EAST WEST UNIVERSITY

RDF Data Clustering Framework

A capstone project report to be submitted in partial fulfillment of the requirements for the degree

of

Bachelor of Science in Computer Science and Engineering

by

Prinom Mojumder

2021-2-60-098

Suraiya Nusrat Tanha

2021-2-60-030

Tasnim Israk Synthia

2021-2-60-097

Nur Nahar Mim

2021-1-60-091

Under the supervision of

Dr. Mohammad Rezwanul Huq

Associate Professor

Department of Computer Science and Engineering East West University

Dhaka, Bangladesh

22 February 2025

DECLARATION

Project Title RDF Data Clustering Framework

Authors Prinom Mojumder, Suraiya Nusrat Tanha, Tasnim Israk Synthia, Nur Nahar Mim

Student IDs ID: 2021-2-60-098, ID: 2021-2-60-030, ID: 2021-2-60-097, ID: 2021-1-60-091

Supervisor Dr. Mohammad Rezwanul Huq

We, hereby, declare that the work presented in this capstone project is the outcome of the investigation performed by us under the supervision of Dr. Mohammad Rezwanul Huq, Associate Professor, Dept. of Computer Science and Engineering, East-West University, Dhaka, Bangladesh. We also declare that no part of this study has been or is being submitted elsewhere for awarding any degree or diploma.

Prinom Mojumder

ID: 2021-2-60-098

(Signature)

Suraiya Nusrat Tnaha

ID: 2021-2-60-030

(Signature)

Tasnim Israk Synthia

ID: 2021-2-60-097

(Signature)

Nur Nahar Mim

ID: 2021-2-60-091

(Signature)

Dept. of Computer Science and Engineering
East West University
Dhaka, Bangladesh

LETTER OF ACCEPTANCE

This is to certify that the capstone project entitled A Deep Learning based System for Distance Estimation and Navigation Assistance for the Visually Impaired People, submitted by Prinom Mojumder (ID: 2021-2-60-098), Suraiya Nusrat Tanha (ID: 2021-2-60-030), Nur Nahar Mim (ID: 2021-1-60-091), Tasnim Israk Synthia (ID: 2021-2-60-097) are undergraduate students of the Dept. of Computer Science and Engineering has been examined. Upon recommendation by the examination committee, we hereby accorded our approval as the presented work and submitted report fulfills the requirements for its acceptance in partial fulfillment for the degree of Bachelor of Science in Computer Science and Engineering.

Dr. Mohammad Rezwanul Huq
Associate Professor

Dept. of Computer Science and Engineering
East West University
Dhaka, Bangladesh

Dr. Maheen Islam
Chairperson, Associate Professor

Dept. of Computer Science and Engineering
East West University
Dhaka, Bangladesh

ACKNOWLEDGEMENTS

In the name of Allah, the Most Merciful, the Most Compassionate, we begin by praising Allah for His blessings and guidance in successfully completing this capstone project.

First, we extend our heartfelt thanks and gratitude to our research supervisor, Dr. Mohammad Rezwanul Huq, for his constant support and invaluable guidance throughout the research process. His enthusiasm, vision, and dedication have been a true source of inspiration. Dr. Huq has taught us how to approach our work methodically and present our findings clearly. It has been a privilege and honor to work under his mentorship.

We are also deeply grateful to the esteemed faculty members of the Department of Computer Science and Engineering at East West University for their teachings. Their guidance has played a significant role in shaping our academic and professional growth.

Our sincere appreciation goes to our parents for their unwavering love, endless prayers, sacrifices, and efforts to ensure our education and future success. We are equally thankful to our friends and relatives for their constant support and care.

Finally, we would like to express our gratitude to everyone who has contributed, either directly or indirectly, to the completion of this project. Your support has been invaluable.

With sincere appreciation,

Prinom Mojumder

Suraiya Nusrat Tanha

Tasnim Israk Synthia

Nur Nahar Mim

Dept. of Computer Science and Engineering

East West University

Dhaka, Bangladesh

ABSTRACT

The Resource Description Framework (RDF) is widely used for representing structured knowledge in the Semantic Web and Knowledge Graphs. However, RDF data is inherently complex, consisting of interconnected triples that make traditional clustering and analysis difficult. This research proposes an RDF Data Clustering Framework to effectively organize and analyze RDF data. Our approach involves converting RDF graphs into sequential representations, applying Natural Language Processing (NLP) techniques such as Word2Vec and BERT to transform these sequences into vector embeddings, and then utilizing clustering algorithms like K-Means, Gaussian Mixture Model (GMM), and Agglomerative Clustering. By leveraging these methods, we uncover hidden semantic relationships within RDF data and enhance its usability for knowledge extraction and semantic search. To evaluate clustering effectiveness, we use multiple performance metrics, including Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and Fowlkes-Mallows Index (FMI). Results demonstrate that NLP-based embeddings significantly improve clustering accuracy compared to direct graph-based approaches. This framework can be applied to various domains, including ontology management, knowledge graph optimization, and intelligent data retrieval.

Keywords: RDF, Knowledge Graph, Clustering, NLP, Embeddings, Machine Learning, Semantic Analysis, Data Mining.

Table of Contents

List of Figures

..

List of Tables

1

Introduction

The world is increasingly driven by large-scale data, and organizing this information efficiently is essential for knowledge discovery, decision-making, and intelligent applications. Among the various ways of structuring and managing data, the Resource Description Framework (RDF) has emerged as a powerful approach. RDF is a graph-based data model designed to represent information in a structured, linked, and meaningful way. It is widely used in semantic web technologies, linked open data, and knowledge graphs, making it easier to connect related pieces of information.

Despite its advantages, RDF data presents significant challenges due to its size, complexity, and interlinked structure. Traditional data analysis methods often struggle to handle RDF-based datasets effectively. One promising approach to making sense of RDF data is clustering, which involves grouping similar data points together based on shared characteristics. Clustering can help uncover patterns, improve data retrieval, and enable efficient knowledge representation.

In this research, we explore RDF data as a knowledge graph, generate sequential representations from it, employ Natural Language Processing (NLP) techniques to convert these sequences into vector embedding's, and apply clustering methods to identify and understand underlying semantic relationships within the data. By integrating these advanced techniques, we aim to develop an effective framework for RDF data clustering, enabling better organization, retrieval, and analysis of RDF knowledge graphs.

This introduction provides an overview of our research, outlining the key questions, objectives, contributions, and the overall organization of this study

1.1 Capstone Project: Research Questions and Objectives

Our research focuses on understanding how to efficiently cluster RDF data while exploring different approaches to uncovering hidden patterns and relationships within knowledge graphs. This section outlines the key research questions and objectives that guide our study.

1.1.1 Research Questions

To better understand RDF data clustering and improve its effectiveness, we investigate the following key questions:

1. **How can RDF data be transformed into a suitable format for clustering?**
   RDF data is typically structured as a knowledge graph, consisting of entities (nodes) and relationships (edges). Since most clustering techniques work with numerical representations, we explore how to convert RDF data into sequential representations and transform these sequences into vector embedding's for clustering.

2. **Which NLP techniques can be used to generate meaningful vector embedding's from RDF sequences?**
   Given that RDF data consists of relationships and semantic links, we examine how Natural Language Processing (NLP) methods, such as Word2Vec, BERT, and sentence embedding's, can be applied to RDF sequences to capture their semantic structure.

3. **What are the most effective clustering techniques for RDF-based vector embedding's?**
   Different clustering methods exist, such as K-Means, Gaussian Mixture Models (GMM), and Agglomerative Clustering. We evaluate which of these methods performs best in grouping RDF-based embedding's while maintaining the semantic structure of knowledge graphs.

4. **How can the quality of RDF clustering be evaluated?**
   Clustering effectiveness is typically measured using metrics such as Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and Fowlkes-Mallows Index (FMI). We determine which evaluation metrics are most appropriate for assessing

RDF clustering.

5. **How can RDF data clustering enhance real-world applications?**
   Finally, we investigate how RDF clustering can be applied in practical scenarios, such as knowledge retrieval, recommendation systems, data summarization, and semantic search.

1.1.2 Research Objectives

To answer the above research questions, our study has the following objectives:

- Develop a framework for transforming RDF data into sequential representations and applying NLP-based embedding techniques.
- Compare different clustering algorithms, including K-Means, GMM, and Agglomerative Clustering, on RDF-based vector embedding's.
- Evaluate various NLP methods, such as Word2Vec and BERT embedding's, to determine the best approach for representing RDF sequences.
- Assess clustering quality using multiple evaluation metrics to identify the most effective method for RDF data organization.
- Demonstrate real-world applications of RDF clustering in semantic search, recommendation systems, and automated knowledge discovery.

1.2 Focus and Contributions

Our research provides new insights and contributions to the field of RDF data clustering. The key focus areas and contributions of this study include:

Focus Areas

- Exploring RDF Data as a Knowledge Graph: RDF data is structured as a graph, and we analyze how its entities and relationships can be transformed into meaningful sequences for clustering.
- Applying NLP Techniques to RDF Sequences: We explore how Word2Vec, BERT can be used to convert RDF sequences into vector representations.
- Comparing Clustering Algorithms on RDF Data: We evaluate different clustering techniques to determine which performs best for RDF-based embedding's.
- Analyzing Semantic Relationships within RDF Data: Clustering RDF data helps identify semantic connections between entities, improving knowledge retrieval and organization.
- Evaluating Clustering Performance: We assess clustering results using multiple evaluation metrics, including precision, recall, and F1-score.

Key Contributions

- A novel RDF data clustering framework that integrates knowledge graph transformation, NLP-based embedding's, and clustering techniques.
- An extensive experimental analysis comparing clustering methods on RDF vector embedding's.
- Evaluation of NLP-based representations to identify the best embedding techniques for RDF sequences.
- A discussion on challenges and solutions in RDF clustering, including handling high-dimensional and interconnected data.
- Practical applications of RDF clustering in semantic web, knowledge graphs, and linked open data.

1.3 Organization of the Book

This research is structured into several comprehensive chapters, each focusing on a specific aspect of RDF data clustering. Chapter 1, Introduction, provides the background, motivation, research questions, objectives, and key contributions of the study. It offers an overview of RDF data clustering and highlights its significance in knowledge representation and data organization. Chapter 2, Related Work, reviews previous research on RDF data clustering, NLP-based embedding's, and knowledge graph analysis. It discusses existing techniques, their strengths, and limitations while identifying gaps in the literature that this study aims to address. Chapter 3, Dataset and Methodology, describes the datasets used in the research and explains the methodology for transforming RDF data into sequential representations. It also explores the embedding techniques and clustering methods applied within the framework. Chapter 4, Experimental Setup and Evaluation Metrics, details the experimental setup, covering data preprocessing steps, parameter selection, and the assessment of clustering quality using various evaluation metrics. This chapter ensures that the methodology is transparent and reproducible. Chapter 5, Results and Analysis, presents the clustering results and provides a comparative analysis of different techniques. It examines the effectiveness of various methods in processing RDF-based embedding's and highlights which approaches yield the best performance. Chapter 6, Discussion and Conclusion, discusses key findings, their broader implications, and potential areas for future research. This chapter summarizes the challenges encountered, possible improvements, and real-world applications of RDF data clustering, emphasizing its impact on knowledge discovery and semantic data analysis.

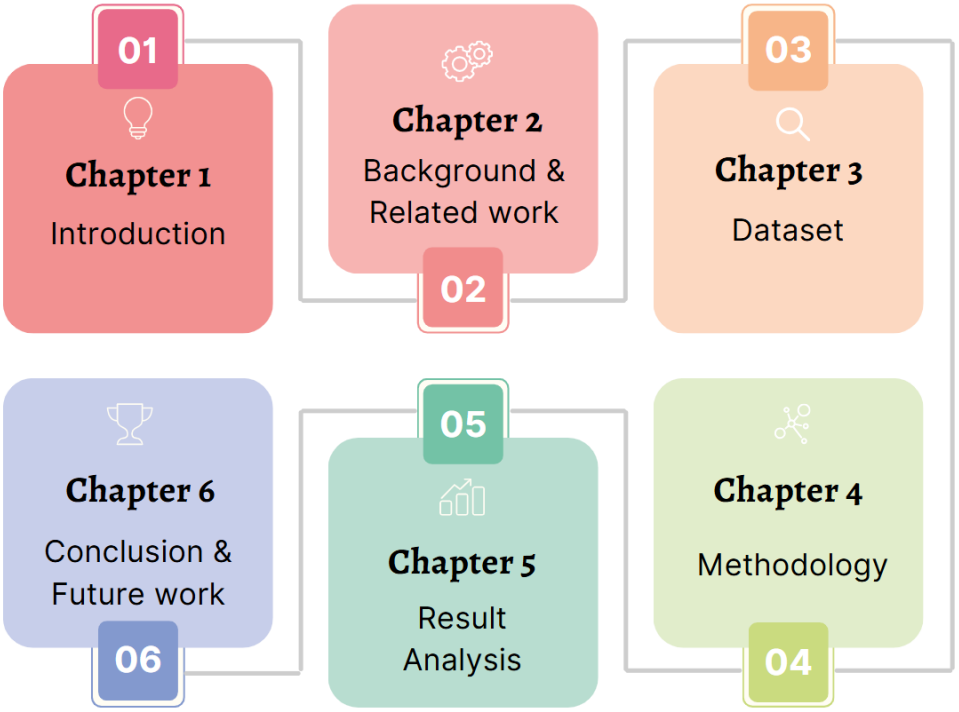

Figure 1: Flowchart of the Index

2

Background and Related Work

2.1 Background of RDF

The Resource Description Framework (RDF) has emerged as a foundational framework in the field of data representation, particularly for the semantic web, enabling interoperability across different data systems and applications [10]. RDF is designed to represent information in a graph-like structure, using subject-predicate-object triples, where the subject represents an entity, the predicate defines a property or relationship, and the object provides a value or another entity. RDF uses Uniform Resource Identifiers (URIs) to uniquely identify resources, ensuring interoperability and consistency across different datasets. Its graph-based structure enables flexible data integration, making it essential for the Semantic Web and Knowledge Graphs.

RDF's ability to capture complex relationships among entities has led to its widespread adoption in domains such as healthcare, finance, e-commerce, and scientific research [11]. However, as RDF datasets grow in size and complexity, managing and analyzing these datasets becomes increasingly challenging. RDF data often exhibits characteristics such as scarcity, high dimensionality, and heterogeneity, which complicate the process of extracting meaningful insights from the data. Efficient data analysis methods, including RDF data clustering, are crucial for improving the scalability and effectiveness of querying and reasoning over RDF data.

- **What is RDF ?**

RDF is a standard model for data interchange on the web. It allows data to be represented in a graph format, where the graph is made up of nodes and edges. Each RDF triple represents a directed edge in the graph between two nodes (resources), where the nodes are the subject and object, and the edge is the predicate.

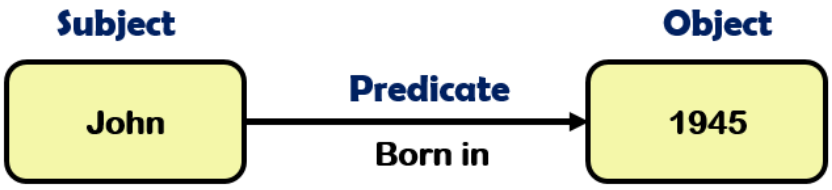

Figure 2: RDF Data Example using Triple

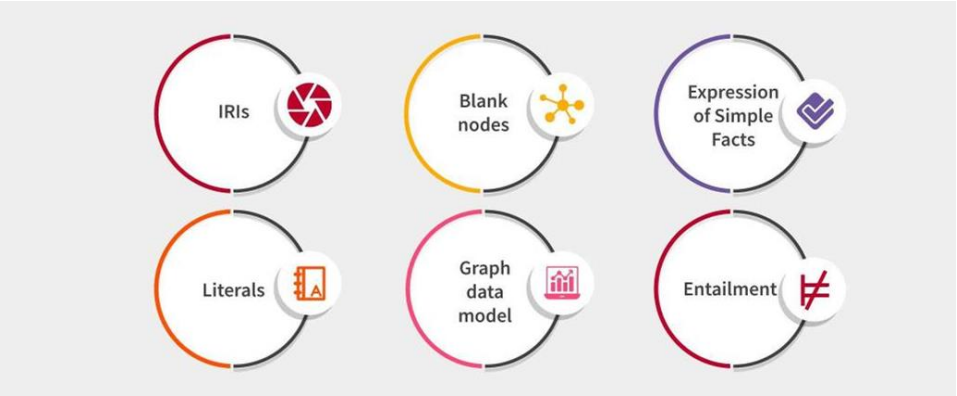

Figure 3: Fundamental concepts of RDF

RDF is a Components of a Triple:

Subject: The subject is usually a resource or an entity. It is identified using a Uniform Resource Identifier (URI). A URI serves as a globally unique reference to the resource. A subject could also be a blank node (a resource that doesn't have a URI, often used to represent an anonymous entity).

Predicate: The predicate defines the relationship or property between the subject and object. It is also identified using a URI. The predicate indicates what kind of relationship the subject

has with the object. Common predicates might represent properties such as "hasAge," "hasName," or "isLocatedIn."

Object: The object can be a literal (such as a number, string, or date) or another resource (which would have a URI). In the case of a literal, the object is just a value (like "25" or "New York"), and in the case of another resource, the object would be identified by a URI (linking to another entity).

2.2 Importance of Data Clustering in RDF

The Data clustering is a crucial technique in the analysis of RDF (Resource Description Framework) data, offering several key advantages that significantly improve the usability, efficiency, and scalability of semantic web applications. As RDF datasets continue to grow in complexity and volume, clustering becomes indispensable in organizing, retrieving, and discovering meaningful patterns within the data. The importance of data clustering in RDF can be explored through the following aspects:

Improved Data Organization and Retrieval: RDF data, by its nature, is often unstructured and consists of triples (subject-predicate-object), which can span diverse domains. As datasets increase in size, querying and retrieving relevant information become more challenging. Clustering addresses this challenge by grouping RDF resources based on shared characteristics, thereby improving the organization of data. This structured grouping enhances data retrieval efficiency, as users can access relevant clusters instead of searching through the entire dataset. Clustering allows for faster and more relevant responses to queries, optimizing computational resources.

Enhanced Knowledge Discovery: One of the primary benefits of data clustering is its ability to uncover hidden patterns within large datasets. In the context of RDF, clustering facilitates the discovery of relationships and semantic connections that may not be immediately apparent. For example, by grouping related entities, clustering can reveal new insights and associations that contribute to knowledge discovery. This is especially valuable in domains such as scientific research, business intelligence, and healthcare, where the identification of hidden relationships can drive informed decision-making and innovation.

Scalability in Large RDF Datasets: The scalability of RDF data systems is a critical consideration as the volume of linked data and knowledge graphs increases. Large RDF datasets pose significant challenges in terms of computational efficiency, especially when it comes to querying and reasoning. Clustering offers a scalable solution by reducing the complexity of these tasks. By grouping related triples into smaller, more manageable clusters, RDF data can be processed more efficiently. This reduction in complexity ensures that RDF-based systems remain responsive and scalable even as data volumes grow exponentially.

Semantic Enrichment: RDF data inherently contains rich semantic information, with entities being linked by meaningful relationships. Data clustering plays a key role in enhancing this semantic structure by grouping related entities that share similar properties or contexts. This semantic enrichment enables more accurate classification, categorization, and tagging of data, which improves the effectiveness of search engines, recommendation systems, and data analytics tools. By organizing RDF data into semantically meaningful clusters, systems can better understand and process new information, leading to more accurate results.

Improved Data Integration and Linking: RDF is widely used in the integration and linking of heterogeneous datasets from multiple sources. However, the challenge of aligning data from different domains often arises due to variations in data format, structure, and semantics. Clustering RDF data can facilitate this integration by grouping related resources from different datasets, enabling more effective data alignment and linking. This clustering-based approach helps create more coherent knowledge graphs by ensuring that data from diverse sources is harmonized and connected, thus improving the overall integrity of the dataset.

Handling Sparsity and Dimensionality: RDF datasets often exhibit Sparsity, with many entities having only a limited number of relationships. Additionally, RDF graphs are typically high-dimensional, which can make data analysis computationally expensive and difficult to manage. Clustering addresses these issues by identifying and isolating dense clusters within the RDF graph, thereby reducing the dimensionality of the data. By focusing on the most relevant clusters, clustering techniques can improve the efficiency of graph-based algorithms and enable faster data processing, while maintaining the richness of the data.

Real-Time Applications and Feedback: In real-time applications such as recommendation systems, personalized content delivery, and adaptive learning platforms, the

ability to process and categorize RDF data quickly is crucial. Clustering provides an effective way to group new information dynamically, ensuring that systems can respond in real-time to changing data. As RDF datasets continue to evolve, clustering allows for continuous adaptation and real-time feedback, keeping systems up to date with the latest information. This dynamic capability is essential for maintaining the accuracy and relevance of RDF-based applications in rapidly changing environments.

2.3 Background of Knowledge Graph

A Knowledge Graph (KG) is a structured and semantic representation of information that organizes data into interconnected entities and relationships, forming a graph-based model. It enables machines to understand, retrieve, and infer knowledge from vast amounts of data by linking concepts, facts, and contextual information in a meaningful way. Unlike traditional relational databases that store data in tabular form, Knowledge Graphs structure information as nodes (representing entities) and edges (representing relationships), allowing for flexible and efficient querying, reasoning, and knowledge discovery.

At the core of a Knowledge Graph is the idea of representing real-world entities such as people, places, organizations, and objects, along with the relationships that connect them. Each entity is described by attributes or properties that provide additional information, such as a person's name, birthdate, or location. These relationships are crucial for understanding the connections between different pieces of information, enabling advanced applications such as semantic search, recommendation systems, and intelligent decision-making.:

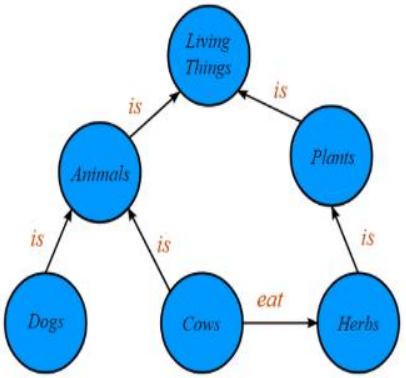

Figure 4: Knowledge Graph

Knowledge Graphs have become essential in various domains, including artificial intelligence (AI), natural language processing (NLP), and search engines. For example, Google's Knowledge Graph enhances search results by providing direct answers and contextual information instead of simple keyword-based matches. Similarly, e-commerce platforms use Knowledge Graphs to improve product recommendations by analyzing customer preferences and purchase history.

One of the key advantages of Knowledge Graphs is their ability to integrate heterogeneous data from multiple sources while preserving the relationships between entities. They use semantic web technologies such as the Resource Description Framework (RDF) and Web Ontology Language (OWL) to ensure interoperability and maintain a standardized representation of knowledge. By leveraging ontologies and taxonomies, Knowledge Graphs help in organizing complex data structures and facilitating automated reasoning.

2.4 Importance of Knowledge Graph

A Knowledge Graphs (KGs) have become a cornerstone of modern data management, playing a pivotal role in a wide range of applications from artificial intelligence (AI) to data integration and semantic search. Their ability to represent and manage vast amounts of interconnected data in a structured and semantically rich manner has revolutionized the way information is organized, accessed, and utilized. The importance of Knowledge Graphs can be highlighted through several key aspects:

Enhanced Data Integration and Interoperability: One of the primary advantages of Knowledge Graphs is their ability to integrate and interconnect data from disparate sources, regardless of format or structure. In today's data-driven world, information is often scattered across multiple platforms and domains, making it difficult to combine or compare. Knowledge Graphs provide a unified, graph-based representation that connects related data, enabling more efficient integration of heterogeneous datasets[14]. This capability is particularly important in fields like healthcare, finance, and e-commerce, where data originates from various sources such as databases, documents, and web pages.

Improved Search and Discovery: Knowledge Graphs significantly enhance the capabilities of search engines and recommendation systems by enabling systems to understand the context and relationships between entities. Unlike traditional keyword-based search, which relies solely on matching terms, Knowledge Graphs facilitate semantic search—allowing systems to understand the underlying meaning behind a query. This results in more relevant and personalized search results, as the system is able to recognize relationships between entities and present information in a way that aligns with user intent. This is particularly important in applications such as web search, content recommendations, and virtual assistants, where understanding user queries is essential for providing accurate results.

Rich Contextual Understanding: The ability to capture the rich, contextual relationships between entities is one of the defining features of Knowledge Graphs. By organizing data as interconnected nodes and edges, KGs provide a more comprehensive and intuitive representation of knowledge. For instance, in a Knowledge Graph representing a company's product catalog, relationships such as "brand," "category," and "price" can be explicitly encoded, allowing for advanced reasoning and querying. This deeper understanding of context enables systems to perform more complex tasks, such as answering detailed questions, inferring missing information, and supporting decision-making processes.

Support for AI and Machine Learning: Knowledge Graphs serve as an essential resource for AI and machine learning models by providing structured, semantic data that machines can process. Many AI applications, such as natural language processing (NLP), computer vision, and recommendation systems, benefit from the structured nature of KGs, which offer a solid foundation for training models. Furthermore, KGs can be used to improve the performance of machine learning algorithms by providing additional context and knowledge about the relationships between entities. For example, in NLP, Knowledge Graphs can assist in understanding word meanings based on their relationships to other words, thereby improving tasks such as sentiment analysis, entity recognition, and language generation.

Enabling Knowledge-Based Reasoning and Inference: One of the significant advantages of Knowledge Graphs is their ability to perform reasoning and inference. By encoding relationships and rules within the graph, KGs enable systems to infer new knowledge from existing data. For example, if a Knowledge Graph contains information about people, places, and their connections, it can infer new relationships based on logical reasoning. This ability to perform inference makes KGs particularly valuable in areas such as expert systems,

automated decision-making, and knowledge discovery, where it is crucial to generate new insights from existing knowledge.

Personalized User Experiences: Knowledge Graphs are instrumental in enabling personalized experiences across various domains. By capturing the preferences, behaviors, and interactions of users, KGs allow systems to offer highly personalized recommendations. For example, in e-commerce platforms, a Knowledge Graph can recommend products based on a user's past behavior, social connections, and contextual information, such as location or time of year. Similarly, in content platforms, Knowledge Graphs can help tailor content recommendations, improving user engagement and satisfaction. This personalized experience is crucial for businesses seeking to enhance customer experience and drive user retention.

Facilitating Data Quality and Consistency: Knowledge Graphs also help improve the quality and consistency of data by enforcing a structured approach to data representation. By utilizing ontologies and controlled vocabularies, KGs ensure that data is consistent across various datasets and sources. This consistency is especially important when integrating data from multiple sources, as it reduces ambiguity and ensures that related entities are treated uniformly. In fields like healthcare, where data consistency is critical, Knowledge Graphs help ensure that medical concepts are accurately represented and linked, contributing to better data governance and quality.

Real-Time Insights and Decision Support: Knowledge Graphs are crucial for providing real-time insights and supporting decision-making processes. Their ability to dynamically update and integrate new data allows for continuous enrichment of the graph, ensuring that decision-makers have access to the most current and relevant information. In fast-paced industries like finance or e-commerce, where timely decisions are essential, Knowledge Graphs help provide real-time, actionable insights that can drive business strategies and outcomes.

2.4 Literature Review

The With the rapid expansion of RDF data, numerous studies have explored various embedding techniques and clustering methodologies to enhance RDF data analysis. This section reviews existing approaches, highlighting their methodologies, contributions, and limitations, and discusses recent advancements in the field.

Several studies have investigated the application of graph-based embeddings for RDF data clustering. Ristoski and Paulheim (2016)[1] introduced RDF2Vec, which generates embeddings using graph walks and Word2Vec. Their approach demonstrated an accuracy of up to 82.5% in machine learning tasks, with Skip-gram outperforming CBOW. However, the reliance on graph walks led to structural loss, and the Weisfeiler-Lehman Kernel proved computationally expensive. To overcome this, Eddamiri et al. (2018)[2] extended the concept by applying Word2Vec and Doc2Vec for RDF clustering using K-means, achieving 79.67% purity on AIFB and 86.99% purity on BGS datasets. While their work pioneered Word2Vec-based RDF clustering, its dataset scope was limited, and computational demands remained high.

Furthering these efforts, Eddamiri et al. (2019)[3] proposed a hierarchical RDF clustering technique leveraging TF score similarity and Weisfeiler-Lehman Kernel, achieving 84.2% accuracy. However, the model's dependency on candidate descriptions increased computational cost, limiting its applicability to large-scale RDF datasets. Other research efforts have explored alternative graph-based clustering methods. Giannini (2013)[4] employed community detection algorithms for RDF clustering on synthetic datasets, achieving 78.5% accuracy. Their work effectively exploited RDF's network structure but lacked validation on real-world datasets, making generalizability a concern.

Qi et al. (2013)[5] implemented k-means and MROC clustering using SPARQL update queries to minimize dataset downloads. The study revealed that k-means offered low communication costs, while MROC proved inefficient due to sequential processing. Despite its novelty, their work was tested on only one dataset, limiting its broader applicability. Similarly, recent advancements have sought to refine RDF clustering pipelines to address these limitations.

Edamiri et al. (2020)[6] introduced an RDF graph mining pipeline integrating Word2Vec, Doc2Vec, TF-IDF, and K-means, achieving 97% purity and over 90% F-measure. Their approach improved clustering and theme identification, though overlapping clusters due to missing rdf:type annotations posed challenges. Another study by Edamiri et al. (2018)[7] examined instance extraction using graph walks and Word2Vec, demonstrating that the Set of Walks approach yielded the best results. However, optimizing instance extraction methods and distance metrics remains an open problem, requiring further refinement for practical applications.

Deep learning techniques have also been employed for RDF data analysis. Pramanik et al. (2024)[8] developed a context graph-based approach using fine-tuned BERT and Group Steiner Trees (GST) for answer extraction. Their model outperformed traditional QA systems in accuracy and interpretability but struggled with ambiguity and lacked granular answers. Similarly, Sasi et al. (2022)[9] explored RDF triplestore representation, analyzing SPARQL query patterns and inefficiencies in RDF stores. Their work provided a comprehensive RDF design space but omitted certain query optimization cases, leaving room for future exploration.

In addition to supervised learning approaches, unsupervised deep learning models have been applied to RDF clustering. Hu et al. (2021)[10] introduced a graph autoencoder model for unsupervised entity clustering in RDF graphs, showing significant improvements in clustering coherence. Their approach leveraged both structural and semantic relationships, addressing some of the key limitations of traditional graph embedding techniques. However, scalability remains a major challenge for deep learning-based RDF clustering methods, particularly when dealing with large-scale knowledge graphs.

Despite these advancements, RDF data clustering continues to face challenges. Existing techniques struggle with semantic complexity, high computational costs, and scalability issues. Traditional clustering algorithms often fail to capture contextual relationships within RDF triples, while deep learning-based methods introduce computational overhead. Furthermore, evaluation metrics for RDF clustering vary widely, making it difficult to standardize performance assessment across different studies.

Our research aims to address these gaps by integrating state-of-the-art embedding techniques—RDF2Vec, Word2Vec, and BERT—with clustering algorithms such as K-means, Gaussian Mixture Models (GMM), and Agglomerative Clustering. By evaluating performance

using ARI, NMI, and Silhouette Score, we seek to refine RDF data clustering and improve knowledge discovery within structured datasets. Additionally, we propose the incorporation of hybrid clustering techniques that combine graph-based embeddings with deep learning models, leveraging self-supervised learning to enhance feature extraction from RDF graphs. This approach aims to improve semantic understanding while mitigating computational costs, paving the way for more efficient and scalable RDF clustering methodologies.

3.2 Limitations

Several limitations exist in the related works on RDF data clustering, which can hinder the development of more robust and scalable approaches.

Petar Ristoski and Heiko Paulheim (2016) [1] highlighted the issue of graph walks losing critical structural relationships within RDF data, which significantly diminishes the effectiveness of clustering when the inherent semantics of the data are not captured adequately. Additionally, the Weisfeiler-Lehman Kernel, though effective in certain applications, was found to be computationally expensive, limiting its scalability for large RDF datasets. The complexity of maintaining graph structure while balancing computational efficiency remains a challenge in this domain.

Eddamiri et al. (2018) [2] faced limitations related to the narrow scope of the datasets used, which affected the generalizability of their approach. Their use of high-dimensional embeddings resulted in substantial computational overheads, particularly when dealing with large-scale datasets. While their work presented significant contributions, it highlighted the need for methods that can handle a wider range of RDF data while being mindful of computational costs. Furthermore, this limitation was partially addressed in their follow-up work in 2019 [3], where they encountered high computational costs in hierarchical clustering. Moreover, their reliance on candidate descriptions introduced biases that could affect the purity and accuracy of clustering results, particularly in cases where the descriptions were incomplete or inconsistent.

Giannini (2013) [4] proposed community detection algorithms that were only validated on synthetic RDF datasets, which introduces uncertainty about their scalability and applicability to real-world RDF data. Although synthetic datasets can effectively demonstrate the capabilities of algorithms, they often lack the complexity and heterogeneity found in real-

world applications, thus limiting the ability to generalize findings to practical use cases.

Qi et al. (2013) [5] demonstrated that MROC clustering was inefficient due to its sequential nature, which introduced delays when processing large-scale datasets. Moreover, the incorporation of SPARQL update queries to handle RDF data added unnecessary complexity to the overall process, further hindering efficiency. Additionally, their approach was only evaluated on a single dataset, making it difficult to assess its robustness and adaptability across different types of RDF data.

Edamiri et al. (2020) [6] encountered challenges related to overlapping clusters, which were caused by missing or multiple rdf:type values. This issue suggested a need for optimization to scale their approach, particularly when dealing with large and incomplete datasets, as such missing or inconsistent type information could lead to ambiguities in cluster formation. Their work highlighted the need for methods that could handle such challenges more effectively.

In their earlier work in 2018 [7], Edamiri et al. acknowledged the necessity for further research on optimal instance extraction methods and improvements in distance metrics. As RDF datasets often contain various forms of heterogeneity, the development of more flexible and accurate distance metrics remains a crucial challenge in RDF data clustering.

Pramanik et al. (2024) [8] encountered limitations related to the granularity of the answers provided by their clustering approach, which ultimately reduced the interpretability of the clustering results. Moreover, their method's reliance on BERT for embedding introduced ambiguity and computational overhead, making it difficult to balance the accuracy and interpretability of the generated clusters with the performance of the system.

Lastly, Sasi et al. (2022) [9] focused solely on explicit query patterns, excluding certain ORDER BY cases. Although they identified inefficiencies in RDF store query performance, they did not fully resolve these issues, leaving room for further optimization. Their work revealed that RDF stores still struggle with query performance under certain conditions, especially when complex queries or large datasets are involved. There is a need for continued research to develop more effective query processing techniques that can handle complex RDF queries without sacrificing performance.

These limitations collectively underscore the need for further advancements in RDF data clustering methodologies, especially in improving computational efficiency, handling larger datasets, and addressing the biases introduced by incomplete or inconsistent RDF data.

3

Dataset Construct

3.1 Data Collection from DBpedia

In this section we involved querying the DBpedia endpoint, a structured data source that contains information extracted from Wikipedia. The DBpedia SPARQL endpoint provides a way to access this structured data using SPARQL, a powerful query language for querying RDF (Resource Description Framework) data. In this process, a SPARQL query was written to retrieve details about famous individuals, specifically focusing on attributes such as their name, birth year, occupation, country, field, and notable achievements.

- **DBpedia endpoint**

The DBpedia endpoint is a web service that allows users to query the DBpedia dataset using SPARQL. DBpedia is a large-scale, structured knowledge base derived from the content of Wikipedia. It contains structured information about various entities such as people, places, events, organizations, and more, which are extracted from Wikipedia articles and represented in RDF format.

The DBpedia SPARQL endpoint provides an interface for querying this structured data using SPARQL. Users can execute queries to retrieve specific information, such as details about famous people, historical events, geographical locations, or relationships between different entities. The endpoint makes DBpedia's knowledge base accessible in a machine-readable way, enabling users to integrate, analyze, and explore this data programmatically.

By using this endpoint, users can write and execute SPARQL queries to interact with the vast amount of data available in DBpedia. The endpoint supports operations like filtering, sorting, and joining data, and it can return results in various formats such as JSON, XML, or CSV. This makes it a valuable resource for anyone looking to work with semantic web data or build applications that require access to structured knowledge from Wikipedia.

- **SPARQL**

SPARQL (SPARQL Protocol and RDF Query Language) is a powerful query language specifically designed to query and manipulate data stored in the Resource Description Framework (RDF) format. RDF is a standard model for representing structured data on the web as triples, where each piece of data is stored as a subject-predicate-object structure. SPARQL enables users to interact with these triples by writing queries to retrieve specific data, perform filtering, and execute complex operations on RDF datasets.

SPARQL queries are structured to select variables, define conditions, and return results that meet those conditions. The language is designed to handle semantic data and is commonly used with datasets like DBpedia, Wikipedia, and other linked open data sources. With SPARQL, users can explore large datasets, retrieve information about entities (such as people, places, events), and even perform operations like joining multiple data sources, grouping results, and sorting data. This makes SPARQL an essential tool for working with linked data on the web.

3.2 Construct Entities Dataset

The process involved extracting structured data from the DBpedia endpoint. DBpedia provides a wide range of information about various entities in the form of Resource Description Framework (RDF), which can be queried using SPARQL. A SPARQL query was designed to retrieve data on famous individuals, specifically gathering key details such as their name, birth year, occupation, country, field of expertise, and notable achievements. The query was carefully constructed to ensure only English-language data was returned, filtering out non-relevant results.

Given the large volume of data required, pagination was implemented to manage the retrieval process efficiently. The query was structured to fetch 1,000 records per request, with the use of OFFSET to collect additional chunks of data. This approach enabled the collection of a total of 10,000 records, ensuring that the server's performance was not impacted while retrieving a substantial dataset. Each record included a link to the corresponding individual's DBpedia page, in addition to the key attributes extracted by the query.

After the data was fetched, it was processed and structured into a consistent format, with each individual's information organized into a table. This tabular structure allowed for

easy analysis and manipulation. The final dataset was then converted into a CSV file, making it suitable for further processing, sharing, and use in different applications. The resultant dataset provides a comprehensive list of famous individuals, complete with detailed attributes and direct links to their respective DBpedia pages.

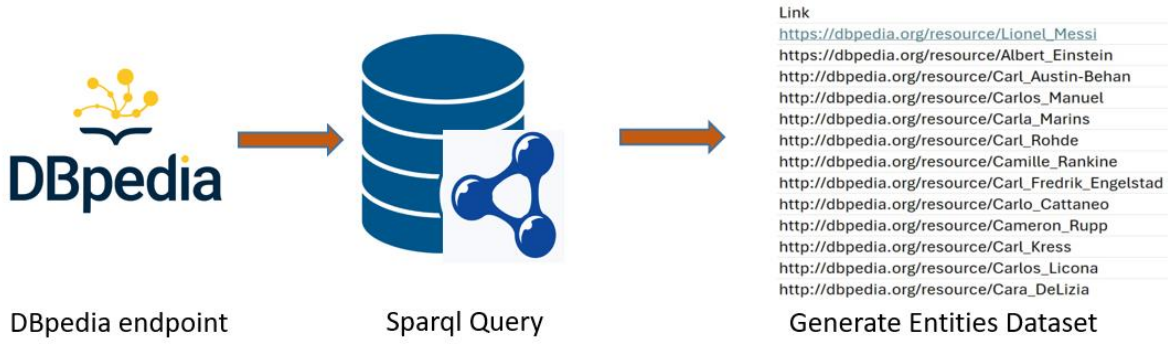

Figure 5: Custom Data Construct

4

Methodology

4.1. Overview

Our approach to entity extraction and analysis from DBpedia is designed to systematically explore and understand entity relationships using structured methodologies.

The first approach leverages the existing DBpedia knowledge graph by executing SPARQL queries to retrieve relevant entities and their relationships. Once extracted, these entities undergo a detailed analytical process involving embedding's, clustering, and visualization. Embedding's help transform relationships into numerical representations, clustering techniques identify patterns and similarities among entities, and visualization methods provide intuitive insights into the data. By utilizing DBpedia's extensive structured knowledge, this approach enables a deeper understanding of entity interactions.

The second approach focuses on constructing a custom knowledge graph by generating entities through SPARQL queries. Instead of relying solely on DBpedia's predefined structure, this method allows us to refine entity relationships and organize data according to specific analytical needs. By building a tailored knowledge graph, we gain greater flexibility in applying advanced analytical techniques, such as embedding models, clustering algorithms, and visualization tools, to extract meaningful insights from the data.

Together, these approaches provide a robust framework for entity extraction and relationship analysis, enabling a structured, data-driven exploration of knowledge networks.

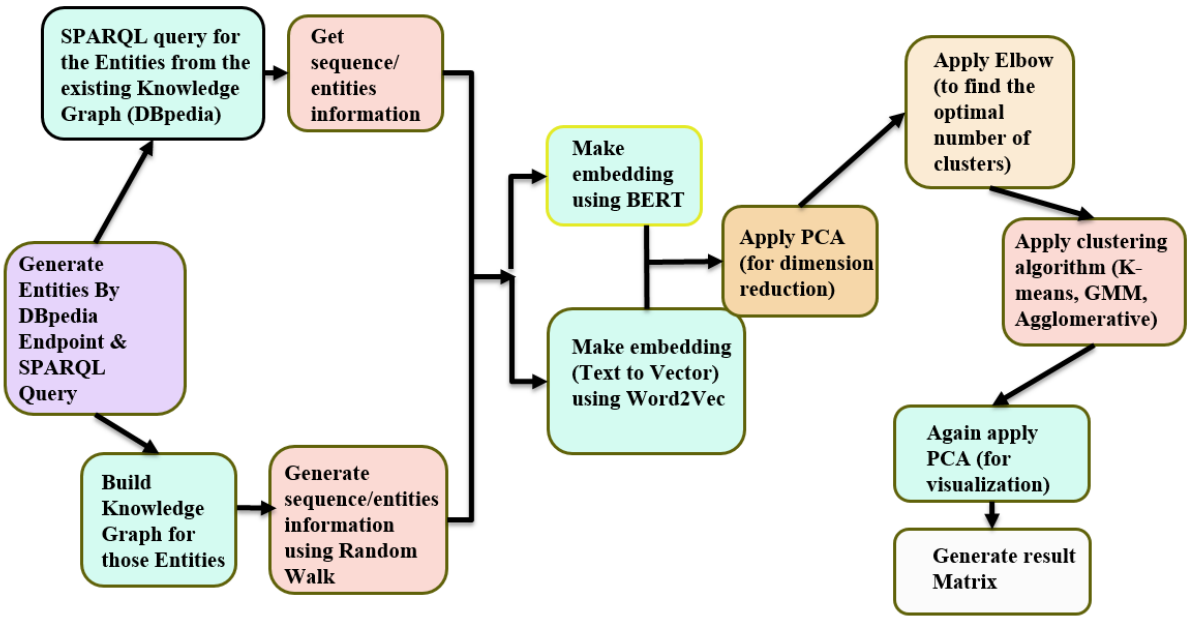

Figure 6: Overview of Methodology

Together, these approaches provide a robust framework for entity extraction and relationship analysis, enabling a structured, data-driven exploration of knowledge networks.

4.2. Data Collection

The process of creating a custom dataset involves extracting structured information from publicly available RDF datasets, such as DBpedia. This source contains rich semantic data covering a wide range of entities, including people, organizations, and locations. The main objective of this dataset creation process is to structure the extracted information, preprocess it for machine learning applications, and generate embeddings that enable effective analysis. By leveraging knowledge graphs, relationships between entities can be visualized and structured efficiently, allowing for a more comprehensive understanding of the data.

4.2.1 Data Preprocessing

To generate the dataset, RDF data was retrieved using SPARQL queries from the DBpedia endpoint. SPARQL, a query language for RDF databases, enables the extraction of triplets in the form of (subject, predicate, object) relationships. The retrieved data consists of structured links pointing to entities such as person, locations, and organizations. Once the data was collected, preprocessing was performed to enhance data quality. This included:

Filtering Noise: Irrelevant or redundant entities were removed to ensure the dataset contained only meaningful information.

Normalizing Relationships: Entity relationships were standardized to maintain consistency across different data sources.

Generating Embedding's: To convert textual and relational data into numerical representations, three embedding techniques were used:

RDF2Vec: A graph-based embedding method designed for RDF knowledge graphs.

Word2Vec: A word embedding technique used for textual information associated with entities.

BERT: A contextualized language model used for semantic representation of entity descriptions.

Following that entitites dataset was generated. This dataset consists of multiple entity links retrieved from DBpedia, each representing a unique entity with associated metadata. The links serve as identifiers for accessing further information about the entities. The extracted data is stored in a structured format, forming the basis for the knowledge graph construction.

4.2.2 Entities Information Extraction

A knowledge graph was built using the generated dataset, where nodes represent different entities (e.g., people, places, organizations). Edges define relationships between these entities (e.g., "Lionel Messi → Plays For → FC Barcelona"). The knowledge graph provides a structured representation of real-world relationships, allowing for efficient querying and retrieval of interconnected information. By leveraging knowledge graphs, complex entity interactions can be visualized, enabling deeper insights into relationships between various entities.

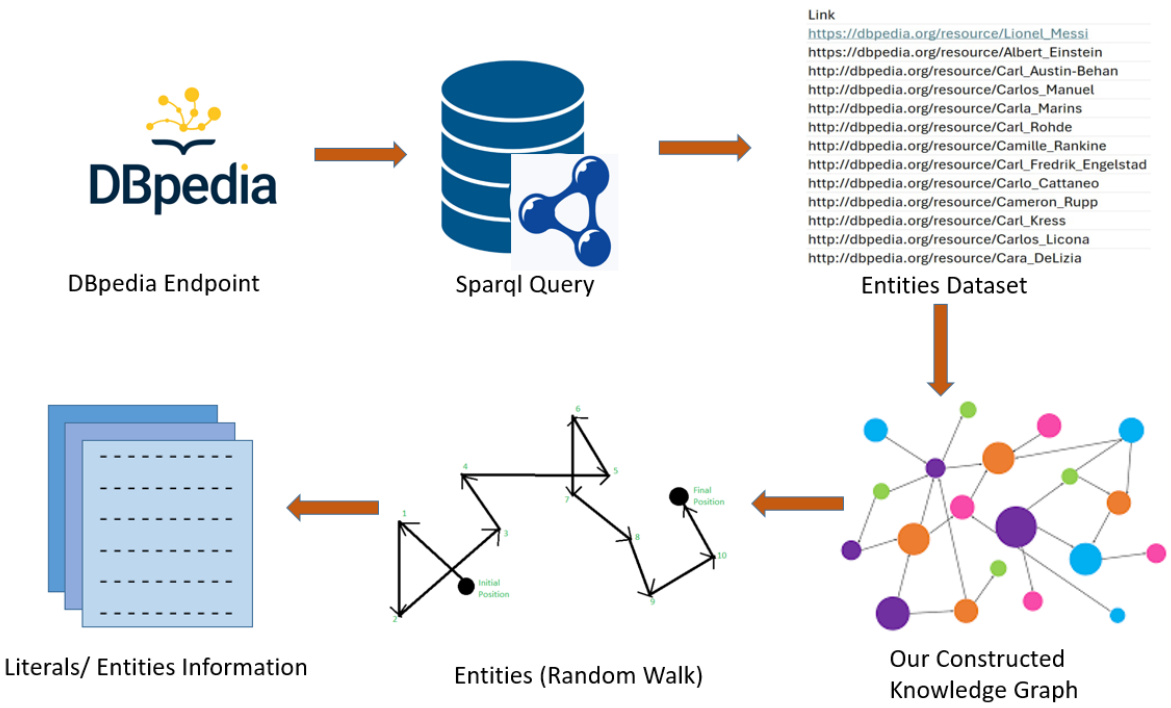

Figure 7: Entities Information Extraction

Once the knowledge graph was constructed, entity attributes were extracted to build a structured database of literal information. Each entity link was resolved to obtain detailed attributes, including[Type (e.g., Person, Organization), Height, Date of Birth, Birthplace, Profession, Associated Teams or Organizations]

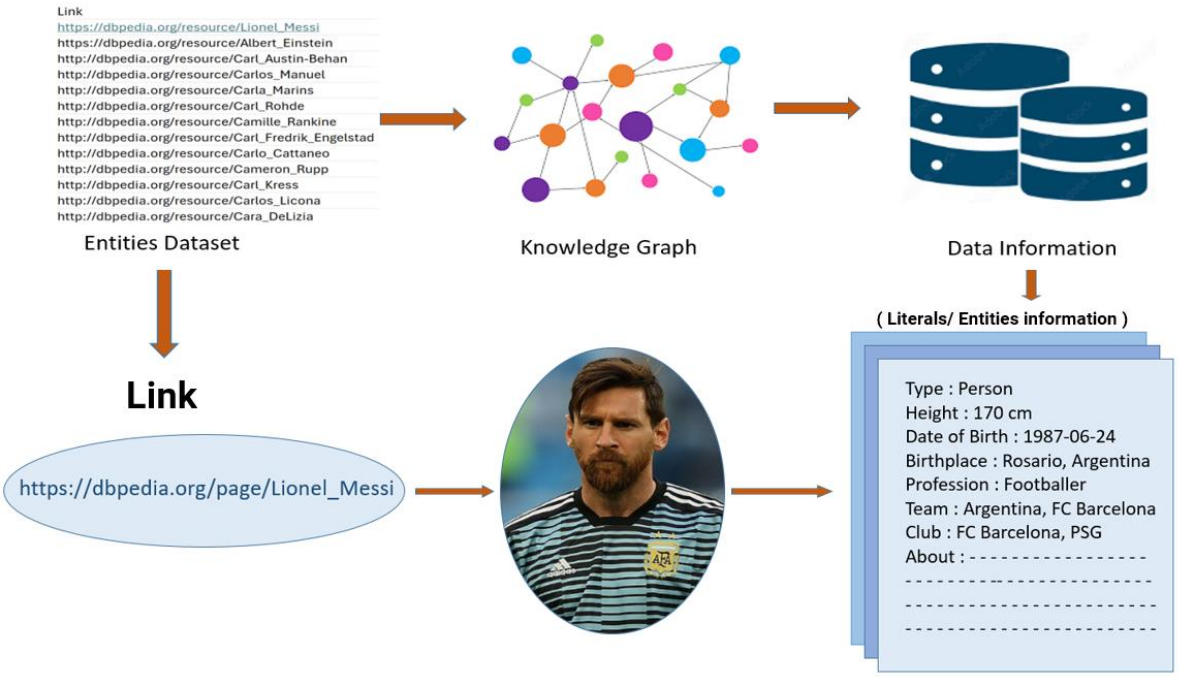

Figure 8: Information Generate from Link

For example, in the case of Lionel Messi, the extracted attributes include [Type: Person, Height: 170 cm, Date of Birth: 1987-06-24, Birthplace: Rosario, Argentina, Profession: Footballer, Teams: Argentina, FC Barcelona, Clubs: FC Barcelona, PSG]

To ensure accessibility and verification, the dataset was linked to real-world knowledge sources. Each entity entry contains a direct link to its DBpedia page, allowing users to explore further information beyond the extracted attributes. This enables seamless integration with external knowledge bases and facilitates machine-readable access to detailed entity descriptions.

4.2.3 Random Walk for Graph-Based Feature Learning

A random walk is a stochastic process that describes a sequence of steps in which each move is determined probabilistically, making the path unpredictable. It is widely used in various fields such as physics, finance, computer science, and artificial intelligence. Random walks can occur in different spaces, including number lines, grids, and graphs, with common types including simple random walks, where each step has an equal probability, and biased random walks, where certain directions are more likely than others.

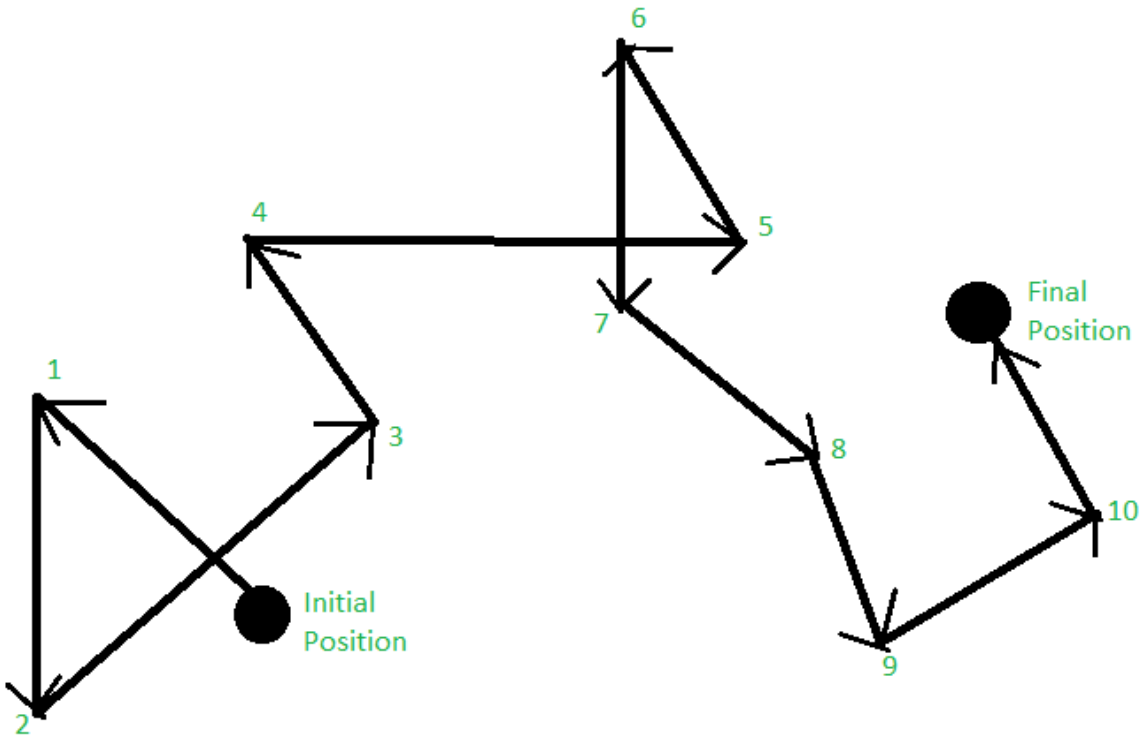

Figure 9: Random Walk

In graph theory, random walks play a crucial role in algorithms like PageRank, which ranks web pages, and RDF2Vec, which generates knowledge graph embeddings. They are also applied in finance to model stock price movements, physics to study diffusion processes, biology to analyze animal movement patterns, and AI for reinforcement learning and network analysis. By capturing uncertainty and randomness, random walks provide valuable insights into complex systems, enabling their use in a wide range of computational and analytical applications.

4.2.4 Entities Information / Literals

In the context of data representation, knowledge graphs, and databases, literals are fundamental data values that store descriptive information about entities. Unlike entities, which represent real-world objects (e.g., a person, location, or organization), literals are concrete values such as numbers, dates, strings, or Boolean values that provide additional details about these entities.

For example, in a knowledge graph, an entity like Lionel Messi would be linked to several literals, such as "170 cm" (height), "1987-06-24" (date of birth), and "Rosario, Argentina" (birthplace). These literals help provide structured and machine-readable information that enhances data retrieval and knowledge inference.

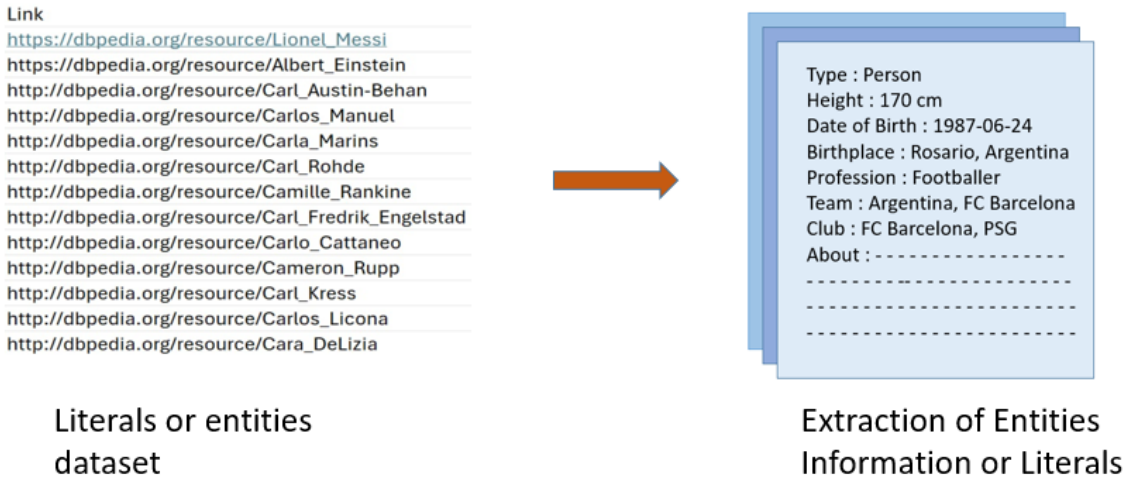

Figure 10: Literals Generate from Link

Literals are typically stored as datatype properties in RDF (Resource Description Framework) models, meaning they have specific data types such as xsd:string, xsd:integer, xsd:dateTime, etc. They are essential for semantic web applications, machine learning models, and knowledge-based systems, as they allow efficient data querying, filtering, and reasoning.

4.3. Embedding

An embedding technique is a method used in machine learning to convert high-dimensional data, such as words, images, or objects, into lower-dimensional numerical representations (vectors). These embeddings capture the essential features and relationships of the input data, allowing similar objects to be positioned closer together in the embedded space. This helps models understand patterns and similarities more effectively. Embeddings are widely used in natural language processing, recommendation systems, and image recognition to improve performance and enable meaningful comparisons between different data points.

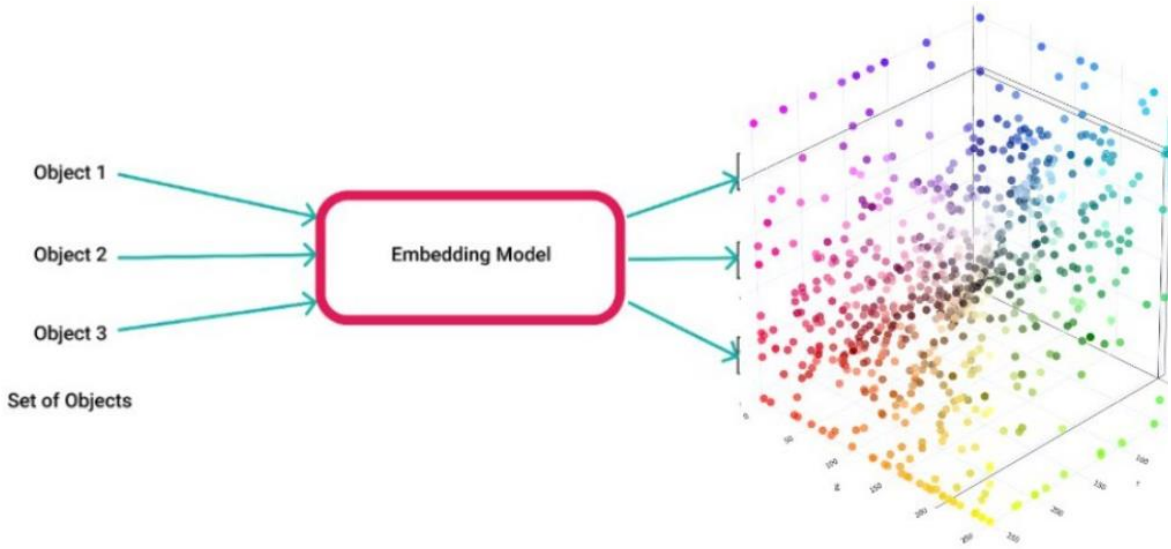

Figure 11: Embedding Process

For example, in the case of knowledge graphs, embedding techniques help convert structured data (entities and relationships) into vector representations that can be used for tasks such as link prediction, entity classification, and clustering.

4.3.1 RDF2Vec: Knowledge Graph Embedding

RDF2Vec is an embedding technique that transforms entities from RDF (Resource Description Framework) graphs into numerical vector representations. RDF graphs store structured information in the form of triples (subject-predicate-object), and RDF2Vec helps convert this symbolic data into a format that machine learning models can process.

- **How It Works:**

Graph Walks: RDF2Vec performs random walks or predefined graph traversals on the RDF graph to create sequences of entities, similar to sentences in natural language.

Embedding Training: These sequences are then used to train a Word2Vec model, where entities are treated like words, and the context of each entity is learned based on its neighbors in the graph.

Vector Representation: The model generates dense vector embeddings that capture both the structure and semantic relationships within the RDF graph.

This allows machine learning algorithms to use graph-based knowledge for tasks like classification, clustering, and link prediction.

In our case, RDF2Vec is a method for generating embeddings from RDF (Resource Description Framework) knowledge graphs. RDF2Vec converts entities and their relationships into vector representations using a process inspired by Word2Vec, a popular NLP model. Instead of processing words from text data, RDF2Vec applies a random walk on the knowledge graph to generate sequences of entities, treating them similarly to word sequences in NLP. These sequences are then used to train embedding models, capturing the structure and semantic meaning of the graph.

4.3.2 Two Approaches in RDF2Vec

- **Word2Vec**

The Word2Vec model, introduced by Google, is an unsupervised learning algorithm that learns vector representations of words based on their context in a corpus. In RDF2Vec, Word2Vec is applied to entity sequences extracted from random walks over the knowledge graph.

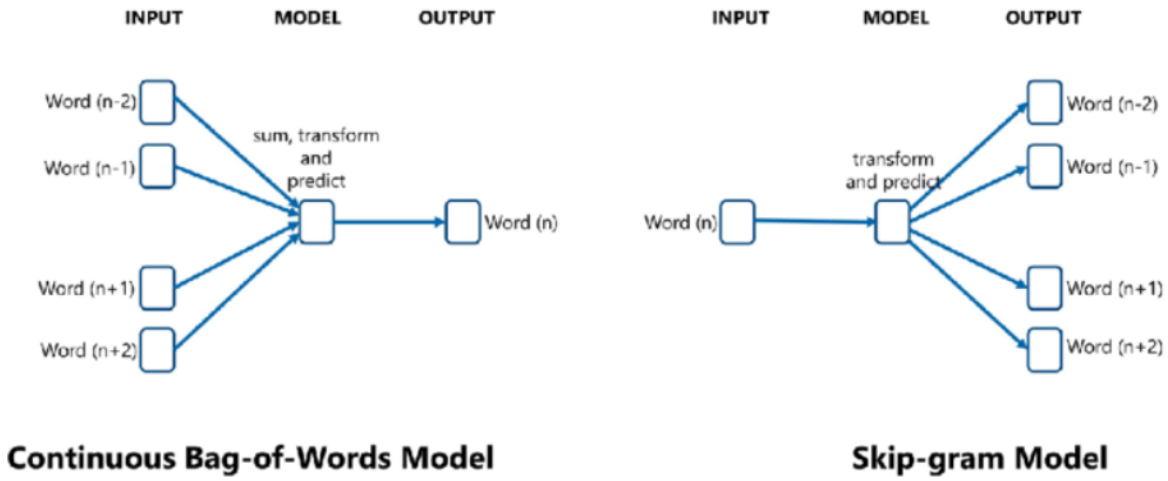

Figure 12: CBOW Model and Skip-gram Model

Two main techniques are used in Word2Vec:

CBOW (Continuous Bag of Words): Predicts a word (or entity) given its surrounding context.

Skip-gram: Predicts surrounding words (or entities) given a target word.

When applied to RDF graphs, Word2Vec learns embeddings where entities with similar semantic relationships are placed closer in the vector space.

- **Bert**

BERT (Bidirectional Encoder Representations from Transformers) is a deep learning model developed by Google that learns contextual representations of words using transformers. Unlike Word2Vec, which considers only local context, BERT processes entire sentences bidirectionally, capturing complex semantic and syntactic relationships. When applied to RDF2Vec, BERT enhances entity embeddings by leveraging deep contextualized understanding, improving performance in tasks such as entity linking, clustering, and classification.

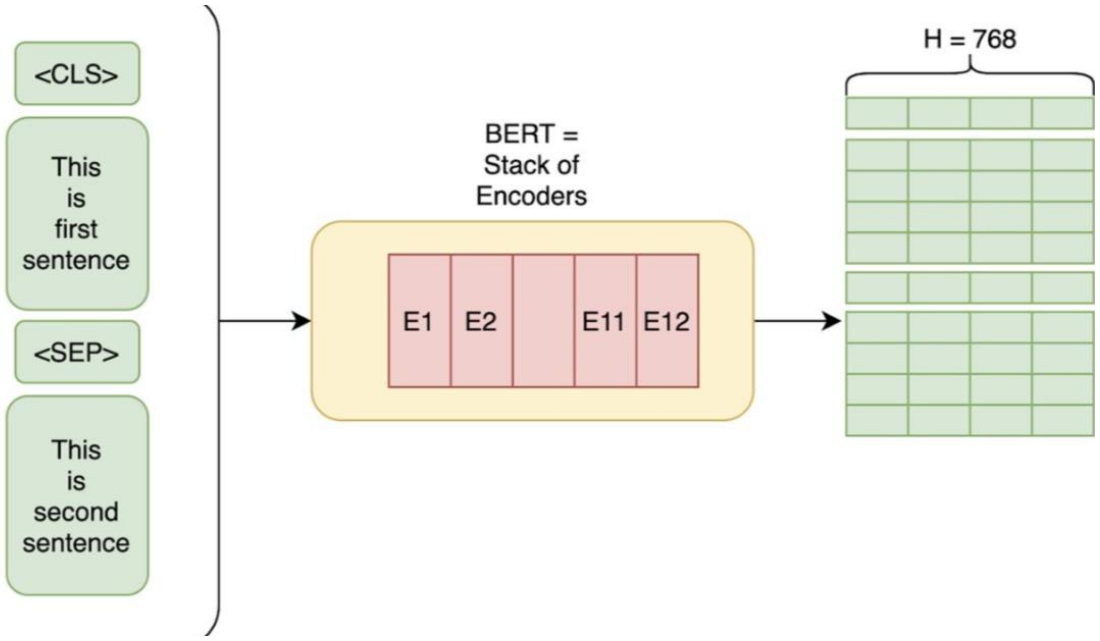

Figure 13: BERT Algorithm Approach

4.4. PCA (Principal Component Analysis)

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional space while preserving as much variance as possible. It works by computing the covariance matrix of the dataset, extracting its eigenvalues and eigenvectors, and selecting the top principal components that capture the most variance. By projecting the data onto these components, PCA reduces complexity, removes redundancy, and improves efficiency in machine learning models. This technique is particularly useful in scenarios where high-dimensional data leads to computational inefficiencies or over fitting. It is commonly applied in areas such as image processing, finance, genetics, and data visualization. PCA enhances interpretability while maintaining the essential structure of the data, making it an essential tool for preprocessing in machine learning and statistical analysis.

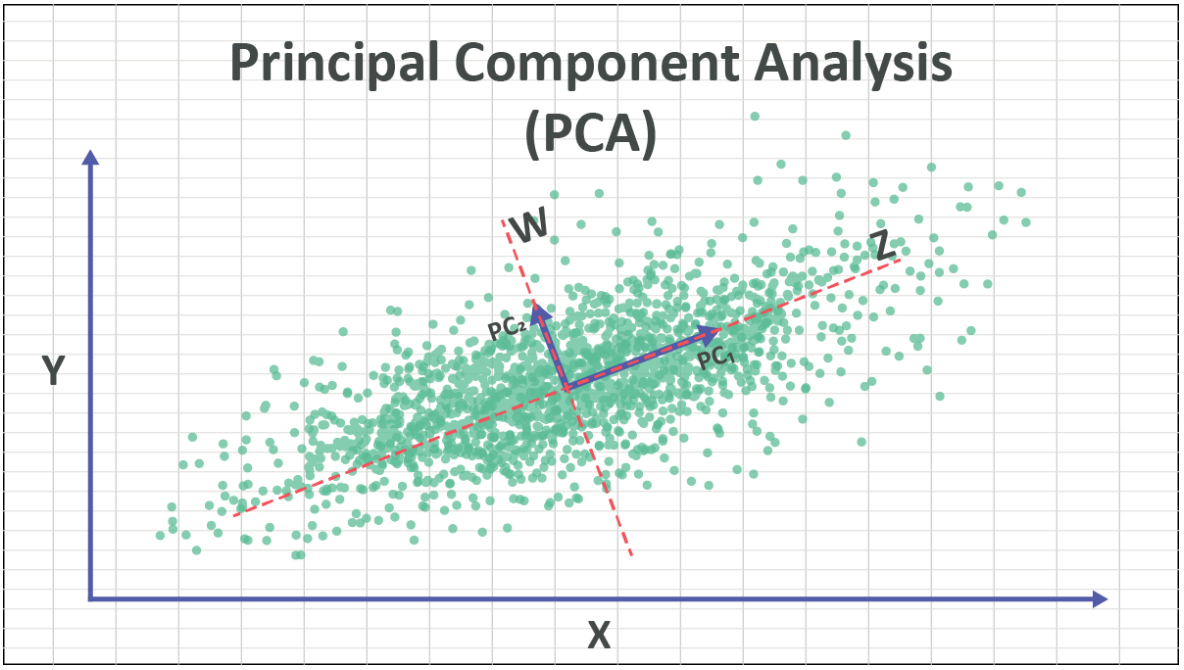

Figure 14: Principal Component Analysis

Key Reasons to Use PCA:

- Dimensionality Reduction: Reduces high-dimensional data while retaining most of the important information.
- Improves Computational Efficiency: Speeds up algorithms by simplifying data without losing accuracy.
- Eliminates Redundancy: Combines correlated features into uncorrelated components, reducing data overlap.
- Noise Reduction: Filters out less significant data, enhancing model quality.
- Better Data Visualization: Reduces data to 2D/3D for clearer visual patterns, clusters, and outliers.
- Feature Extraction: Creates new, meaningful features by combining variables to highlight important aspects.

4.5. Elbow Method

The Elbow Method is a popular technique used in K-Means clustering to determine the optimal number of clusters (K) in a dataset. It works by running the K-Means algorithm for different values of K and plotting the within-cluster sum of squares (WCSS), also known as inertia, against the number of clusters. WCSS measures the compactness of the clusters, representing the sum of squared distances between each data point and its assigned cluster center.

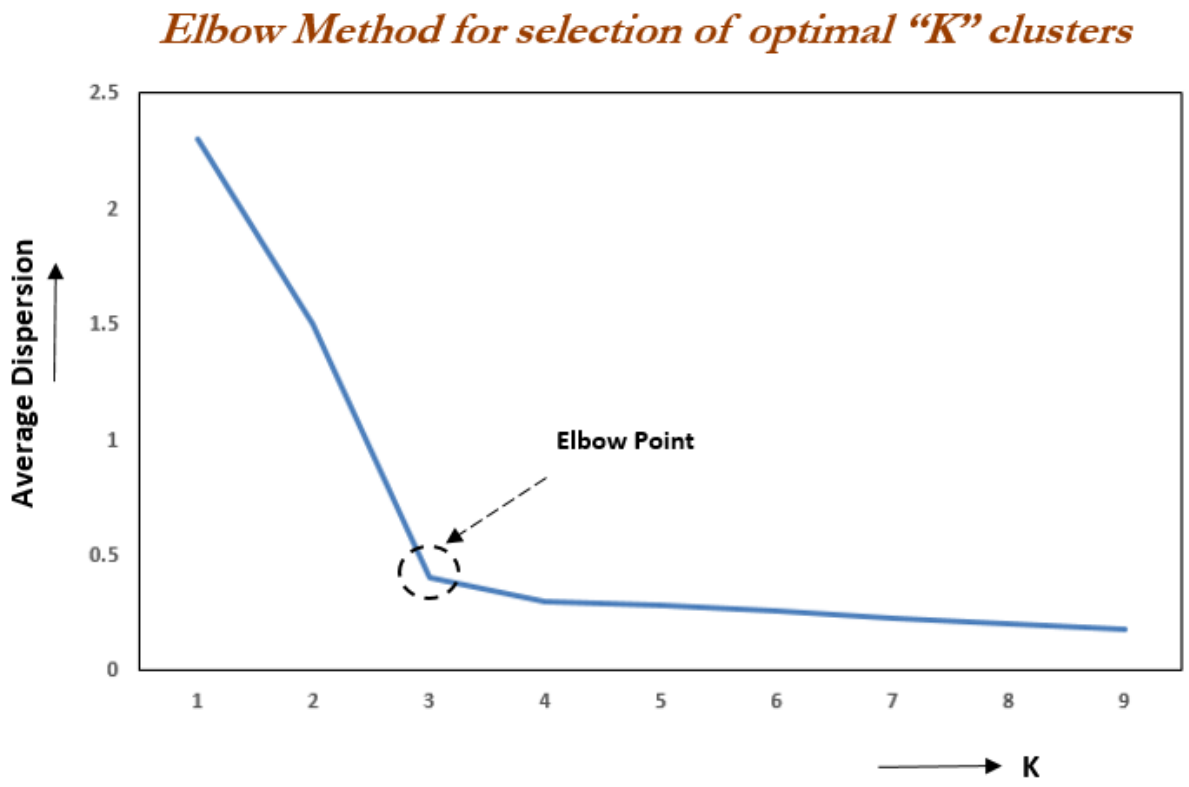

Figure 15: Elbow Method for Finding Optimal Cluster

As K increases, WCSS decreases because clusters become smaller and more compact. However, after a certain point, the rate of decrease slows down, forming an "elbow" shape in the plot. The optimal K is chosen at this elbow point, where adding more clusters no longer provides significant improvement in reducing WCSS. This helps balance cluster compactness and model efficiency, preventing both under fitting (too few clusters) and over fitting (too many clusters). The Elbow Method is widely used in unsupervised learning applications such as market segmentation, anomaly detection, and document clustering.

4.6. Clustering

Clustering is an unsupervised learning technique that groups similar data points into clusters based on their characteristics. Among various clustering methods, K-Means, Gaussian Mixture Model (GMM), and Agglomerative Clustering are widely used due to their effectiveness in different scenarios. K-Means is a partition-based algorithm that divides data into K clusters by iteratively assigning points to the nearest cluster centroid and updating centroids until convergence. It is efficient for large datasets but requires specifying K in advance and struggles with non-spherical clusters. GMM, a probabilistic model-based clustering technique, assumes that data points are generated from multiple Gaussian distributions. It assigns each point a probability of belonging to a cluster, making it more flexible than K-Means in handling overlapping clusters. However, GMM can be computationally expensive and sensitive to initialization. Agglomerative Clustering, a hierarchical approach, starts with each point as its own cluster and merges the closest clusters iteratively based on a similarity measure. Unlike K-Means, it does not require K to be predefined and provides a dendrogram for better interpretability, but it becomes computationally expensive for large datasets. Each of these methods has strengths and weaknesses, making their selection dependent on the dataset structure and clustering requirements.

4.6.1 K-means

K-means clustering is an unsupervised learning algorithm used to divide a dataset into K distinct groups or clusters, based on the similarity of data points. The key idea behind K-means is that it tries to minimize the variance within each cluster by grouping similar data points together. This is done by finding the centroids (the central points) of each cluster and assigning data points to the closest centroid.

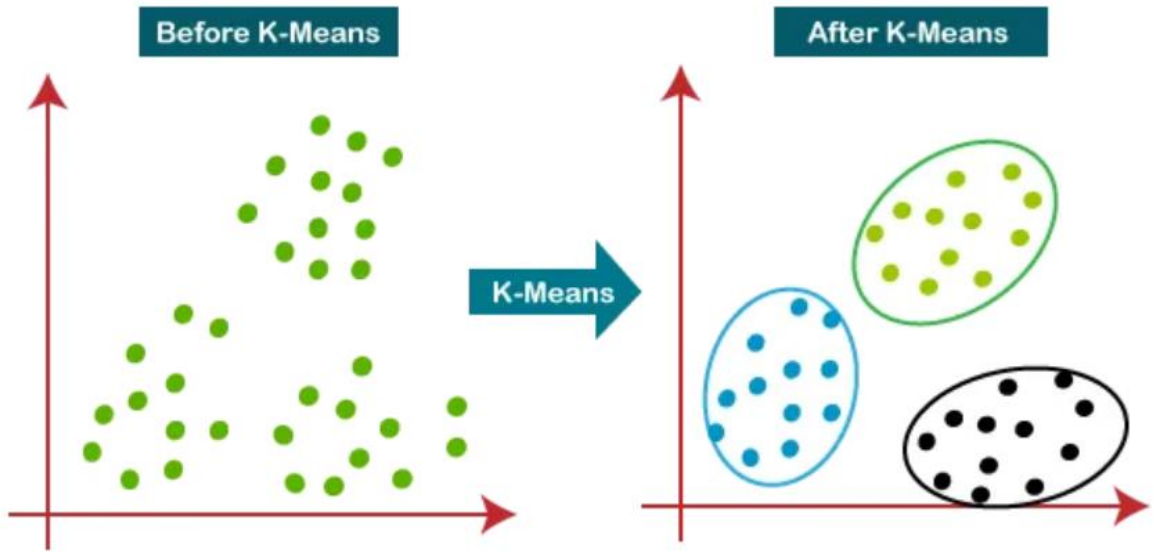

Figure 16: K-means Algorithm Approach

Why Use K-means Clustering?

K-means clustering is a widely used algorithm due to its simplicity, efficiency, and interpretability. It is simple to implement and computationally efficient, making it an excellent choice for working with large datasets. K-means also scales well with the number of data points, ensuring fast performance even with extensive datasets. Moreover, the clusters produced by K-means are often easy to interpret and analyze, which makes the results practical for decision-making.

When to Use K-means Clustering?

K-means is particularly useful when you have unlabeled data and wish to uncover underlying patterns or groupings. For example, it can be applied in customer segmentation to identify distinct customer groups based on purchasing behavior. K-means is also effective when the number of clusters (K) is either already known or can be estimated, using techniques such as the Elbow Method. It is most suited for datasets containing numerical data, particularly when features have been scaled or normalized to ensure the algorithm performs optimally.

Limitations

- The number of clusters (K) must be pre-defined, which might not always be obvious.
- Different initial centroid choices can lead to different final clusters.
- K-means assumes that clusters are spherical and roughly equal in size, which may not be true in all cases.
- K-means can be affected by outliers or noisy data points.

4.6.2 Gaussian Mixture Model (GMM)

Gaussian Mixture Model (GMM) is a probabilistic model that assumes data is generated from a mixture of multiple Gaussian distributions. Unlike K-means, which assigns each data point to one cluster, GMM assigns probabilities for each point belonging to each cluster. This allows GMM to model more complex, overlapping, and non-spherical clusters.

GMM uses the Expectation-Maximization (EM) algorithm to iteratively estimate the parameters (mean, covariance) of each Gaussian distribution and the probabilities of data points belonging to clusters.

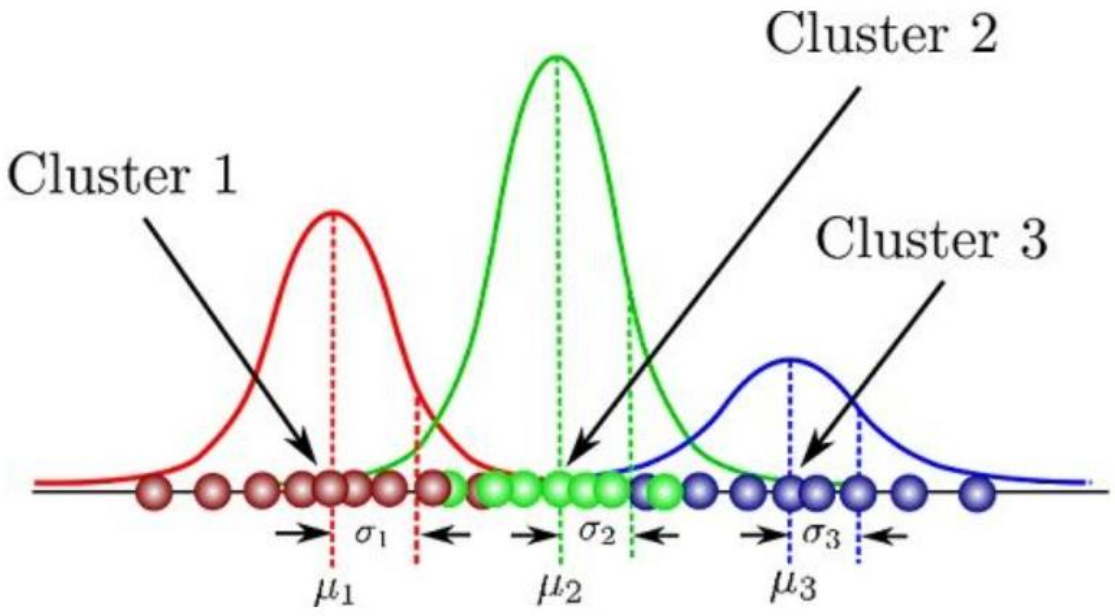

Figure 17: Gaussian Mixture Model (GMM) Algorithm Approach

Why Use Gaussian Mixture Models (GMM)?

GMM is a flexible clustering algorithm that extends K-means by modeling data as a mixture of several Gaussian distributions. Unlike K-means, GMM can capture elliptical clusters and assigns a probability to each data point for belonging to each cluster, making it more robust to noise and outliers. It also models clusters with different variances, offering a probabilistic approach to clustering.

When to Use GMM?

GMM is ideal when clusters are non-spherical or have varying shapes and sizes. It is especially useful when the number of clusters is unknown and when you need to capture uncertainty in cluster assignments. GMM is also effective for overlapping clusters and works well with numerical data, offering a probabilistic interpretation for more accurate modeling.

Limitations

- Works best when clusters follow Gaussian distribution.
- Can give different results based on initial parameters.
- More complex and slower than K-means.
- Can overfit with too many clusters.
- Needs careful selection of the number of clusters.
- Less effective for high-dimensional data due to covariance estimation challenges.

4.6.3 Agglomerative Clustering

Agglomerative Clustering is a hierarchical clustering algorithm that builds a hierarchy of clusters by iteratively merging the closest pairs of clusters. It starts with each data point as its own cluster and then merges the closest clusters step by step until all points are grouped into a single cluster or a desired number of clusters is reached. The merging process is based on a distance metric (e.g., Euclidean distance) and a linkage criterion, which determines how the distance between clusters is calculated (e.g., single, complete, or average linkage).

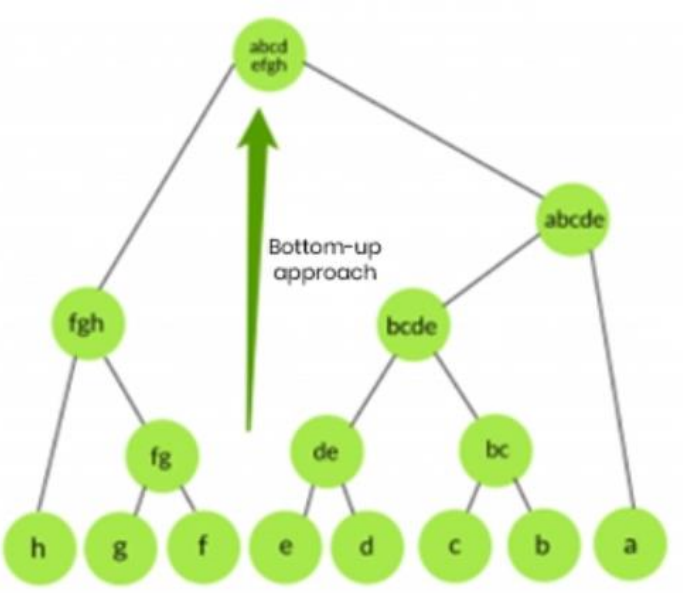

Figure 18: Agglomerative Algorithm Approach

Why Use Agglomerative Clustering?

Agglomerative Clustering is useful because it builds a hierarchical tree of clusters, allowing you to understand how data points group together at different levels. This hierarchical structure, known as a dendrogram, helps in visualizing the relationships between clusters. Unlike K-means, you don't need to predefine the number of clusters, which is beneficial when the optimal number of clusters is unknown. The flexibility in choosing distance metrics and linkage methods also makes it adaptable to various types of data and clustering needs.

When to Use Agglomerative Clustering?

Agglomerative Clustering is ideal when you don't know the number of clusters and want to explore the natural structure of your data. It is especially useful when you need to understand hierarchical relationships, such as in taxonomy or nested groupings, where the data naturally forms a tree-like structure. This algorithm is also well-suited for small to medium-sized datasets, where you need a detailed clustering structure to analyze how data points group together at different levels. Additionally, Agglomerative Clustering allows you to visualize the data at various granularities through a dendrogram, helping you decide on the most meaningful number of clusters.

Limitations

- The algorithm has a time complexity of making it slower for large datasets.
- Like many clustering algorithms, it can be sensitive to noisy data and outliers.
- Not suitable for very large datasets due to its high computational cost.
- The dendrogram can become complex and hard to interpret if there are many data points.

4.7. Approach 1: Custom Knowledge Graph Construction

In our first approach, we begin by querying the DBpedia endpoint to collect a structured dataset of entities. This dataset serves as the foundation for constructing a knowledge graph (KG), where entities are represented as nodes and their relationships as edges. Once the knowledge graph is established, we employ a random walk method to navigate through the graph, uncovering meaningful connections between entities. This exploration helps in identifying relevant entities based on their contextual relationships within the graph. Finally, we extract detailed information about these entities, enabling a deeper understanding of their attributes and associations within the knowledge graph.

4.8. Approach 2: Utilizing an Existing Knowledge Graph

In our second approach, we leverage our existing knowledge graph (KG) in DBpedia by executing SPARQL queries to retrieve structured information about entities and their relationships. Instead of constructing a new KG, we directly interact with the DBpedia knowledge base, extracting meaningful sequences of entities based on predefined query criteria. This process allows us to obtain relevant entity details, including attributes, connections, and contextual associations within the graph. By querying the existing KG, we efficiently access structured knowledge without the need for additional graph-building steps, enabling a streamlined approach to entity exploration and analysis.

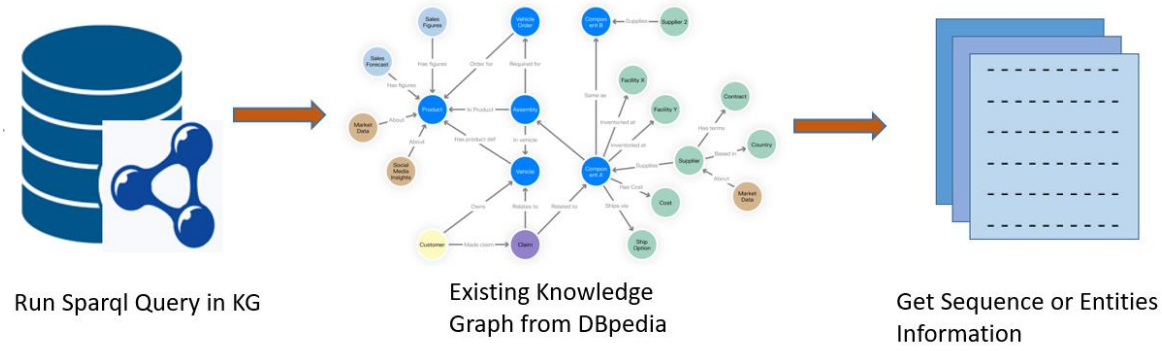

Figure 19: Generating Entities from Query

In both approaches, we retrieve entity information using different methods. However, once the entity data is extracted, the subsequent processing steps remain the same.

After retrieving entity information, we apply two key techniques for generating embedding's:

Word2Vec – This method captures semantic relationships between entities by representing them as vectors in a continuous space, where similar entities are positioned closer to each other based on contextual similarity.

BERT – This transformer-based model helps in capturing deeper contextual and relational meaning between entities, leveraging bidirectional understanding to enhance entity representations.

Once we obtain embedding's from both methods, we combine the results to generate a unified numerical representation of the entities. These embedding's serve as a structured and meaningful way to represent entities in a vector space, enabling further analysis such as clustering, classification, and visualization.

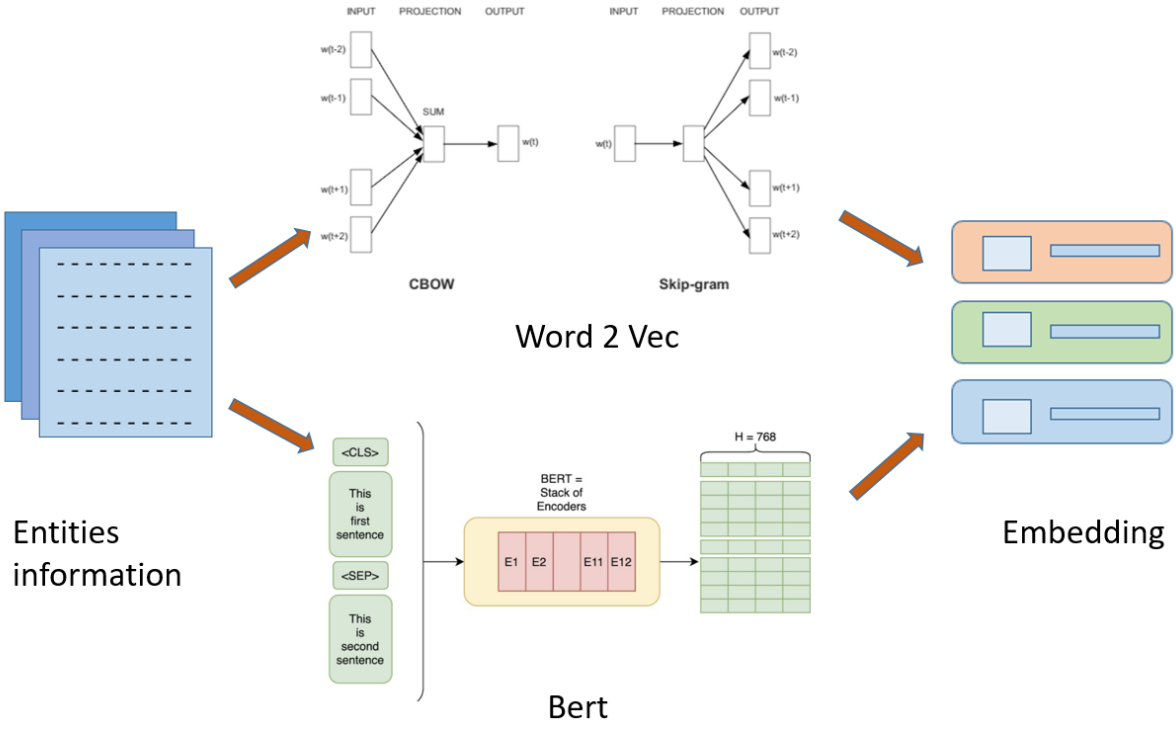

Figure 20: Creating Embedding

Once After generating the embedding's, we apply Principal Component Analysis (PCA) to reduce the dimensionality of the data while preserving its essential patterns and relationships. This step helps in improving computational efficiency and visualization while minimizing information loss.

Next, we determine the optimal number of clusters using the Elbow Method, which analyzes the variance within clusters to identify the point where adding more clusters does not significantly improve the clustering performance. This ensures that we choose a well-balanced number of clusters for meaningful grouping.

Once the optimal number of clusters is determined, we perform clustering using three different algorithms:

K-Means – A centroid-based clustering method that partitions entities into distinct groups by minimizing intra-cluster variance.

Gaussian Mixture Model (GMM) – A probabilistic approach that assumes data points belong to multiple Gaussian distributions, providing flexibility in capturing complex cluster structures.

Agglomerative Clustering – A hierarchical clustering method that iteratively merges smaller clusters into larger ones based on similarity, forming a tree-like structure for data organization.

These clustering techniques help in categorizing entities into meaningful groups, enabling deeper insights into their relationships and distributions within the knowledge graph.

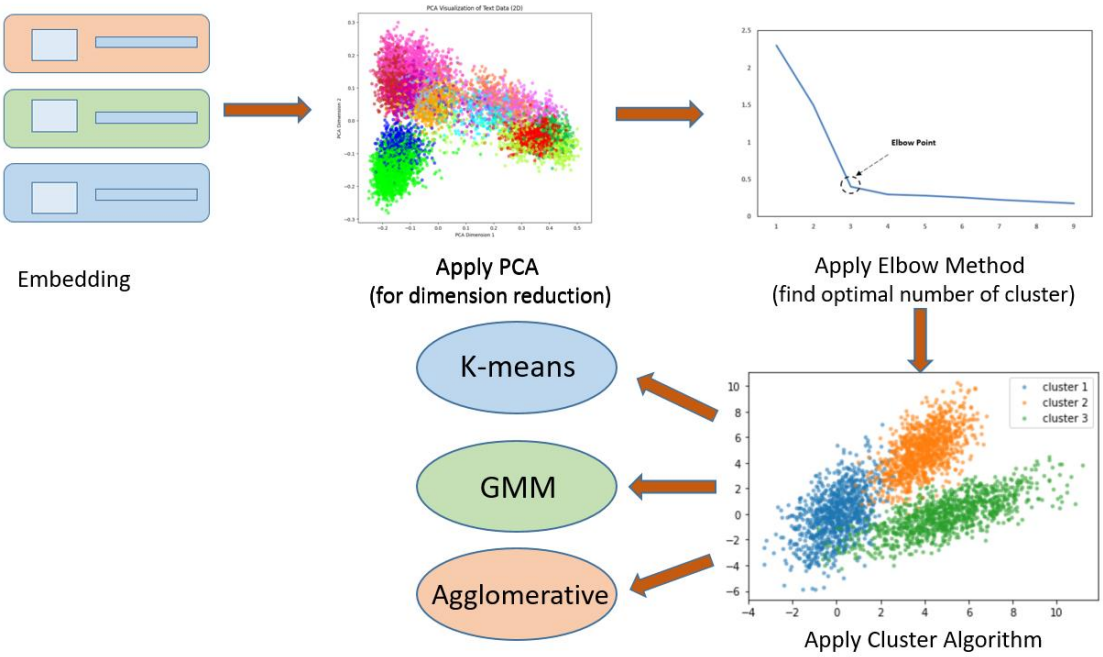

Figure 21: Apply Cluster Algorithms

After applying clustering, we use PCA again, but this time for visualization purposes. By reducing the dimensionality of the clustered data, we can represent it in a lower-dimensional space—typically 2D or 3D—making it easier to interpret and analyze. This step helps in visually distinguishing clusters and understanding their distributions within the dataset.

Finally, we perform a detailed analysis of the clustered results to extract meaningful insights. We examine the structure and coherence of the clusters, evaluate the effectiveness of the applied clustering techniques, and interpret the relationships between entities. This final step allows us to refine our understanding of the entity groupings, ensuring that the results align with our objectives. Through this structured approach, we obtain a well-organized and meaningful representation of entities, facilitating deeper exploration and knowledge discovery.

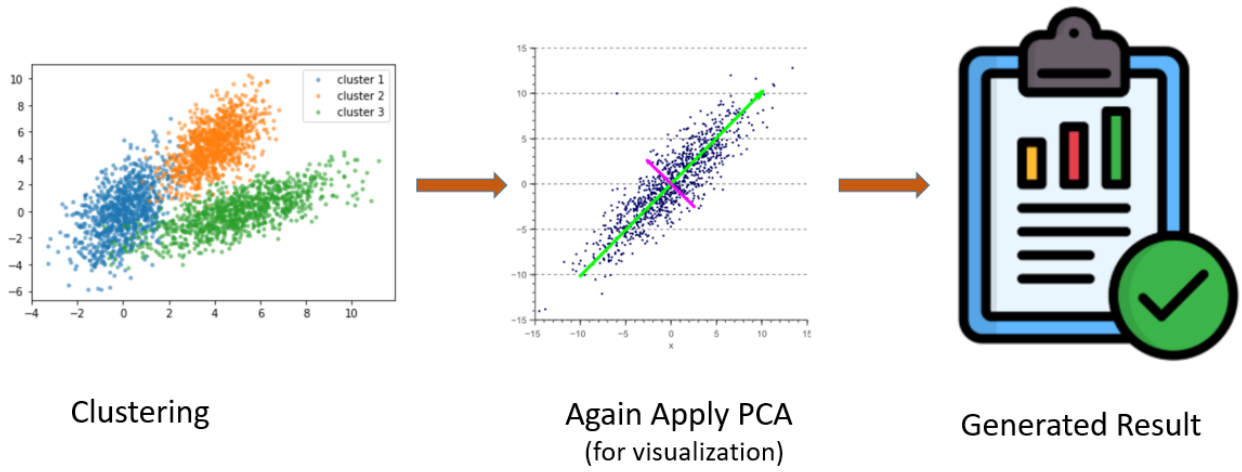

Figure 22: Generating Result from Clustering

By following the above steps, we can try two approaches to achieve our expected results

5

Result Analysis

5.1. Clustering Algorithms for Constructed Knowledge Graph (KG)

The constructed knowledge graph (KG) was built by extracting entities from DBpedia using SPARQL queries and generating entity relationships through a Random Walk technique. This approach allowed us to create a new structure of interconnected entities, which were then transformed into vector embeddings using Word2Vec and BERT. After applying Principal Component Analysis (PCA) for dimensionality reduction, we implemented three clustering algorithms—K-Means, Gaussian Mixture Model (GMM), and Agglomerative Clustering—to group similar entities.

- **Result Metrics of Clustering Algos (Constructed KG) - Word2Vec**

	ARI	NMI	AMI	FMI	Homogeneity	Completeness	V-Measure	Silhouette Score	Davies-Bouldin Index
K-Means	0.148116	0.186606	0.181359	0.458050	0.178609	0.195353	0.186606	0.445111	0.720934
GMM	0.148116	0.186606	0.181359	0.458050	0.178609	0.195353	0.186606	0.445111	0.720934
Agglomerative	0.150356	0.193200	0.187838	0.476606	0.179772	0.208795	0.193200	0.453618	0.697912

Table 1: Result Metrics of Clustering Algos (Constructed KG) - Word2Vec

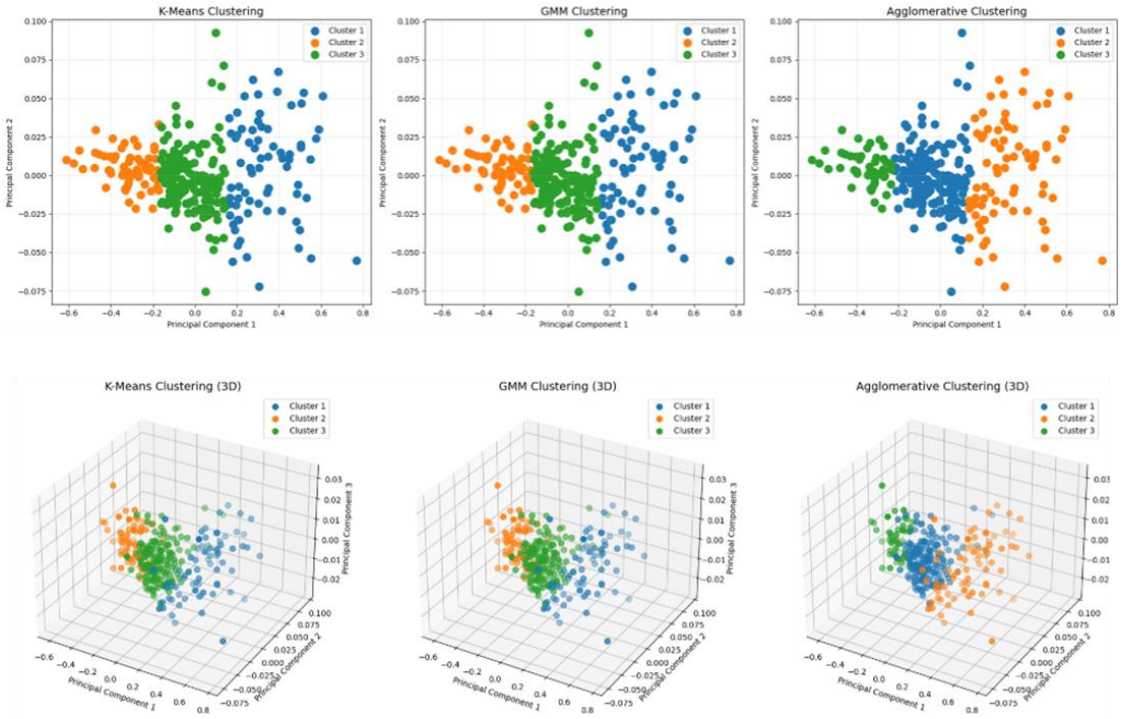

Figure 23: Visualization of Metrics of Clustering Algos (Constructed KG)- Word2Vec

- **Result Metrics of Clustering Algos (Constructed KG) – Bert**

	ARI	NMI	AMI	FMI	Homogeneity	Completeness	V-Measure	Silhouette Score	Davies-Bouldin Index
K-Means	0.660588	0.751441	0.749856	0.781736	0.727506	0.777004	0.751441	0.425399	1.126185
GMM	0.397002	0.575875	0.572891	0.656290	0.508059	0.664583	0.575875	0.425399	1.126185
Agglomerative	0.660588	0.751441	0.749856	0.781736	0.727506	0.777004	0.751441	0.425399	1.126185

Table 2: Result Metrics of Clustering Algos (Constructed KG) – Bert

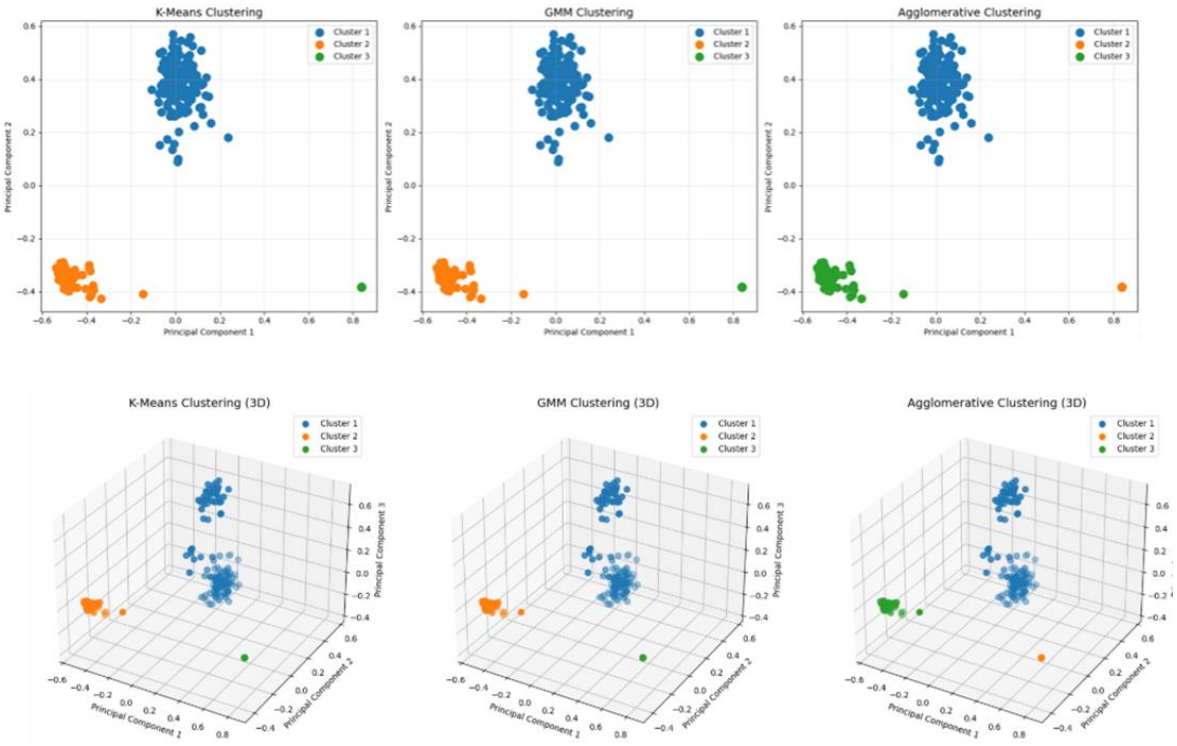

Figure 24: Visualization of Metrics of Clustering Algos (Constructed KG)- Bert

The results indicate that clustering performance in the constructed KG is less structured compared to an existing KG. BERT embedding's demonstrated better clustering quality than Word2Vec, as seen in higher Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and homogeneity scores. K-Means and Agglomerative Clustering performed comparably well, creating well-defined clusters, while GMM produced slightly less distinct groupings due to its probabilistic assignment of entities. The silhouette scores and Davies-Bouldin Index suggest that while clusters are forming, they are not as well-defined as those in the existing KG. The reason for this is the relatively unstructured nature of the newly built KG, which lacks some of the inherent semantic richness present in established KGs. However, the results still demonstrate the effectiveness of embedding-based clustering in discovering meaningful entity relationships.

5.2. Clustering Algorithms for Existing Knowledge Graph (KG)

When working with an existing KG, the structure and predefined relationships between entities significantly enhanced clustering performance. Unlike the constructed KG, the existing KG provides well-defined entity connections, allowing for more accurate embedding generation and better clustering outcomes. We applied the same set of methodologies—embedding generation using Word2Vec and BERT, dimensionality reduction via PCA, and clustering using K-Means, GMM, and Agglomerative Clustering.

- **Result Metrics of Clustering Algos (Existing KG) – BERT**

	ARI	NMI	AMI	FMI	Homogeneity	Completeness	V-Measure	Silhouette Score	Davies-Bouldin Index
K-Means	0.950943	0.939365	0.938992	0.967200	0.939009	0.939722	0.939365	0.173035	2.556783
GMM	0.950943	0.939365	0.938992	0.967200	0.939009	0.939722	0.939365	0.173035	2.556783
Agglomerative	0.970252	0.953246	0.952958	0.952958	0.953232	0.953261	0.953246	0.172293	2.585212

Table 3: Result Metrics of Clustering Algos (Existing KG) – BERT

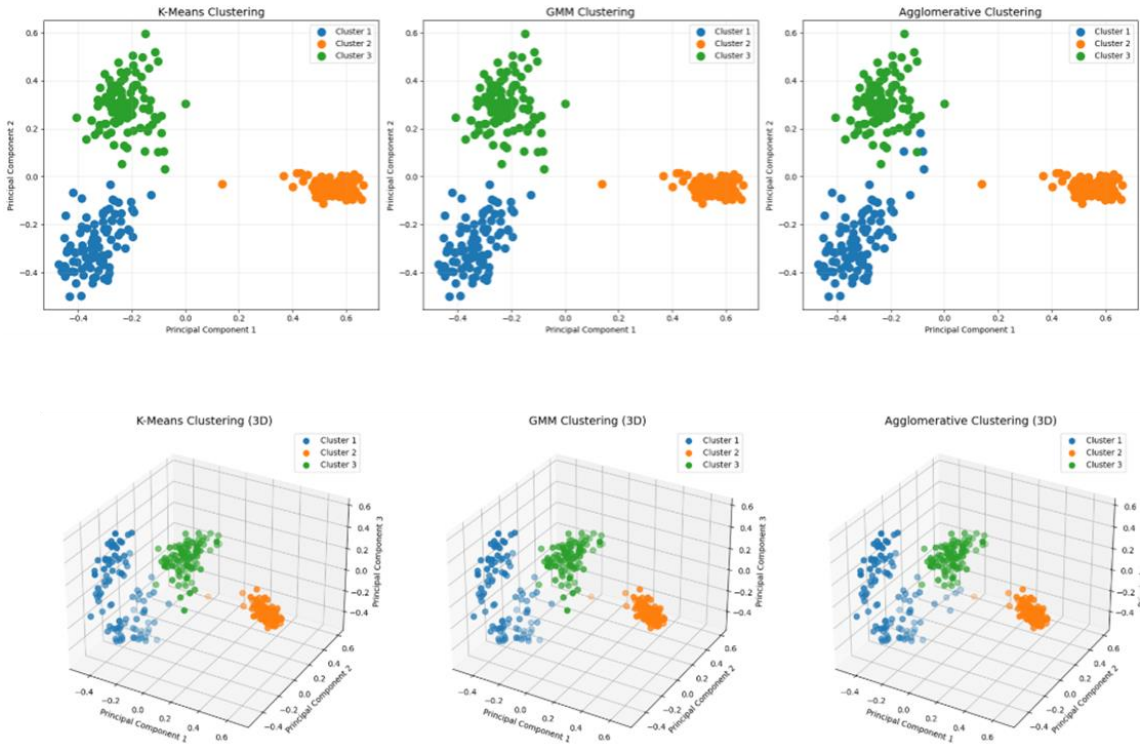

Figure 25: Visualization of Result Metrics of Clustering Algos (Existing KG) – BERT

- **Result Metrics of Clustering Algos (Existing KG) - Word2Vec**

	ARI	NMI	AMI	FMI	Homogeneity	Completeness	V-Measure	Silhouette Score	Davies-Bouldin Index
K-Means	0.549818	0.685855	0.683472	0.748802	0.582248	0.834317	0.685855	0.221246	1.499028
GMM	0.555259	0.700728	0.698130	0.762249	0.564530	0.923539	0.700728	0.200932	1.152938
Agglomerative	0.562336	0.708506	0.706268	0.757056	0.598952	0.867109	0.708506	0.220264	1.505110

Table 4: Result Metrics of Clustering Algos (Existing KG) - Word2Vec

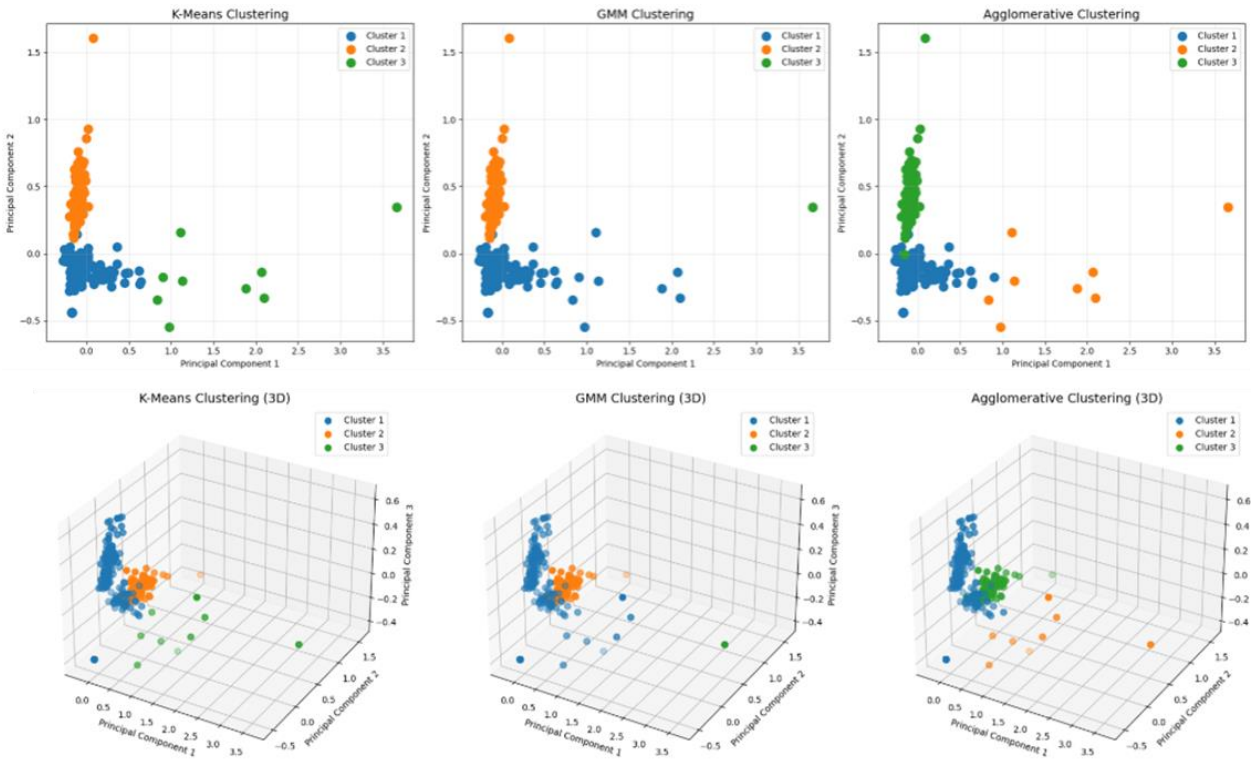

Figure 26: *Visualization of Result Metrics of Clustering Algos (Existing KG) - Word2Vec*

The results show that clustering in the existing KG achieved significantly higher accuracy compared to the constructed KG. The presence of structured semantic relationships contributed to better-defined clusters, as reflected in high homogeneity, completeness, and V-Measure scores. BERT-based embeddings again outperformed Word2Vec, demonstrating stronger entity relationship capture. Among the clustering techniques, K-Means and Agglomerative Clustering produced the most well-separated clusters, whereas GMM, though effective, showed slightly less distinct boundaries due to its probabilistic approach. The silhouette scores further confirm that the existing KG results in clearer and more meaningful clustering, highlighting the importance of structured relationships in knowledge graph analysis.

5.3 Performance Metrics Analysis

We evaluated the clustering algorithms using metrics such as Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), Adjusted Mutual Information (AMI), Fowlkes-Mallows Index (FMI), Homogeneity, Completeness, V-Measure, Silhouette Score, and Davies-Bouldin Index.

- **BERT-based embedding's (Existing KG vs. Constructed KG)**

1. For the existing KG, K-Means and GMM performed similarly, achieving high clustering performance across all metrics. Agglomerative clustering performed slightly better in terms of ARI (0.970252) and completeness (0.953261).
2. In the constructed KG, K-Means outperformed GMM significantly in ARI (0.660588 vs. 0.397002), while Agglomerative clustering yielded the same ARI as K-Means but slightly better completeness.
3. Overall, K-Means and Agglomerative clustering worked well with BERT embedding's on the constructed KG.

- **Word2Vec-based embedding's (Existing KG vs. Constructed KG)**

1. The existing KG showed moderate clustering quality, with K-Means performing slightly worse than GMM and Agglomerative Clustering.
2. The constructed KG exhibited lower clustering quality, with all three algorithms performing below expectations compared to BERT-based embedding's. Agglomerative clustering had the highest ARI (0.150356) and slightly better completeness.

5.4 2D and 3D Visualization

The visual analysis of clustering in 2D and 3D demonstrates the clustering behavior of the different algorithms:

- In the existing KG, the clusters are more clearly separated using BERT-based embedding's compared to Word2Vec.

- In the constructed KG, clusters appear more compact using BERT, whereas Word2Vec shows more overlap, leading to reduced clustering effectiveness.

5.5 Comparison between Existing KG & Constructed KG

From the comparative analysis:

- The existing KG provides well-structured relationships, leading to higher clustering performance.

- The constructed KG is more challenging to cluster due to the nature of generated links, but BERT-based embedding's still achieve reasonable results.

- K-Means and Agglomerative clustering outperform GMM in most cases, especially when dealing with constructed KG.

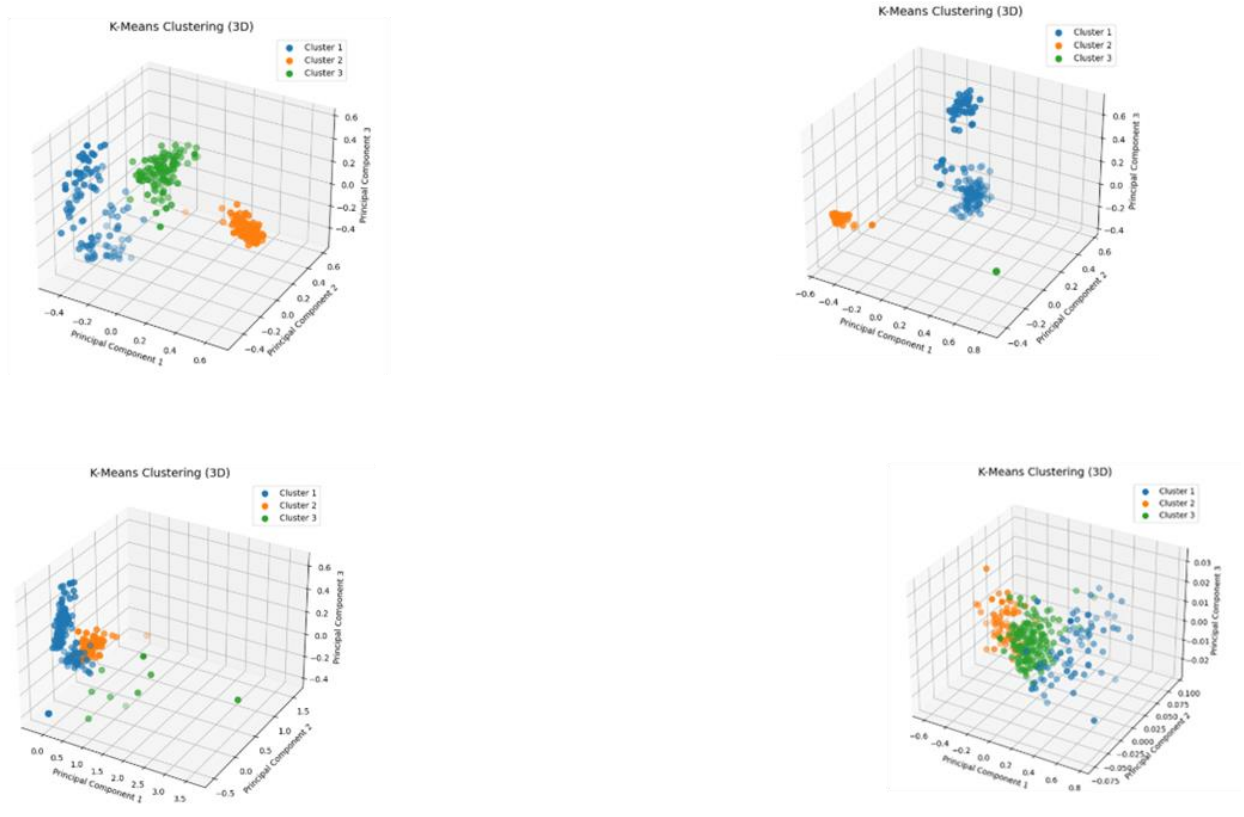

Figure 27: Estimated distance for class Building Piler and Stairs

6

Web Application

6.1. System Overview

The primary objective of this web application is to develop a recommendation system that intelligently suggests similar individuals based on clustering techniques and real-time data extraction from DBpedia. The system focuses on retrieving structured information about players, singers, or actors through SPARQL queries and presenting it in an intuitive and organized manner to enhance user interaction and experience.

The recommendation engine not only extracts comprehensive details about the searched individual but also offers relevant suggestions of other personalities who share similarities with the individual, based on clustering results. This approach enhances the exploratory aspect of the application, allowing users to discover new individuals who might interest them.

The core functionality revolves around utilizing the DBpedia endpoint, which serves as a reliable source of structured information about various public figures. By combining clustering algorithms with this rich dataset, the system efficiently identifies and groups similar entities, allowing users to explore new connections seamlessly. This combination of clustering and real-time data retrieval ensures that the application remains dynamic, adaptable, and capable of delivering meaningful recommendations.

The recommendation system leverages advanced machine learning models and knowledge graphs to provide accurate results. The interaction between the user and the system is designed to be smooth and responsive, enabling users to obtain results almost instantly. The intuitive design of the platform ensures that users can easily navigate the application, view comprehensive profiles, and explore related individuals, enhancing user engagement and satisfaction.

6.2. System Implementation

The implementation process is built upon a robust architecture that integrates data extraction, clustering, and recommendation generation into a seamless flow. The goal is to develop a system capable of efficiently handling user queries and returning accurate search results along with relevant recommendations. The implementation process consists of several interconnected stages, starting from data collection to visualization.

Each phase in the implementation is meticulously designed to ensure that the system operates effectively, delivering high-quality recommendations and an intuitive user experience. The flow begins with extracting valuable data from DBpedia, processing this information through advanced clustering algorithms, and concluding with a user-friendly presentation of the results.

6.2.1 System Setup

The initial setup phase is critical in laying the groundwork for the entire recommendation system. The process begins by collecting relevant data from DBpedia using SPARQL queries. This structured querying language allows the system to extract detailed and specific information about various individuals from a large, interconnected dataset. Key attributes such as [Name, Birth Year, Occupation, Country, Field of Work, Notable Achievements] are extracted during this phase, ensuring a comprehensive dataset for analysis. Once the raw data is retrieved, an entities dataset is constructed, forming the foundation of the knowledge graph. The knowledge graph represents the relationships and connections between entities and literals, providing a contextual understanding of the extracted data.

By using knowledge graphs, the system can understand complex relationships between different attributes, allowing for more accurate clustering and better recommendations. The extracted data is refined and structured to ensure that all relevant information is captured in a format suitable for analysis. Once the relevant data is gathered, the clustering process begins.
Following data extraction, the setup continues with the following steps:

Install Dependencies: Key libraries such as Streamlit (for web interface development), NumPy, and SPARQLWrapper (for querying DBpedia) are installed.

File Preparation: The embeddings generated through BERT are saved into an embedding.npy file, ensuring that the numerical representations of the data are preserved for future use.

Clustering Results Storage: The clustering results, which categorize individuals based on similarities, are stored in a cluster.txt file for efficient retrieval during user searches.

Data Integration: The application loads the prepared files (embedding.npy and cluster.txt) into memory to facilitate real-time recommendations based on previously processed data.

The full implementation flow follows the steps outlined in the flowchart:

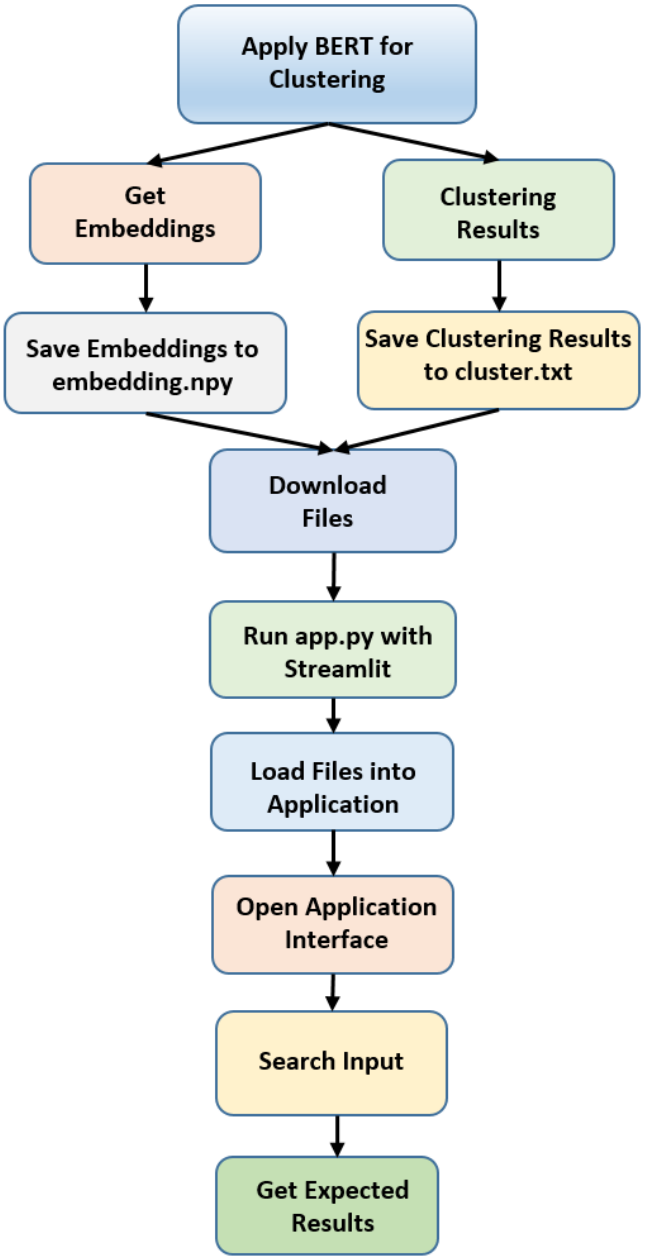

Figure 28: Flowchart of web application

6.2.2 Workflow

The application integrates all the previously discussed components into a fully functional system that delivers relevant recommendations based on clustering and real-time data extraction. It begins by loading the necessary files (embedding.npy and cluster.txt) to ensure that all pre-processed data is ready for use.

The application workflow consists of several crucial steps:

- The application retrieves detailed information from DBpedia using SPARQL queries. This information is then structured into a knowledge graph to facilitate deeper analysis.

- Using BERT embeddings, the system applies clustering algorithms to group individuals based on shared attributes. This step helps identify relationships between different personalities and generates meaningful recommendations.

- Users can search for a specific player, singer, or actor. Upon receiving the query, the system extracts the relevant profile information from DBpedia.

- Based on the clustering outcomes, the system identifies similar individuals and presents them as recommendations. This feature enhances the user experience by offering relevant suggestions for further exploration.

- The final results are displayed in an organized format, with detailed profiles of the searched individual and recommended individuals presented side by side for easy comparison.

This dynamic system effectively merges advanced clustering techniques with real-time data extraction from DBpedia, delivering a powerful and intuitive recommendation engine. The seamless integration of these technologies ensures that users receive accurate results promptly, enhancing their overall experience and encouraging exploration.

By combining clustering with knowledge graph-based data extraction, the system offers an engaging platform where users can discover new personalities and learn more about their favorite players, singers, or actors. The recommendation engine's accuracy and efficiency make it a valuable tool for users seeking to explore related profiles and expand their knowledge base.

6.2.3 Web Design

The design of the application is user-centric, focusing on delivering an engaging, responsive, and intuitive experience. The interface is built using Streamlit, providing a clean and interactive environment where users can search for individuals and view their detailed profiles. The design ensures that the system displays comprehensive information about any searched individual while also offering a list of similar individuals based on clustering results. This dual-display approach enhances the exploration experience and allows users to discover new, relevant individuals seamlessly.

Key features of the application's design include:

Interactive Interface: The Streamlit-based interface is simple yet effective, providing users with a responsive and easy-to-use platform.

Comprehensive Information Display: All relevant details, such as name, occupation, achievements, and country of origin, are presented in an organized manner.

Recommendation Section: Recommendations are generated based on clustering results, allowing users to discover individuals with similar attributes.

Visual Appeal: The layout is designed for clarity and ease of navigation, ensuring that users can quickly find the information they need.

The design philosophy prioritizes ease of use and efficiency, ensuring that users can engage with the system without requiring technical knowledge. The integration of real-time data extraction and clustering-based recommendations further enhances user interaction and satisfaction.

5.2.4 Web Interface

The web interface is built using Streamlit for an intuitive, user-friendly experience. Users can search for players, singers, or actors and view detailed profiles, such as name, occupation, country, and achievements. For example, if a user searches for "Lionel Messi," the interface

will display his name, occupation (footballer), country (Argentina), and notable achievements (e.g., Ballon d'Or wins). Based on clustering results, the system also recommends similar individuals. For instance, after searching for "Messi," it might suggest other footballers like "Cristiano Ronaldo" or "Neymar," grouped based on shared attributes.

The interface displays both the searched individual's profile and related recommendations side by side, making it easy for users to compare and explore similar personalities, enhancing the discovery experience.

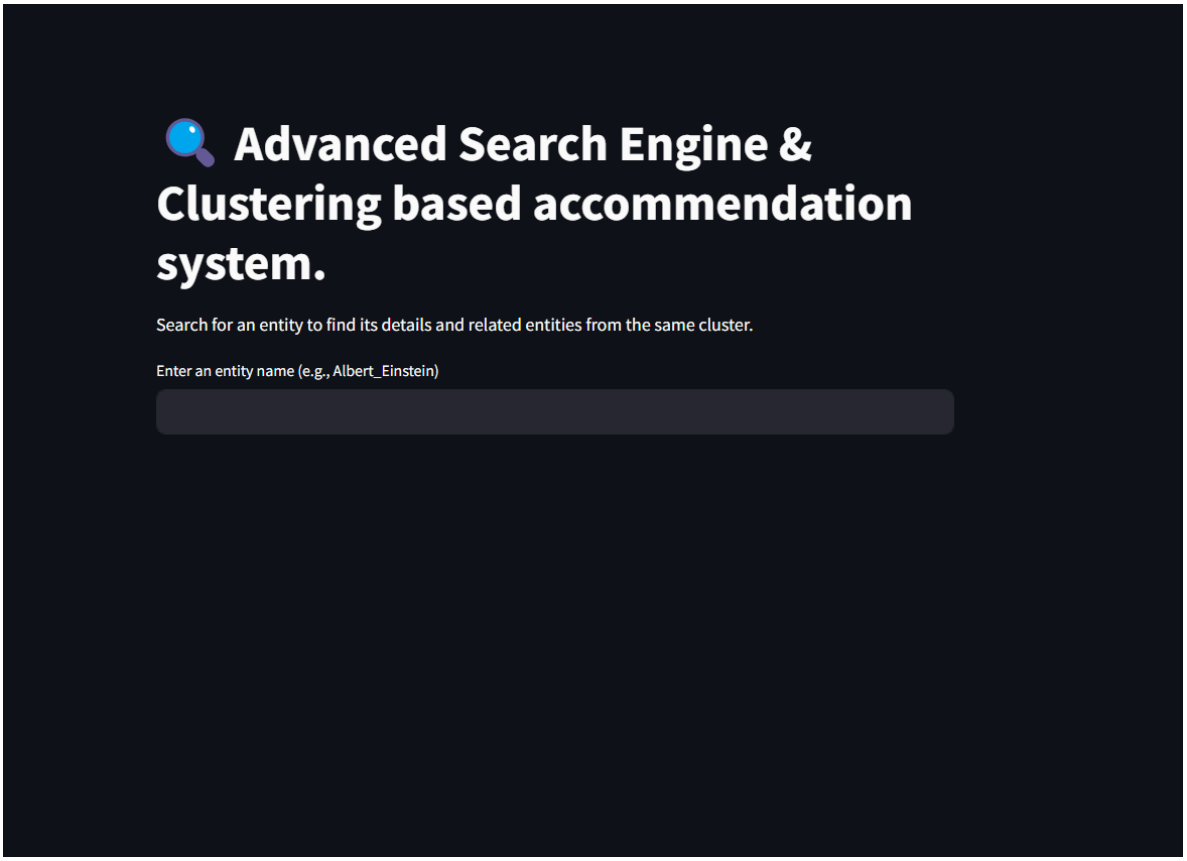

Figure 29: Web Interface

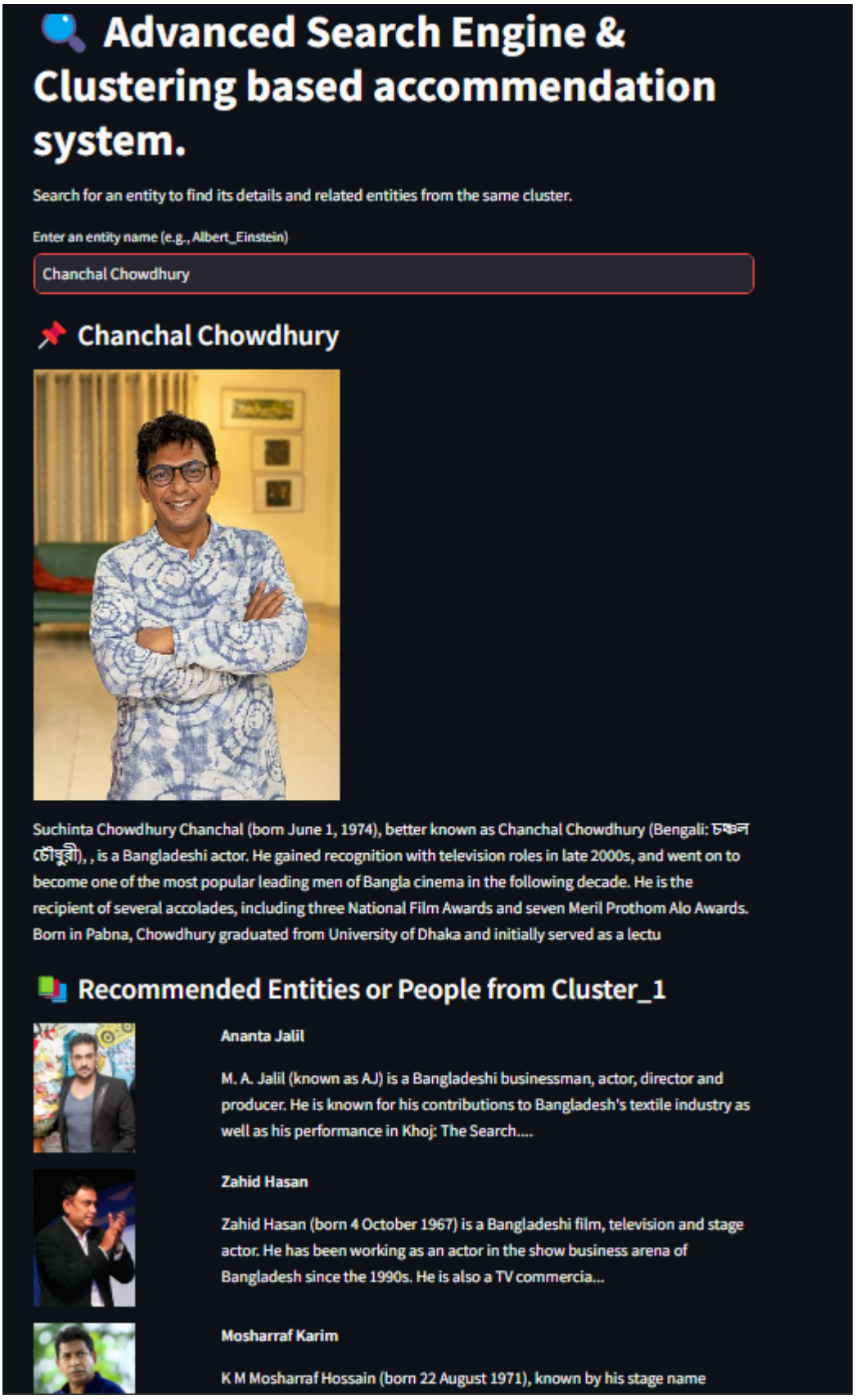

Figure 30: Recommendation for Actor

Figure 31: Recommendation for Scientist

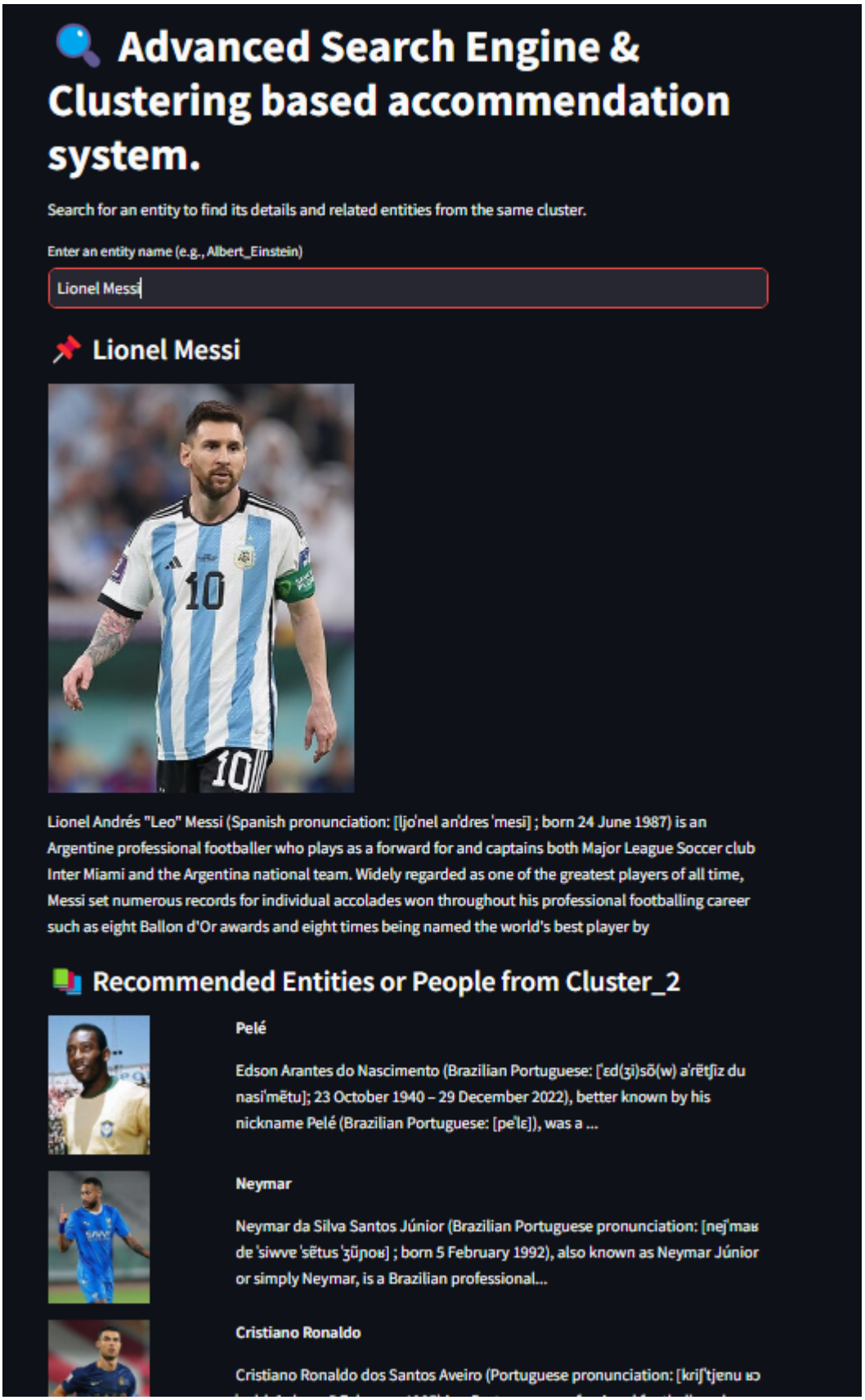

Figure 32: Recommendation for player

6

Conclusion and Future work

This research presented an RDF Data Clustering Framework that transforms RDF data into structured formats, applies Natural Language Processing (NLP) techniques for embedding generation, and utilizes clustering algorithms to identify meaningful patterns in knowledge graphs. By converting RDF structures into sequential representations and leveraging models like Word2Vec and BERT, we enhanced the semantic understanding of RDF data. Our evaluation using clustering metrics such as ARI, NMI, FMI, Precision, Recall, and F1-Score demonstrated that NLP-based embeddings improve clustering performance compared to traditional methods.

Although our framework has shown promising results, there are several areas for improvement: Future research can explore more advanced embeddings using transformers (e.g., BERT, GPT) and Graph Neural Networks (GNNs) for better context-aware representations.
Implementing distributed processing (e.g., Apache Spark, TensorFlow) can help handle large-scale RDF datasets efficiently. Applying the framework in bioinformatics, recommendation systems, and semantic search engines will help validate its practical impact. Evaluating the approach on multi-domain RDF datasets will enhance its generalizability and benchmark it against state-of-the-art clustering methods.
By addressing these areas, we aim to refine RDF clustering techniques, making them more robust, scalable, and applicable to diverse knowledge-driven applications.
Throughout this study, we gained valuable insights into the challenges of data preprocessing, embedding selection, and clustering efficiency. We observed how different embedding techniques impact clustering accuracy and realized the importance of integrating graph structures with machine learning for better knowledge representation. This research provided a strong foundation for semantic clustering of RDF data, making it more useful for applications such as ontology learning, data integration, and knowledge-based AI systems.

References

[1] P. Ristoski and H. Paulheim, "RDF2Vec: RDF graph embeddings for data mining," in Proceedings of the International Semantic Web Conference (ISWC), Mannheim, Germany, 2016, pp. 498–514. DOI: 10.1007/978-3-319-46547-0_30.

[2] S. Eddamiri, E. Zemmouri, and A. Benghabrit, "RDF Data Clustering based on Resource and Predicate Embeddings," in Proceedings of the 10th Int. Joint Conf. on Knowledge Discovery, Lisbon, Portugal, 2018, pp. 367–373. DOI: 10.5220/0007228903670373.

[3] S. Eddamiri, E. M. Zemmouri, and A. Benghabrit, "An improved RDF data clustering algorithm," in Procedia Computer Science, vol. 148, pp. 208–217, Jan. 2019. DOI: 10.1016/j.procs.2019.01.038.

[4] S. Giannini, "RDF data clustering," in Proceedings of the BIS 2013 Workshop, W. Abramowicz, Ed., Lecture Notes in Business Information Processing, vol. 160, Berlin, Germany: Springer, 2013, pp. 220–231.

[5] L. Qi, H. T. Lin, and V. Honavar, "Clustering remote RDF data using SPARQL update queries," in Proceedings of the 29th IEEE International Conference on Data Engineering Workshops (ICDEW), Apr. 2013. DOI: 10.1109/ICDEW.2013.6547456.

[6] [6] S. Eddamiri, A. Benghabrit, and E. Zemmouri, "RDF graph mining for cluster-based theme identification," International Journal of Web Information Systems, vol. ahead-of-print, Apr. 2020. DOI: 10.1108/IJWIS-10-2019-0048.

[7] S. Eddamiri, E. M. Zemmouri, and A. Benghabrit, "RDF data clustering based on resource and predicate embeddings," in Proceedings of the 10th International Conference on Knowledge Discovery and Information Retrieval (KDIR), Jan. 2018. DOI: 10.5220/0007228903670373.

[8] S. Pramanik, J. Alabi, R. S. Roy, and G. Weikum, "Uniqorn: Unified question answering over RDF knowledge graphs and natural language text," Journal of Web Semantics, vol. 83, p. 100833, Dec. 2024. DOI: 10.1016/j.websem.2024.100833.

[9] T. Sagi, M. Lissandrini, T. B. Pedersen, and K. Hose, "A design space for RDF data representations," The VLDB Journal, vol. 31, pp. 347–373, Jan. 2022.

[10] K. S. Candan, H. Liu, and R. Suvarna, "Resource description framework: Metadata and its applications," ACM SIGKDD Explorations Newsletter, vol. 3, no. 1, pp. 6–19, Jul. 2001. DOI: 10.1145/507533.507536.

[11] R. G. Nehe, R. Menon, S. D. Khedkar, and N. S. Mujumdar, "A statistical analysis of knowledge graph and its applications," Panamerican Mathematical Journal, vol. 34, no. 3, 2024.

[12] B. Ellefi, Z. Bellahsene, J. G. Breslin, E. Demidova, S. Dietze, J. Szymański, and K. Todorov, "RDF dataset profiling – a survey of features, methods, vocabularies and applications," Semantic Web, vol. 9, no. 5, pp. 677–705, 2018, doi: 10.3233/SW-180294.

[13] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," IEEE Trans. Neural Networks Learn. Syst., vol. 33, no. 2, pp. 494–514, Feb. 2022, doi: 10.1109/TNNLS.2021.3070302.

[14] Z. Ye, Y. J. Kumar, G. O. Sing, F. Song, and J. Wang, "A comprehensive survey of graph neural networks for knowledge graphs," IEEE Access, vol. 10, pp. 75729–75741, Jul. 2022, doi: 10.1109/ACCESS.2022.3191784.

[15] C. Gutierrez, C. A. Hurtado, and A. Vaisman, "Introducing time into RDF," IEEE Trans. Knowl. Data Eng., vol. 19, no. 2, pp. 207–218, Feb. 2007, doi: 10.1109/TKDE.2007.34.

[16] M. Arenas, C. Gutierrez, and J. Pérez, "Foundations of RDF databases," in Reasoning Web 2009, S. Tessaris et al., Eds., vol. 5689, Lecture Notes in Computer Science, Berlin, Germany: Springer-Verlag, 2009, pp. 158–204.

[17] J. Dokulil and J. Katreniakova, "Using clusters in RDF visualization," in Proc. 2009 Third Int. Conf. Adv. Semantic Process. (SEMAPRO), Sliema, Malta, Oct. 2009, doi: 10.1109/SEMAPRO.2009.19.

[18] S. A. Bamatraf and R. A. BinThalab, "Clustering RDF data using K-medoids," in Proc. 2019 First Int. Conf. Intell. Comput. Eng. (ICOICE), Hadhramout, Yemen, Dec. 2019, doi: 10.1109/ICOICE48418.2019.9035160.

[19] L. Ma and Y. Zhang, "Using Word2Vec to process big text data," in Proc. 2015 IEEE Int. Conf. Big Data (Big Data), Santa Clara, CA, USA, Oct. 2015, doi: 10.1109/BigData .2015.7364114.

[20] G. Di Gennaro, A. Buonanno, and F. A. N. Palmieri, "Considerations about learning Word2Vec, " Neural Comput. Appl., vol. 77, pp. 12320–12335, Apr. 2021.

Appendix

Addressing Course Outcomes (CO), Knowledge Profile (K), Complex Engineering Problem (EP), and Program Outcome (PO):

CO	**CO Descriptions**	**K (Knowledge Profile)**	**EP (Engineering Problem)**	**PO (Program Outcome)**
CO1	Understand RDF data representation and knowledge graphs	K1: Demonstrates understanding of RDF triplets and their application in data modeling.	EP1: Identifies challenges in RDF data clustering and representation.	PO1: Develops structured methods for analyzing and extracting meaningful insights from RDF knowledge graphs.
CO2	Apply embedding techniques for RDF data clustering	K2: Knowledge of word embeddings such as BERT and Word2Vec for transforming RDF data.	EP2: Implements embedding-based clustering models for RDF data analysis.	PO2: Demonstrates capability to leverage advanced NLP and machine learning models for knowledge graph clustering.
CO3	Evaluate clustering techniques on RDF data	K3: Understands different clustering algorithms such as K-Means, GMM, and Agglomerative.	EP3: Compares clustering performance using evaluation metrics like Silhouette Score and ARI.	PO3: Enhances RDF-based applications by identifying the most efficient clustering methods for structured knowledge representation.
CO4	Optimize clustering performance and scalability	K4: Awareness of computational challenges in large-scale RDF data clustering.	EP4: Develops strategies to improve clustering efficiency using dimensionality reduction techniques such as PCA.	PO4: Integrates optimization techniques to ensure scalable and efficient RDF data clustering pipelines.
CO5	Apply RDF clustering in web applications	K5: Understands the impact of RDF clustering in recommendation systems.	EP5: Adapts RDF clustering techniques for structured and similar data retrieval.	PO5: Enhances knowledge-based applications by applying clustering methods for recommendation systems.