People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University Centre SALHI Ahmed of Naama
Institute of Science
Department of Computer Science

Final Year Project Thesis

In partial fulfillment of the requirements for the degree

Master in Computer Science

Specialty : **Information Systems**

Web Services Clustering Using Unsupervised Learning Algorithms

Presented by :

- SMAIL Samiha

Supervised by :

- Dr. SIDAOUI Boutkhil
- Dr.KAOUAN Moussa

Academic Year : **2024-2025**

Acknowledgment

First and foremost, I express my deepest gratitude to Allah, whose blessings and guidance have been my constant source of strength and inspiration throughout this journey.

I would like to extend my heartfelt thanks to my esteemed advisor,**Mr. Sidaoui Boutkhil** , for his unwavering support, valuable guidance, and continual encouragement. His expertise and dedication have played a crucial role in the successful completion of this work.

To my beloved parents, I offer my sincerest gratitude for their unconditional love, patience, and constant support. Your faith in me has been the foundation of my motivation and achievements.

I am also truly grateful to the family **SMAIL** for their constant encouragement and support throughout this journey.

Finally, I extend my sincere thanks to all those who have contributed in any way to my academic and personal development. Whether through encouragement, guidance, or assistance, your support has meant a lot to me.

Thank you all.

Dedication

I dedicate this thesis to my beloved parents, **Fatima** and **Slimane**, for their unconditional love, strength, and unwavering support .

To my dear siblings, for always being by my side. A special thank you to my friend **Youcef**, one of the few who share my passion for programming. I'm deeply grateful for his unwavering support and invaluable advice, which have continually inspired me to grow and improve.

To all the friends I've met along the way—thank you for being part of this journey. I also dedicate this work to my cherished group of friends who have been a constant source of support and motivation throughout my university life:
Zoulikha ,Marwa ,Sabrine .

The moments we shared, from late-night, last-minute study sessions filled with coffee and laughter, to makeup exams and spontaneous adventures, have left a lasting imprint on my life. Through every high and low, your loyalty, joy, and encouragement made this path not only bearable but genuinely memorable. I am forever grateful for each of you and the beautiful memories we've built together.
To all those I love, and all those who love me—**this is for you**.

Abstract

The rapid proliferation of web services on the Internet has created significant challenges in their classification, discovery, and efficient management. Traditional methods often rely on manual annotation or keyword-based techniques, which are limited in scalability and semantic understanding. This project proposes an unsupervised learning approach for the clustering of web services using machine learning algorithms. The methodology involves preprocessing web service descriptions, extracting key features using Term Frequency–Inverse Document Frequency (TF-IDF), and applying clustering algorithms such as K-Means, DBSCAN, and Agglomerative Hierarchical Clustering to group services based on functional similarities. The proposed system is implemented within a web-based platform that enables users to upload service datasets, perform clustering, and visualize results through interactive charts and dimensionality reduction techniques such as WordNet. Experimental results highlight the effectiveness of clustering in revealing latent groupings and improving the organization and accessibility of large-scale web services. This work contributes to the advancement of intelligent web service management and lays the groundwork for future integration with semantic web technologies and automated service discovery systems.

Keywords— web services, clustering

CONTENTS

LIST OF FIGURES

LIST OF TABLES

General Introduction

Web services refer to software systems, applications, or cloud-based technologies that use standardized web protocols—typically HTTP or HTTPS—to enable communication, interaction, and data exchange over the Internet. These services primarily use XML (Extensible Markup Language) for message formatting and are designed to support application-to-application (A2A) integration.

In essence, web services serve as platforms for exchanging structured data between different systems, allowing programs, documents, messages, or objects to be shared seamlessly. One of the core strengths of web services is their language-independence: applications developed in different programming languages can still communicate effectively by sending and receiving XML-formatted requests and responses over the web.

Typically, a client invokes a web service by sending a request in XML format, and the server processes this request and returns an XML response. Web services are also closely tied to the concept of Service-Oriented Architecture (SOA), where individual services are loosely coupled and reusable, enabling modular, scalable, and interoperable system designs. [1].

The primary objective of Web Services Clustering is to automatically group semantically or functionally similar web services into clusters, based on features such as textual descriptions, input/output parameters, or domain-specific metadata. This facilitates efficient service discovery, organization, and recommendation within large-scale service repositories. By reducing the search space and structuring services according to their functional similarities, clustering improves the scalability, navigability, and reusability of web services. It also supports semantic reasoning, enabling more accurate service composition and classification [2].

The structure of this document is organized as follows.

- **Chapter 1:** introduces the fundamentals of web services, covering their architecture, principles, and key techniques. It also explains clustering methods for classifying web services and introduces TF-IDF, a method for evaluating the importance of terms in text.
- **Chapter 2:** presents the basic concepts of Machine Learning (ML), explaining their definitions, relationship, and common algorithms. It sets the theoretical foundation for understanding how systems can learn and improve from data.
- **Chapter 3:** details the tools and methods used for clustering web services. It describes the use of Python and libraries like pandas and scikit-learn for data preprocessing and clustering (K-Means, DBSCAN, Hierarchical), using TF-IDF on a CSV dataset of web service descriptions.

Chapter 1

CHAPTER 1

WEB SERVICES

1.1 Introduction

In today's digital era, web services have become essential to enable seamless interaction and data exchange between various online systems. With their growing number and variety, classifying these services has become crucial for efficient discovery and utilization [3]. This chapter explores the concept of web services classification, focusing on techniques such as clustering algorithms to group services based on their features and functions. We will highlight the importance of clustering, its challenges and how modern methods can enhance decision-making in dynamic web environments [4].

1.2 Web and Web Services

1.2.1 What is the web?

The Web, commonly referred to as the World Wide Web (WWW), is a globally distributed information system that revolutionized the way individuals and organizations access, share, and disseminate information. Initially conceived and developed by Tim Berners-Lee in 1989 while working at CERN, the Web was designed to facilitate the automatic sharing of information among researchers by linking documents through hypertext. Since its inception, it has evolved into one of the most transformative technological innovations in modern history [5]. The Web is built upon the foundational infrastructure of the Internet—a massive, decentralized network of interconnected computers that facilitates the transmission of data across the globe. However, it is important to understand that the Web is only one of several services that operate on top of the Internet, alongside others such as email, file transfer (FTP), Voice over IP (VoIP), and more [6].

The Web comprises a vast collection of multimedia documents and digital resources, including text, images, videos, and interactive applications, all of which are accessible through web browsers. These resources are interconnected using hyperlinks, forming a complex, navigable structure of information that enables users to transition seamlessly from one document or website to another. At the core of the Web's functionality is the Hypertext Transfer Protocol (HTTP), which governs the communication between web clients (browsers) and servers. Through Uniform Resource Locators (URLs), each web resource is uniquely identified and retrievable, contributing to the Web's role as a global repository of knowledge and services [7].

While the Web relies on the Internet for its technical operations—such as routing, addressing, and data transfer—it differs conceptually in purpose and design. The Internet provides the physical and logical infrastructure for data exchange, but the Web adds a semantic layer that allows users

to access and interact with that data meaningfully. The introduction of the Web has profoundly impacted nearly every aspect of human life, from education and commerce to communication and entertainment, fostering unprecedented levels of connectivity, innovation, and information democratization [8].

1.2.2 What are Web Services?

A Web Service is a modular, self-contained software component specifically designed to perform a set of well-defined operations over a network. It serves as a crucial bridge for enabling interoperability among distributed systems and heterogeneous applications, even when they are developed using different programming languages or deployed across various hardware and software platforms. This is achieved by leveraging standardized communication protocols such as HTTP or HTTPS, and utilizing platform-neutral data representation formats like XML (Extensible Markup Language) or JSON (JavaScript Object Notation), which ensure that systems can communicate effectively regardless of their internal structure or architecture. The primary advantage of web services lies in their ability to encapsulate business logic and expose it through well-defined application programming interfaces (APIs), which can be accessed and consumed over the internet by remote clients [9].
In the context of cloud computing, web services play a foundational role by offering on-demand access to computing resources and application functionalities. They support dynamic discovery, meaning that services can be published in registries and located by other systems as needed, enabling loose coupling and enhanced system modularity. Furthermore, the remote invocation of these services allows for scalable and elastic solutions that can automatically adapt to changing workloads and user demands. This architecture not only facilitates reuse and composition of services to form more complex workflows, but also promotes interoperability and integration of diverse systems across organizational and geographic boundaries. The widespread adoption of Service-Oriented Architecture (SOA) and more recently, microservices architecture, further underscores the pivotal role web services play in modern software engineering, especially in designing resilient, flexible, and maintainable distributed systems [10].

1.2.3 Evolution and Architecture of Web Services

Web services have their roots in the Remote Procedure Call (RPC) mechanism developed within the Distributed Computing Environment (DCE), a software framework introduced in the early 1990s. DCE featured components such as a distributed file system (DCE/DFS) and a Kerberos-based authentication system. While DCE originated in the Unix ecosystem, Microsoft created its own version, known as MSRPC, which became the backbone of interprocess communication in Windows. Technologies like COM (Component Object Model) and OLE (Object Linking and Embedding) were built upon this MSRPC foundation. Interestingly, although RPC in DCE was originally intended for distributed computing across multiple machines, Microsoft repurposed it for communication between processes on a single device, using it as the core of its COM infrastructure [11] .

Early distributed object system frameworks, including CORBA (Common Object Request Broker Architecture) and Microsoft's DCOM (Distributed COM), were based on the procedural model of DCE/RPC. Similarly, Java's RMI (Remote Method Invocation) evolved from DCE/RPC, and its usage is evident in Java EE (Enterprise Edition), particularly within Session and Entity EJBs (Enterprise JavaBeans), where RMI facilitates method invocation. Java EE (formerly J2EE) and Microsoft's .NET platform represent the second generation of distributed object frameworks, continuing the legacy of DCE/RPC. Notably, DCE/RPC remains relevant today, with tools like Samba (which supports file and print services for Windows clients) still relying on it [11] .

1.2.4 Key Characteristics of Web Services [12]

- Based on standard protocols such as **HTTP** and **HTTPS**
- Use **XML** for message formatting and data exchange
- Platform-independent and language-neutral
- Promote **interoperability** between diverse systems
- Can be discovered and invoked remotely over the Internet

1.2.5 Types of Web Services

- **XML-RPC (Remote Procedure Call):** is the most basic XML protocol to exchange data between a wide variety of devices on a network. It uses HTTP to quickly and easily transfer data and communicate other information from client to server [13].

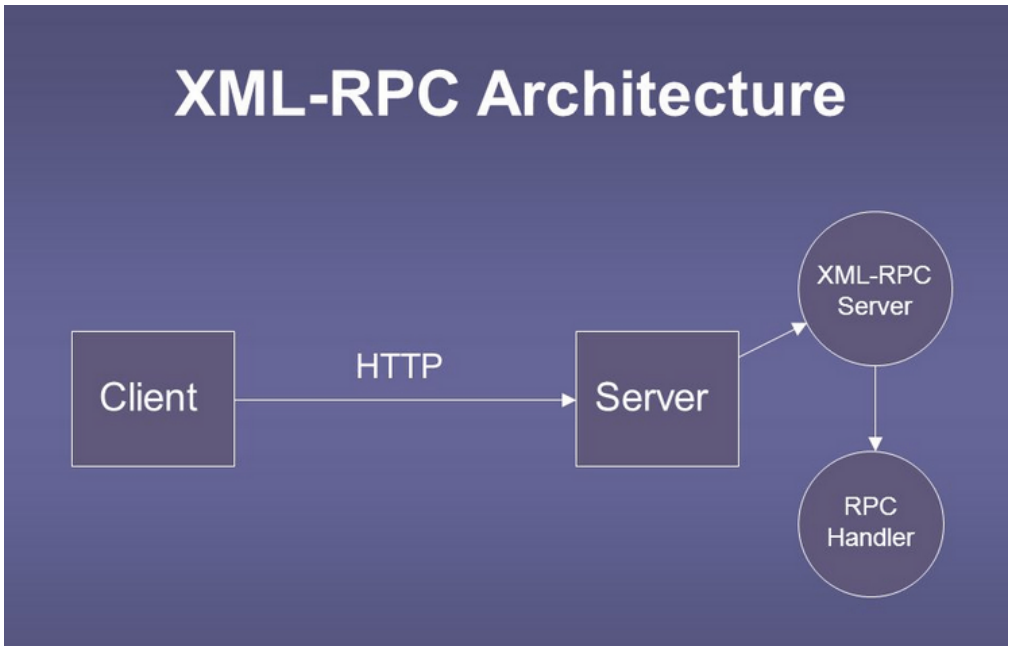


Figure 1.1: XML-RPC Architecture

- **UDDI (Universal Description, Discovery, and Integration):** is a standard that uses XML to enable the publication, description, and discovery of web services. Acting as an online directory, it allows organizations globally to simplify digital interactions and enhance e-commerce by making it easier for systems to locate and connect with each other.

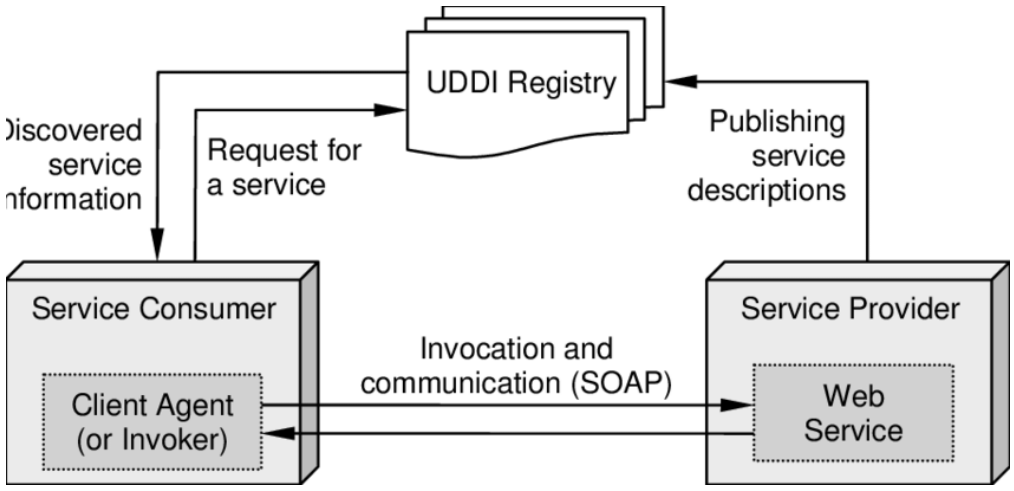


Figure 1.2: Role of UDDI registries and other Web Services Technologies.

- **SOAP (Simple Object Access Protocol):**SOAP is an XML-based protocol designed for exchanging structured data and documents over HTTP or SMTP. It enables independent processes running on different systems to communicate using XML. (SOAP will be discussed in more detail later[14].
- **REST (Representational State Transfer):**REST provides communication between devices and the internet, primarily for API-based interactions. Most RESTful services rely on HTTP as the transport protocol. (Further details on REST will be presented later[15].)

1. **How Does Web Service Work?**

   (a) A client sends a request (usually in XML format) to the web service.

   (b) The server processes the request.

   (c) The server then returns a structured XML response to the client

   This model allows systems to communicate and share functionality without being tightly coupled [12]

1.2.6 Key Components in Web Service Communication

- Client: Sends requests using remote procedure calls (RPCs)
- Server: Hosts the web service and processes the requests.
- XML: Used as the universal data exchange format.
- SOAP: Protocol for sending XML messages over HTTP

  This design allows platform and language independence, making web services a powerful solution for integrating distributed applications over the internet [2].

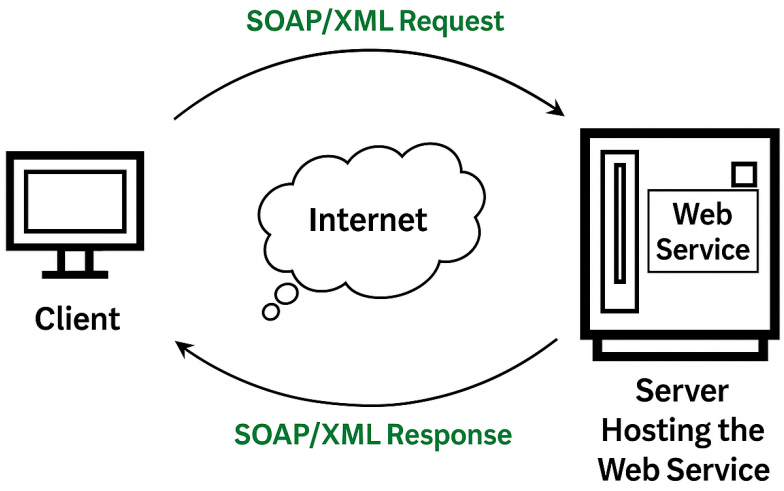


Figure 1.3: Client-Server Communication in Web Services using SOAP/XML

1.2.7 Key Standards

One of the fundamental characteristics of Internet standards is their emphasis on protocols rather than specific implementations. The Internet comprises diverse technologies that achieve interoperability through shared communication protocols. This approach prevents any single vendor from imposing proprietary standards on the Internet ecosystem. Moreover, open source software development plays a critical role in maintaining interoperability by ensuring that vendor implementations adhere to these standards[16].

Several key standards underpin the architecture and functioning of web services. These include Universal Description, Discovery, and Integration (UDDI), Web Services Description Language (WSDL), Web Services Inspection Language (WSIL), Simple Object Access Protocol (SOAP), and Web Services Interoperability (WS-I). The interrelationship among these standards is illustrated in Figure bellow[2].

- UDDI defines open, platform-independent protocols that facilitate the global sharing of business information. It provides a registry framework where organizations can publish, discover, and define interactions with web services over the Internet[17].
- WSIL is an XML-based specification designed for distributed service discovery. It enables the referencing of service descriptions directly at the service provider's location by specifying mechanisms to inspect web sites for available services. WSIL complements UDDI by allowing discovery of services hosted on web sites that may not yet be registered in a UDDI directory [18].
- WSDL is an extensible XML-based language used to describe web service interfaces and their network endpoints. It specifies the operations offered by a web service as well as the formats and protocols used for communication. WSDL documents can be published through UDDI, WSIL, or disseminated directly via URLs, email, or web sites [19].
- SOAP is a lightweight, XML-based protocol for exchanging structured information across decentralized, distributed systems, typically over HTTP or other Internet protocols. The SOAP specification consists of three main components:

An envelope that defines the message structure and processing rules[20].

A set of encoding rules that define how application-defined data types are represented [20].

A convention for encoding remote procedure calls and their responses [20].

- SOAP facilitates the binding and invocation of discovered web services by defining standardized message routing paths. Additionally, SOAP can be employed to query UDDI registries for available web services [21].

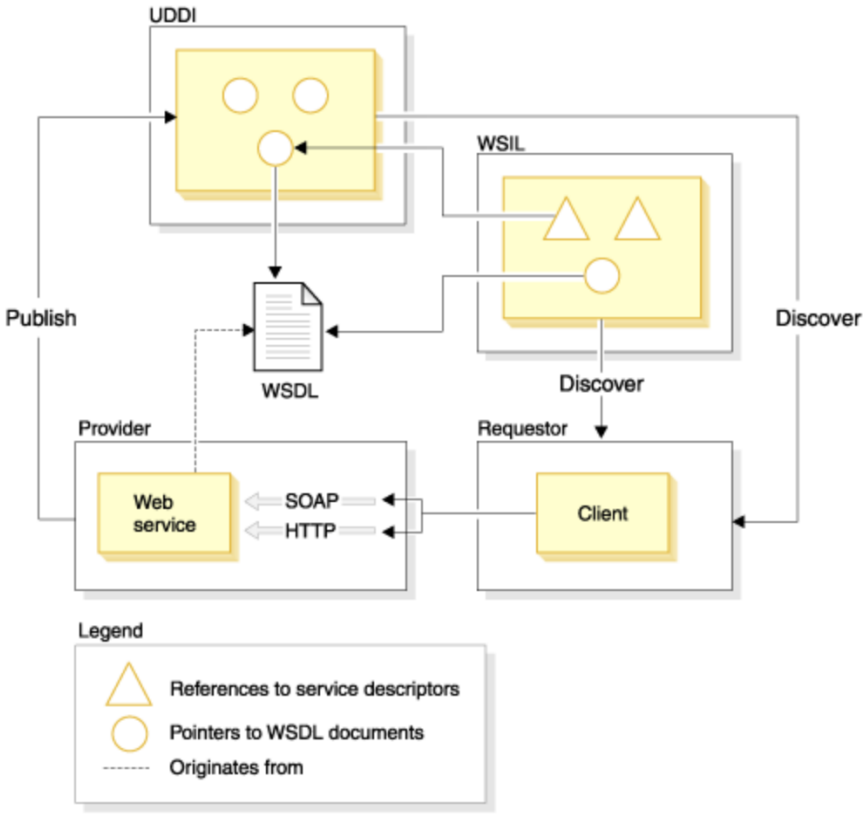


Figure 1.4: Relationships between SOAP, UDDI, WSIL and WSDL.

1.3 Web services clustering

Clustering web services has emerged as a powerful and efficient strategy for addressing a wide range of challenges in the domain of service computing. These challenges include service discovery, selection, recommendation, and composition, which have become increasingly complex due to the growing number of web services available on the internet. By grouping similar services based on their functionalities and descriptions, clustering facilitates better organization, management, and retrieval of relevant services. The foundational idea behind this approach involves extracting meaningful feature vectors from web service description documents, such as WSDL (Web Services Description Language) files or natural language-based service summaries. These feature vectors serve as the numerical representation of the services and form the basis for clustering algorithms [22] .

However, constructing these feature vectors is not a straightforward task. One of the primary obstacles lies in the typically brief and often ambiguous nature of natural language service descriptions. Unlike structured data, these textual descriptions are concise and sometimes lack the context or detail necessary for traditional feature extraction techniques to perform effectively. This limitation

can significantly impact the quality of the resulting clusters, leading to reduced performance in downstream tasks such as service matchmaking or recommendation [23].

1.3.1 Motivation and Benefits

With the proliferation of Web services across various domains and platforms, discovering, organizing, and managing these services has become increasingly challenging. Traditional discovery mechanisms, which typically rely on syntactic keyword matching or centralized registries, are often insufficient for handling the scale, heterogeneity, and semantic diversity of modern service ecosystems. In this context, Web services clustering has emerged as a promising solution that aims to improve the scalability, efficiency, and accuracy of Web service discovery and composition by automatically grouping similar services based on functional and/or semantic characteristics[24].

Motivation

- **Scalability in Large-Scale Environments** The growing number of publicly available Web services has made flat or unstructured service repositories inefficient. Clustering provides a hierarchical or grouped structure, which reduces the computational complexity associated with browsing or searching large service sets[25]
- **Improved Service Discovery**By grouping services with similar functionality or meaning, clustering narrows the search space, thereby enhancing the performance of discovery mechanisms. Clients can focus their queries within the most relevant clusters, resulting in faster and more accurate retrieval [26]
- **Facilitation of Automated Composition** In scenarios where composite services are required, clustering assists in identifying compatible or complementary services that can be combined to fulfill complex workflows[27]. This is particularly beneficial in dynamic environments where services may be frequently updated
- **Semantic Interoperability** Services often differ in terminology due to being developed by independent providers. Clustering techniques that incorporate semantic similarity can bridge lexical gaps, enabling the discovery of services that are semantically equivalent but syntactically diverse [26]
- **Enhanced Quality of Service (QoS) Management** Clustering can also be extended to include non-functional attributes such as response time, availability, and throughput. Grouping services based on QoS metrics facilitates more informed decision-making during service selection [27]

Benefits

- **Improved Efficiency:**Clustering reduces the computational cost of service discovery by allowing search operations to be confined within relevant groups [28].
- **Increased Accuracy:** When combined with semantic analysis, clustering enhances the precision of retrieved services by considering contextual meanings[29].
- **Support for Dynamic Environments:**Clustering models can be incrementally updated to reflect changes in the service repository, making the system more adaptive[30].
- **Better Organization and Maintenance:**Grouping services into coherent categories improves repository structure and simplifies administrative tasks such as monitoring and updating services[31].

1.4 Functional vs. Semantic Similarity

In the domain of Web service clustering, the determination of similarity between services is a foundational step that directly influences the quality and relevance of the resulting clusters. Two primary forms of similarity are typically considered: functional similarity and semantic similarity. Each serves a distinct purpose in characterizing Web services and impacts the clustering process differently [32].

(a) **Functional Similarity:**Functional similarity refers to the degree to which two Web services perform comparable operations or tasks. It is typically assessed by analyzing syntactic features such as the service's input and output parameters, operation names, and interface definitions as described in the Web Services Description Language (WSDL)[33]. Techniques commonly employed to measure functional similarity include:

- Keyword matching in service operation names and parameters [34].
- Vector space models using TF-IDF and cosine similarity [35] .
- Structural comparison of WSDL documents [35].

Although functional similarity provides a foundational approximation of service comparability, it is often limited by vocabulary mismatches and syntactic variations. Services that perform similar tasks but use different terminologies may not be grouped together under purely functional criteria [36].

(b) **Semantic Similarity** Semantic similarity, in contrast, involves the comparison of Web services based on the meaning of their descriptions, rather than their syntactic representations. This approach leverages natural language processing and semantic technologies such as:

- Ontologies (e.g., WordNet, domain-specific taxonomies)[36].
- Semantic annotation of service parameters and operations[37].
- Word embedding models (e.g., Word2Vec, BERT) for context-aware similarity[38].

Semantic similarity enables the clustering of services that are functionally alike but differ in lexical representation. This is particularly beneficial in heterogeneous and dynamic service environments, where naming conventions are not standardized[38].

1.5 Feature Extraction

Feature extraction using TF-IDF identifies and quantifies the importance of terms within the text by considering both their frequency in individual documents and their rarity across the entire corpus. This method helps reduce noise by filtering out common, less informative words and highlights the most meaningful terms for further processing and analysis[39].

1.5.1 TF-IDF

(Term Frequency–Inverse Document Frequency) is a statistical measure used to evaluate how important a word is to a particular document within a larger collection of documents, known as a corpus . It helps highlight terms that are frequent in a specific document but rare across the entire corpus, making them more significant for identifying the content of that document[39].

- **Term Frequency (TF):** Term Frequency quantifies how often a word appears in a document. It is calculated by dividing the number of times the word occurs by the total number of words in the document. This helps identify terms that are more prominent within a single document[40].

$$TF(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \tag{1.1}$$

- **Inverse Document Frequency (IDF)**:Inverse Document Frequency measures how rare or unique a term is across a set of documents. It is calculated using the logarithm of the ratio between the total number of documents and the number of documents that contain the term. This helps reduce the weight of commonly used words and highlight terms that are more distinctive to specific documents[40].

$$IDF(t,D) = \log\left(\frac{\text{Total number of documents in the corpus } N}{\text{Number of documents containing term } t}\right) \tag{1.2}$$

1.5.2 Example Calculations

: **Sentence 1:** good boy
Sentence 2: good girl
Sentence 3: boy girl good

- **TF (Term Frequency)**
TF = (Number of times the word appears in the sentence) / (Total number of words in the sentence)

Word	Sentence 1	Sentence 2	Sentence 3
good	$\frac{1}{2} = 0.5$	$\frac{1}{2} = 0.5$	$\frac{1}{3} + 0.33$
boy	$\frac{1}{2} = 0.5$	0	$\frac{1}{3} = 0.33$
girl	0	$\frac{1}{2} = 0.5$	$\frac{1}{3} = 0.33$

Table 1.1: Term Frequency (TF) calculation

- **IDF (Inverse Document Frequency)**
IDF = log(Total number of sentences / Number of sentences containing the word)
- **Total sentences** =3

Word	Appears in	IDF Calculation
good	3	$\log\left(\frac{3}{3}\right) = \log(1) = 0$
boy	2	$\log\left(\frac{3}{2}\right) \approx 0.176$
girl	2	$\log\left(\frac{3}{2}\right) \approx 0.176$

Table 1.2: Inverse Document Frequency (IDF) calculation

- **TF-IDF**:

$$TF - IDF = TFIDF \tag{1.3}$$

Word	Sentence 1	Sentence 2	Sentence 3
good	$0.5 \times 0 = 0$	$0.5 \times 0 \approx 0$	$0.33 \times 0 \approx 0$
Boy	$0.5 \times 0.176 \approx 0.088$	$0 \times 0.176 \approx 0$	$0.33 \times 0.176 \approx 0.058$
Girl	$0 \times 0.176 = 0$	$0.5 \times 0.176 \approx 0.088$	$0.33 \times 0.176 \approx 0.058$

Table 1.3: TF-IDF Calculation

1.5.3 Using TF-IDF in Web Service Clustering

In our approach to clustering Web Services, TF-IDF is applied to the Input or Output text fields extracted from service description files. This transforms the descriptive text into numerical vectors, which are then fed into clustering algorithms such as K-Means, DBSCAN, or Hierarchical Clustering. These clusters group web services based on their functional similarity. This vectorization process ensures that the clustering is semantically meaningful—services with similar input/output descriptions are grouped together, even if they use different vocabulary [41].

1.5.4 Feature Reduction

1.5.5 Techniques

- **WordNet-Based Grouping**
  WordNet is a lexical database that organizes English words into sets of synonyms called synsets and captures semantic relationships such as hypernymy (generalization), hyponymy (specialization), and meronymy (part-whole). In the context of web service descriptions, WordNet-based grouping exploits these semantic relations to cluster terms with similar meanings into single features. This reduces the feature space dimensionality by merging semantically related words. For example, the words "purchase," "buy," and "acquire" may be grouped under a single semantic concept, thus preventing redundant features representing the same action. This semantic grouping also improves clustering results by focusing on the underlying concepts rather than surface lexical differences[42].
- **Principal Component Analysis (PCA)**
  PCA is a statistical technique that transforms a large set of correlated features into a smaller set of uncorrelated components, called principal components, which retain most of the data's variance. Applied to textual data represented as vectors (e.g., TF-IDF vectors), PCA identifies the directions (components) where the data varies the most, allowing for dimensionality reduction while preserving essential information. For instance, a web service dataset with 1000 textual features can often be reduced to 50-100 principal components without significant loss of information. This reduction helps to alleviate computational overhead and improves clustering stability. PCA is often visualized via scree plots or explained variance ratios, aiding in selecting the optimal number of components [43].

By integrating semantic grouping via WordNet and statistical reduction via PCA, researchers can efficiently preprocess web service textual data, leading to more robust and meaningful clustering results. These techniques address the challenges posed by high-dimensional, sparse, and noisy textual representations[44].

1.6 Evaluation of Clustering Performance

Evaluating the performance of clustering algorithms in unsupervised learning is inherently challenging, as there are no ground truth labels to directly measure accuracy. Instead, internal validation metrics are used to assess the compactness and separation of clusters. In this study, we rely on two widely used evaluation metrics: the **Silhouette Score** and the **Calinski-Harabasz** Index. Additionally, we discuss the general limitations of evaluating clustering results in the absence of labeled data.

1.6.1 Silhouette Score

The Silhouette Score measures how similar an object is to its own cluster compared to other clusters. It ranges from -1 to 1, where a higher score indicates that the sample is well matched to its own cluster and poorly matched to neighboring clusters. A score close to 1 implies clear cluster separation, while values near 0 suggest overlapping clusters. In our application, the Silhouette Score was computed for each clustering method across multiple domains to assess the quality and coherence of the resulting clusters[45].

1.6.2 Calinski-Harabasz Index

The Calinski-Harabasz Index, also known as the Variance Ratio Criterion, evaluates the ratio between the between-cluster dispersion and the within-cluster dispersion. Higher values of this index indicate better-defined clusters. It is particularly useful for comparing clustering results across different configurations or algorithms. This index was used alongside the Silhouette Score to provide a more comprehensive view of clustering performance within our dataset[46].

1.6.3 Elbow Score

The Elbow Method is a graphical technique used to determine the optimal number of clusters in clustering algorithms such as KMeans. It involves plotting the Within-Cluster Sum of Squares (WCSS) against the number of clusters. The point at which the decrease in WCSS becomes marginal—forming an “elbow” in the curve—is considered the optimal number of clusters. This point reflects a balance between minimizing intra-cluster distance and avoiding overfitting[47].

1.7 State of the Art

The clustering of web services based on textual descriptions is a pivotal task in service-oriented computing, enabling the organization and discovery of services within complex, heterogeneous ecosystems. This section reviews prior academic contributions to web services clustering, focusing on text preprocessing, semantic feature extraction, and unsupervised clustering techniques. By synthesizing key studies, we identify limitations in existing approaches, particularly in semantic enhancement and domain-specific analysis, which our system addresses through TF-IDF vectorization with WordNet-based synonym merging and threshold-based noise reduction.

1.7.1 Syntactic-Based Approaches

Early clustering techniques primarily relied on syntactic features extracted from service descriptions such as WSDL. These features were processed using statistical text mining methods like TF-IDF. Researchers such as Dong et al. (2004) and Crasso et al. (2010) applied classic algorithms such as K-Means and Hierarchical Agglomerative Clustering, treating service descriptions as plain text. However, these approaches lacked the ability to capture semantic similarity, making them sensitive to vocabulary variation and less effective in real-world applications[48].

1.7.2 Semantic-Based Approaches

Semantic clustering leverages ontologies and external knowledge bases—such as WordNet—to enrich the understanding of service descriptions beyond their textual form. Unlike syntactic methods that rely on surface-level tokens, semantic techniques interpret the conceptual meaning of terms, enabling the identification of deeper relationships and similarities between services. This enhancement leads to more accurate and meaningful clustering outcomes, especially in cases where synonymy or polysemy might obscure similarity in purely lexical approaches[49].

1.7.3 Hybrid Approaches

Hybrid approaches integrate both syntactic and semantic methods to capitalize on the strengths of each. Typically, this involves the semantic enrichment of textual data (e.g., using ontologies or word embeddings) followed by the application of traditional vector-based clustering algorithms such as KMeans, DBSCAN, or Hierarchical Clustering. By combining these layers, hybrid models improve both the granularity and relevance of resulting clusters, offering a more robust framework for web service classification and discovery[50].

1.7.4 Topic Modeling and Deep Learning

In recent years, topic modeling and deep learning have emerged as powerful tools for web services clustering. Techniques such as Latent Dirichlet Allocation (LDA) uncover latent topics within service descriptions, allowing for more abstract representation and grouping. Additionally, deep learning architectures—including autoencoders, word embeddings (e.g., Word2Vec, GloVe), and transformer-based models (e.g., BERT)—have been applied to encode contextual information from service texts. These models aim to capture nuanced semantics and contextual dependencies, resulting in improved clustering quality, especially in large-scale and dynamic service repositories [51].

Table 1.4: Comparison of Web Services Clustering Approaches

Authors	**Methodology**	**Evaluation Metrics**
Liu, W.; Wong, W. (2009)	Text mining + KMeans	Silhouette: 0.50
Elgazzar, K.; Hassan, A.; Martin, P. (2010)	WSDL-based keyword clustering	Silhouette: 0.48
Chen, L.; Wang, Y.; Yu, Q.; Zheng, Z.; Wu, J. (2013)	WT-LDA (User Tagging + LDA + KMeans)	Precision: 0.75, F-score: 0.72
Elshater, Y.; Elgazzar, K.; Martin, P. (2015)	LDA + Topic modeling	Silhouette: 0.55
Wang, H.; Shi, Y. (2018)	TF-IDF + Hierarchical Clustering	Silhouette: 0.52
Nguyen, C. (2021)	Clustering-based web service recommendation	Not Reported

1.8 Conclusion

architectural elements and standards. We explored the importance of clustering techniques in organizing web services, enabling better discovery and reusability. Through the use of TF-IDF for textual feature extraction, we demonstrated how web service descriptions can be transformed into numerical representations suitable for clustering. We also discussed various evaluation metrics such as Silhouette Score and Calinski-Harabasz Index, as well as dimensionality reduction methods to enhance clustering performance. Finally, we examined state-of-the-art approaches including syntactic, semantic, and hybrid models, setting the stage for more advanced research in web service clustering.

Chapter 2

CHAPTER 2

CLUSTERING ALGORITHMS

2.1 Introduction

Artificial Intelligence (AI) refers to computer software designed to mimic human cognitive abilities, enabling machines to perform complex tasks like decision-making and language translation. Unlike traditional automated systems that follow fixed instructions, AI systems can learn and improve through experience. AI is a broad field that includes various specialized subfields [52].

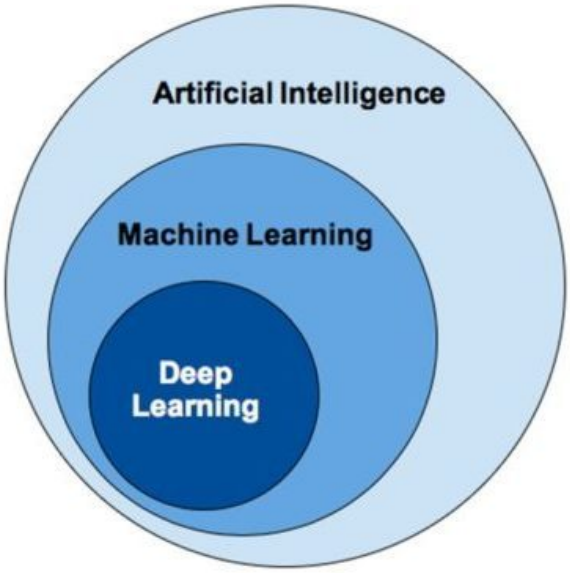


Figure 2.1: AI,ML,DL

2.2 Machine learning

It is a branch of artificial intelligence (AI) that aims to enable computers and machines to replicate human learning, carry out tasks automatically, and continuously improve their performance and accuracy through experience and exposure to growing amounts of data. Machine learning is the process of exposing a machine to large volumes of data, allowing it to learn, make predictions,

identify patterns, and classify information. The three main types of machine learning are supervised learning, unsupervised learning, and reinforcement learning [53].

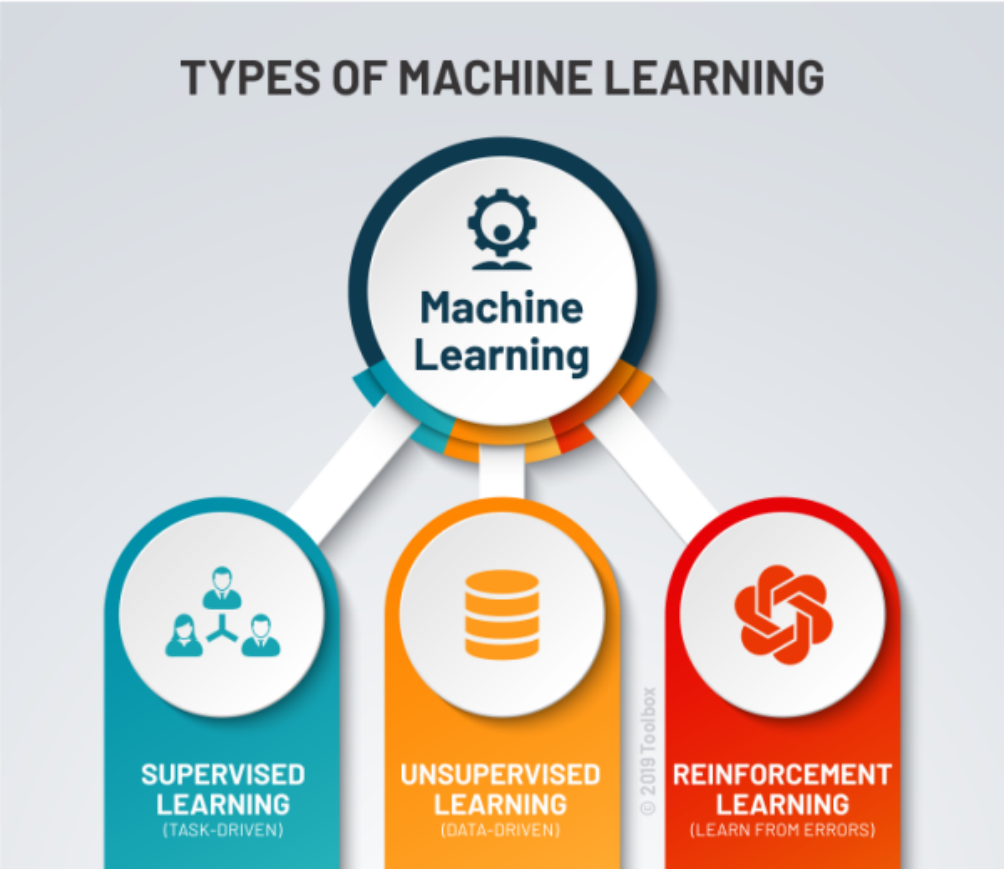


Figure 2.2: types of machine learning

2.2.1 Supervised learning

is a type of machine learning where models are trained on labeled datasets (input-output pairs) to learn the relationship between inputs (features) and outputs (labels/targets). The goal is to predict accurate outcomes for new, unseen data [54]

- **Input and Output Data:** The learning process begins with a dataset containing:
  - **Input features (X):** These are the variables or characteristics used for making predictions.
  - **Output labels (Y):** These are the correct answers the model should learn to predict.
  - **Example:** In an email classification system, the model is trained with emails (input features) labeled as either ”spam” or ”not spam” (output labels).
- **Tasks of Supervised Learning in Machine Learning**
  1. **Classification:**
     - Predicts categorical outputs (discrete labels).
     - **Examples:** Spam vs. non-spam emails, disease diagnosis (yes/no).
  2. **Regression:**

- Predicts continuous numerical outputs.
- **Examples:** House prices, stock market trends.

- **Common Supervised Learning Algorithms**
  - K-Nearest Neighbors (KNN)
  - Naive Bayes
  - Linear Regression
  - Decision Trees

2.2.2 Unsupervised Learning

Unsupervised learning is a machine learning technique that utilizes artificial intelligence (AI) algorithms to identify patterns in datasets that do not have predefined labels or classifications.

Unlike supervised learning, unsupervised models function without human guidance or preset categories during training. This makes them particularly effective for discovering hidden structures, groupings, and anomalies in unstructured data[55].

1. **Working of Unsupervised Learning**

   The working of unsupervised learning can be understood through the following diagram:

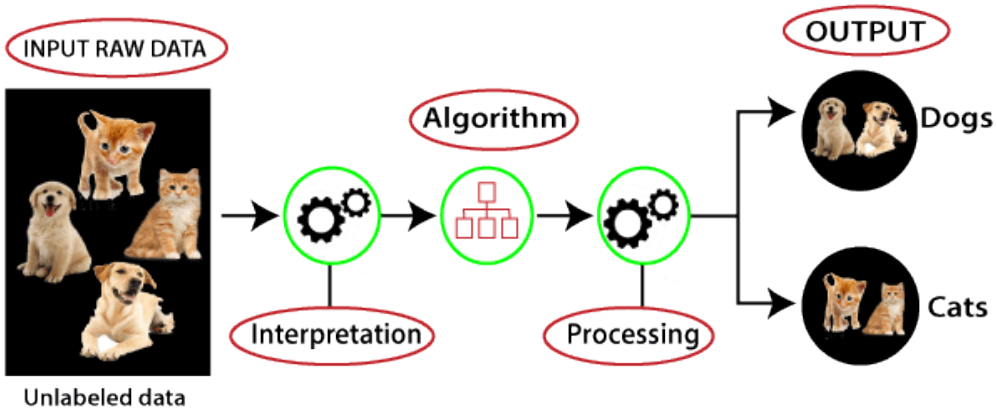


Figure 2.3: working of unsupervised Learning

2.3 Core Concepts in Clustering

Unsupervised learning primarily involves techniques such as clustering and association rule mining. These methods organize data into structures that reveal insights and patterns [56].

2.3.1 Association Rule Learning

Association rule learning is another technique within unsupervised learning used to discover interesting relationships among variables in large datasets[57].

Objective: To identify dependencies among variables and generate meaningful rules from large-scale, unlabeled data.

Popular algorithms:

- Apriori Algorithm
- Eclat Algorithm
- FP-Growth Algorithm

2.3.2 Clustering

Clustering is the process of dividing a dataset into subsets (clusters) where data points within each subset share similar characteristics. Unlike classification, clustering does not rely on predefined labels[58].

1. **Goals of Clustering**

   - **Discover structure:**Reveal inherent patterns in the data.
   - **Simplify data:**Reduce the complexity of large datasets by summarizing them with representative clusters.
   - **Enable insight:**Facilitate understanding of data distributions and relationships.
   - **Improve efficiency:**Enhance the performance of subsequent tasks such as retrieval, classification, or recommendation.

2. **Applications in Web Services**

   - **Service Discovery:**Automatically organize services into thematic or functional groups.
   - **Service Composition:**Identify compatible services based on clustered capabilities.
   - **Recommendation Systems:** Suggest services based on cluster proximity.
   - **Ontology Construction:** Build semantic groupings that reflect domain knowledge.

2.3.3 Distance Metrics and Similarity Measures

The quality of clustering heavily depends on the measure used to assess similarity or dissimilarity between data points. These metrics define how the distance or closeness of two data points is calculated, which directly influences the resulting cluster structures [59].

1. **Common Measures**

   - **Euclidean Distance:** Measures the straight-line distance between two points in a multidimensional space. Suitable for continuous numeric data[60].
   - **Cosine Similarity:** Measures the cosine of the angle between two vectors in a vector space. Commonly used for textual data represented by TF-IDF vectors, as it focuses on orientation rather than magnitude [60].
   - **Jaccard Similarity:**Evaluates similarity based on shared elements over the union of elements. Suitable for binary or set-like data[60].
   - **Manhattan Distance:**Calculates the sum of absolute differences between dimensions. Often used when dealing with high-dimensional or grid-like data [60].
   - **Semantic Similarity**Goes beyond surface forms and compares meanings of terms using ontologies or semantic networks, particularly effective in textual clustering tasks[60].

2.4 Clustering Algorithms

Clustering algorithms are the core tools in unsupervised learning used to discover groups of similar objects within a dataset. These algorithms operate based on a similarity or distance metric and aim to optimize a specific objective function, such as intra-cluster cohesion and inter-cluster separation. In the context of Web Services Clustering, these methods help organize services based on their functional or semantic similarities derived from textual descriptions like WSDL files [57]. The main families of clustering algorithms include:

2.4.1 K-means Clustering

K-means is a partitioning algorithm that divides a dataset into K pre-specified clusters by minimizing the variance (sum of squared distances) between data points and their assigned cluster centroids[58].

How it Works

- Initialize K centroids (randomly or using methods like K-means++).
- Assign each data point to the nearest centroid based on a distance metric (typically Euclidean).
- Update centroids by computing the mean of all points assigned to each cluster.
- Repeat assignment and update steps until centroids stabilize or a maximum number of iterations is reached.

Key Parameters

- K: Number of clusters (must be chosen beforehand).
- Distance metric (e.g., Euclidean, Manhattan).

Strengths

- Simple and computationally efficient (O(nKi), where n is the number of points, K is clusters, i is iterations).
- Works well for spherical, well-separated clusters.
- Scales well to large datasets.

Weaknesses

- Requires pre-specifying K (can use methods like the elbow method to estimate).
- Sensitive to outliers, as they can skew centroids.
- Assumes clusters are spherical and of similar size, which may not hold for complex data.
- Sensitive to initial centroid placement (random initialization can lead to suboptimal solutions).

Use Cases

- Image segmentation (grouping pixels by color).
- Customer segmentation (e.g., grouping by purchasing behavior).

- Document clustering (grouping similar texts).

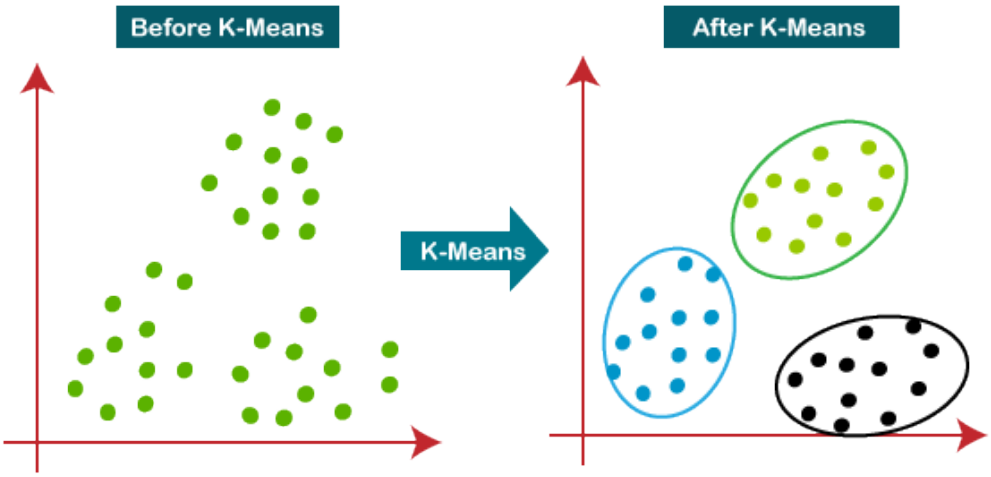


Figure 2.4: Unsupervised Learning: K-Means Clustering Process

2.4.2 Hierarchical Clustering

Builds a hierarchy of clusters either by progressively merging smaller groups (agglomerative) or by splitting larger ones (divisive), resulting in a dendrogram structure[58].

How it Works

- **Agglomerative (Bottom-Up)**
  - Start with each data point as its own cluster.
  - Compute pairwise distances (e.g., Euclidean) between clusters.
  - Merge the two closest clusters based on a linkage criterion (e.g., single, complete, average, or Ward's method).
  - Repeat until all points are in one cluster or a stopping criterion is met.
- **Divisive (Top-Down)**
  - Start with all points in one cluster.
  - Recursively split clusters (e.g., using K-means or other methods) until each point is its own cluster or a stopping criterion is met.

  The dendrogram shows the merging/splitting order and can be cut at a desired level to obtain clusters.
  Key Parameters
- **Linkage Criterion:** Determines how distance between clusters is measured:
  - Single: Minimum distance between points in clusters.
  - Complete: Maximum distance.

  - Average: Average distance.
  - Ward's: Minimizes variance increase when merging clusters.

- Distance metric (e.g., Euclidean, Manhattan, cosine).
- Number of clusters (if cutting the dendrogram).
  Strengths
  - No need to specify the number of clusters upfront.
  - Dendrogram provides a visual hierarchy, useful for understanding data structure.
  - Can capture non-spherical clusters (depending on linkage).

  Weaknesses
  - Computationally expensive ($O(n^2)$ for agglomerative, $O(n^3)$ for some implementations).
  - Sensitive to noise and outliers (especially single linkage).
  - Once clusters are merged/split, decisions are irreversible (no reassignment).

  Use Cases
  - Taxonomy creation (e.g., biological species classification).
  - Social network analysis (grouping users by connections).
  - Gene expression analysis (grouping genes with similar expression patterns).

2.4.3 Density-Based Clustering (DBSCAN)

Identifies clusters in regions of high data density and classifies isolated points as noise[58]. **How it Works**

- Define two parameters: ϵ(radius of neighborhood) and MinPts (minimum number of points to form a dense region).
- Label points as:
  - **Core points:**Have at least MinPts points (including itself) within distance .
  - **Border points:** Within of a core point but have fewer than MinPts neighbors.
  - **Noise points:**Neither core nor border points.
  - Start with a core point, form a cluster by including all density-connected points (core points within ϵ of each other and their border points).
  - Repeat for unvisited core points; unassigned points are noise.

Key Parameters

- ϵ: Maximum distance for points to be considered neighbors.
- MinPts: Minimum points to define a core point.

Strengths

- Automatically detects the number of clusters (no need to specify K).
- Can find arbitrarily shaped clusters.
- Robust to outliers (marks them as noise).
- No assumption about cluster shape or size.

Weaknesses

- Sensitive to and MinPts choice (poor choices can lead to under- or over-clustering).
- Struggles with varying density clusters (single may not suit all regions).
- Computationally intensive for large datasets ($O(n^2)$ without indexing, $O(n \log n)$ with spatial indexing).

Use Cases

- Anomaly detection (e.g., identifying fraudulent transactions).
- Spatial data analysis (e.g., grouping geographic points like crime hotspots).
- Image processing (e.g., identifying object boundaries).

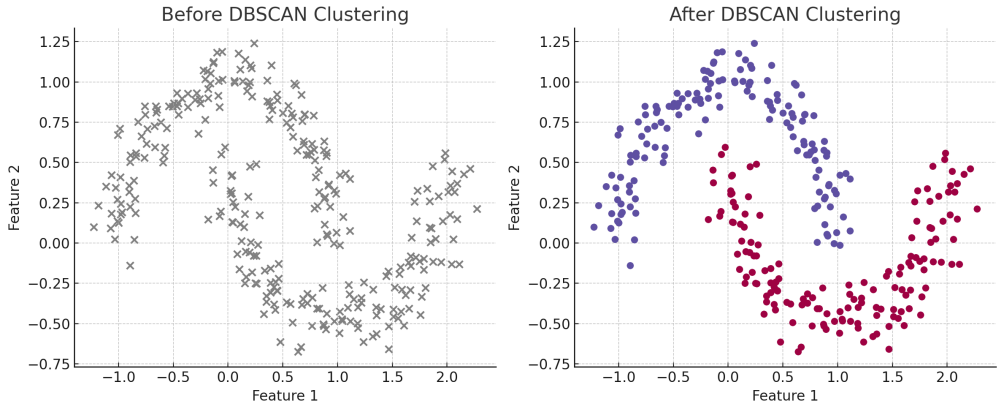


Figure 2.5: DBSCAN Clustering: Before and After

2.4.4 Mean-Shift Clustering

Mean-Shift Clustering locates clusters by shifting points toward the densest regions in the data without predefining the number of clusters[58].

How it Works

- For each data point, define a window (bandwidth) around it.
- Compute the mean of the points within the window (weighted by a kernel, e.g., Gaussian).
- Shift the point to this mean (this is the "mean shift" step).
- Repeat the process until convergence (i.e., points move to local density maxima).

- Points that converge to the same mode form a cluster.

Key Parameters

- **Bandwidth:** Size of the window (kernel radius). Can be fixed or adaptive.
- Kernel type (e.g., Gaussian, flat).

Strengths

- No need to specify the number of clusters.
- Can detect arbitrarily shaped clusters.
- Robust to outliers (they don't converge to dense regions).
- Non-parametric, flexible to data distribution.

Weaknesses

- Computationally expensive ($O(n^2)$ per iteration without optimization).
- Sensitive to bandwidth choice (too large: merges clusters; too small: creates too many).
- Struggles with high-dimensional data due to sparse density.

Use Cases

- Image segmentation (e.g., grouping pixels by color or texture).
- Object tracking in computer vision.
- Clustering spatial data (e.g., geographic points).

2.4.5 Spectral Clustering

Spectral Clustering forms clusters based on relationships between data points using graph theory and the eigenvalues of similarity matrices [58].

How it Works

- Construct a similarity graph where nodes are data points, and edges represent similarity (e.g., based on Gaussian kernel: $\exp(-\|x_i - x_j\|^2/2\sigma^2)$).
- Compute the Laplacian matrix ($L = D - W$, where W is the similarity matrix, D is the degree matrix).
- Calculate the k smallest eigenvectors of the Laplacian (excluding the trivial eigenvector).
- Use these eigenvectors as a lower-dimensional representation of the data.
- Apply a clustering algorithm (e.g., K-means) to the eigenvector space to form clusters.

Key Parameters

- k: Number of clusters.
- σ: Scaling parameter for the similarity kernel.
- Graph construction method (e.g., k-nearest neighbors, fully connected).

Strengths

- Excels at finding non-convex, complex-shaped clusters.
- Effective for high-dimensional data (after dimensionality reduction via eigenvectors).
- Captures global structure through graph connectivity.

Weaknesses

- Computationally expensive ($O(n^3)$ for eigenvalue decomposition, though approximations exist).
- Requires specifying the number of clusters.
- Sensitive to similarity measure and σ (poor choices can degrade performance).
- Memory-intensive for large datasets due to the similarity matrix.

Use Cases

- Image segmentation (e.g., separating objects with complex boundaries).
- Social network analysis (e.g., community detection).
- Text clustering (e.g., grouping documents by topic).

2.5 Advantages and Disadvantages

1. **Advantages of unsupervised learning**

    - We can use these algorithms for complicated tasks as compared to the supervised ones because these algorithms work on unlabeled datasets and do not require large datasets [61].
    - Unsupervised algorithms are preferable for various tasks as getting the unlabeled dataset is easier as compared to the labeled dataset for the training of these algorithms [61].

2. **Disadvantages of Unsupervised Learning**

    - An unsupervised algorithm may result in less accurate outputs as the dataset is not labeled, and we have not trained the algorithms with the exact output beforehand[62].
    - Working with Unsupervised learning is more difficult as compared to other types as it works with the unlabelled dataset that does not map with the output [62].

2.6 Conclusion

In this chapter, we introduced a branch of AI by presenting its definition and explaining its different types: supervised, unsupervised. This theoretical background is essential for understanding how intelligent systems can analyze data, learn from it, and improve performance over time—concepts that will be directly applied in the clustering of web services in the following chapters

Chapter 3

CHAPTER 3

METHODOLOGY AND IMPLIMENTATION

3.1 Introduction

In this chapter, we present the architecture designed for filtering and clustering Web services. Before implementing the final application, several essential steps were taken to prepare the data and define the appropriate analytical approach. These steps included the collection of Web Services data, the cleaning and preprocessing of the dataset, and the selection of relevant features such as inputs, outputs, and domain-specific terms. Additionally, a thorough exploration of text analysis techniques was performed, particularly focusing on the application of TF-IDF (Term Frequency-Inverse Document Frequency) to identify significant keywords within service descriptions. To effectively group similar services, different clustering algorithms such as K-Means, DBSCAN, and Hierarchical Clustering were tested and evaluated based on their performance and interpretability. This groundwork provided a strong foundation for designing a modular and efficient architecture capable of supporting interactive analysis, filtering, and classification of Web Services within a web-based application.

3.2 Tools and Technologies

To develop this system, we used the following technologies:

- **Programming Language:** Python 3.7.4
- **Framework:**Flask (for backend routing and web interaction)
- **IDE:**PyCharm (for development)
- **Frontend:**Tailwind CSS + HTML
- **Data Handling:**pandas, numpy
- **Machine Learning:**scikit-learn (TF-IDF, clustering)
- **Visualization:**matplotlib, plotly
- **Other Libraries:**os, io, base64 (for image rendering)

3.3 Dataset Description and Structure

The dataset used in this project contains a collection of Web Services categorized by domain. Each Web Service is described using four key attributes: the file name, the domain (or Domaine), the input, and the output. These attributes provide a structured representation of the functional and contextual aspects of each service. The dataset is organized in a tabular format, where:

- **File:**Refers to the OWL-S file name describing the Web Service.
- **Domaine:** Indicates the thematic category (e.g., communication, travel, etc.).
- **Input:** Describes the parameters or data expected by the service.
- **Output:**Describes the result or response returned by the service. This structured format allows efficient text processing and analysis using Natural Language Processing (NLP) techniques such as TF-IDF for vectorization, followed by clustering algorithms for classification.

  Figure 3.33 below illustrates a sample of the dataset, focusing on services in the communication domain.

	File,Domaine,Input,output
1	
2	\communication\title_sciencefictionfilmrecommendedpricequality_service (1).owls,communication,Title,Science fiction film Recommended price Quality
3	\communication\title_sciencefictionfilmrecommendedpricequality_service (10).owls,communication,Title,Action film Tax free price Quality
4	\communication\title_sciencefictionfilmrecommendedpricequality_service (11).owls,communication,Title,Video Tax free price Quality
5	\communication\title_sciencefictionfilmrecommendedpricequality_service (12).owls,communication,Title,Film Recommended price Quality
6	\communication\title_sciencefictionfilmrecommendedpricequality_service (13).owls,communication,Title,Film Recommended price Quality
7	\communication\title_sciencefictionfilmrecommendedpricequality_service (14).owls,communication,Title,Action film Taxed price Quality
8	\communication\title_sciencefictionfilmrecommendedpricequality_service (15).owls,communication,Title,Comedy film Taxed price Quality
9	\communication\title_sciencefictionfilmrecommendedpricequality_service (16).owls,communication,Title,Video Taxed price Quality
10	\communication\title_sciencefictionfilmrecommendedpricequality_service (17).owls,communication,Title,Action film Max price Quality
11	\communication\title_sciencefictionfilmrecommendedpricequality_service (18).owls,communication,Title,Comedy film Max price Quality
12	\communication\title_sciencefictionfilmrecommendedpricequality_service (19).owls,communication,Title,Video Max price Quality
13	\communication\title_sciencefictionfilmrecommendedpricequality_service (2).owls,communication,Title,Science fiction film Tax free price Quality
14	\communication\title_sciencefictionfilmrecommendedpricequality_service (20).owls,communication,Title,Film Tax free price Quality
15	\communication\title_sciencefictionfilmrecommendedpricequality_service (21).owls,communication,Linguistic expression,User
16	\communication\title_sciencefictionfilmrecommendedpricequality_service (22).owls,communication,Title,Film Tax free price Quality
17	\communication\title_sciencefictionfilmrecommendedpricequality_service (23).owls,communication,Title,Film Taxed price Quality
18	\communication\title_sciencefictionfilmrecommendedpricequality_service (24).owls,communication,Title,Action film Price Quality
19	\communication\title_sciencefictionfilmrecommendedpricequality_service (25).owls,communication,Title,Comedy film Price Quality
20	\communication\title_sciencefictionfilmrecommendedpricequality_service (26).owls,communication,Title,Film Taxed price Quality
21	\communication\title_sciencefictionfilmrecommendedpricequality_service (27).owls,communication,Title,Video Price Quality
22	\communication\title_sciencefictionfilmrecommendedpricequality_service (28).owls,communication,Title,Film Max price Quality
23	\communication\title_sciencefictionfilmrecommendedpricequality_service (29).owls,communication,Title,Film Max price Quality
24	\communication\title_sciencefictionfilmrecommendedpricequality_service (3).owls,communication,Title,Science fiction film Taxed price Quality
25	\communication\title_sciencefictionfilmrecommendedpricequality_service (30).owls,communication,Title,Film Price Quality
26	\communication\title_sciencefictionfilmrecommendedpricequality_service (31).owls,communication,Title,Film Price Quality
27	\communication\title_sciencefictionfilmrecommendedpricequality_service (32).owls,communication,Title,Cd Price Software

Figure 3.1: Example Rows from the Web Services Dataset

3.4 Workflow Proposed

This diagram illustrates a data preprocessing workflow for web services clustering. It shows five sequential steps that transform raw service descriptions into structured numerical data. The process starts with basic text cleaning, creates a TF-IDF matrix for numerical representation, extracts synonyms using WordNet, builds a synonym mapping system, and finally merges related terms to reduce redundancy. The workflow addresses the challenge of semantic variations in service descriptions by consolidating similar terms like ”film” and ”movie” into single features, resulting in cleaner data that improves clustering accuracy.

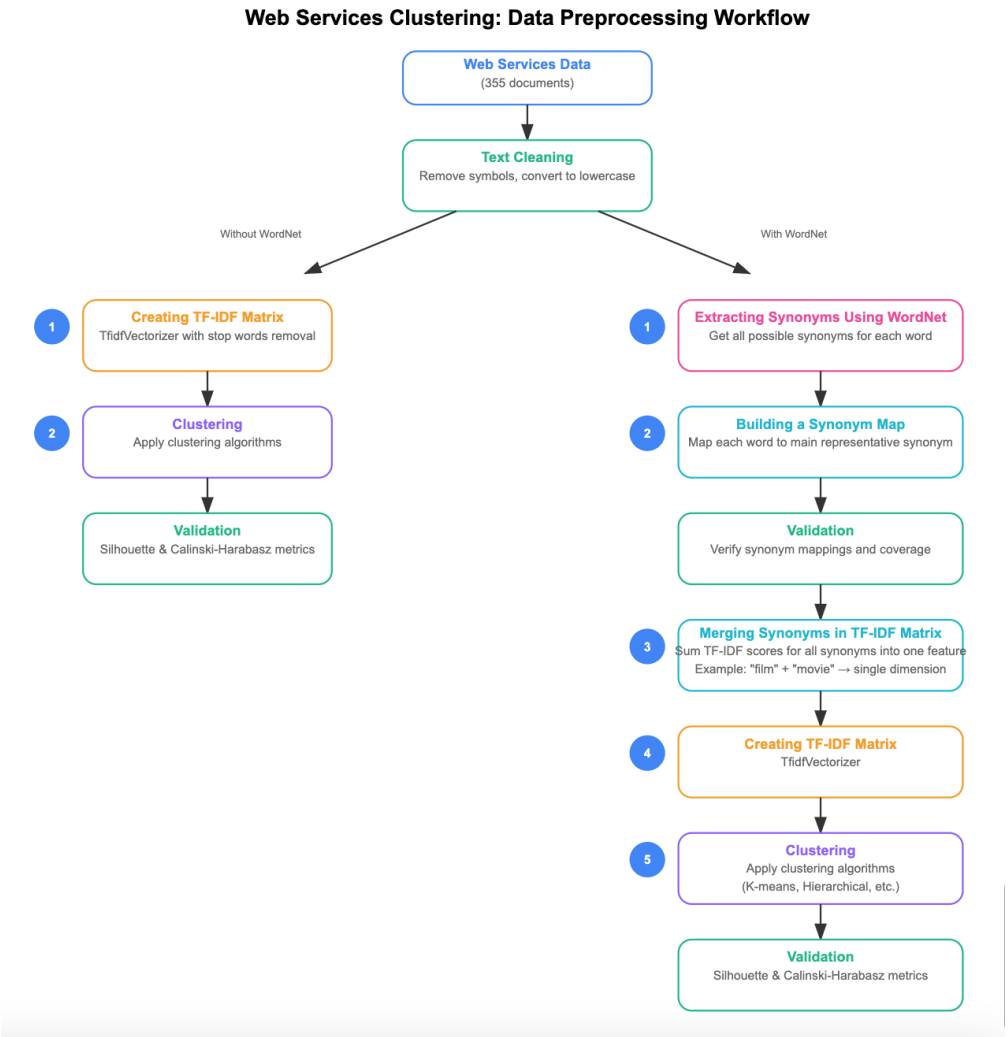


Figure 3.2: Workflow schema

3.5 Data Cleaning and Preprocessing

The provided dataset contained raw textual information extracted from OWL-S Web Services files. Before applying any analytical or machine learning techniques, it was necessary to clean and standardize the data to ensure consistency and accuracy. The main preprocessing steps included:

1. **Text Cleaning (Preprocessing):** All symbols are removed except letters, and all characters are converted to lowercase to unify the input.

    - **stop_words='english'**: Removes common words like "the", "and", etc.
    - **`min_df=2`**: Ignores terms that appear in fewer than 2 documents.

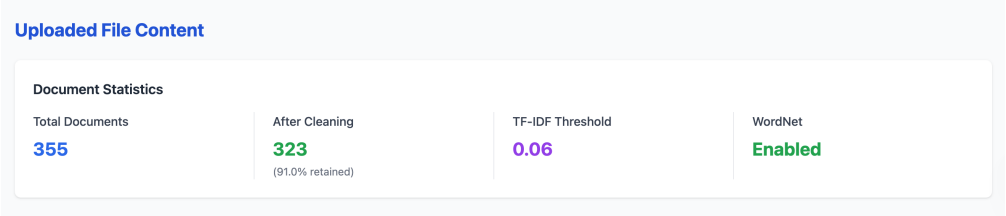


Figure 3.3: Results of Cleaning data

3.6 Clustering without Reduction

In the second phase of the Web Services Clustering application, we implement unsupervised clustering algorithms to discover latent structures within the processed TF-IDF representations of services. This step is critical for grouping semantically similar web services based on their textual descriptions

3.6.1 Features extruction by TF-IDF

In this phase, we extract features directly from the raw text using the TfidfVectorizer from Scikit-learn, without applying any semantic enrichment such as WordNet. This approach focuses purely on the statistical importance of terms across the corpus, based on their frequency and inverse document frequency. It allows us to identify and weigh the most relevant terms while ignoring common or less informative words. After applying TF-IDF to the input data in our application, a total of 34 distinct features were extracted. These features correspond to the most significant terms present across the selected domain, and they form the basis of the document-term matrix used for further analysis. The following screenshot (see below) illustrates the resulting set of features, showing the terms selected and their respective TF-IDF weights. This output serves as a baseline for comparison with the feature extraction process that incorporates semantic enrichment via WordNet. In this step, a **threshold** value of 0.06 was applied to filter out less relevant terms, and the resulting features are shown below.

DOCUMENT	APPLE	APPLE FOOD	BISCUIT	BISCUIT QUANTITY	BREAD	BREAD BISCUIT	BUTTER	BUTTER QUANTITY	FOOD FOOD	FOOD QUANTITY	PREPARED	PREPAR
Doc 1	0	0	1.0000	1.0000	1.0000	1.0000	0	0	0	0	0	
Doc 2	0	0	0	0	0	0	0	0	0	0	0	
Doc 3	0	0	0	0	0	0	0.5539	1.0000	0	0	0	
Doc 4	0	0	0	0	0	0	0.5539	1.0000	0	0	0	
Doc 5	0	0	0	0	0	0	0	0	0	1.0000	0	
Doc 6	0	0	0	0	0	0	1.0000	0	0	0	0	
Doc 7	0	0	0	0	0	0	0	0	0	0	0	
Doc 8	0	0	0	0	0	0	0	0	0	0	0	
Doc 9	0	0	0	0	0	0	0	0	0.6347	0	1.0000	1.0
Doc 10	0	0	0	0	0	0	0	0	0.6347	0	1.0000	1.0
Doc 11	0	0	1.0000	1.0000	1.0000	1.0000	0	0	0	0	0	

Figure 3.4: Results TF-IDF without WordNet

3.6.2 Clustering step

In this step, clustering algorithms are applied to group similar web service descriptions based on their extracted features, such as TF-IDF vectors. The goal is to discover underlying patterns and

relationships within the data that are not explicitly labeled. By organizing services into coherent clusters, this process facilitates better understanding, analysis, and classification. Common clustering techniques used include KMeans, DBSCAN, and Hierarchical Clustering.

1. Clustering by kmeans

    - **Objective:** Partition documents into k clusters that minimize intra-cluster variance.
    - **Functionality:**Utilizes the apply_kmeans() function, which applies the KMeans algorithm with a predefined number of clusters.
    - **Visualization:**Cluster distribution is illustrated using bar charts.

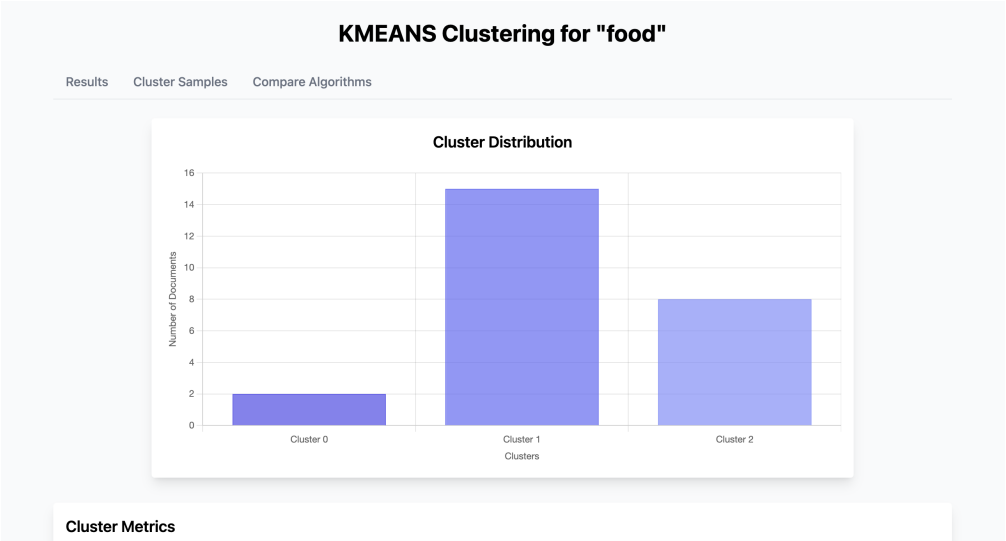


Figure 3.5: K-means Bar char results

The bar chart in Figure 3.5 presents the clustering distribution of documents in the ”food” domain using the K-Means algorithm. The clustering was performed without incorporating WordNet-based semantic preprocessing. The documents were grouped into three clusters, with the following distribution:

Cluster 0: 2 documents

Cluster 1: 15 documents

Cluster 2: 8 documents

The results show a clear imbalance between clusters, with Cluster 1 containing the majority of the documents. This suggests that most documents share highly similar surface-level vocabulary, which allowed them to be grouped together. Conversely, Clusters 0 and 2 contain fewer documents, possibly indicating outliers or documents with distinct lexical patterns.

It is important to note that this clustering was done without using WordNet, which means that semantic relationships between words (such as synonyms or lemmas) were not taken into account. As a result, documents that are semantically similar but use different wording may have been placed in separate clusters. For example, words such as ”meal”, ”dish”, and ”cuisine” may be treated as unrelated, despite referring to the same concept.

The absence of WordNet likely led to:

Increased feature sparsity due to unmerged synonyms.

Reduced semantic cohesion within clusters.

A higher possibility of conceptually similar documents being split across clusters.

Overall, while the clustering shows some coherent grouping, the lack of semantic normalization reduces the overall quality and interpretability of the results. Integrating WordNet in future steps is expected to improve clustering performance by unifying conceptually related terms and enhancing the semantic representation of documents.

2. **Clustering By DBSCAN**

- **Objective:**Discover arbitrarily shaped clusters based on density without predefining the number of clusters.
- **Functionality:**The apply_dbscan() function automatically identifies core points, border points, and noise.

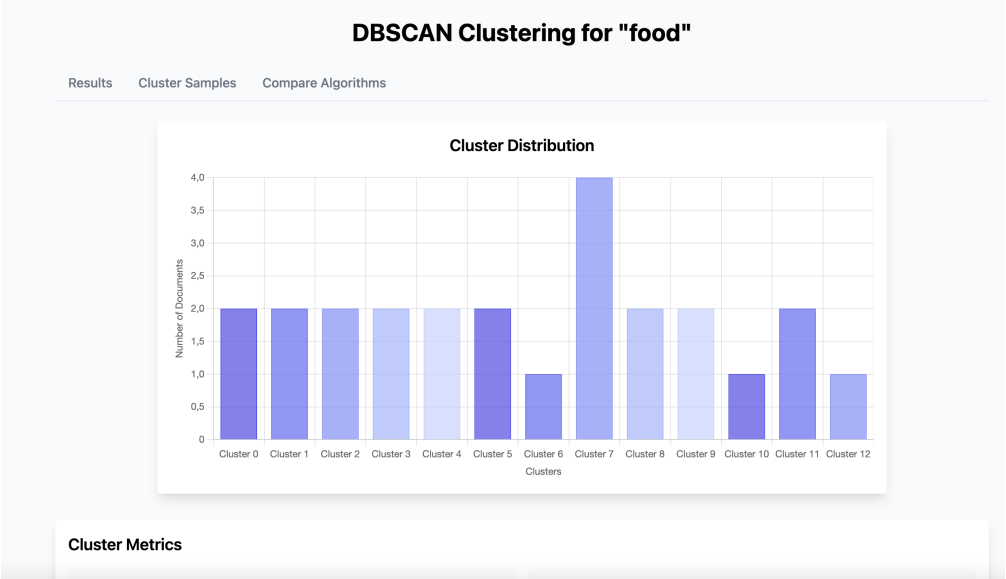


Figure 3.6: DBSCAN Bar char results

Figure 3.6 presents the clustering output obtained using the DBSCAN algorithm applied to the ”food” domain, without the incorporation of semantic enrichment via WordNet. The resulting distribution reveals a total of 13 clusters (Cluster 0 to Cluster 12), with the number of documents per cluster ranging from 1 to 4. Notably, most clusters contain only 1 or 2 documents, with Cluster 7 being the most populated (4 documents).

This high degree of fragmentation indicates that DBSCAN identified numerous low-density regions in the feature space, each forming a separate cluster. The absence of WordNet likely contributed to this outcome, as semantically similar terms (e.g., synonyms or variations) were treated as distinct features. Consequently, documents that are conceptually related may have appeared lexically dissimilar, preventing them from being grouped together.

Moreover, the lack of synonym normalization reduced the overall density of document vectors, which is critical for DBSCAN’s performance. As a density-based algorithm, DBSCAN requires closely grouped data points to form meaningful clusters. In sparse feature spaces, where semantically related documents use different vocabularies.

These results highlight a key limitation of unsupervised clustering on raw textual data: the reliance on surface-level lexical similarities. Without semantic preprocessing techniques such

as those offered by WordNet, clustering algorithms may fail to capture the true thematic structure of the data.

In summary, the DBSCAN results underscore the importance of integrating semantic knowledge to improve cluster cohesion and reduce fragmentation. Subsequent experiments that incorporate WordNet are expected to yield more semantically meaningful and compact clusters.

3. **Agglomerative (Hierarchical) Clustering:**

   - **Objective:** Create a nested hierarchy of clusters using a bottom-up (agglomerative) approach.
   - **Visualization:** The results are visualized via a dendrogram using plot_dendrogram().

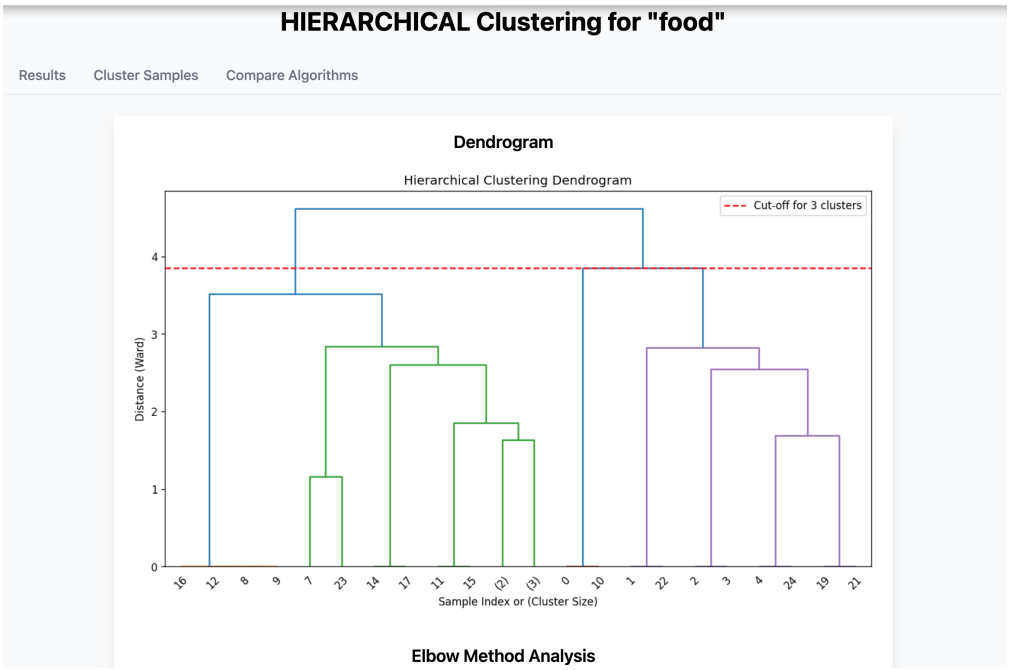


Figure 3.7: Hierarchical Bar char results

Figure 3.7 illustrates the hierarchical clustering dendrogram generated for the ”food” domain. The clustering was performed using an agglomerative approach, and the dendrogram visualizes the hierarchical relationships between documents based on their pairwise distances.

The vertical axis (Distance) represents the linkage distance (Ward’s method), and the horizontal axis corresponds to individual documents or merged clusters. A red dashed line marks the cut-off threshold, resulting in the identification of three major clusters. This cut-off was determined using the elbow method, balancing intra-cluster compactness and inter-cluster separation.

The dendrogram structure reveals how individual documents are iteratively merged into clusters, with shorter branches indicating higher similarity between merged items. The three primary clusters formed after the cut-off suggest that, unlike DBSCAN, the hierarchical method achieves better document grouping, potentially due to its ability to detect nested cluster structures and adapt to variable densities.

However, similar to other clustering algorithms, the absence of semantic enrichment may still lead to the separation of conceptually related items if their lexical features are sparse or

inconsistent. Despite this, the hierarchical model offers a more interpretable representation of document relationships compared to flat clustering methods.

This dendrogram provides valuable insight into the internal cluster composition, supporting qualitative analysis and offering flexibility in selecting the number of clusters based on analytical needs.

3.6.3 Evaluation step

To assess the quality of the clustering results, we applied an evaluation phase using internal validation metrics. This step aims to measure how well the clustering algorithms grouped semantically similar web services based on the TF-IDF features. The results obtained through this evaluation will help in comparing the effectiveness of each algorithm. In our case, we used the Silhouette Score and Calinski-Harabasz Index, which are discussed in detail in a separate chapter.

ALGORITHM	SILHOUETTE	CALINSKI-HARABASZ	CLUSTERS
KMEANS	0.330	7.5	3
DBSCAN	0.880	1.0	13
HIERARCHICAL	0.330	7.5	3

Figure 3.8: Evaluation results without reduction

3.6.4 Comparison of Clustering Algorithms

An extended analysis is available through the compare_clustering_algorithms() function, which applies and evaluates all three clustering algorithms on the same data:

Visual Analytics : a comparative visual plot is generated to help users select the most appropriate clustering method for their data:

- **Purpose:**Allows intuitive comparison of clustering performance across algorithms
- **Outcome:**Supports evidence-based selection of the optimal clustering technique.

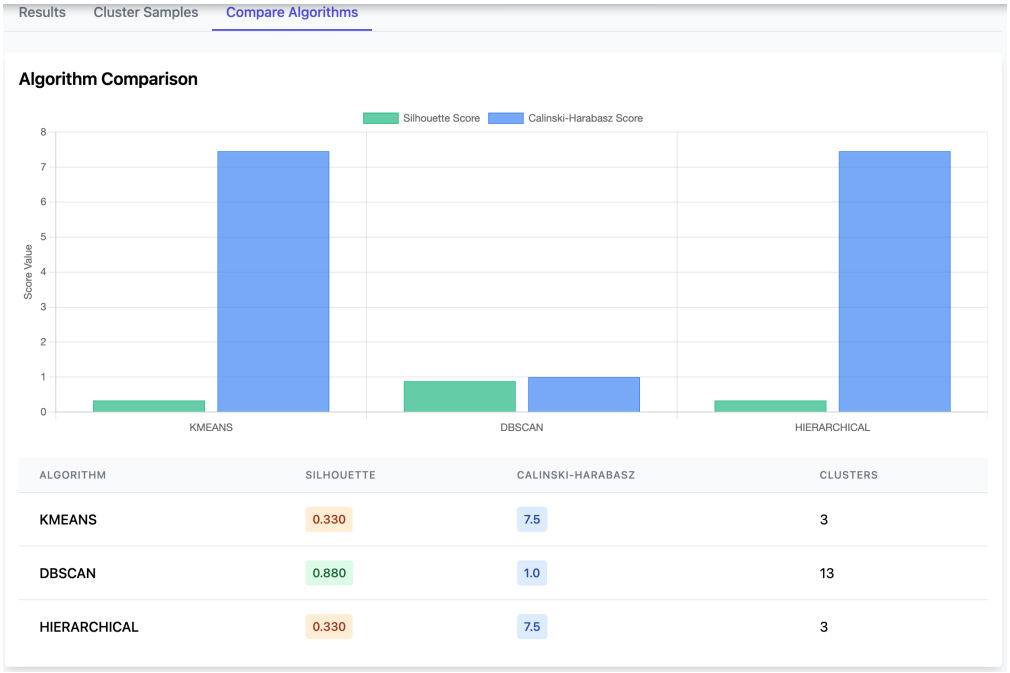


Figure 3.9: Comparison results

- **Cluster Samples** The figures below illustrate representative samples of web services grouped into three distinct clusters using the KMeans clustering algorithm. Each cluster highlights services that share similar features based on their input and output descriptions.

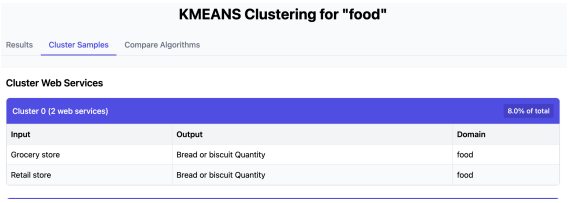


Figure 3.10: Web Services in Cluster 0

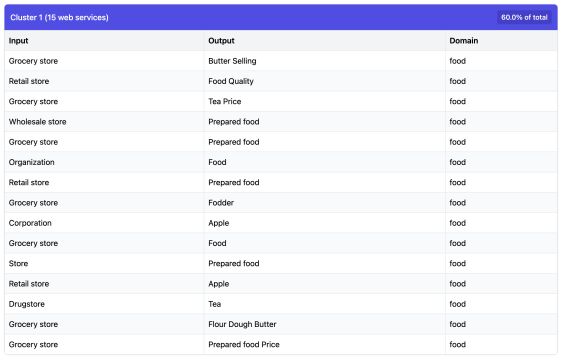


Figure 3.11: Web Services in Cluster 1

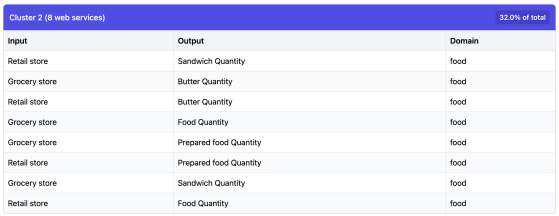


Figure 3.12: Web Services in Cluster 2

3.7 Clustering with Reduction

In the second phase of the Web Services Clustering application, we implement unsupervised clustering algorithms to discover latent structures within the processed TF-IDF representations of services.

This step is critical for grouping semantically similar web services based on their textual descriptions

3.7.1 Features extruction using TF-IDF with WordNet

1. **Extracting Synonyms Using WordNet:**
   We use NLTK's WordNet corpus to extract all possible synonyms for each word rpus to extract all possible synonyms for each word
   Example:for the word film, we get synonyms like movie, etc.

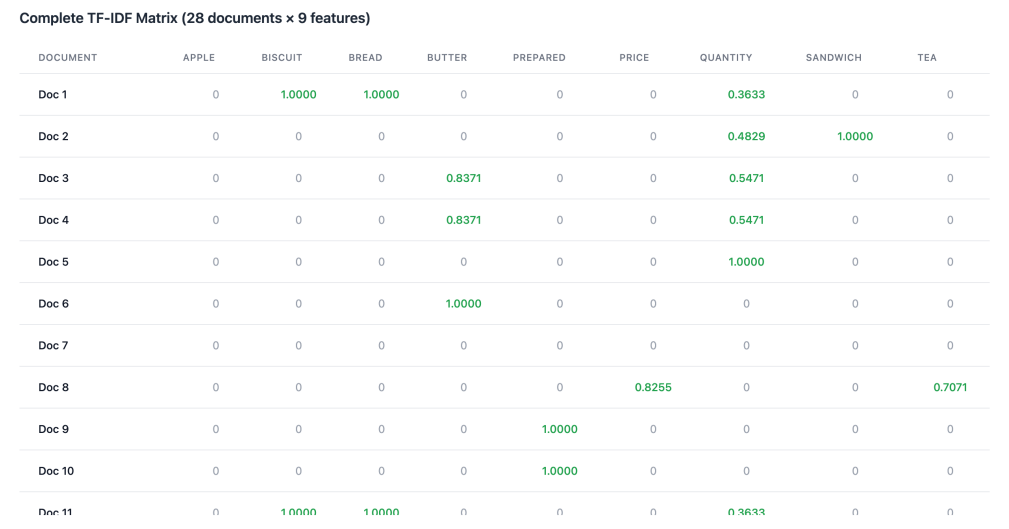
2. **Building a Synonym Map:** A dictionary is created to map each word to a main representative synonym (we choose the shortest one)

3. **Merging Synonyms in the TF-IDF Matrix:** For each group of synonyms, we sum their TF-IDF scores into one feature

4. **Applying a Threshold**

   To reduce noise and eliminate low-importance terms, we apply a threshold. In our implementation, a threshold of 0.06 was used, meaning any value below this threshold is set to zero
   Expected Results:

   - A more semantic numeric representation instead of pure word-based.
   - Dimensionality reduction by merging semantically similar words.
   - Enhanced performance and clarity in later classification/clustering steps



Complete TF-IDF Matrix (28 documents × 9 features)

DOCUMENT	APPLE	BISCUIT	BREAD	BUTTER	PREPARED	PRICE	QUANTITY	SANDWICH	TEA
Doc 1	0	1.0000	1.0000	0	0	0	0.3633	0	0
Doc 2	0	0	0	0	0	0	0.4829	1.0000	0
Doc 3	0	0	0	0.8371	0	0	0.5471	0	0
Doc 4	0	0	0	0.8371	0	0	0.5471	0	0
Doc 5	0	0	0	0	0	0	1.0000	0	0
Doc 6	0	0	0	1.0000	0	0	0	0	0
Doc 7	0	0	0	0	0	0	0	0	0
Doc 8	0	0	0	0	0	0.8255	0	0	0.7071
Doc 9	0	0	0	0	1.0000	0	0	0	0
Doc 10	0	0	0	0	1.0000	0	0	0	0
Doc 11	0	1.0000	1.0000	0	0	0	0.3633	0	0

Figure 3.13: Results of TF-IDF with WordNet

3.7.2 Clustering by kmeans

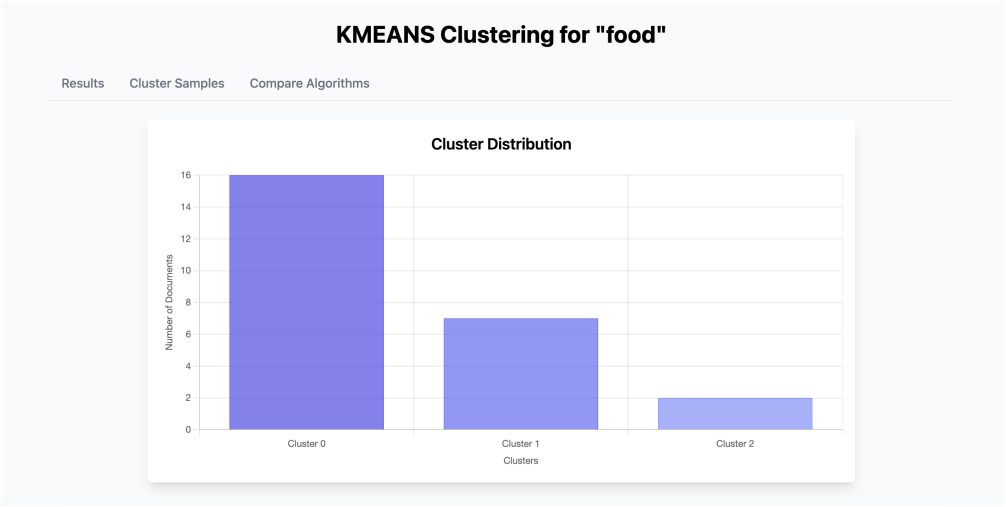


Figure 3.14: K-means Bar char results

Figure 3.14: K-means Bar Chart Results, displays the distribution of documents across clusters generated by the K-means clustering algorithm applied to a ”food” dataset, utilizing WordNet. The bar chart illustrates the number of documents in three clusters: Cluster 0, Cluster 1, and Cluster 2. Cluster 0 contains the highest number of documents, approximately 16, indicating a significant concentration of data points. Cluster 1 follows with around 7 documents, while Cluster 2 has the fewest, with approximately 2 documents. This distribution suggests that the K-means algorithm, enhanced by WordNet for semantic analysis, has effectively grouped the majority of the ”food”-related data into Cluster 0, with more balanced but smaller representations in Clusters 1 and 2. The variation in cluster sizes may reflect the diversity of food-related concepts captured by the algorithm.

3.7.3 Clustering By DBSCAN

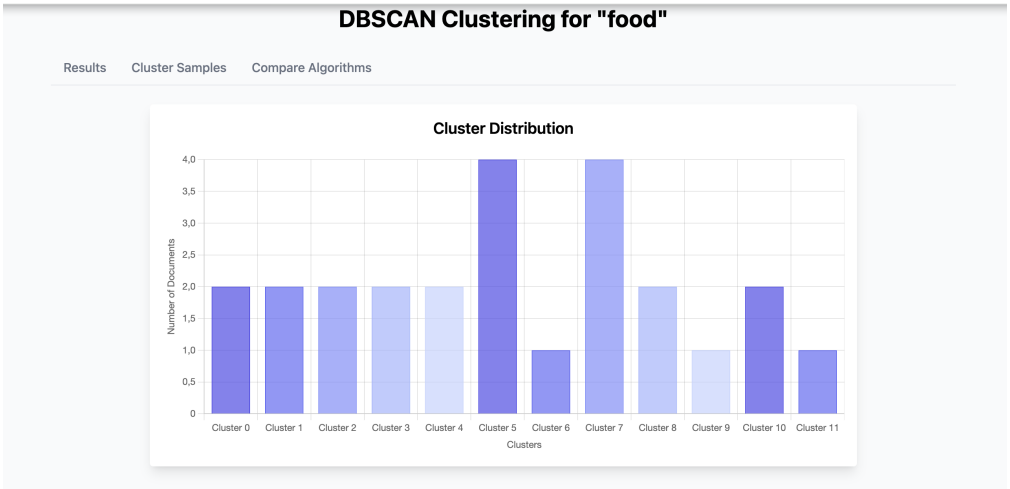


Figure 3.15: DBSCAN Bar char results

Figure 3.6 presents the distribution of documents across clusters resulting from the application of the DBSCAN clustering algorithm to the ”food” domain, enhanced with semantic similarity

measures using WordNet. The bar chart illustrates the presence of documents in 13 distinct clusters, labeled from Cluster 0 to Cluster 12.

Among these, Cluster 5 exhibits the highest density, containing approximately 3.5 documents, signifying a dominant grouping of semantically similar items. Clusters 0 through 4 and Cluster 7 each contain around 2 documents, indicating moderate concentrations. In contrast, Clusters 6, 8, 9, 10, 11, and 12 contain approximately 1 document each, suggesting the presence of either outliers or sparsely connected instances.

This distribution pattern reflects the effectiveness of DBSCAN, particularly when combined with WordNet for capturing semantic relationships. The presence of both densely populated and sparsely filled clusters demonstrates DBSCAN's ability to identify core regions of semantic similarity while simultaneously isolating less coherent or noisy data points within the dataset. This highlights the algorithm's suitability for detecting irregular cluster shapes and managing noise in semantically enriched textual data.

3.7.4 Agglomerative (Hierarchical) Clustering:

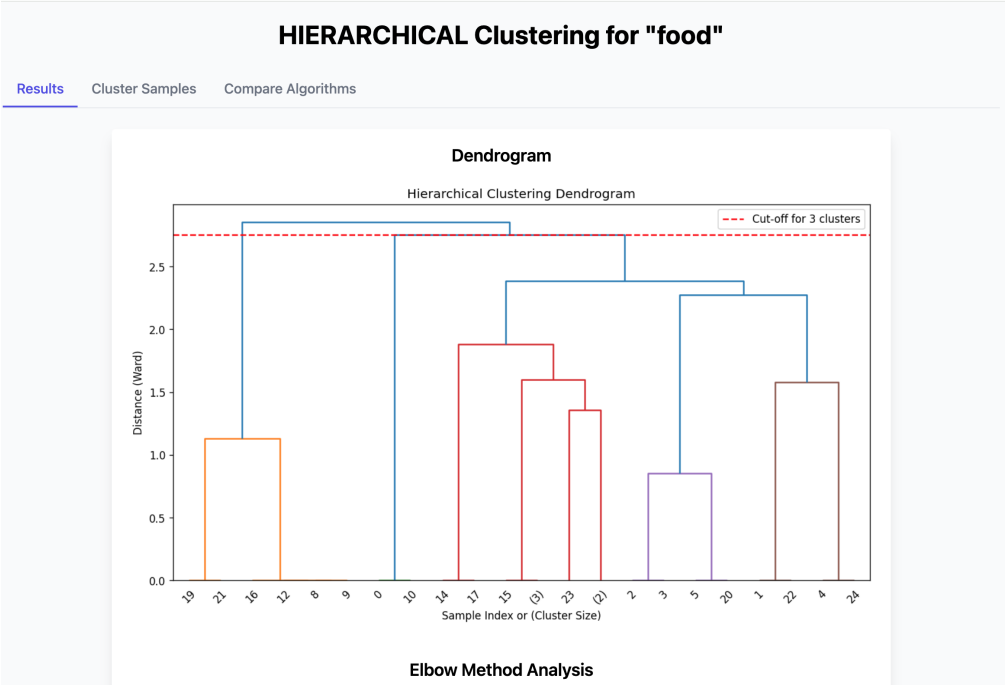


Figure 3.16: Hierarchical Bar char results

Figure 3.16 presents a dendrogram generated using the Hierarchical Clustering algorithm applied to a "food" dataset, incorporating semantic similarity through WordNet. The visualization includes an elbow method analysis and represents the hierarchical relationships between data points based on Ward's linkage distance.

The x-axis denotes the sample indices or cluster sizes, while the y-axis reflects the distance metric used to determine the dissimilarity between merged clusters. A red dashed line marks the cut-off threshold corresponding to three clusters, as determined through elbow method analysis.

The dendrogram reveals multiple merging levels, with key distance values ranging approximately from 1.0 to 2.5, indicating varying degrees of semantic similarity among the "food"-related samples. The chosen cut-off point reflects the optimal number of clusters, where the

rate of increase in inter-cluster distance begins to diminish, achieving a balance between cluster cohesion and interpretability. The use of WordNet enhances the semantic understanding of the dataset, contributing to more meaningful and contextually accurate clustering outcomes.

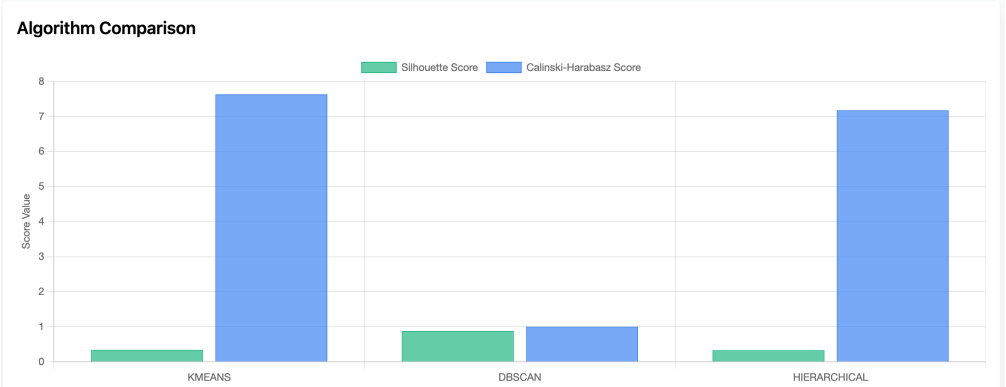
3.7.5 Evaluation step

Post-clustering, we compute quantitative evaluation metrics to assess the effectiveness of the clustering process this is the results with reduction threshold=0.06

ALGORITHM	SILHOUETTE	CALINSKI-HARABASZ	CLUSTERS
KMEANS	0.340	7.6	3
DBSCAN	0.880	1.0	12
HIERARCHICAL	0.330	7.2	3

Figure 3.17: Evalution results with reduction

3.7.6 comparison



ALGORITHM	SILHOUETTE	CALINSKI-HARABASZ	CLUSTERS
KMEANS	0.340	7.6	3
DBSCAN	0.880	1.0	12
HIERARCHICAL	0.330	7.2	3

Figure 3.18: Comparison results with reduction

- **Cluster Samples**

- **Evaluation Metrics**
  We used internal metrics to evaluate cluster quality:

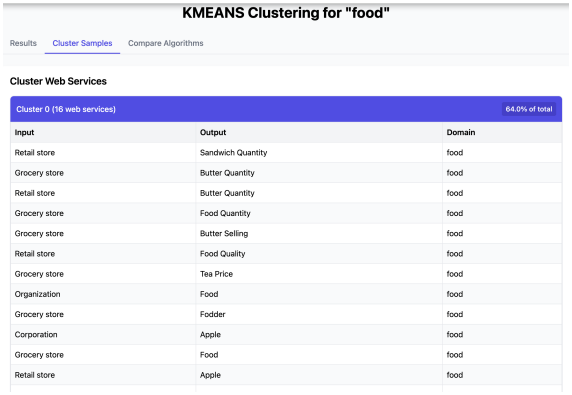


Figure 3.19: Web Services in Cluster 0

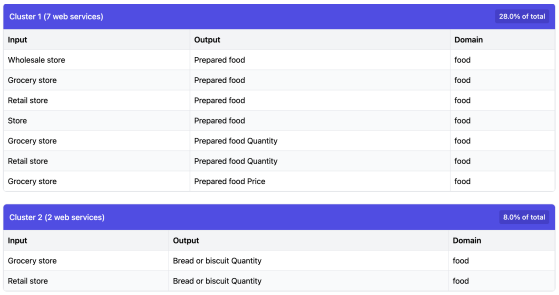


Figure 3.20: Web Services in Cluster 1 and 2

Metric	Meaning
Silhouette Score	How similar an object is to its own cluster vs others
Calinski-Harabasz	Ratio of between-cluster to within-cluster dispersion

Table 3.1: Clustering Evaluation Metrics

3.7.7 Analysis of results

To evaluate the impact of semantic enrichment on clustering performance, we conducted a visual comparison between the clustering results obtained with WordNet-based synonym merging and those generated without semantic enrichment. Both approaches used the same threshold value of 0.06 to filter out low-weight terms.

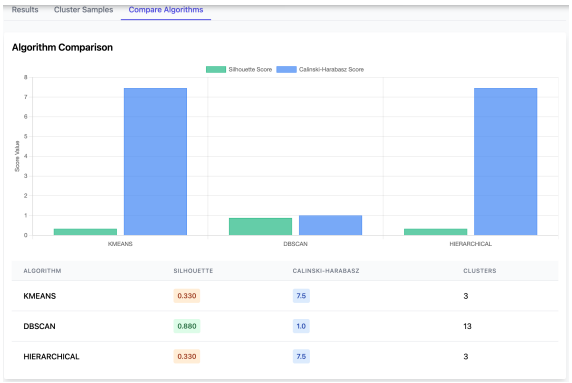


Figure 3.21: Algorithm comparison without WordNet

Figure 3.22: Algorithm comparison with WordNet

Table 3.2: Number of Clusters by Algorithm

Domain	K-means	DBSCAN (With/Without WN)	Hierarchical
Communication	3	33/34	2
Food	3	12/13	3
Travel	3	25/26	4/5
Weapon	3	6/8	2
Geography	3	33/37	2

This table summarizes the number of clusters generated by three clustering algorithms (K-means, DBSCAN, and Hierarchical) across five domains, both with and without the use of WordNet (WN). The values indicate that the use of WordNet generally has minimal effect on K-means and Hierarchical clustering results, while DBSCAN shows slight variations, suggesting sensitivity to semantic preprocessing.

3.8 Comperative study with the state of art

Table 3.2 compares the current results with prior studies. The accuracy is competitive, but the Silhouette Score and Calinski-Harabasz Index suggest room for improvement in cluster definition.

Table 3.3: Comparison of Web Service Clustering Approaches

Authors	Methodology	Silhouette
Aliyu, Wong (2009)	Text Extraction + KMeans	0.50
Elgazzar et al. (2010)	WSDL Keyword Clustering	0.48
Chen et al. (2013)	WT-LDA (Tags + LDA + KMeans)	–
Elshater et al. (2015)	LDA + Topic Modeling	0.55
Wang, Shi (2018)	TF-IDF + Hierarchical Clustering	0.52
Nguyen (2021)	Recommendation via Clustering	–
Our Work (KMeans)	TF-IDF + WordNet + Thresholds + KMeans	0.4499
Our Work (DBSCAN)	TF-IDF + WordNet + Thresholds + DBSCAN	0.6113
Our Work (Agglomerative)	TF-IDF + WordNet + Thresholds + Hierarchical	0.4544

3.9 Conclusion

The Web Services Clustering system transforms OWL-S Web Services text data through cleaning, TF-IDF vectorization, and WordNet synonym merging, creating a semantic, compact feature space. It applies K-means, DBSCAN, and Hierarchical clustering via a Flask interface to group similar services. Silhouette Score and dendrogram visualizations ensure robust cluster evaluation. Built with Python and scikit-learn, it enhances service discovery and supports downstream tasks, providing an effective tool for semantic analysis.

General Conclusion

In today's interconnected digital world, web services have become essential for enabling seamless communication and data exchange between distributed applications, regardless of platform or programming language.
However, the rapid growth and diversity of web services have introduced significant challenges in their discovery, selection, and organization. This project focused on addressing these challenges through Web Services Clustering, a technique that groups semantically or functionally similar services based on features such as textual descriptions, input/output parameters, and domain-specific metadata. By applying methods such as TF-IDF analysis and clustering algorithms (e.g., KMeans, DBSCAN, Hierarchical Clustering), we aimed to improve the structure and accessibility of large service repositories.
The results demonstrated that clustering facilitates more efficient service discovery, enhances reusability, and supports semantic reasoning for better composition and classification of services. In addition, the project highlighted the importance of combining machine learning techniques with domain knowledge to achieve scalable and intelligent service management solutions. Ultimately, this work contributes to the development of smarter, more navigable web service ecosystems, paving the way for advanced applications in service-oriented architectures and cloud computing environments.

Bibliography

BIBLIOGRAPHY

[1] Alonso Gustavo et al. "Web services: concepts, architectures and applications." In: (2004).

[2] Thomas Erl. *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 1900.

[3] Neha Agarwal, Geeta Sikka, and Lalit Kumar Awasthi. "A systematic literature review on web service clustering approaches to enhance service discovery, selection and recommendation." In: *Computer Science Review* 45 (2022), p. 100498.

[4] Waeal J Obidallah, Bijan Raahemi, and Umar Ruhi. "Clustering and association rules for web service discovery and recommendation: A systematic literature review." In: *SN Computer Science* 1.1 (2020), p. 27.

[5] T Berners-Lee et al. *Electronic Networking: Research, Applications and Policy, Vol. 1 No. 2*. 1992.

[6] Timothy J Berners-Lee. *Information management: A proposal*. Tech. rep. 1989.

[7] Tim Berners-Lee. *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*. Harper San Francisco, 1999.

[8] Tim Berners-Lee et al. "The world-wide web." In: *Communications of the ACM* 37.8 (1994), pp. 76–82.

[9] Markus Ast and Martin Gaedke. "Self-contained web components through serverless computing." In: *Proceedings of the 2nd International Workshop on Serverless Computing*. 2017, pp. 28–33.

[10] Hanna Bahemia, John Sillince, and Wim Vanhaverbeke. "The timing of openness in a radical innovation project, a temporal and loose coupling perspective." In: *Research Policy* 47.10 (2018), pp. 2066–2076.

[11] Ying Sun. "Measuring and modelling RPC performance in OSF DCE." PhD thesis. University of Saskatchewan, 1997.

[12] Gustavo Alonso et al. *Web services*. Springer, 2004.

[13] D Winer. "XML-RPC Specification. UserLand Software." In: *Inc. http://www. xmlrpc. com/spec* (1999).

[14] Don Box et al. *Simple object access protocol (SOAP) 1.1*. 2000.

[15] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.

[16] David D Clark et al. “Tussle in cyberspace: defining tomorrow’s internet.” In: *IEEE/ACM transactions on networking* 13.3 (2005), pp. 462–475.

[17] Francisco Curbera et al. “Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI.” In: *IEEE Internet computing* 6.2 (2002), pp. 86–93.

[18] Keith S. Ballinger, Francisco Curbera, and Rania Govindarajan. “Web Services Inspection Language (WSIL) and UDDI: Complementary Discovery.” In: *IEEE Internet Computing* 6.5 (2002), pp. 90–93. DOI: 10.1109/MIC.2002.1036036.

[19] Erik Christensen et al. *Web Services Description Language (WSDL) 1.1*. https://www.w3.org/TR/2001/NOTE-wsdl-20010315. W3C Note. 2001.

[20] Don Box et al. *Simple Object Access Protocol (SOAP) 1.1*. https://www.w3.org/TR/2000/NOTE-SOAP-20000508/. W3C Note. 2000.

[21] Tony Bellwood, Luc Clément, and Claus von Riegen. *UDDI Version 3.0*. https://docs.oasis-open.org/uddi/v3.0/uddi-v3.0.2-20041019.htm. OASIS Specification. 2002.

[22] Wei Liu and W.Y. Wong. “Web service clustering using text mining techniques.” In: *International Journal of Agent-Oriented Software Engineering* 3.1 (2009), pp. 6–26. DOI: 10.1504/IJAOSE.2009.022944.

[23] Mustapha Aznag et al. “Web Services Discovery and Recommendation Based on Information Extraction and Symbolic Reputation.” In: *arXiv preprint arXiv:1304.3268* (2013).

[24] Eyhab Al-Masri and Qusay H Mahmoud. “Discovering web services based on functional clustering.” In: *2007 IEEE International Conference on Web Services*. IEEE. 2007, pp. 601–608.

[25] Y Taher and M Kayed. “Improving Web service discovery using clustering.” In: *2007 IEEE International Conference on Web Services*. IEEE. 2007, pp. 561–569.

[26] M Klusch, B Fries, and K Sycara. “Automated semantic web service discovery with OWLS-MX.” In: *The Semantic Web–ISWC 2006*. Springer, 2006, pp. 329–343.

[27] L Zeng et al. “QoS-aware middleware for web services composition.” In: *IEEE Transactions on Software Engineering* 30.5 (2004), pp. 311–327.

[28] Shuai Zhao et al. “IoT service clustering for dynamic service matchmaking.” In: *Sensors* 17.8 (2017), p. 1727.

[29] Ha Huy Cuong Nguyen et al. “An effective method for clustering-based web service recommendation.” In: *International Journal of Electrical and Computer Engineering (IJECE)* 12.2 (2022), pp. 1571–1578.

[30] Ismail Nadim and Yassine Elghayoumi. “Semantic discovery architecture for dynamic environments of Web of Things.” In: *Sensors* 23.1 (2023), pp. 1–15.

[31] Hai Dong, Farookh Khadeer Hussain, and Elizabeth Chang. “A human-centered semantic service platform for the digital ecosystems environment.” In: *World Wide Web* 13.1 (2010), pp. 75–103.

[32] Ahmed Khedr and Khaled Shaalan. “Ontology-based Web service clustering and retrieval.” In: *Journal of Network and Computer Applications* 41 (2014), pp. 373–388.

[33] J. S. Dong and S. Kambhampati. “Discovery of composite Web services using graph search algorithms.” In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence*. 2003, pp. 1476–1478.

[34] O. Tibermacine, C. Tibermacine, and F. Cherif. "A Practical Approach to the Measurement of Similarity between WSDL-based Web Services." In: *Proceedings of the RNTI Conference*. 2014, pp. 1–16.

[35] O. Tibermacine and F. Cherif. "WSSim: A Tool for the Measurement of Web Service Interface Similarity." In: *Proceedings of the CAL Conference*. 2013, pp. 1–12.

[36] M. Paolucci et al. "Semantic Matching of Web Services Capabilities." In: *International Semantic Web Conference*. 2002, pp. 333–347.

[37] L. Zhou and Y. Chen. "A Semantic Similarity Measure for Web Services Based on Ontology and Word Embedding." In: *IEEE Access* 6 (2018), pp. 42609–42620.

[38] E. Al-Masri and Q. H. Mahmoud. "Discovering Semantic Similarity of Web Services Based on Their Descriptions." In: *Proceedings of the 17th International Conference on World Wide Web*. 2008, pp. 737–746.

[39] Gerard Salton and Chris Buckley. "Term-weighting approaches in automatic text retrieval." In: *Information Processing Management* 24.5 (1988), pp. 513–523. DOI: 10.1016/0306-4573(88)90021-0.

[40] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. ISBN: 9780521865715.

[41] Li Xu, Chuan Wu, and Huiqun Yu. "A clustering algorithm for Web service classification." In: *IEEE International Conference on Services Computing (SCC)* (2007), pp. 287–294. DOI: 10.1109/SCC.2007.78.

[42] Li Fang and Farshad Fotouhi. "Measuring Semantic Similarity in Ontologies: A New Model." In: *Proceedings of the IEEE International Conference on Granular Computing (GRC)*. 2007, pp. 289–294. DOI: 10.1109/GRC.2007.84.

[43] I.T. Jolliffe. "Principal Component Analysis." In: *Springer Series in Statistics* (2002). DOI: 10.1007/b98835.

[44] J. M. Gómez-Pérez, A. Gangemi, and M. C. Suárez-Figueroa. "Ontology Engineering in a Networked World." In: *Springer* (2012).

[45] Peter J Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis." In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65.

[46] Tadeusz Caliński and Jerzy Harabasz. "A dendrite method for cluster analysis." In: *Communications in Statistics* 3.1 (1974), pp. 1–27.

[47] Robert L Thorndike. "Who belongs in the family?" In: *Psychometrika* 18.4 (1953), pp. 267–276.

[48] J. S. Dong, A. Kinoshita, and F. Ishikawa. "Clustering web services by their functional attributes." In: *International Journal of Web Services Research* 1.3 (2004), pp. 69–86.

[49] H. Ma, X. Zeng, and S. Gao. "Semantic similarity based web service clustering." In: *IEEE International Conference on Web Services*. IEEE. 2011, pp. 197–204.

[50] S. Bansal, M. Kumar, and P. Kaur. "Web service clustering based on semantic and syntactic features." In: *International Journal of Computer Applications* 102.17 (2014), pp. 17–22.

[51] J. Devlin et al. "BERT: Pre-training of deep bidirectional transformers for language understanding." In: *Proceedings of NAACL-HLT*. 2019, pp. 4171–4186.

[52] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 4th. Pearson, 2020.

[53] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[54] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. "Supervised Learning." In: *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*. Springer, 2008, pp. 21–49. DOI: 10.1007/978-3-540-75171-7_2.

[55] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. "Unsupervised Learning." In: *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*. Springer, 2008, pp. 51–70. DOI: 10.1007/978-3-540-75171-7_3.

[56] Anil K Jain. "Data clustering: 50 years beyond K-means." In: *Pattern Recognition Letters* 31.8 (2010), pp. 651–666. DOI: 10.1016/j.patrec.2009.09.011.

[57] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. 2nd. Pearson, 2018.

[58] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. "Data clustering: A review." In: *ACM computing surveys (CSUR)* 31.3 (1999), pp. 264–323. DOI: 10.1145/331499.331504.

[59] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. "On the surprising behavior of distance metrics in high dimensional space." In: *International Conference on Database Theory*. Springer, 2001, pp. 420–434. DOI: 10.1007/3-540-44503-X_27.

[60] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. ISBN: 9780521865715.

[61] Muhammad Usama et al. "A survey of clustering algorithms for big data: Taxonomy and empirical analysis." In: *IEEE Transactions on Emerging Topics in Computing* 9.3 (2019), pp. 1302–1321.

[62] Terrence J Sejnowski. *Unsupervised learning: Foundations of neural computation*. MIT press, 1992.

ANNEX

1. Example of Execution

   (a) **System Architecture:**
   The system follows a modular structure, divided into the following layers:

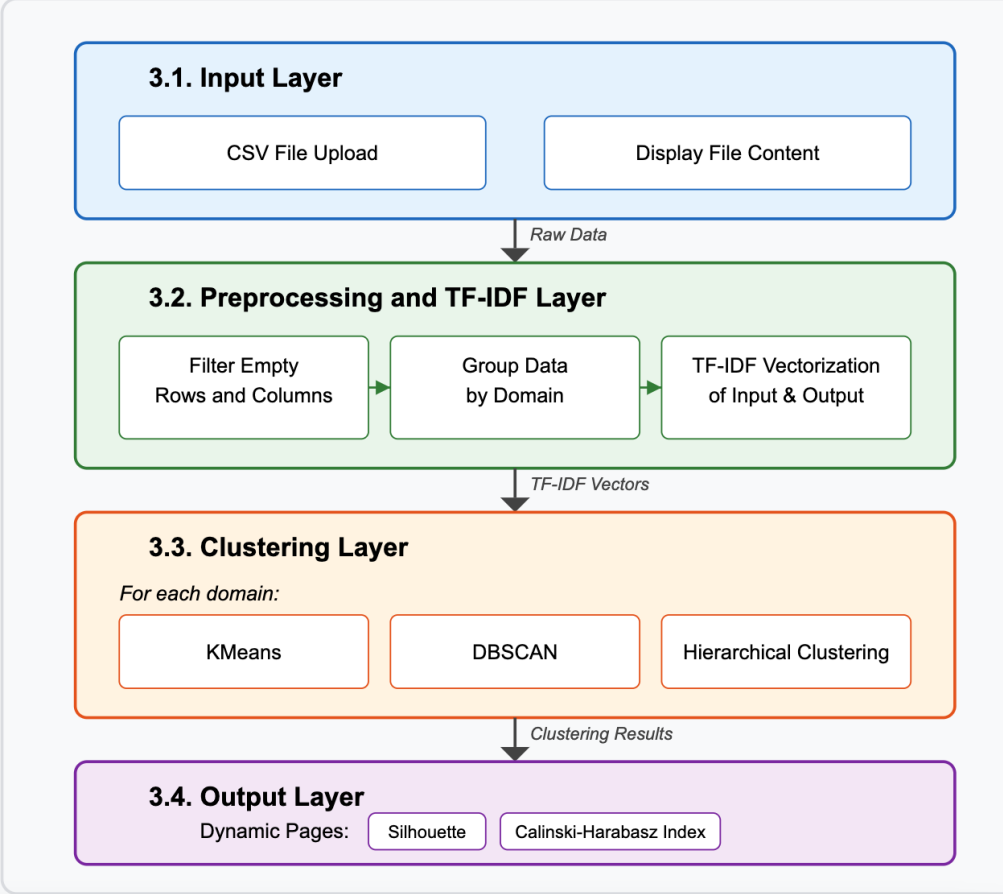


Figure 3.23: Web Services Clustering System Architecture

Results steps With WordNet

- **Step 1: Upload Dataset**
  The user is presented with a clean interface to upload a .csv file containing web service descriptions, with columns such as File, Domain, Input, Output.

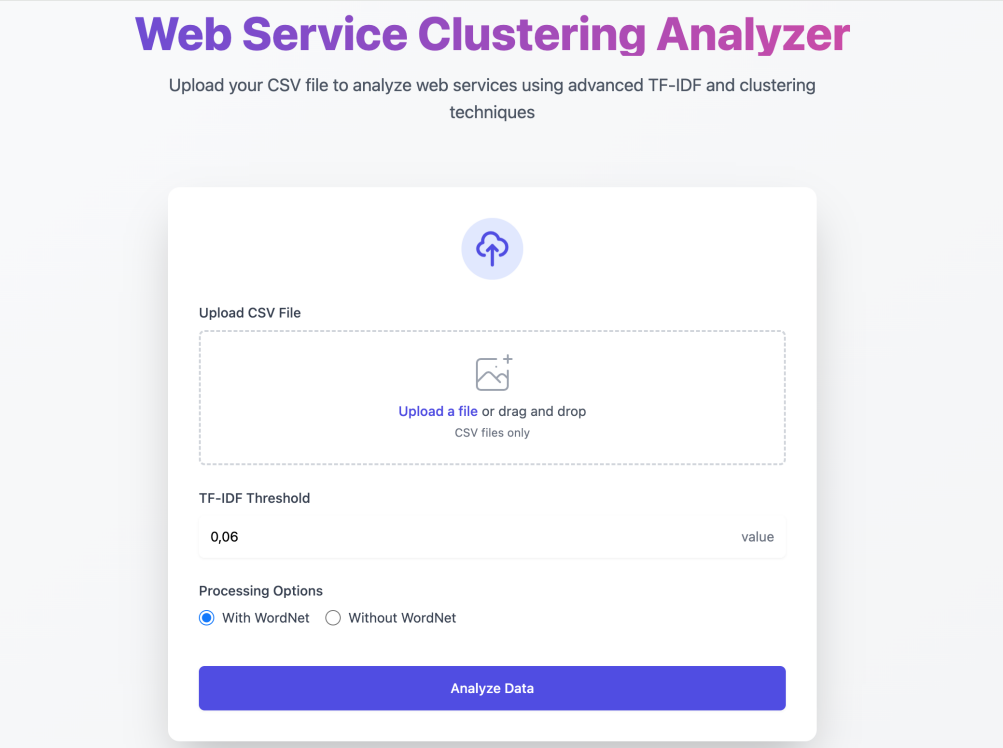


Figure 3.24: upload and threshold

- **Step 2: Display Data**
  After uploading, the application displays a preview of the file content to ensure correctness.

- **Step 3: TF-IDF Analysis by Domain**
  The application extracts the domains automatically.

Complete TF-IDF Matrix (49 documents × 13 features)

DOCUMENT	ACTION	COMEDY	FICTION	FILM	FREE	HOME	MAX	PRICE	QUALITY	RECOMMENDED	SCIENCE	TAX	VIDEO
Doc 1	0	0	0.8719	0.2297	0	0	0	0.2448	0.4219	0.6464	0.8719	0	0
Doc 2	0.5714	0	0	0.2336	0.8532	0	0	0.2489	0.4290	0	0	0.6753	0
Doc 3	0	0	0	0	0.9289	0	0	0.2710	0.4670	0	0	0.7352	0.4493
Doc 4	0	0	0	0.3554	0	0	0	0.3786	0.6526	1.0000	0	0	0
Doc 5	0	0	0	0.3554	0	0	0	0.3786	0.6526	1.0000	0	0	0
Doc 6	0.6697	0	0	0.2738	0	0	0	0.2917	0.5028	0	0	0.7915	0
Doc 7	0	0.6195	0	0.2894	0	0	0	0.3084	0.5315	0	0	0.8367	0
Doc 8	0	0	0	0	0	0	0	0.3291	0.5673	0	0	0.8931	0.5458
Doc 9	0.6697	0	0	0.2738	0	0	0.7915	0.2917	0.5028	0	0	0	0
Doc 10	0	0.6195	0	0.2894	0	0	0.8367	0.3084	0.5315	0	0	0	0
Doc 11	0	0	0	0	0	0	0.8931	0.3291	0.5673	0	0	0	0.5458

Figure 3.25: Résults of tfidf domain comminication with WordNet

For each domain, TF-IDF is computed on the combined text (Input + Output). Displays most significant terms.

- **Step 4: Apply Clustering**
  For each domain, the user can apply any of the supported algorithms:

- **KMeans:** User-defined number of clusters.
- **DBSCAN:** Detects clusters and outliers automatically
- **Hierarchical:**Builds tree-based clusters.

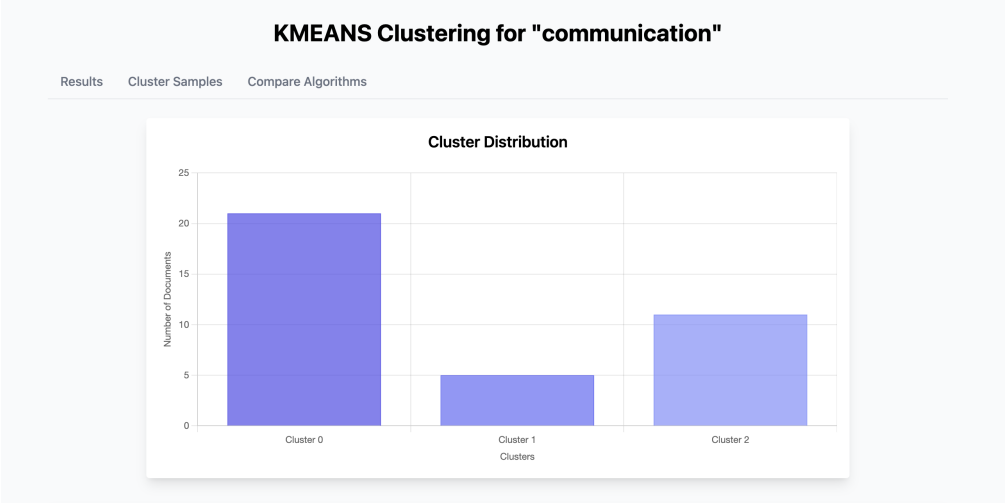


Figure 3.26: KMEANS Clustering Résults" comminication "with WordNet

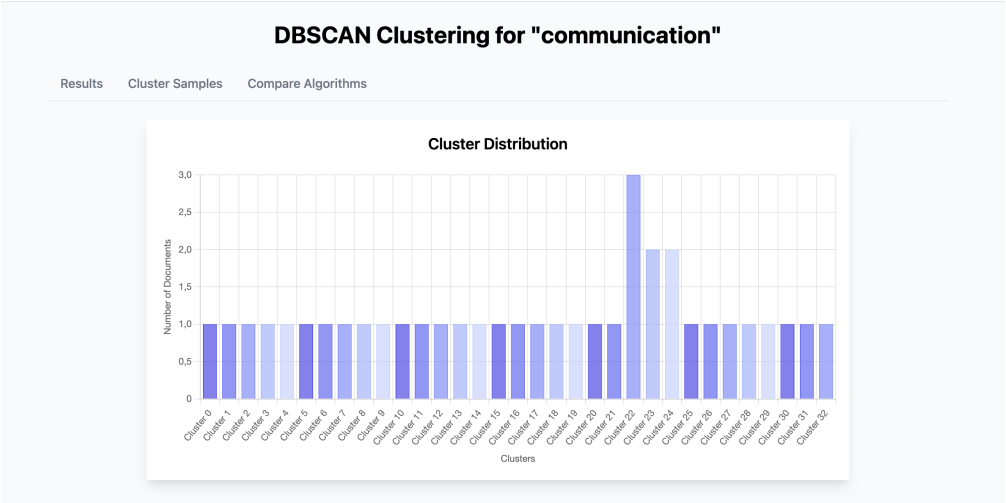


Figure 3.27: DBSCAN Clustering Résults" comminication "with WordNet

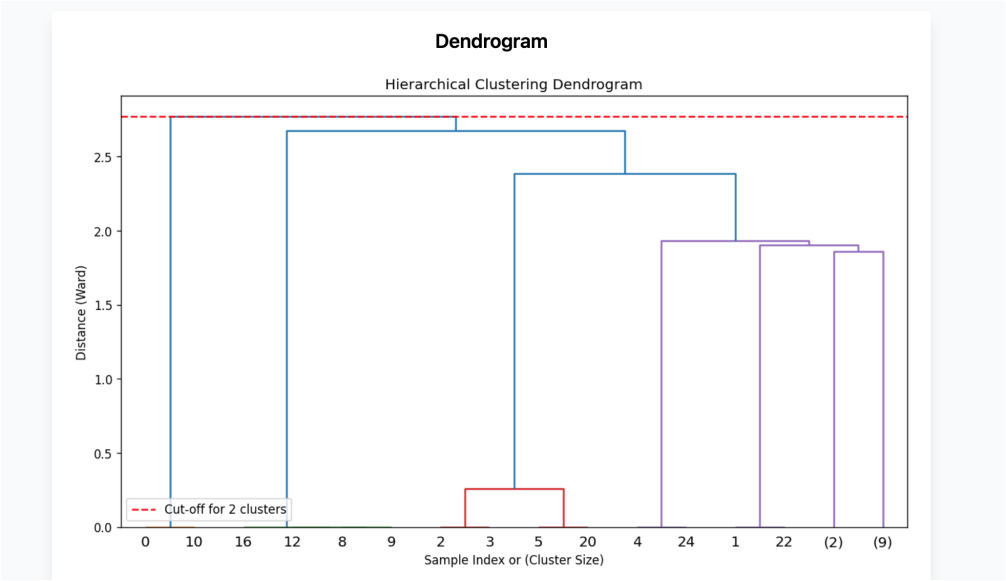


Figure 3.28: HIERARCHICAL Clustering Résults" comminication "with WordNet

- Results steps Without WordNet In this phase, the features were extracted from the raw output data without applying any semantic enrichment techniques such as WordNet. The TF-IDF method was used to represent the text, followed by the application of three clustering algorithms: KMeans, DBSCAN, and Agglomerative Clustering. To evaluate the quality and consistency of the clustering results, two evaluation metrics were used: the Elbow method to estimate the optimal number of clusters, and the Silhouette score to assess the cohesion and separation of the clusters.

  - step1 : Upload Dataset (Without WordNet) a .csv file containing web service descriptions is uploaded, including fields like File, Domain, Input, and Output. The text is processed without semantic enrichment, using only raw data and TF-IDF for feature extraction.

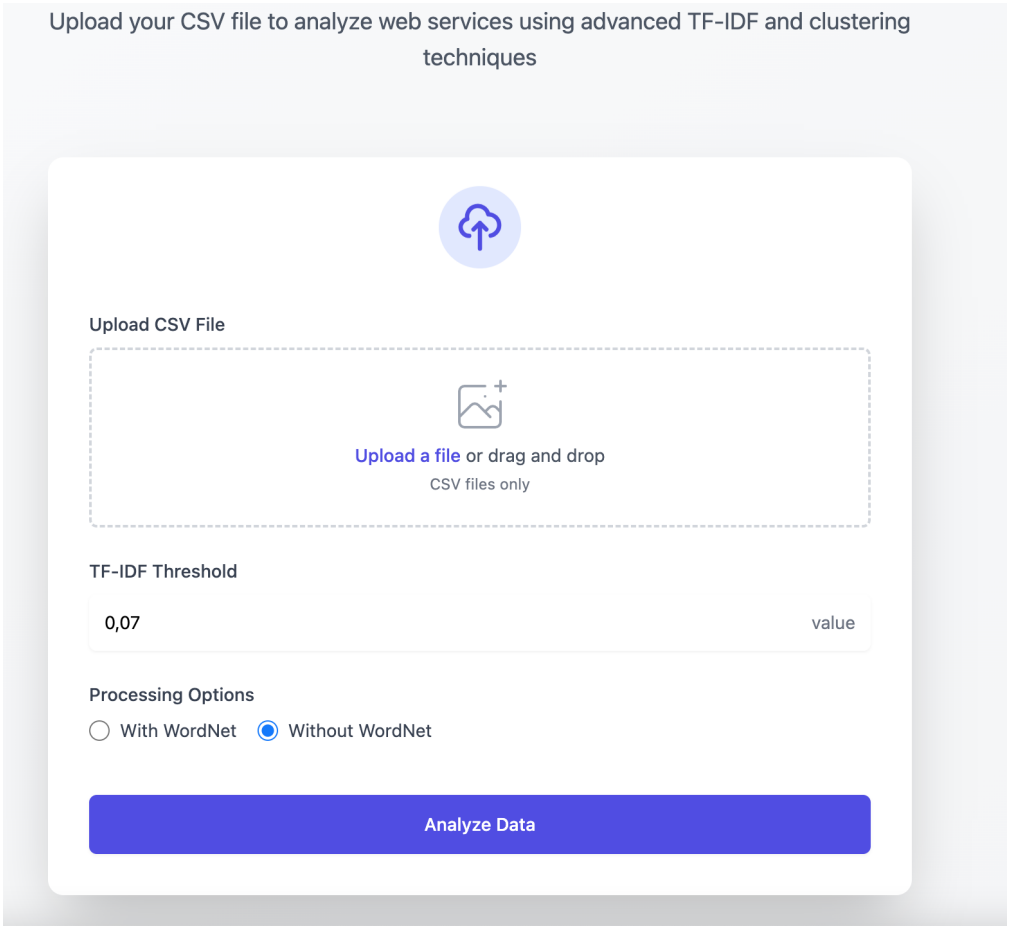
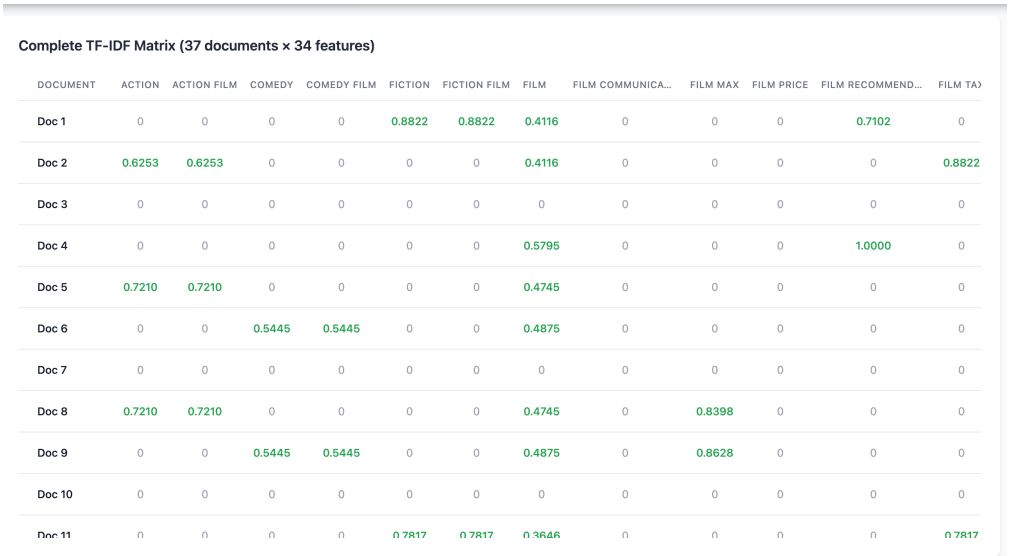


Figure 3.29: upload and threshold"without WordNet"

– Step 2: Display Data After uploading, the application displays a preview of the file content to ensure correctness.

– Step 3: TF-IDF Analysis by Domain The application extracts the domains automatically.



Complete TF-IDF Matrix (37 documents × 34 features)

DOCUMENT	ACTION	ACTION FILM	COMEDY	COMEDY FILM	FICTION	FICTION FILM	FILM	FILM COMMUNICA...	FILM MAX	FILM PRICE	FILM RECOMMEND...	FILM TA
Doc 1	0	0	0	0	0.8822	0.8822	0.4116	0	0	0	0.7102	0
Doc 2	0.6253	0.6253	0	0	0	0	0.4116	0	0	0	0	0.8822
Doc 3	0	0	0	0	0	0	0	0	0	0	0	0
Doc 4	0	0	0	0	0	0	0.5795	0	0	0	1.0000	0
Doc 5	0.7210	0.7210	0	0	0	0	0.4745	0	0	0	0	0
Doc 6	0	0	0.5445	0.5445	0	0	0.4875	0	0	0	0	0
Doc 7	0	0	0	0	0	0	0	0	0	0	0	0
Doc 8	0.7210	0.7210	0	0	0	0	0.4745	0	0.8398	0	0	0
Doc 9	0	0	0.5445	0.5445	0	0	0.4875	0	0.8628	0	0	0
Doc 10	0	0	0	0	0	0	0	0	0	0	0	0
Doc 11	0	0	0	0	0.7817	0.7817	0.3646	0	0	0	0	0.7817

Figure 3.30: Results of tfidf domain communication"without WordNet"

– Step 4 : Apply Clustering

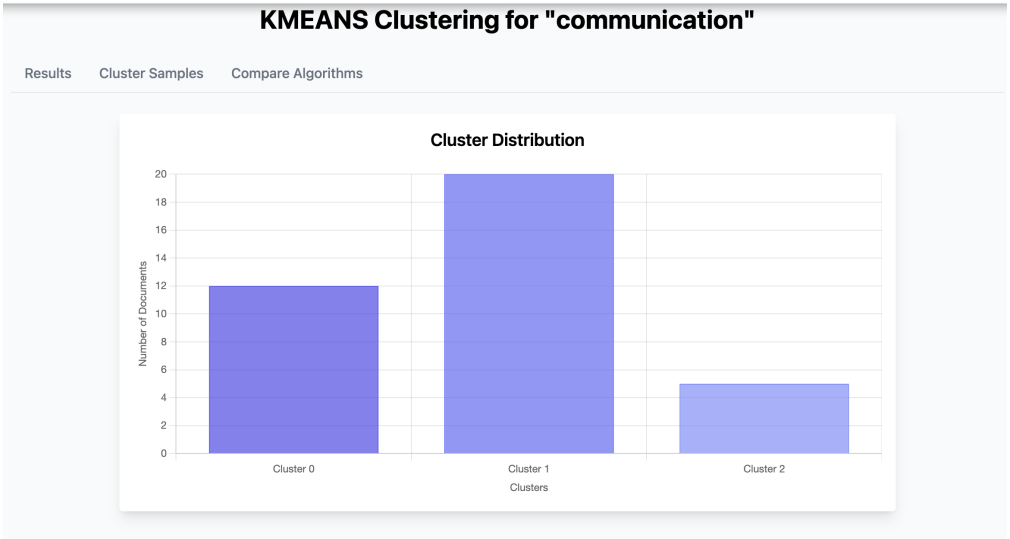


Figure 3.31: KMEANS Clustering Résults comminication ”without WordNet”

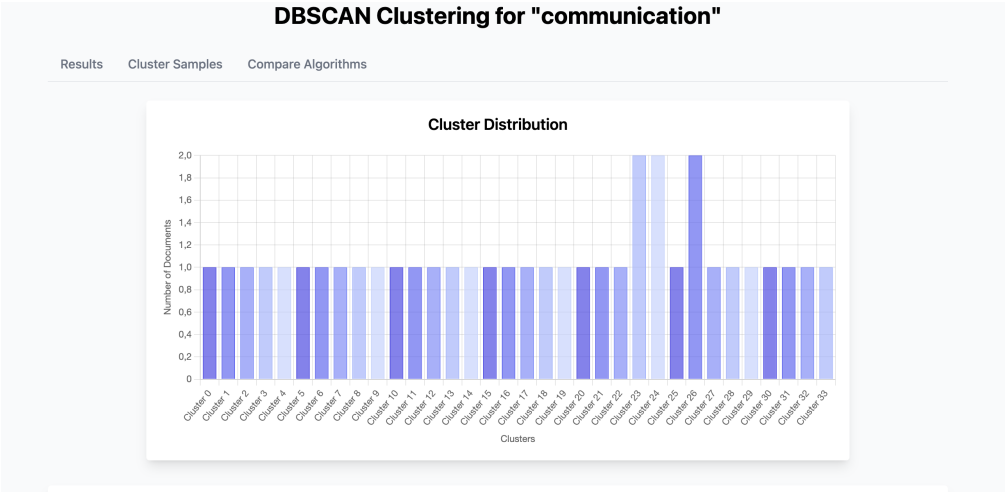


Figure 3.32: DBSCAN Clustering Résults comminication ”without WordNet”

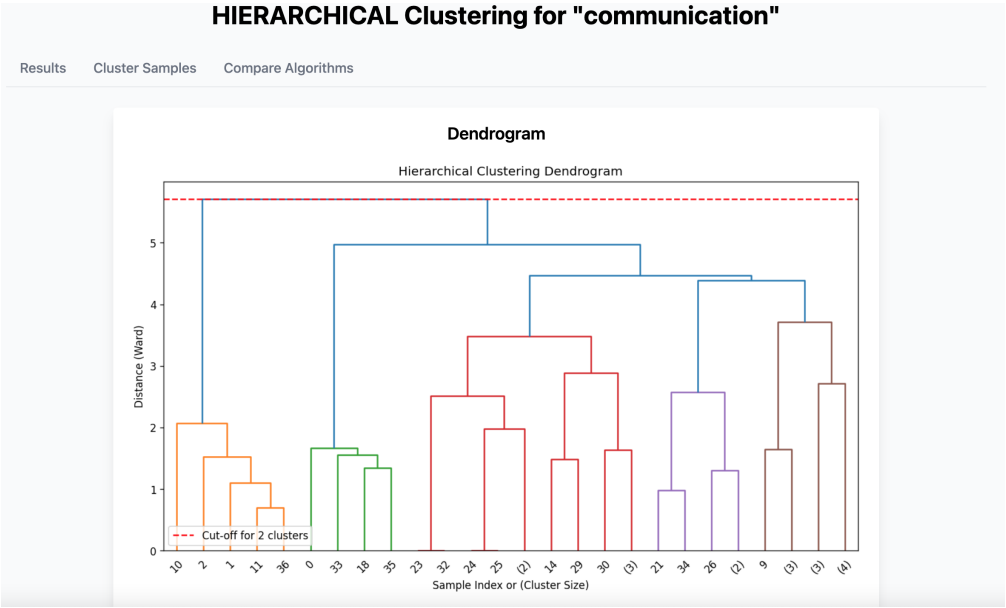


Figure 3.33: HIERARCHICAL Clustering Résults comminication "without WordNet"