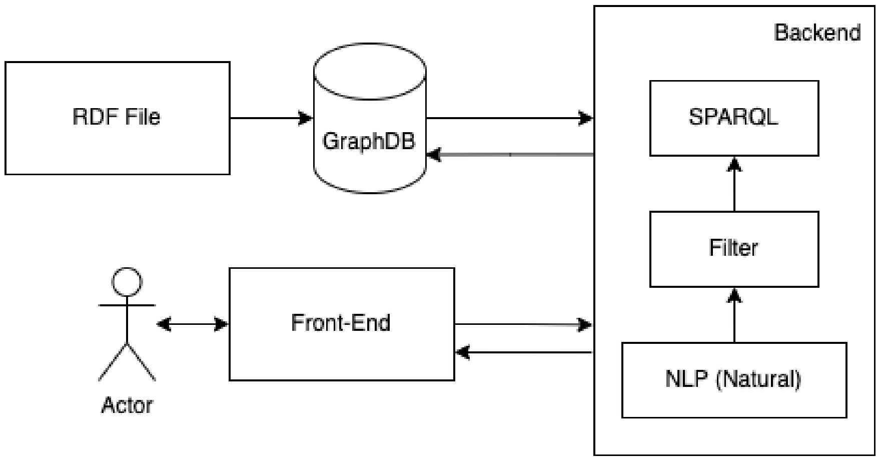


Figure 37: Execution Part

5.2.6.1 Define Search API Endpoint

we set up a POST route at "/search" to handle search requests from Frontend. It retrieves the search sentence from the request body, sets up a SPARQL query to perform semantic search, and connects to the GraphDB repository using the EnapsoGraphDBClient library. Once the query results are obtained, they are filtered based on the search sentence using the filterResults function, and the filtered results are sent as the response.

```
app.post("/search", (req, res) => {
  // Request body contains the search sentence
  const sentence = req.body.sentence;

  // SPARQL query for semantic search
  const query = ‘select distinct ?g ?make ?2020_Census_Tract ...‘;

  // Connect to the GraphDB repository

  // Perform the SPARQL query and filter the results
  graphDBEndpoint
    .query(query, { transform: "toJSON" })
    .then((result) => {
      const filteredResults = filterResults(result.records, sentence);
      res.send(filteredResults);
    })
  // Functions to filter the results based on the search sentence
});
```

5.2.6.2 Database Connection

The provided code demonstrates how to establish a database connection between Node.js and GraphDB using the EnapsoGraphDBClient library. After that we configure connection settings for connecting to GraphDB. we specify the base URL of our GraphDB instance and the name of the repository we want to connect to. Then we used the graphDBEndpoint object to connect to the GraphDB repository and transformed our output as JSON.

```
// Connect to RDF dataset
  let graphDBEndpoint = new EnapsoGraphDBClient.Endpoint({
    baseURL: "http://localhost:7200",
    repository: "customRepo",
  });
```

5.2.6.3 Helper Functions

The code includes two custom helper functions, filterResults() (section 5.2.6.4) and filterResultsBySentences() (section 5.2.6.5), which are used to filter the query results based on the search sentence and specific patterns or keywords. These helper functions work together to perform additional filtering on the query results, based on both individual keywords and specific patterns or properties derived from the search sentence. This allows for more refined and accurate filtering of the results to match the user's search criteria.

5.2.6.4 filterResults()

This function takes in two parameters. results (the query results from the GraphDB repository) and sentence (the search sentence from front end). It filters the results based on specific properties and keywords present in the sentence. It tokenizes the sentence into individual words using the natural.WordTokenizer() from the natural module. Then it iterates over each result in the results array and checks if any of the keywords extracted from the sentence are present in specific properties of the result object, such as modelYear, Electric_Utility, make, etc. If any of the keywords match, the result is considered a valid match and is included in the filtered results. The function returns an array of filtered results that match the keywords present in the search sentence and send it to the front end.

```
function filterResults(results, sentence) {
    const tokenizer = new natural.WordTokenizer();
    const keywords = tokenizer.tokenize(sentence);

    return results.filter((result) => {
      return (
        keywords.includes(result.modelYear) ||
        keywords.includes(result.Electric_Utility) ||
        keywords.includes(result.make) ||
        keywords.includes(result.Base_MSRP) ||
        keywords.includes(result.Clean_Alternative_Fuel_Vehicle) ||
        keywords.includes(result.DOL_Vehicle_ID) ||
        keywords.includes(result.electricRange) ||
        keywords.includes(result.model) ||
        keywords.includes(result.modelYear) ||
        keywords.includes(result.type) ||
        keywords.includes(result.VIN)
      );
    });
  }
```

5.2.6.5 filterResultsBySentences()

This function takes in two parameters: vehicles (an array of vehicles) and sentence (the search sentence). It further filters the vehicles based on specific patterns or regular expressions extracted from the sentence. After that it defines a regular expression pattern for each property that needs to be matched in the sentence. For example, it defines a pattern for matching the "Electric_Utility" property, "Base_MSRP", "Clean_Alternative_Fuel_Vehicle", etc. Then it iterates over each property and corresponding regular expression pattern in the regexMap. Then it attempts to match each regular expression pattern with the sentence. If a match is found, it captures the corresponding value (e.g., the electric utility name, base MSRP amount, etc.) from the sentence. It creates a filterObj object and populates it with the captured values for each property. Finally, it calls the filterObjects function, passing the filterObj and vehicles array, to filter the vehicles based on the captured property values. The filterObjects function checks if all the properties in the filterObj match the corresponding properties in each vehicle, and returns only the vehicles that satisfy all the property matches.

```
function filterResultsBySentences(vehicles, sentence) {
    const regexMap = {
      Electric_Utility: /Electric Utility/i,
      Base_MSRP: /\$\d+(?:,\d+)*(?:\.\d+)?/,
      Clean_Alternative_Fuel_Vehicle: /Clean Alternative Fuel Vehicle/i,
      DOL_Vehicle_ID: /DOL Vehicle ID of (\d+)/i,
      electricRange: /Electric Range of (\d+) miles/i,
      model: /Model is ([\w−]+)/i,
      modelYear: /(\d{4}) model year/i,
      type: /(sedan|hatchback|suv)/i,
      make: /make is ([\w\s]+)/i,
      VIN: /with VIN (\w{17})/i,
    };

    const filterObj = {};

    for (const [key, regex] of Object.entries(regexMap)) {
      const match = sentence.match(regex);
      if (match) {
        filterObj[key] = match[1] || match[0];
      }
    }
    return filterObjects(filterObj, vehicles);
  }
```

5.2.7 Frontend

The frontend part of the application is responsible for rendering an input form and displaying the search results in a table format.

- It imports the necessary dependencies, including React and the required styles.
- React functional component called InputSection using the arrow function syntax.
- Two state variables are declared using the useState hook where *sentence*: It represents the user's input sentence for the search and *results*: It stores the search results retrieved from the backend.

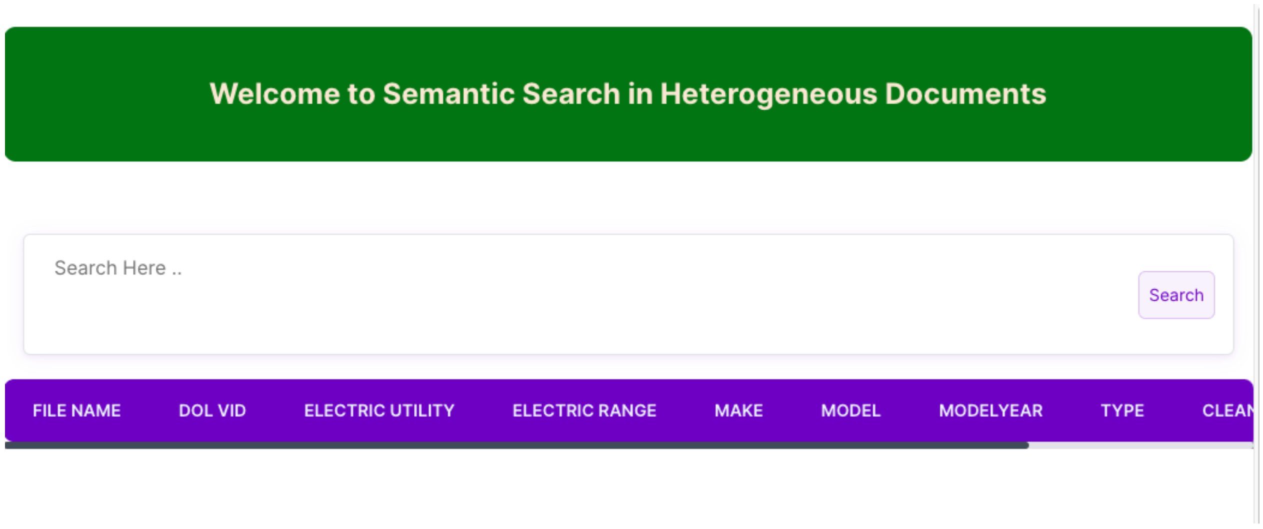


Figure 38: Frontend with React.js

- The handleSubmit function is defined as an asynchronous function that is triggered when the user submits the form.
- Afer user click submit, it performs a POST request to the backend server at "http://localhost:9000/search" using the fetch API where the request includes the sentence in the request body as a JSON string.
- If the response from the server is successful (response.ok), the response JSON is extracted and stored in the responseObj variable using response.json().
- The return statement defines the JSX markup that will be rendered by the component
- JSX includes an input section with a search form. The user can enter a sentence in the input field.
- When the form is submitted, the handleSubmit function is called.
- The table is displayed below the form, where the search results are rendered.
- The table has a header row with column headings for various properties such as "File Name," "DOL VID," "Electric Utility," etc.

See (section 5.3) for the results.

5.3 Results

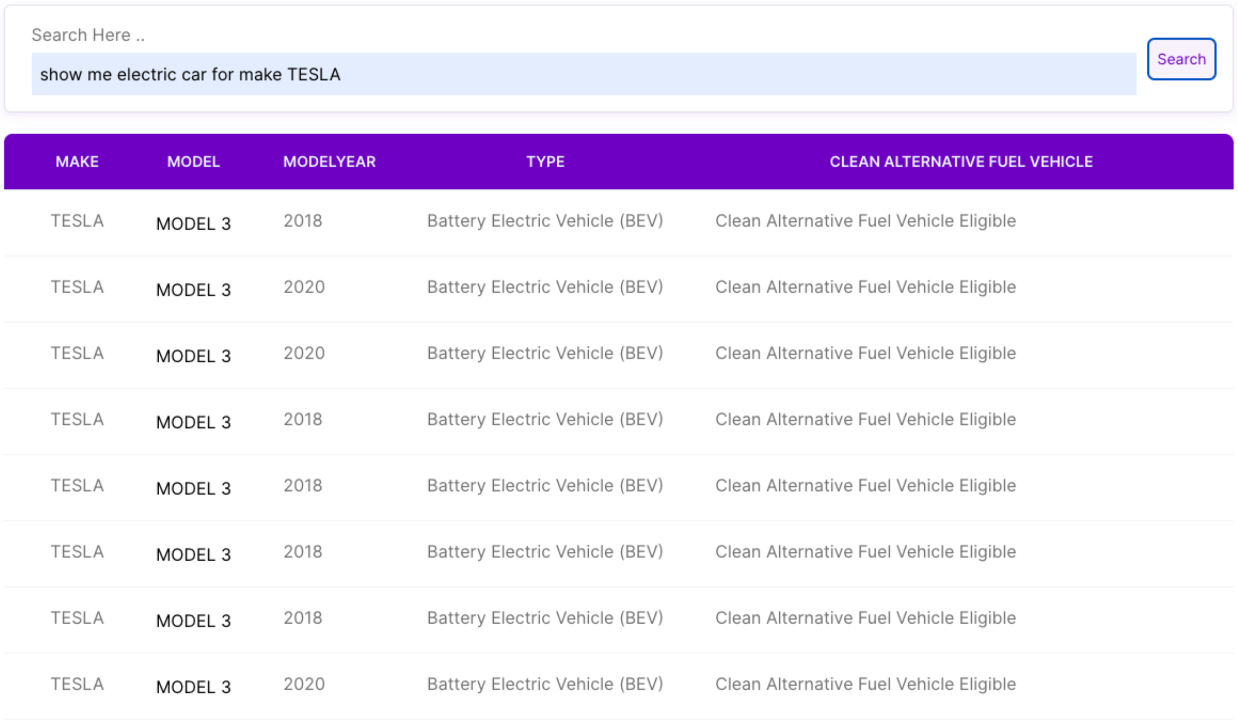
In a case study experiment, we modeled with Protege, converted data into RDF format using RML mapping rules, and then conducted a semantic search using a graph database (GraphDB) and SPARQL. This method enables extensive querying and the examination of inter-entity relationships, making it especially suitable for complicated data structures. By implementing this experimental setup, we aimed to demonstrate the effectiveness and scalability of our semantic search solution for diverse document types. We expected our approach to deliver highly accurate and relevant search results, empowering users to perform complex searches, integrate data from different sources, and extract valuable insights. See (section 6) for the discussion.

5.3.1 Experiment Summery

In this case study, our objective was to develop and showcase an effective method for performing semantic searches on a diverse range of document types. We aimed to demonstrate the power of semantic technologies and sophisticated search algorithms in providing users with highly relevant and accurate results, even when dealing with complex data structures and different formats such as docx, pdf, xml, json, excel, and csv files. To achieve this, we employed several key components in our experiment. We utilized Protege, a widely used ontology modeling tool, to create a comprehensive model for representing domain-specific knowledge. Additionally, we employed RML mapping rules to transform the data from various formats into RDF (Resource Description Framework), a standard semantic data representation format. The transformed data was then stored in a graph database, GraphDB, which enabled efficient storage, retrieval, and querying of the semantic data. For conducting the semantic search, we leveraged SPARQL (SPARQL Protocol and RDF Query Language), a powerful query language designed specifically for querying RDF data. SPARQL allowed us to perform sophisticated analysis of the relationships between entities, enabling flexible and insightful searching capabilities. To provide a user-friendly interface, we developed a web-based frontend using React.js, a popular JavaScript library for building interactive UI components. The backend was implemented using Node.js, a runtime environment for server-side JavaScript execution. This architecture allowed users to input search queries and view the results seamlessly.

5.3.2 Example Query Search

In this example, a user wants to obtain data on Tesla from heterogeneous documents. They type the search request "**Show me electric cars from Tesla**" into the search interface. The question is sent to the backend, which uses Node.js to implement it, and this starts the semantic search procedure. The method searches the graph database (GraphDB), which stores the modified semantic data from multiple file kinds, using SPARQL and the Protege ontology model. The program retrieves all pertinent information about Tesla electric automobiles by examining the links between entities. Then, it primarily focuses on electric cars as it filters the outcomes according to the requirements of the query. The user is presented with the retrieved information in a user-friendly way via the front end, which was created with React.js. Users can browse the results, get thorough details on each vehicle, and examine relevant files. This example illustrates how semantic search efficiently obtains and provides highly relevant information across various document types, powered by advanced algorithms and semantic technologies.



MAKE	MODEL	MODELYEAR	TYPE	CLEAN ALTERNATIVE FUEL VEHICLE
TESLA	MODEL 3	2018	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible
TESLA	MODEL 3	2020	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible
TESLA	MODEL 3	2020	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible
TESLA	MODEL 3	2018	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible
TESLA	MODEL 3	2018	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible
TESLA	MODEL 3	2018	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible
TESLA	MODEL 3	2018	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible
TESLA	MODEL 3	2020	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible

Figure 39: Frontend Part

5.4 Evaluation and Discussion

The Evaluation and Discussion highlight the strengths, weaknesses, and effectiveness of the experiment and its approach to performing a semantic search at an enterprise level. These strengths contribute to the experiment's potential effectiveness in delivering relevant and accurate results. The experiment showcases the promising potential for semantic search in enterprise settings, and further research and evaluation may be necessary to validate its suitability and performance.

5.5 Comparison with Our RDF Ontology Approach with Other Relevant Approaches

In comparison to other relevant approaches, the experiment offers several distinct advantages and differentiating factors.

5.5.1 Semantic Technologies and RDF

The experiment specifically focuses on leveraging semantic technologies, such as RDF ontology modeling and SPARQL querying.
In the experiment, we are managing an extensive dataset consisting of various electric car details like model, year, make and vehicle type, and so on. Using traditional keyword-based searches often leads to unsatisfactory outcomes since it fails to differentiate between different contexts of words related to our query term; for instance 'Steel body, aluminum hood' can appear in both 'AUDI' and 'TESLA'. An RDF ontology-based method offers an innovative solution that allows the establishment of meaningful associations between car entities and their corresponding attributes on the basis of defined structures governing them. For instance, defining Properties like bodyType along with multiple models per Vehicle makes Entity ensures greater accuracy as it helps tailor our search more efficiently- querying all relevant cars having a specific bodyType such as 'Steel body, aluminum hood'. Employing SPARQL's advantages will ensure that the results generated are strictly appropriate without any confusion caused by i.e., other texts with keywords related to but not indicating body type as 'Steel body, aluminum hood'. An illustration of the benefits of the RDF ontology-based approach is how it improves both the accuracy and meaning of searches by utilizing semantic connections between entities and properties. This emphasis on semantic representation allows for a more structured and meaningful organization of data, enabling advanced search capabilities and analysis of relationships

between entities. This approach provides a semantic layer that enhances search accuracy and relevance compared to traditional keyword-based approaches.

5.5.2 Diverse Document Types

Unlike some existing approaches that may specialize in specific document types or formats, the experiment addresses the challenge of searching through a wide range of heterogeneous documents, including docx, pdf, xml, json, excel, and csv files. By accommodating diverse data formats, the experiment offers a more comprehensive solution suitable for enterprises with varied document sources.

5.5.3 Graph Database Integration

The use of a graph database (GraphDB) for storing and querying semantic data is a notable aspect of the experiment. By employing a graph database, it enables efficient storage, retrieval, and traversal of relationships between entities, which is especially advantageous when dealing with complex and interconnected data structures. This approach enhances the scalability and performance of the semantic search solution compared to traditional relational or document-oriented databases.

5.5.4 Web-based Interface

The experiment incorporates a user-friendly web-based interface using React.js and Node.js. This frontend and backend combination allows users to easily input search queries and view the results, enhancing usability and accessibility for a wide range of users.

6 Discussion: Comparative Analysis of Semantic Search Approaches

The objective of this master's thesis is to develop a semantic search system specifically designed for equipment-related information. The system aims to enhance the retrieval process by utilizing advanced techniques that understand the context and meaning of the search queries and documents. This enables users to find the desired information quickly and accurately, saving time and effort. To accomplish this goal, two experiments were conducted as part of this research. The first experiment focused on retrieving information solely from structured datasets, primarily in CSV or Excel formats. These datasets are commonly used to store equipment-related information, such as specifications, models, and technical details. By analyzing and indexing these datasets, the system can provide precise search results based on the user's queries.

The second experiment aimed to expand the search capabilities to include unstructured documents, such as PDF, DOCX, Excel, JSON, XML, and CSV files. These document formats are prevalent in various industries and research fields, containing valuable equipment-related information in different formats. By implementing sophisticated techniques, the system can extract meaningful data from these documents and incorporate them into the search process, enabling comprehensive retrieval. In this section, we will be focusing on comparisons between experiments and approaches.

6.1 Comparisons Between Vector Embedding (Approach 1) vs RDF-Based Ontology (Approach 2)

6.1.1 Similarities Between Vector Embedding (Approach 1) vs RDF-Based Ontology (Approach 2)

Similarities between the two approaches (entity extraction with vector embedding and RDF ontology with RML) in the experiments:

6.1.1.1 Semantic Representation

Both approaches aim to represent the semantic information contained within the documents. They go beyond simple keyword matching and enable capturing the meaning, context, and relationships present in the data.

6.1.1.2 Support for Semantic Search

Both approaches are intended to facilitate semantic search by permitting users to retrieve pertinent documents based on their semantic content. In lieu of relying solely on keyword matching, they provide means to query and investigate the dataset using semantic criteria.

6.1.1.3 Semantic Relationships

Both approaches consider and leverage semantic relationships between entities within the documents. While vector embedding capture similarities between words or phrases, RDF ontology and RML enable modeling and representing complex relationships, hierarchies, and ontological concepts.

6.1.1.4 Use of Technology Stack

Both approaches use modern technologies such as machine learning, NLP techniques, graph databases, and semantic web technologies.

6.1.2 Variations between Vector Embedding (Approach 1) vs RDF-Based Ontology (Approach 2)

Variations between the two approaches:

6.1.2.1 Data Representation

Vector embedding represent documents as high-dimensional vectors in a continuous vector space, quantifying the semantic data. To model and convey semantic relationships, RDF ontology with RML represents documents as nodes and edges in a graph-based structure.

6.1.2.2 Querying Mechanism

Vector embedding frequently employ similarity-based queries, in which documents are retrieved according to their similarity to a given query. Complex queries can traverse the graph structure to retrieve documents based on their relationships and attributes when RDF ontology is combined with RML. The use of SPARQL in RDF ontology with RML enables the formulation and execution of more complex queries.

6.1.2.3 Semantic Expressiveness

RDF ontology with RML provides a more expressive representation by allowing the definition of ontologies, classes, properties, and relationships between entities. It offers the ability to capture nuanced semantics and domain-specific knowledge. Vector embedding, while effective in capturing similarities, may not provide the same level of semantic expressiveness.

6.1.2.4 Flexibility

Only the second approach - Semantic Search Through Heterogeneous Structured, Semi-Structured and Un-Structured Documents using RDF Ontology (section 5) is flexible and can handle a variety of file formats, including structured (CSV, Excel) and unstructured (JSON, XML, DOCX, PDF) data. This flexibility allows for semantic search across diverse document types. But the vector embedding approach can only handle structured (CSV, Excel, datasets).

6.1.2.5 Implementation Complexity

The implementation complexity varies between the two approaches. Vector embedding require expertise in NLP techniques, such as entity extraction, feature extraction, and similarity calculation algorithms. RDF ontology with RML involves knowledge modeling, ontology design, and mapping of data to the ontology using RML, which requires understanding semantic web technologies. The second approach also involves additional implementation complexity for building the frontend and backend using React.js and Node.js.

6.1.2.6 Scalability

Scalability considerations may differ between the two approaches. Vector embedding can handle large-scale document collections efficiently due to parallelized similarity calculations. Graph databases used with RDF ontology and RML can scale well, leveraging indexing and optimization techniques specific to graph databases, but may require additional optimizations for extremely large datasets.

6.1.3 Database Comparison

Down below in Table 1, an overview of Vector Database Vs Graph Database is given.

Topic	**Vector Database**	**Graph Database**
Data Representation.	Data is represented as high-dimensional vectors in a vector space. Each document is transformed into a numerical vector.	Data is represented as nodes, and their semantic connections are represented as edges in a graph which is a flexible and rich representation of semantic relations.
Flexibility and Expressiveness	Provide a convenient way to capture semantic similarities between data using vector embedding. However, may struggle with complex semantic relationships beyond simple similarity calculations.	Captures complex semantic relationships. Graph-based models allow for the representation of various types of relationships, making them suitable for representing heterogeneous document collections.
Querying Capabilities	Involve similarity-based queries. Given a query, calculates the similarity between the query vector and the data vector to retrieve the most similar data.	Offer more advanced querying capabilities that can traverse the graph structure to discover and retrieve based on relationships, enabling exploration of interconnected documents or data.
Scalability	Can efficiently handle large-scale document collections due perform parallelized vector similarity calculations. If the dimensionality and size of the vectors increase, the storage and computational requirements also increase.	Graph databases can scale well with the growth of the document collection. They can handle large graphs and still maintain efficient query performance, thanks to indexing and optimization techniques specific to graph databases.
Integration of Heterogeneous Data	Excel in integrating and searching across homogeneous data. Handling heterogeneous data sources requires additional preprocessing and normalization steps to ensure compatibility.	Naturally accommodate the integration of heterogeneous data. Relationships can be modeled. This flexibility makes graph databases well-suited for semantic search on heterogeneous documents.
The complexity of the Implementation	Requires expertise in vector embedding, dimensionality reduction techniques, and similarity calculation algorithms.	Involves data modeling, graph representation, and query optimization. May require additional effort to design and maintain the graph structure and relationships accurately.

Table 1: VectorDB Vs GraphDB

Although the choice between a vector database and a graph database for semantic search on heterogeneous documents depends on the specific requirements of the application, the complexity of semantic relationships, and the scalability needs, in our cases, a combination of both approaches may be beneficial, leveraging the strengths of vector databases for similarity-based retrieval and graph databases for capturing complex semantic relationships.

6.2 Risk Analysis

Down below in Table 2, Risk Analysis for Semantic Search on Equipment from Heterogeneous Documents is given.

Risk Description	Severity	Probability	Impact	Mitigation
Extracting relevant equipment-related information from unstructured documents, such as PDFs or DOCX files, can be challenging due to their complex and diverse formats.	Critical	Probable	Intolerable	Utilized advanced natural language processing (NLP) techniques, entity recognition to improve the accuracy and completeness of information extraction.
The system should be capable of handling increasing volumes of data without compromising search speed and efficiency.	Critical	Frequent	Undesirable	Conducted manual testing and performance tuning to identify and address bottlenecks. Continuously monitored and optimize system performance as the volume of data and document sources increase.
Unauthorized access to equipment specifications, technical details, or intellectual property can have severe consequences for organizations or individuals	Catastrophic	Occasional	Undesirable	Robust security measures should be implemented, such as encryption, access controls, and user authentication, to protect sensitive information which is out of the scope of this thesis.
Can affect the accuracy and reliability of the search results within the heterogeneous documents.	Catastrophic	Probable	Intolerable	Conducted verification of the system's ability to handle diverse data sources effectively..
Wrong Knowledge Mapping	Catastrophic	Probable	Intolerable	Implemented RDF/ Owl ontology

Table 2: Risk Analysis

6.3 Effectiveness in Terms of Overall Semantic Search on Equipment

6.3.1 Effectiveness of Vector embedding

The effectiveness of vector embedding for semantic search, highlighting their advantages, impact on information retrieval, and challenges encountered during implementation are discussed below.

6.3.1.1 Advantages and Disadvantages of Using Vector embedding

The use of vector embedding for semantic search offers several advantages in capturing semantic relationships and similarities between words or phrases in documents. One notable advantage is the ability of vector embedding to represent words or phrases as dense vectors in a high-dimensional space. This representation enables the capture of semantic information by encoding similarities based on contextual usage. By leveraging the distributional hypothesis, which suggests that words appearing in similar contexts tend to have similar meanings, vector embedding can capture semantic relationships and similarities.

6.3.1.2 Ability of Vector Embedding to Capture Semantic Relationships and Similarities

The effectiveness of vector embedding in semantic search has been demonstrated through popular techniques such as word2vec. These methods have shown promising results in capturing semantic information and facilitating information retrieval from various file formats. For instance, by representing words or phrases as vectors, these embedding techniques can enable retrieval systems to identify documents that contain similar or related content, even if the specific terms used in the query may not exactly match the terms in the documents.

6.3.1.3 Impact of Vector Embedding Techniques

The impact of vector embedding techniques cannot be understated when it comes to extracting data from different file formats. The provision for unified representation pertaining to textual context facilitates efficient analysis by semantic search systems even on documents that are available in multiple diverse formats including but not limited to docx, pdf, JSON, XML, CSV, and excel files. However, acquiring vector representations is what enables the seamless resolution of heterogeneity issues associated with specific file categories allowing quick access to a vast range of document types.

6.3.1.4 Challenges and Limitations

However, implementing vector embedding in the semantic search process is not without challenges and limitations. One challenge lies in the computational requirements associated with training and using vector embedding, especially when working with large corpora or extensive vocabularies. Training high-quality embedding may require substantial computational resources and time. Additionally, the effectiveness of vector embedding heavily relies on the availability of large, representative training data to capture the diverse semantic relationships accurately. Another limitation is that vector embedding may struggle with capturing context-specific or domain-specific meanings. While they excel at identifying general semantic relationships, capturing fine-grained nuances can be challenging. For instance, vector embedding may struggle with distinguishing between polysemous words (words with multiple meanings) or terms specific to a particular domain that may not occur frequently in the training data. This limitation can impact the precision and recall of the semantic search system when dealing with specialized or domain-specific documents.

6.3.2 Effectiveness of RDF Ontology with RDFS/OWL Modeling and RML

Several elements contribute to the success of this experiment's approach for semantic search at the enterprise level. These include precise result delivery handling diverse data formats with ease offering advanced relationship analysis capabilities ensuring scalable operations and providing a user-friendly interface for easy information retrieval and analysis. Together they constitute an effective solution that helps businesses extract knowledge from their data sources while streamlining information management.
The effectiveness of this experiment and its approach in terms of semantic search at an enterprise level can be evaluated based on several key factors:

6.3.2.1 Efficiency and Advantages of RDF-based Approaches for Extracting and Retrieving Information

Relevance and Accuracy of Results: The experiment intends to provide highly pertinent and accurate search results by utilizing semantic technologies and sophisticated search algorithms. The application of RDF ontology modeling permits a structured representation of domain-specific knowledge, thereby facilitating a more thorough comprehension of data and relationships. This improves the relevance and precision of search results, which is essential for enterprise-level information retrieval.

Handling Diverse Data Formats: The experiment addresses the challenge of searching through a variety of document types, including docx, pdf, xml, json, excel, and csv files. The ability to handle diverse data formats is essential in enterprise environments where data sources can be heterogeneous. This approach provides a comprehensive solution that can accommodate the wide range of document types commonly encountered in enterprises.

Sophisticated Analysis of Relationships: The combination of a graph database (GraphDB) and SPARQL querying permits sophisticated analysis of entity relationships. This capability enables deeper insights and a more comprehensive comprehension of data relationships, facilitating knowledge discovery and enabling enterprise-level advanced data analysis tasks.

Scalability and Efficiency: The use of a graph database and semantic technologies provides a robust and scalable solution for large-scale enterprise applications. The experiment's approach ensures efficient storage, retrieval, and querying of semantic data, supporting enterprises dealing with substantial amounts of data across various document formats.

Usability and Accessibility: Usability and accessibility are enhanced by the deployment of a web-based interface using React.js and Node.js. It enables users with varying levels of technical expertise to readily interact with the semantic search system, enter search queries, and view results. This aspect is essential in enterprise environments where a large number of users may require access to and use of semantic search capabilities.

6.3.2.2 Challenges and Limitations

Data Quality and Integration: Assuring the quality and consistency of the integrated data is a challenge for RDF-based semantic search on heterogeneous documents. Depending on the data integrity of documents from various sources, mapping them to RDF triples may require careful consideration. In addition, integrating data from various formats may necessitate overcoming format-specific obstacles, such as coping with nested structures or inconsistent data representations.

Ontology Design: Designing an effective and comprehensive ontology for representing the semantic relationships in the documents can be a complex task. It requires expertise in domain modeling and ontology engineering. Inadequate ontology design can lead to incomplete or inaccurate representation of semantic relationships, impacting the effectiveness of the semantic search.

Performance and Scalability: While RDF-based approaches offer scalability benefits, handling large-scale document collections can still pose performance challenges. Processing and indexing RDF triples, especially with complex ontologies, may require efficient data storage and retrieval mechanisms. Optimizations, such as caching, indexing, and parallel processing, may be necessary to achieve satisfactory performance in semantic search systems.

6.4 Practical Application Ability and Scalability

6.4.1 Practical Application Ability

Two methods have emerged as possible solutions to carry out semantic searches across varying formats of enterprise documents: entity extraction using vector embedding and RDF ontology in conjunction with RML techniques.

Using vector embedding to perform entity extraction renders structured document search particularly effortless - CSV files and Excel sheets are ideal examples - while retaining scalable performance levels by utilizing high-performing databases like Pinecone. This approach has an immense advantage in its effortless implementation that may also scale up quite effortlessly. To facilitate searching across multiple types of document sources including structured and unstructured ones (such as DOCX or PDF files) RDF ontology combined with RML techniques are an optimal alternative.

It allows capturing intricate relationships between entities much more accurately than standard approaches while storing all data within graph databases such as GraphDB delivers extremely responsive querying functionality using advanced query languages like SPARQL even across large datasets.

6.4.2 Scalability

Regarding scalability, both methods are capable of handling large datasets. Entity extraction with vector embedding can utilize distributed indexing and search mechanisms, while RDF ontology with RML can utilize graph partitioning and clustering techniques.

6.5 Challenges and Limitations Encountered

6.5.1 Limited Data Representation

Due to the lack of data, our thesis relies on data that are collected from the internet, which may not accurately represent the domain or context of the domain we intended to study. This limitation may impact the applicability and relevance of our results to real-world situations.

6.5.2 Data Quality and Reliability

The data collected from the internet may vary in terms of quality, reliability, and accuracy. It is challenging to verify the authenticity and correctness of the information gathered, potentially introducing biases and errors into our experiments and results.

6.5.3 Lack of Domain Specificity

Without access to company-specific data, the applicability of our findings may be limited. Each organization has its own unique set of data, vocabulary, and knowledge representation, and not having access to such data restricts the ability to analyze and evaluate semantic searches within a specific domain.

6.5.4 Weak Knowledge Map

The unavailability of company data negatively impacts the strength of our knowledge map. A robust knowledge map requires a comprehensive and accurate representation of the domain-specific information, which could not be achieved solely through internet data collection. This limitation undermines the effectiveness of our semantic search implementation and hinders its ability to deliver accurate and relevant results.

6.5.5 Limited RDF Ontology

While we conducted an experiment using RDF ontology, the effectiveness of this approach may be hindered by the unavailability of comprehensive data. RDF graphs rely on rich and well-defined ontologies, which may not be adequately established with the limited data collected. This limitation can limit the expressiveness and precision of the ontology and, subsequently, the overall performance of the semantic search system.

6.5.6 Modeling and Processing Challenges

However, Building an RDF index for a graph database involves both modeling and processing challenges. The modeling aspect requires designing a suitable ontology that accurately represents the domain and relationships between entities. This involves careful consideration of concepts, properties, and relationships. Additionally, integrating data from various sources can pose challenges in mapping different data models to RDF and resolving conflicts. On the processing side, data transformation into RDF format, especially for large datasets, requires extracting relevant information, structuring it as RDF triples, and ensuring data quality. Scalability and performance are crucial considerations, requiring optimization of indexing and querying mechanisms. However, there are automatic means available to generate a graph DB from RDF data. RDF data management systems, ontology-based mapping tools, and semantic web frameworks provide functionalities and APIs to automate the process. Although these tools can streamline certain aspects, user involvement remains essential to ensure accurate modeling decisions and address data quality issues for an effective RDF index.

6.6 Identifying the Optimal Approach

The experiments conducted to evaluate the effectiveness of two different approaches, entity extraction with vector embedding and RDF ontology with RML, in the context of semantic search on heterogeneous documents, have provided valuable insights. After the consideration of the results, according to our case studies the RDF ontology experiment appears to be superior for semantic search in a large enterprise-level setting on heterogeneous documents.

The RDF ontology experiment showed a number of significant benefits that let it handle semantic searches on a variety of file types. First, the modeling and representation of intricate semantic connections, hierarchies, and ontological ideas were made possible through the usage of RDF ontology and RML. This adaptability allows for a more sophisticated comprehension of the context and content of the documents, producing more precise and insightful search results. The RDF-based method also offered a reliable and expandable foundation for managing heterogeneous documents. The RDF ontology experiment demonstrated its capacity to smoothly integrate various data sources by supporting both structured (CSV, Excel) and unstructured (JSON, XML, DOCX, PDF) file formats. This capability is especially useful at the enterprise level when documents are frequently obtained from different systems and formats. The RDF ontology experiment also has sophisticated SPARQL querying capabilities, which is a big advantage. The retrieval of information based on intricate relationships and attributes within the documents was made possible by the graph-based querying strategy, which allowed for more accurate and flexible searches. For enterprise-level applications where correct information retrieval is critical, this improved querying technique delivered a more focused and refined search experience.

Additionally, the RDF ontology experiment demonstrated a higher level of semantic expressiveness compared to entity extraction with the vector embedding approach. The ability to define ontologies, classes, and properties allowed for the capture and utilization of domain-specific knowledge. This expressiveness facilitated a deeper understanding of the document's semantics and enabled more sophisticated and context-aware search operations. In section 6.6.1, we discuss choosing the ideal approach for the right enterprise environment.

6.6.1 Choosing the Optimal Method in Enterprise Environments

The choice between the two approaches (sections 4 and 5), vector embedding (section 4), and RDF ontology (section 5), depends on various factors, including the scale of the application, budget considerations, and the capabilities of the organization.

6.6.1.1 Data Complexity

If the data has a high degree of semantic relationships, hierarchies, and interconnections, RDF ontology may provide a more accurate and comprehensive representation. On the other hand, if the data is relatively simpler and can be effectively represented through numerical vector embedding, that approach may be more suitable.

6.6.1.2 Search Flexibility

If the target application demands advanced search functionalities based on complex semantic patterns, RDF ontology might be a better fit. However, if the search requirements are focused on similarity-based retrieval, vector embedding can be more efficient.

6.6.1.3 Data Integration

Vector embedding can be easily integrated into different applications and existing systems, allowing for seamless integration and retrieval of information. RDF ontology, on the other hand, requires mapping and transforming data into the RDF triple format, which may involve additional efforts for integration, especially when dealing with multiple file formats and sources.

6.6.1.4 Expertise and Resource Availability

Implementing and maintaining RDF ontology requires expertise and knowledge of semantic technologies, ontology modeling, and RDF tools like Protégé. If the target organization has expertise in semantic technologies, ontology modeling, and RDF tools like Protégé and the necessary resources to build and manage the ontology, it can be a favorable factor for choosing RDF ontology. On the other hand, if the organization has a team proficient in machine learning and natural language processing techniques, vector embedding may align better with the existing expertise.

In conclusion, despite the fact that both approaches have practical applications for semantic search on heterogeneous documents in an enterprise context, the choice between them depends on the specific domain requirements, size and complexity of the data, search capabilities, and scalability requirements. Based on these requirements and the preceding experiments, we hypothesize that using the appropriate data and documents, the RDF ontology with the RML approach may be more suitable for larger, more complex datasets with rich semantic relationships, whereas the entity extraction with vector embedding approach may be more suitable for smaller, less complex structures where fast indexing and retrieval are essential.

7 Conclusion and Future Work

7.1 Conclusion

The efforts made to examine various different approaches to conducting semantic searches on heterogeneous documents located within enterprise-level settings have provided some remarkable insights into what is possible and pointed toward many exciting avenues for possible future research endeavors. One particularly noteworthy result from our experiments (section 5) involves an RDF ontology that outperformed its peer by offering highly expressive results through sophisticated queries that span multiple file formats and multi-level semantic hierarchy while modeling complex semantic relationships. This clearly demonstrates how this approach - along with its associated technologies (such as RML) - can yield valuable outcomes for identifying insightful results accurately. For accuracy, we refer to the extent of relevance and precision with which a search result conforms to an intended meaning or user's query. At the same time, we also discovered how critical ontological knowledge was in creating useful enhancements that added context awareness across users' search experiences, something which is crucial for improving the overall correctness of the retrieval and productivity levels.

However, our entity extraction with vector embedding technique displayed great promise by capturing document representations as powerful vectors displaying subtle points of similarity in semantics across various documents. Unfortunately, though, this method struggled somewhat when dealing with complex semantic relationships or obtaining a more complete understanding of each document's context. One potential advantage of combining entity extraction with vector embedding in semantic search methodologies is the ability to handle unstructured or semi-structured data more effectively compared to RDF-based approaches. RDF relies on predefined ontologies and explicit modeling of relationships, which may be challenging or time-consuming to define for large, diverse, or rapidly evolving datasets. It seems quite obvious that both techniques possess their respective benefits and limitations; however, it is feasible to combine these methods into a single cohesive approach when designing new semantic search methodologies in the future.

Reflecting on this thesis, it becomes clear that the field of semantic search for heterogeneous documents in enterprise settings is a complex and challenging area of research. The experiments conducted have shed light on the importance of context, ontological knowledge, and the diverse nature of document sources. The findings provide a foundation for further exploration and the development of novel techniques that can address the limitations and harness the strengths of different semantic search approaches. This thesis expands our comprehension of semantic search in the presence of diverse documents and emphasizes the possibility for advancements in the field. By uniting different methods and incorporating their unique advantages, researchers strive to achieve more effective, precise, and context-aware solutions for enterprise-level information retrieval.

7.2 Future Research Scope: Combined Semantic Search Approach with RDF and Vector embedding

The experiments conducted in this research have explored the effectiveness of two different approaches, RDF ontology with RML and entity extraction with vector embedding, for semantic search on heterogeneous documents in an enterprise-level setting. As a result, a promising avenue for future research is to investigate the potential of merging these two approaches to leverage their respective strengths and address their limitations. This combined approach can offer a more comprehensive and powerful semantic search solution. With Hybrid Semantic Representation, we can explore ways to combine the semantic representations derived from RDF ontology and vector embedding. This can involve incorporating the vector embedding as additional features within the RDF graph or integrating the RDF ontologies into the vector space model. Investigate techniques to leverage the strengths of both representations, such as the rich semantic relationships captured by RDF and the ability of vector embedding to capture semantic similarity. To enhance Querying Mechanisms, we can develop advanced querying mechanisms that leverage both RDF graph-based querying and vector similarity-based retrieval. Explore ways to seamlessly integrate SPARQL queries with similarity-based search techniques to enable more precise and flexible searches.

By merging RDF ontology with RML and vector embedding, future research can aim to create a holistic semantic search solution for enterprise-level applications. Such a combined approach has the potential to leverage the strengths of both techniques, offering enhanced semantic representation, more advanced querying mechanisms, and improved retrieval accuracy for heterogeneous document collections.

References

Abass, O. A., & Arowolo, O. A. (2018). Information retrieval models, techniques and applications. *International Journal of Computer Applications*, *179*(6), 35–43. Retrieved from `https://www.ijcaonline.org/archives/volume179/number6/abass-2018-ijca-918609.pdf` doi: 10.5120/ijca2018917609

Ancona, D., Franceschini, L., Ferrando, A., & Mascardi, V. (2021, 05). Rml: Theory and practice of a domain-specific language for runtime verification. *Science of Computer Programming*, *205*, 102610. doi: 10.1016/j.scico.2021.102610

Anjomshoa, A., Karim, S., Shayeganfar, F., & Tjoa, A. M. (2020). Exploitation of semantic web technology in erp systems. In *Proceedings of the 23rd international conference on extending database technology* (pp. 443–446). Retrieved from `https://openproceedings.org/2020/conf/edbt/paper_292.pdf`

Asfand-e yar, M., & Ali, R. (2016). Semantic integration of heterogeneous databases of same domain using ontology. *International Journal of Software Engineering & Applications*, *7*(1), 101–110.

Badr, N. M., Elabd, E., & Abdelkader, H. M. (2016). A semantic-based framework for facilitating integration in erp systems. In *Proceedings of the international conference on computer science, information systems and telecommunications* (pp. 24–32).

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked data-the story so far. *International Journal on Semantic Web and Information Systems*, *5*(3), 1–22. Retrieved from `https://www.researchgate.net/profile/Christian_Bizer/publication/220781830_Linked_Data_-_The_Story_So_Far/links/0c9605292483fb69b9000000/Linked-Data-The-Story-So-Far.pdf` doi: 10.4018/jswis.2009081901

Boulos, M. N. K. (2004). A first look at healthcybermap medical semantic subject search engine. *Technology and Health Care*, *12*(1), 33–41. Retrieved from `https://content.iospress.com/articles/technology-and-health-care/thc00320`

Buriro, S., Siddiqui, I. F., Arain, Q., Shaikh, U., & Babar, N. (2015). Semantic web based healthcare data management. *International Journal of Web & Semantic Technology*, *6*, 1–10.

Caputo, A., Basile, P., & Semeraro, G. (2017). Integrating named entities in a semantic search engine. In *International conference on applications of natural language to information systems* (pp. 92–103).

Castells, P., Fernández, M., & Vallet, D. (2002). An adaptation of the vector-space model for ontology-based information retrieval. In *European conference on information retrieval* (pp. 197–206).

Charbel, N. (2018). Semantic representation of a heterogeneous document corpus for an innovative information retrieval model: Application to the construction industry. *International Journal of Information Management*, *43*, 94–107. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0268401218302731` doi: 10.1016/j.ijinfomgt.2018.07.013

Chung, H. (2019). A retrieval framework and implementation for electronic documents with similar layouts. *Journal of Intelligent & Fuzzy Systems*, *36*(2), 1151–1163. Retrieved from `https://content.iospress.com/articles/journal-of-intelligent-and-fuzzy-systems/ifs179296` doi: 10.3233/JIFS-179296

Galhotra, S., & Khurana, U. (2016). Semantic search over structured data. In *Proceedings of the 25th international conference companion on world wide web* (pp. 237–242).

Ganguly, D., Mitra, M., Roy, D., & Jones, G. J. (2015). A word embedding-based generalized language model for information retrieval. In *Proceedings of the 38th international acm sigir conference on research and development in information retrieval* (pp. 795–798).

Hagelien, T. F. (2019). A framework for ontology based semantic search. In *Proceedings of the 20th international conference on distributed computing and networking* (pp. 1–8).

Holter, O. M. (2015). Semantic embeddings for owl 2 ontologies. *Journal of Web Semantics*, *35*, 194-205. Retrieved from `https://www.sciencedirect.com/science/article/pii/S1570826815000708` doi: 10.1016/j.websem.2015.08.002

Jain, R., Duhan, N., & Sharma, A. (2021). Comparative study on semantic search engines. *International Journal of Computer Applications*, *180*(40), 30–34. doi: 10.5120/ijca2021921494

Jain, V., & Singh, M. (2021). Ontology development and query retrieval using protégétool. *International Journal of Advanced Research in Computer Science*, *12*(3), 143–147. doi: 10.26483/ijarcs.v12i3.7859

Jiang, S., Hagelien, T. F., Natvig, M., & Li, J. (2015). Ontology-based semantic search for open government data. In *Proceedings of the 9th international conference on theory and practice of electronic governance* (pp. 332–335).

Jirkovsky, V., & Obitko, M. (2018). Semantic heterogeneity reduction for big data in industrial automation. *IEEE Transactions on Industrial Informatics*, *14*(10), 4544–4553. Retrieved from `https://ieeexplore.ieee.org/abstract/document/8337021` doi: 10.1109/TII.2018.2831279

Kivikangas, P., & Ishizuka, M. (2020). Improving semantic queries by utilizing unl ontology and a graph database. In *2020 international conference on computational science and computational intelligence (csci)* (pp. 467–472).

Klinger, P., Gampe, S., Tolle, K., & Peter, U. (2018). Semantic search based on natural language processing–a numismatic example. *Semantic Web*, *9*(3), 371–391.

Kohn, A., Bry, F., & Manta, A. (2016). Semantic search on unstructured data: Explicit knowledge through data recycling. In *2016 ieee 16th international conference on data mining workshops (icdmw)* (pp. 258–264). Retrieved from `https://ieeexplore.ieee.org/abstract/document/7836616` doi: 10.1109/ICDMW.2016.0043

Kumar, S., Singh, M., & De, A. (2017). Owl-based ontology indexing and retrieving algorithms for semantic search engine. *Journal of Web Engineering*, *16*(5-6), 349–368. Retrieved from `https://www.rintonpress.com/journals/jwe/abstract/1540-9589_16_5-6_349` doi: 10.13052/jwe1540-9589.1654

Løvdahl, C. (2018). A comparison of information modeling in orm and owl. In *International conference on conceptual modeling* (pp. 181–190). Retrieved from `https://doi.org/10.1007/978-3-030-00722-8_14` doi: 10.1007/978-3-030-00722-8_14

Mrabet, Y., Bennacer, N., Pernelle, N., & Thiam, M. (2018). Supporting semantic search on heterogeneous semi-structured documents. *Information Processing & Management*, *54*(3), 439–459.

Munir, K., Odeh, M., McClatchey, R., Khan, S., & Habib, I. (2014). Semantic information retrieval from distributed heterogeneous data sources. *Journal of Computer and System Sciences*, *80*(6), 1066–1079.

Mäkelä, E. (2021). Survey of semantic search research. *Journal of Information Retrieval*, *27*(3), 237–264. Retrieved from `https://link.springer.com/article/10.1007/s10791-021-09484-9` doi: 10.1007/s10791-021-09484-9

Nachouki, G., Nachouki, M., & Chastang, M.-P. (2015). Semantic reconciliation in peer multi-data source management system. *International Journal of Database Theory and Application*, *8*(2), 123–136.

Nešić, S., Crestani, F., Jazayeri, M., & Gašević, D. (2010). Search and navigation in semantically integrated document collections. In *International conference on advanced information systems engineering* (pp. 79–94).

Noah, S. A., Alias, N. A. R., Osman, N. A., Abdullah, Z., Omar, N., Yahya, Y., & Yusof, M. M. (2013). Ontology-driven semantic digital library. *Journal of Information and Knowledge Management*, *12*(1), 1250003. doi: 10.1142/S0219649212500032

Obiniyi, A. A., Oyelade, O. N., & Junaidu, S. B. (2014). Enhancing reasoning and retrieval performance on owl using sqwrl. In *Proceedings of the 2nd international conference on advanced data and information engineering (daeng-2014)* (pp. 165–172). Retrieved from `https://link.springer.com/chapter/10.1007/978-3-319-06764-3_18` doi: 10.1007/978-3-319-06764-3_18

Pappu, A., Blanco, R., Mehdad, Y., Stent, A., & Thadani, K. (2019). Lightweight multilingual entity extraction and linking. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* (pp. 2694–2704).

Popadić, D., Iglesias, E., Sakor, A., Janev, V., & Vidal, M.-E. (2019). Towards a solution for an energy knowledge graph. In *Proceedings of the international conference on web intelligence* (pp. 399–403).

Pressat-Laffouilh'ere, T., Balay'e, P., Dahamna, B., Lelong, R., Billey, K., Darmoni, S. J., & Grosjean, J. (2018). Evaluation of doc'eds: a french semantic search tool to query health documents from a clinical data warehouse. *BMC medical informatics and decision making*, *18*(1), 91. Retrieved from `https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-018-0652-4` doi: 10.1186/s12911-018-0652-4

Rekabsaz, N. (2017). Enhancing information retrieval with adapted word embedding. In *Proceedings of the 40th international acm sigir conference on research and development in information retrieval* (pp. 1245–1248).

Ristoski, P., Rosati, J., Di Noia, T., De Leone, R., & Paulheim, H. (2018). Rdf2vec: Rdf graph embeddings and their applications. In *European semantic web conference* (pp. 498–514). Retrieved from `https://link.springer.com/chapter/10.1007/978-3-319-93417-4_30` doi: 10.1007/978-3-319-93417-4_30

Rusinol, M., Aldavert, D., Toledo, R., & Llados, J. (2017). Browsing heterogeneous document collections by a segmentation-free word spotting method. In *2017 14th iapr international conference on document analysis and recognition (icdar)* (pp. 245–250).

Ryen, V. (2021). Semantic knowledge graph creation from structured data: a systematic literature review. *Journal of Web Semantics*, *71*, 101094. Retrieved from `https://www.sciencedirect.com/science/article/pii/S1570826821000226` doi: 10.1016/j.websem.2021.101094

Salim, F. A., Muller, G., Ali, H. B., & Falk, K. (2019). User-centered data-driven approach to enhance information exploration, communication, and traceability in a complex systems engineering environment. *International Journal of Information Management*, *49*, 416–432. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0268401218301115` doi: 10.1016/j.ijinfomgt.2019.03.005

Samuelsen, S. D. (2016). Representing and storing semantic data in a multi-model database. *Journal of Web Semantics*, *40*, 58–71. Retrieved from `https://www.sciencedirect.com/science/article/pii/S157082681630035X` doi: 10.1016/j.websem.2016.06.001

Shekhar, M., & Saravanaguru, R. (2021). A case study on semantic web search using ontology modeling. *International Journal of Advanced Science and Technology*, *30*(03), 291–300. doi: 10.32777/ijast.30.03.291

Shi, C., Li, Y., Zhang, J., Sun, Y., & Yu, P. S. (2015). A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, *27*(2), 339–362.

Tang, X., Wang, X., Feng, Z., & Jiang, L. (2021). Ontology-based semantic search for large-scale rdf data. *Journal of Intelligent & Fuzzy Systems*, *40*(5), 10331–10341. doi: 10.3233/JIFS-210508

Wang, L., Liu, Y., & Zhang, Q. (2019). Effective techniques for retrieving information from heterogeneous industrial data sources. *Journal of Industrial Information Integration*, *14*, 55-63. Retrieved from `https://www.sciencedirect.com/science/article/pii/S2452337019300460` doi: 10.1016/j.jii.2019.02.003

Wrigley, S. N., Elbedweihy, K., Reinhard, D., Bernstein, A., & Ciravegna, F. (2013). Evaluating semantic search tools using the seals platform. In *European semantic web conference* (pp. 91–105). Retrieved from `https://doi.org/10.1007/978-3-642-38288-8_7` doi: 10.1007/978-3-642-38288-8_7

Zaman, G., Mahdin, H., Hussain, K., Rahman, A.-u., Abawajy, J., & Mostafa, S. A. (2018). An ontological framework for information extraction from diverse scientific sources. *IEEE Access*, *6*, 11482–11494. Retrieved from `https://ieeexplore.ieee.org/abstract/document/8323797` doi: 10.1109/ACCESS.2018.2804053

Zende, D. A., & Baban, C. G. (2015). Effective semantic search over huge rdf data. *International Journal of Computer Applications*, *114*(10), 6–10. Retrieved from `https://www.ijcaonline.org/archives/volume114/number10/20483-2015098579` doi: 10.5120/20150-9857

Appendices

Appendix A Appendix

Here is the GitHub Code Repository link -

`https://github.com/habibulmursaleen/semantic_search_w_rdf_owl_rml`