Multi-Layer Web Services Discovery using Word Embedding and Clustering Techniques

Waeal Obidallah

A thesis submitted in partial fulfillment of
the requirements for the Doctorate of Philosophy degree in

Digital Transformation and Innovation

uOttawa

Faculty of Engineering
University of Ottawa
Ottawa, Ontario, Canada

Abstract

Web services discovery is the process of finding the right Web services that best match the end-users' functional and non-functional requirements. Artificial intelligence, natural language processing, data mining, and text mining techniques have been applied by researchers in Web services discovery to facilitate the process of matchmaking. This thesis contributes to the area of Web services discovery and recommendation, adopting the Design Science Research Methodology to guide the development of useful knowledge, including design theory and artifacts.

The lack of a comprehensive review of Web services discovery and recommendation in the literature motivated us to conduct a systematic literature review. Our main purpose in conducting the systematic literature review was to identify and systematically compare current clustering and association rules techniques for Web services discovery and recommendation by providing answers to various research questions, investigating the prior knowledge, and identifying gaps in the related literature.

We then propose a conceptual model and a typology of Web services discovery systems. The conceptual model provides a high-level representation of Web services discovery systems, including their various elements, tasks, and relationships. The proposed typology of Web services discovery systems is composed of five groups of characteristics: storage and location characteristics, formalization characteristics, matchmaking characteristics, automation characteristics, and selection characteristics. We reference the typology to compare Web services discovery methods and architectures from the extant literature by linking them to the five proposed characteristics.

We employ the proposed conceptual model with its specified characteristics to design and develop the multi-layer data mining architecture for Web services discovery using word embedding and clustering techniques. The proposed architecture consists of five layers: Web services description and data preprocessing; word embedding and representation; syntactic similarity; semantic similarity; and clustering. In the first layer, we identify the steps to parse and preprocess the Web services documents. Bag of Words with Term Frequency–Inverse

Document Frequency and three word-embedding models are employed for Web services representation in the second layer. Then in the third layer, four distance measures, including Cosine, Euclidean, Minkowski, and Word Mover, are studied to find the similarities between Web services documents. In layer four, WordNet and Normalized Google Distance are employed to represent and find the similarity between Web services documents. Finally, in the fifth layer, three clustering algorithms, including affinity propagation, K-means, and hierarchical agglomerative clustering, are investigated to cluster Web services based on the observed documents' similarities. We demonstrate how each component of the five layers is employed in the process of Web services clustering using randomly selected Web services documents.

We conduct experimental analysis to cluster Web services using a collected dataset of Web services documents and evaluating their clustering performances. Using a ground truth for evaluation purposes, we observe that clusters built based on the word embedding models performed better compared to those built using the Bag of Words with Term Frequency–Inverse Document Frequency model. Among the three word embedding models, the pre-trained Word2Vec's skip-gram model reported higher performance in clustering Web services. Among the three semantic similarity measures, path-based WordNet similarity reported higher clustering performance. By considering the different words representations models and syntactic and semantic similarity measures, the affinity propagation clustering technique performed better in discovering similarities among Web services.

Acknowledgments

First, I would like to thank Allah for giving me strength and guide throughout my life. Completing this doctoral dissertation was possible with the support and encouragement from several people to whom I would like to express my sincere gratitude.

I would like to express my special and most profound appreciation to my supervisor Professor Bijan Raahemi for his valuable guidance and constant supports. I have always been enjoying his friendly relationship and useful advice. I have been incredibly privileged to have a supervisor who gave me the freedom to explore on my own, and at the same time, provided guidance and directions to recover when my steps faltered. Your patience, support, and motivation helped me overcome many obstacles and helped me finish this dissertation. Thank you for your advice on research as well as on my career and life.

I am grateful to my thesis advisory committee Professor Liam Peyton and Professor Umar Ruhi for their valuable comments, encouragement, and guidance for the past years. I am thankful to Professor Daniel Amyot and Professor Ali Reza Montazemi for their comments and feedback on this dissertation.

Most importantly, none of this would have been possible without the love and patience of my family. My family, to whom this dissertation is dedicated, has been a constant source of love, concern, support, and strength all these years. I want to express my heartfelt gratitude to my wife Bushra and the kids for their endless love, patience, and support. I am grateful to my father Dr.Jomaah, and my mother Nourulhuda for their patience and sacrifices they have made on my behalf, which led me to be the one I am today. I am also thankful to all my brothers and sisters for their constant inspiration and encouragement.

Finally, I want to express my gratitude to KDD lab members for being there with their unfailing support and assistance when needed. With a special mention to Fatemeh, Mohammad, Elnaz, Mahdi, and Shahrzad. Special gratitude to my friends Ibrahim AlMohimeed and Faisal Arafsha for their friendship and support during hard times.

Dedication

To My Father Dr. Jomaah Obaidullah.

To My Mother Nourulhuda Bin Saleh.

To My Beloved Wife Bushra Krieyeh and Kids.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ACID	Atomicity, Consistency, Isolation, Durability
AI	Artificial Intelligence
AP	Affinity Propagation
API	Application Programming Interface
BOW	Bag of Words
CBOW	Continuous Bag of Words
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DIY	Do It Yourself
DL	Description Language
DM	Data Mining
DOM	Degree of Matching
DSRM	Design Science Research Methodology
EDA	Exploratory Data Analysis
FTP	File Transfer Protocol
GloVe	Global Vectors for Word Representation
HAC	Hierarchical Agglomerative Clustering
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
Hydra	Hypermedia-Driven Web APIs
IC	Information Content
ICT	Information and Communication Technologies
IOPE	Input and Output Parameters, Preconditions, and Effects
IR	Information Retrieval
JSON	JavaScript Object Notation
NFP	Non-Functional Parameters
NGD	Normalized Google Distance
NLP	Natural Language Processing
NS	Numerical Space
OAS	Open API Specification
OASIS	Organization for the Advancement of Structured Information Standards
OWL	Web Ontology Language

OWL-S	OWL Semantic Markup for Web Services
P2P	Peer-to-Peer
PSWS	Progressive Size Working Set
PW	Programmable Web
QoS	Quality of Service
RAML	RESTful API Modeling Language
REST	Representational State Transfer
RPC	Remote Procedure Call
RSDL	RESTful Service Description Language
SA-REST	Semantic Annotation of Web Resources
SAWSDL	Semantic Annotations for WSDL and XML Schema
SEREDASJ	Semantic Restful Data Services JSON
SLA	Service-Level Agreement
SLR	Systematic Literature Review
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SOVO	Service-Oriented Virtual Organization
UDDI	Universal Description Discovery and Integration
UDP	User Datagram Protocol
UI	User Interface
VO	Virtual Organization
VSM	Vector Space Model
W3C	World Wide Web Consortium
WADL	Web Application Description Language
WMD	Word Mover's Distance
WS	Web Service
WSDL	Web Services Description Language
WSMO	Web Service Modeling Ontology
WUP	Wu-Palmer semantic similarity
XML	Extensible Markup Language
YAML	Yet Ain't Markup Language

Chapter 1. Introduction

Web services are sets of loosely coupled software components that are developed, described, published, discovered, used, and reused to achieve specific functionalities [1]. The World Wide Web Consortium (W3C) states that a Web service is "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Services Description Language (WSDL)). Other systems interact with the Web service in a manner prescribed by its description using Simple Object Access Protocol (SOAP)-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards [2]."

Web services discovery is the process of finding and locating existing Web services based on the service requester's requirements. The discovery process must match the services' functional and non-functional descriptions [3, 4], and retrieve Web services descriptions, in formats such as WSDL and WADL (Web Application Description Language), published by a service provider. Service requesters are looking for Web services that match their functional and non-functional requirements.

Functional requirements refer to the elements that indicate the system's capabilities, including interfaces, operations, and protocol bindings. These are typically described in the service profile [4, 5] and its description. *Non-functional requirements* refer to the elements that indicate the system's performance parameters, which include quality of service (QoS) considerations and service policies, such as security features and service cost [4, 5]. The QoS can include various parameters such as availability, privacy, reputation, price, response time, and usability.

Web services recommendation systems help Web services consumers in selecting Web services with QoS parameters playing a significant role in ranking Web services [6]. Since many Web services have similar functionalities, the service requester must make their service selection without prior knowledge about the list of Web services candidates.

Web services recommendation systems provide leverage by suggesting and ranking Web services to help service requesters during the selection process. Web services recommendation systems have been employed to improve Web services discovery systems and as an additional and supportive step to personalize the users' request.

Web services discovery and recommendation have faced several challenges and limitations in the last few decades. In this thesis, a conceptual model and a typology of Web services discovery systems are presented based on five identified characteristics. In addition, a multi-layer Web services discovery architecture is presented based on various clustering algorithms considering different similarity measures on features extracted from text descriptions using different natural language processing models and text mining techniques.

1.1. Challenges and Motivation

The discovery problem is inherently difficult because of the large scale of dynamic, changing, and uncertain service-oriented architecture (SOA) systems. These include Web services, mobile services, and cloud services. The scope of services in SOA is rapidly changing as new services are added, and old ones are deleted or modified. As a new standard emerges, the older one is abandoned by developers. Technology-agnostic SOA approaches are not tied to a particular technology. However, Web services are commonly employed to constitute the foundational building blocks of SOA [7].

Web services are of no use if they cannot be discovered and utilized. To find a Web service for individual use, service requesters (e.g., software developers) spend much time discover, compare, and select the appropriate services for their requirements. Service requesters usually begin the discovery process by looking at public search engines (e.g., Google or Bing). However, these initial searches can lead to confusion due to the number of returned Web services. Furthermore, the user will be unsure about the functionalities and quality of service (QoS) of the retrieved Web services.

Figure 1.1 demonstrates how service requesters search for a Web service by using various search engines and how the service requesters may be confused by the lengthy list of returned services. It also shows some of the limitations of public search engines in solving problems related to Web services discovery. Search engines respond to the user query

by matching its syntax against the indexed list of Web services. The result is a list of Web services matching the user's query but without any recommendation or personalization based on the user's preferences and without any semantic or contextual relation. The result will not be optimum to help the user find or select an appropriate Web service. Furthermore, the services' returned list may not reflect the user's functional requirements based on the query syntax. In addition, the whole process depends on which Web services have been indexed by the search engine.

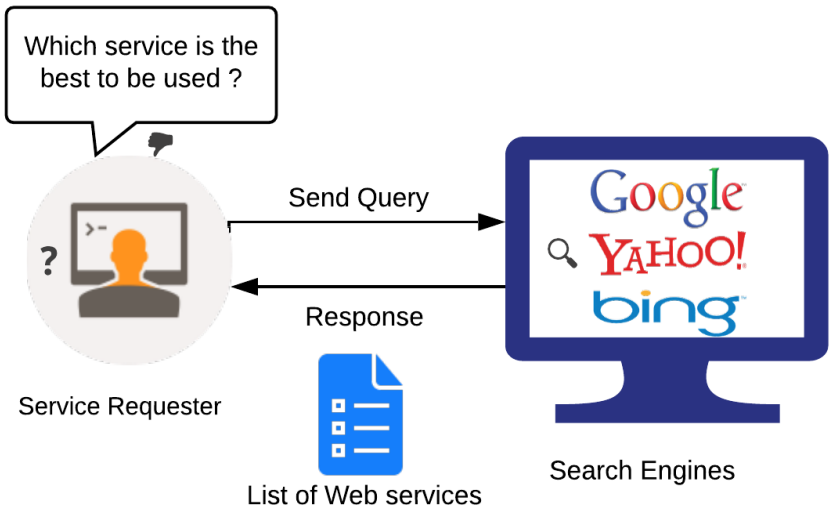


Figure 1-1: A Scenario of a Service Requester Searching for a Web Service

Researchers and industry practitioners have investigated the problem of Web services discovery and recommendation on how to make Web services available and accessible to service requesters with best-fit user functional and non-functional requirements. One of the first efforts was the Universal Description Discovery and Integration [8] (UDDI) registry, the OASIS standard for discovering Web services. However, public UDDI registries, which were created by large companies (i.e., IBM, Microsoft, SAP, and NTT), were shut down in 2006 [9].

UDDI exclusively supports keyword searches and a category-based discovery mechanism suitable for syntax-based discovery. UDDI lacks semantic representation of Web services and does not specify the non-functional requirements of the service requester; thus, service requesters are overwhelmed by a large number of irrelevant returned Web

services because user intentions, context, and semantics are ignored [10–13]. UDDI limitations drew researchers and industry practitioners from the standard Web services discovery mechanism into do-it-yourself (DIY) approaches [14].

In general, Web services architecture consists of three components: 1) service provider; 2) service requestor; and 3) service registry. The service provider develops services and publishes service descriptions (e.g., WSDL) in service registries (e.g., UDDI). The service registries advertise the published service description. This process allows the service requestor to search the registry's directory and all published service descriptions. Figure 1.2 illustrates the Web services architecture and its components.

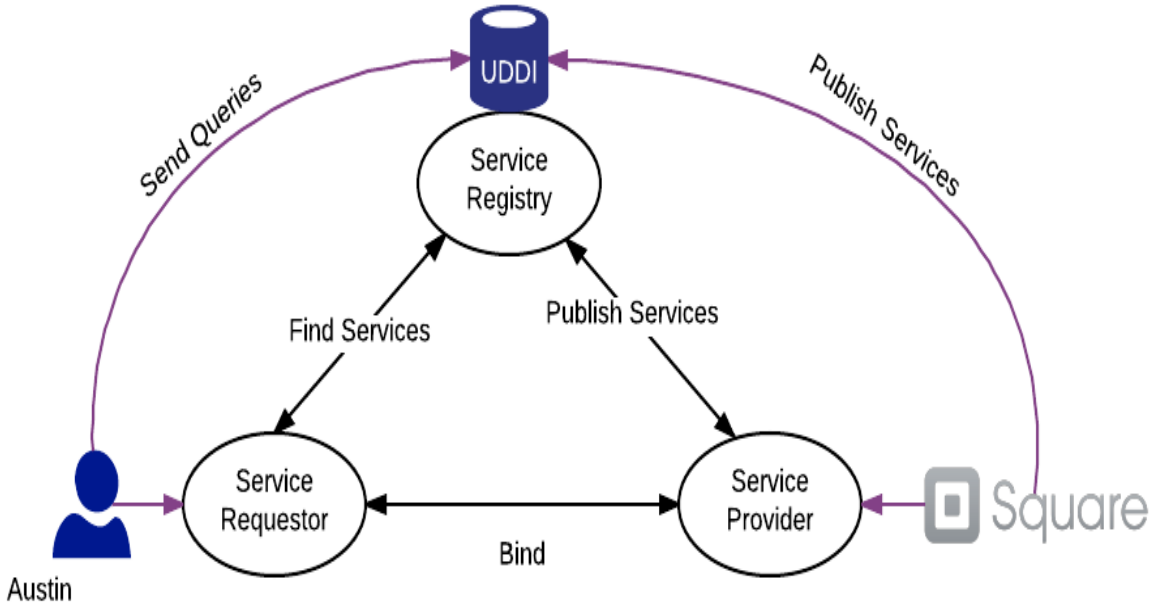


Figure 1-2: Web Services Architecture and Components (with an Example Scenario)

Figure 1.2 shows that the service requestor (Austin) wants to locate a payment service to handle transactions on his new application. In its directory of service descriptions, the service registry has PayPal, Stripe, and Square as providers of such a service (i.e., based on the functionality). Austin invokes Square's payment service through three operations in the Web services components: 1) publish service, 2) find service; and 3) bind. The publish operation is the contract between the service registry (i.e., UDDI) and service provider (i.e., Square). After Square publishes its services, the service descriptions move to the registry directory structure. The find operation is the contract between the service requestor (i.e.,

Austin) and the service registry. The requestor states the search query criteria, and the registry matches the criteria against its directory based on the employed matching mechanism. Based on the search query (i.e., keyword-based on functionality), Square's service is found in the matching process. Finally, the bind operation allows Austin to invoke Square's service. Based on this scenario, only functional characteristics of Web services are considered in matching keywords in the search query. This limitation motivated researchers to produce systems based on DIY approaches that try to understand both functional and non-functional user requirements and add more information to increase the matching process's performance.

Web services are increasingly scattered due to the growing number of internet Web services, the increasing demand for and use of Web services, and the discontinuation or limitations of public UDDI registries. Therefore, Web services providers began publishing services on their websites and Web portals (e.g., Xmethods[1] and WebServiceX[2]). According to [4], Seekda,[3] which was launched in 2006 as a crawler-based Web portal to publish Web services, reported 30,000 Web services in November 2011. The public directory website ProgrammableWeb (PW)[4] reported approximately 16,000 single or composite RESTful Web services as of March 2013 [4] and more than 22,000 as of June 2019. However, such Web services repositories have limitations. For example, they only support keyword-based searches in which a user's specified keywords are searched within the Web services description, and they provide browsing through a limited number of coarse-grained categories. Therefore, they are unable to express the characteristic features of gathered Web services and offer limited multifaceted queries [15].

Recently, Web services marketplaces are providing more capabilities for discovering and using Web services. Marketplaces such as Amazon Marketplace Web Service (Amazon MWS)[5] or Kong[6] provide a better way to discover and utilize Web services and build a business model around the available Web services, initiating the evolving API economy. The term API economy has emerged when Web APIs became part of the business models

[1] http://www.xmethods.com/
[2] http://www.webservicex.net
[3] http://www.seekda.com
[4] https://www.programmableweb.com
[5] https://aws.amazon.com/marketplace
[6] https://konghq.com/

where the public, private and partner APIs published by API providers has been the strategic enablers for such models. The APIs economy is the commercial exchange of business processes, functions, capabilities, and competencies in the form of services by using web APIs [16]. Enterprises use an API strategy to monetize their business capabilities [17] and share them with several audiences through API marketplaces. Enterprises that do not embrace an APIs strategy of their capabilities and services will likely lag behind.

Many Web services developers and providers describe and publish their Web services in ad hoc websites, as we can see with large companies such as Google and Facebook [18]. Most Web services search engines rely on keyword-based matching. Therefore, the search may suffer from a lack of keywords in the description. Web services files do not contain an adequate number of words for indexed terms or features. Moreover, the small numbers of words in the Web services files are erratic and unreliable [19]. The limited number of functionalities offered by Web services can lead to insufficient terms in descriptions, which differs from a regular text document. Furthermore, a vast and irrelevant number of Web services will be returned due to the ample search space. However, clustering Web services based on different similarities will help in minimizing the search space [20]. This clustering serves as a pre-matching stage of the discovery process.

The massive number of sources from which Web services descriptions are available introduced a challenge for Web services discovery as Web crawlers looking for Web services from different sources find duplicated services and invalid descriptions [21]. The descriptions of most publicly available Web services are in the form of short text, which weakens Web services clustering quality due to the sparseness of useful information. Using semantic-based Web services discovery approaches can accurately cluster Web services, but they cannot be widely adopted since they are more complex and depend on ontologies [22]. Without a semantic-aware search method, finding Web services through UDDI registries or search engines can be ineffective as the query may return many irrelevant results and miss the desirable ones; however, semantic-based approaches have their own limitations [14].

Approaches to Web services recommendation help in understanding the user's requirements when selecting a Web service. Although Web services recommendation is chal-

lenging and time-consuming due to the large search space, clustering techniques can minimize this space [23]. When QoS information, which plays the significant part in Web services recommendation, is limited, then Web services recommendation approaches usually either fail to make predictions or make inferior ones. Due to user disparity, QoS values evaluated by one user cannot be directly employed by another in Web services recommendation. However, user-specific QoS information is hard to obtain because service requesters often request and invoke very few of the available Web services [24]. Furthermore, it is impractical, costly, and time-consuming for service requesters to try all candidate Web services and acquire QoS information [24–26]. Therefore, it is necessary to either identify associations between service requester or cluster them based on their similarities.

Web services technology is an efficient and effective way to deliver services to users in E-business and E-commerce. Companies across the globe use Web services; they are developed and published to leverage the companies' and enterprises' business models. However, the Web services concept is unfamiliar to end-users who continue to struggle in discovering and selecting the appropriate Web services. End-users need advanced mechanisms to assist them in discovery and selection services. This situation has led many researchers to propose solutions, both theoretical and experimental levels, to enhance Web services discovery and recommendation. Data analytics, text mining, NLP, information retrieval (IR), and machine learning techniques provide solutions to facilitate the process of Web services discovery and recommendation, by applying clustering, classification, association rules, or collaborative filtering algorithms.

This thesis is motivated to facilitate the process of Web services discovery and recommendation with a combination of theory and experimental approaches. We propose a conceptual model and typology of Web services discovery systems based on five identified characteristics. Furthermore, we propose the use of Web services data and similarity analysis techniques that more precisely use word representation and embedding models, syntactic and semantic similarity measures, and clustering techniques to facilitate the process of discovery before the matching process. For this purpose, we conduct a systematic literature review (SLR) for Web services discovery and recommendation, where we identify research gaps and future directions. Furthermore, we state the need for comprehensive and neutral Web services discovery and recommendation solutions. In these solutions, NLP,

text and data mining, IR, collaborative tagging, and service information extraction techniques can identify useful patterns, clusters, and group-related Web services.

1.1.1 Motivation in the Context of E-Business

Web services enable E-businesses to publish their competencies to allow developers to add business models and leverage the overall E-business ecosystems. However, identifying the right Web services to meet developers' functional and non-functional requirements is difficult. Web services requesters look for Web services in different venues and sources, including Mashape (Kong), PW, XMethods, and WebServiceList. Furthermore, word of mouth plays a significant role because developers trust their professional communities when looking for appropriate Web services. Developers use these sources to search for, compare, select, retrieve, and use Web services. On the other side, service providers publish their Web services in various venues, which provide different functionalities and diverse qualities. For service requesters, finding and selecting the right Web service in this context is a challenge.

For example, consider a service requestor looking for payment services by searching Web services repositories, as in Figure 1.3. The requestor sends the search query "Payment service" through the Web service broker portal, an interface and entry point connecting different Web services repositories and providers. There are different Web services types, including SOAP-based, REST-based, XML-RPC, and JSON-RPC [27]. Furthermore, Web services are described in various formats, including WSDL, WADL, WSMO, OWL-S, and text. There are massive numbers of Web services published by different providers, and it will take a long time to search and match the requestor's query because it requires reviewing all the published Web services looking for ones related to payment services. However, considering the functional clustering of Web services to categorize them can mitigate this problem and speed up the process. A list of payment Web services can be matched by looking into the clusters of Web services. The requestor must then compare all the retrieved payment Web services based on different QoS requirements and preferences. The requestor can achieve a better solution by having Web service recommendation systems that understand their service preferences.

Figure 1.3 illustrates a scenario where clustering Web services minimizes the search space based on different similarity measures. The retrieved list of Web services has no QoS parameters or ranking, which hinders the process of selecting the right Web services. We believe that a better solution is possible by clustering Web services based on finding similarities with syntactic and semantic similarity measurements for Web services discovery. Furthermore, considering Web services recommendation systems to recommend a Web service based on clustering consuming user preferences or location helps service requesters select the right Web service.

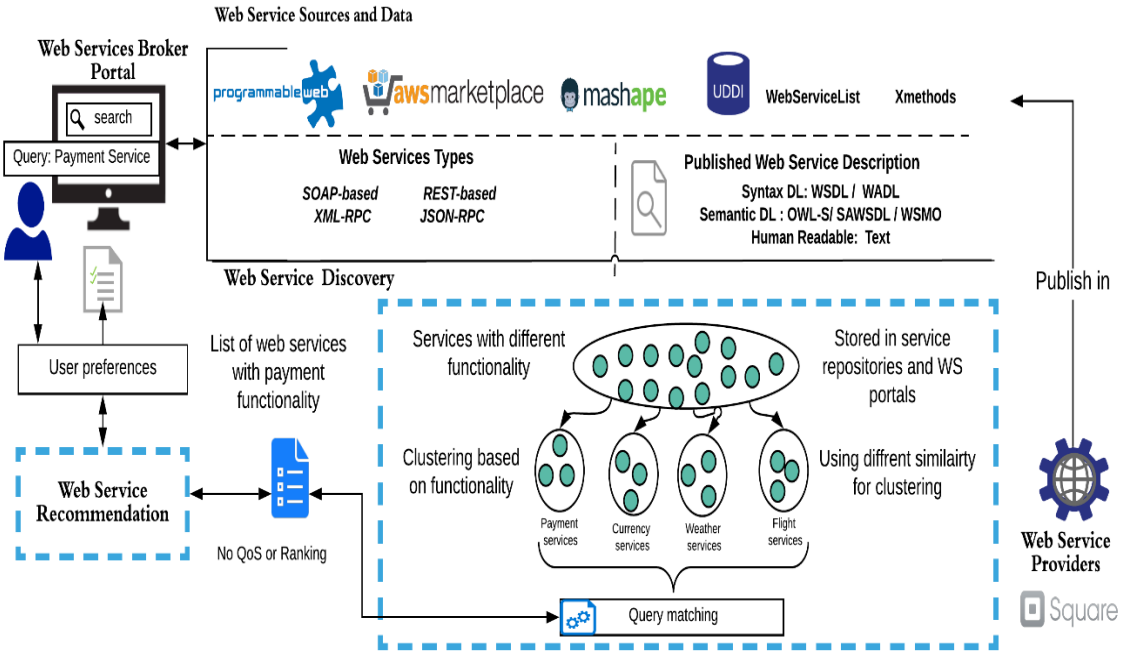


Figure 1-3: Research Motivation, a Scenario in E-Business

1.1.2 Motivation in the Context of Service-Oriented Virtual Organizations

The motivation for this study began while working on a Service-Oriented Virtual Organization (SOVO) project [28]. SOVO aligns a virtual organization (VO) with SOA to form a SOVO. A SOVO is a dynamic temporal consortium of autonomous and legally independent organizations that collaborate to achieve objectives and meet business needs relying on services [29]. This is orchestrated as shared business processes, which could combine several Web services functionalities. During the first phase of a SOVO (the crea-

tion phase), a VO looks for, discovers, and selects VO partners (service providers) to formalize a collaborative business opportunity based on functionality (Web services) and service-level agreements (i.e., QoS) [28]. This is a domain-specific service-discovery problem during the VO's creation phase within the collaborative business process. Rather than investigate the problem in a SOVO domain, this study examines an original area by examining Web services discovery in the context of E-business. Figure 1-4 illustrates the problem of Web services discovery in the context of SOVO.

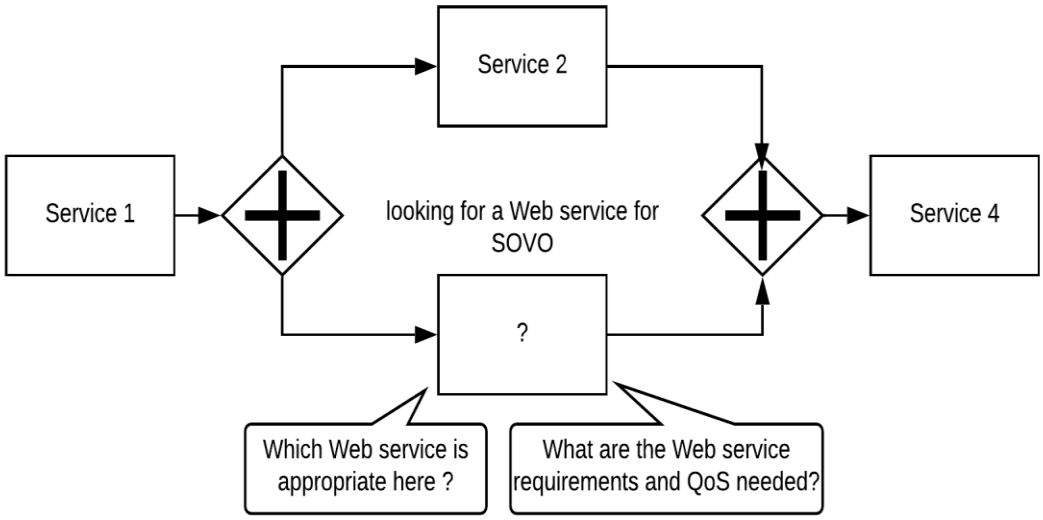


Figure 1-4: Web Services Discovery within SOVO Collaborative Business Process

In Figure 1-4, a SOVO consumes Web services as part of its collaborative business process by employing three Web services while seeking a new service. The new service (VO partner) should meet the VO's functional and non-functional requirements [30]. From the VO point of view, the problem is discovering and looking for a VO partner that provides service functionality that meets the VO's business processes. The VO is looking for specific service functionality (functional Web service requirement) with SLAs (non-functional Web service requirements). This VO service discovery problem is the same as when an individual (e.g., Austin in the prior example) is looking for a Web service that meets specific functional and non-functional requirements in a different context. Based on this research problem, we investigated Web services discovery and recommendation in enabling E-business and E-commerce.

1.1.3 Why Do Clustering Web Services Help in the Process of Discovery and Recommendation?

Web services providers publish their newly developed Web services in different online repositories and portals to be consumed by users. The newly developed Web services are added to the list of Web services, which had previously identified groups/categories on the portal. In the absence of automation, the process of adding the new Web services to the right category within the portal is carried out manually by a domain expert who goes over the Web services and categorizes them based on their functionalities. Thus, this process takes more time and effort as the number of added Web services increases.

In a real scenario, a Web services aggregator or collector gathers the description of Web services with their details from different repositories and portals so service requesters can access them. After the Web services descriptions are collected, the Web services files need to be indexed and categorized. The process of categorizing collected Web services based on their domain (e.g., functional and non-functional similarity) is done by a domain expert. Artificial intelligence, data and text mining, and NLP techniques can be employed instead of the domain expert to cluster the collected Web services considering their descriptions (e.g., WSDL, text, etc.) into categories. The clustering techniques can automatically cluster Web services based on different similarity measures where similar Web services are assigned to the same cluster. Furthermore, having clusters of Web services based on their functional or non-functional specifications will minimize the search space for the query matching algorithm.

1.2. Research Objectives

The primary objectives of this study are to:

- Conduct a systematic literature review in the area of Web services discovery and recommendation where clustering and association rules techniques are employed;
- Build a comprehensive understanding of Web services discovery systems on the conceptual level by identifying their components and the functionalities of and relationships between those components;

- Develop and define a typology of Web services discovery systems based on the observation of Web services discovery characteristics after investigating different systems, supporting techniques, and methods;
- Investigate a general approach and framework in applying text and data mining techniques for Web services clustering;
- Improve the process of Web services discovery by proposing and developing a multi-layer data mining architecture for Web services discovery;
- Study the use of various Web service text representations, syntactic and semantic similarity measures, and clustering algorithms in the proposed architecture; and
- Implement and evaluate the performance of the proposed architecture for Web services clustering while exploring various similarity measures, datasets, and performance evaluation metrics employed in Web services discovery and recommendation approaches.

1.3. Research Questions

Table 1-1 frames the research problem statement, questions, and other details. Our primary research questions are:

- What are the components of Web services discovery systems, their functionalities, and how they interact with each other?
- What are the characteristics of Web services discovery systems and how they can be used to differentiate the developed Web services discovery systems from other systems?
- Which clustering algorithm among the employed algorithms is the best to minimize the search space by clustering similar Web services?
- What type of similarity measures are more accurate to measure the similarity between Web services (for clustering) and how?
- What is the best performance for Web service clustering among the multiple layers (considering different models) of word representations, similarity measures, and clustering algorithms?

Table 1-1: Framing the Research Problem and Objectives

Steps	*Description*
Observation	• Web services are the enabling technology of E-business and E-commerce. • There are different types of Web services with various ways to describe them. • Web services service requesters are searching to find the right Web services to meet their functional and non-functional requirements. • Service requesters select Web services based on service quality and functionality. However, not all Web services discovery and recommendation systems consider these criteria. • The process of Web services discovery takes a long time and considerable effort. In the process of Web services discovery, the search space to find the right Web service can be minimized by clustering similar Web services.
Thesis	A systematic literature review of clustering and association rules techniques for Web services discovery and recommendation is conducted to identify methods, algorithms, similarity measures, and datasets utilized. A conceptual model and typology for Web services discovery systems are developed based on five identified characteristics, which include storage and location, formalization, matchmaking, automation, and selection characteristics. This provides a foundation for the development of a multi-layer Web service discovery architecture to cluster Web services for the purpose of minimizing the discovery search space: an architecture with various layers to identify useful data and parse them, perform text preprocessing steps, text representation, and embedding and clustering Web services documents based on syntactic and semantic similarities measures.
Enthymeme	Several Web services discovery and recommendation methods and techniques have been proposed in the last decade. Therefore, a thorough and comprehensive systematic literature review (SLR) is needed to identify these methods and their algorithms, similarity measures, and datasets. Also, the research gaps and future directions need to be identified. Some service requesters are not familiar with Web services and how and where to look for and discover them. Therefore, developing a conceptual model and a typology to identify Web services' characteristics will help service requesters better understand Web services and their discovery mechanisms. Web services are utilized to build different applications by composing various Web services together and adding more functionalities. Therefore, Web services requesters are looking for the right Web services that meet their functional and non-functional requirements. The proposed solutions group/cluster Web services based on their syntactic and semantic similarities to minimize the search space because of the massive number of Web services published in different repositories with various functionalities.

Problem Statement	Web services are a commonly utilized technology to constitute the foundation of SOA building blocks. The services are of no use if they cannot be discovered and utilized by users. To identify Web services for individual use, service requesters (e.g., software developers) spend significant time discovering, comparing, and selecting services to satisfy their requirements. The discovery process requires time and effort due to the massive number of Web services published and scattered throughout sources with different functionalities. Searching for the right Web service based on the user requirements takes a long time due to the ample search space. However, clustering Web services based on their syntactic and semantic similarities can minimize the search space to facilitate the discovery process.
Objectives	• Conduct a systematic literature review in the area of Web services discovery and recommendation where clustering and association rules techniques are employed; • Build a comprehensive understanding of Web services discovery systems on the conceptual level by identifying the systems' components and their functionalities and relationships; • Develop and define a typology of Web services discovery systems based on the observation of Web services discovery characteristics after investigating different systems, supporting techniques, and methods; • Investigate a general approach and framework for applying text and data mining techniques for Web service clustering; • Improve the process of Web services discovery by proposing and developing a multi-layer data mining architecture for Web services discovery; • Study the use of various Web service text representations, syntactic and semantic similarity measures, and clustering algorithms in the proposed architecture; and • Implement and evaluate the performance of the proposed architecture for Web services clustering while exploring various similarity measures, datasets, and evaluation metrics employed in Web services discovery and recommendation approaches.
Research Questions	• What are the components of Web services discovery systems, their functionalities, and how they interact with each others? • What are the characteristics of Web services discovery systems and how they can be used to differentiate the developed Web services discovery systems from other systems? • Which clustering algorithm among the employed algorithms is the best to minimize the search space by clustering similar Web services? • What type of similarity measures are more accurate to measure the similarity between Web services (for clustering) and how? • What is the best performance for Web service clustering among the multiple layers (considering different models) of word representations, similarity measures, and clustering algorithms?

1.4. Thesis Contributions

The major contributions of this thesis are:

1. A systematic literature review (SLR) of clustering and association rules techniques for Web services discovery and recommendation. It aims to identify and compare existing techniques, methods, and algorithms employed to facilitate the process of Web services discovery and recommendation. A classification of Web services discovery and recommendation techniques considering clustering and association rules is presented where the most cited algorithms relating to the various similarity measures are identified and discussed. Furthermore, the SLR identifies benchmark data collections utilized to validate the proposed methods. The syntactic and semantic similarities for each method are identified and compared. The trends and future research directions are also discussed. Future researchers can use its information and data to identify research gaps based on synthesis and analysis, as well as identify different algorithms, similarity measures, and datasets in Web services discovery and recommendation;
2. A conceptual model and a typology of Web services discovery approaches based on five identified characteristics, including storage and location, formalization, matchmaking, automation, and selection characteristics. The proposed typology is derived from observation of the wide-ranging literature about Web services discovery systems. The typology provides a high-level view of where researchers conducted their investigations to improve the process of Web services discovery. A comparison of different Web services discovery mechanisms based on the proposed typology is also conducted;
3. A multi-layer data mining architecture for Web services discovery includes various similarity-based Web services clustering methods and consists of 5 layers, starting with the data preprocessing layer; representation, embedding, and transformation layer; syntactic similarity layer; semantic similarity layer; and clustering layer. The implementation and performance evaluation are discussed considering different evaluation measures. The aim is to cluster Web services according to different similarities to minimize the search space and measure the performance; and
4. A new solution introduced, which exploits Affinity propagation (AP) clustering algorithm to cluster Web services based on various syntactic and semantic similarity

measures, reporting higher performance than the most cited clustering algorithms (baseline) in the literature for clustering Web services.

1.5. Publications Resulting from this Research

The following publications were made during this thesis research:

1. **W. J. Obidallah**, B. Raahemi, and U. Ruhi, "Clustering and Association Rules for Web Service Discovery and Recommendation: A Systematic Literature Review," SN Comput. Sci., vol. 1, no. 1, p. 27, Jan. 2020, [31];
2. **W. J. Obidallah** and B. Raahemi, "A survey on web service discovery approaches," in *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing - ICC '17*, 2017, pp. 1–8, [32];
3. **W. J. Obidallah** and B. Raahemi, "Managing Changes in Service Oriented Virtual Organizations," J. Electron. Commer. Organ., vol. 15, no. 1, pp. 59–83, Jan. 2017, [29];
4. F. Firozbakht, **W. J. Obidallah**, and B. Raahemi, "Cloud computing service discovery framework for IaaS and PaaS models," in Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing - ICC '17, 2017, pp. 1–6, [33];
5. **W. J. Obidallah**, U. Ruhi, and B. Raahemi, "Current Landscape of Web Service Discovery: A Typology Based on Five Characteristics," in 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2016, pp. 678–683, [14];
6. **W. J. Obidallah** and B. Raahemi, "A Taxonomy to Characterize Web Service Discovery Approaches, Looking at Five Perspectives," in 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE), 2016, pp. 458–459, [27]; and
7. M. M. Mohammadi, B. Raahemi, F. Cheraghchi, **W. J. Obidallah**, and E. Bigdeli, "Big data analytics using Hadoop," in Proceedings of 24th Annual International Conference on Computer Science and Software Engineering, 2014, pp. 323–325 [34].

1.6. Thesis Outline

This thesis is organized into eight chapters, as follows:

- Chapter 2 provides background information. It presents the concepts related to Web services design, description languages and documentation, types of similarity measures, and clustering algorithms;
- Chapter 3 presents the research design and methodology employed throughout this thesis, which includes an explanation of different research design layers and the Design Science Research Methodology (DSRM), including its steps;
- Chapter 4 introduces the systematic literature review of clustering and association rules techniques to support the process of Web services discovery and recommendation. It shows the review methodology where we identify the need for the systematic literature review, define the search questions, develop the review protocol, and discuss the review results;
- Chapter 5 presents the proposed conceptual model and typology of Web services discovery approaches based on five identified characteristics, including storage and location, formalization, matchmaking, automation, and selection characteristics;
- Chapter 6 describes the proposed multi-layer based data mining architecture, which uses various similarity measures for Web services clustering. The proposed architecture consists of five layers, including the text preprocessing layer; text representation, embedding and transformation layer; syntactic similarity measure layer; semantic similarity measure layer; and clustering layer. This chapter provides detailed descriptions of the functionality and method in each layer;
- Chapter 7 presents the implementation and evaluation of the proposed multi-layer data mining architecture. The dataset, which includes the Web Service Description Language (machine-readable) and auxiliary text (human-readable), are discussed. The evaluation procedures and performance measures are presented, and the experimental results are analyzed and discussed; and
- Finally, Chapter 8 offers the thesis summary, conclusions, and future works.

Chapter 2. Background

This chapter provides a background of the related concepts and an overview of some of the theories and techniques required for this research. We briefly describe Web services, including Web services architecture, design, and description languages. Furthermore, an overview of clustering algorithms, including partition-based and hierarchical-based clustering, is presented, followed by a background on similarity measures.

2.1. Web Services

Web services are computational software with the ability to achieve users' objectives with a remote invocation, allowing applications written in different programming languages to communicate seamlessly using standard protocols. Web services are sets of loosely coupled software components that are developed, described, published, discovered, used, and reused to achieve specific functionalities [1]. Web services have been utilized in a broad range of applications ranging from simple (e.g., currency exchange) to complex (e.g., payment, weather, and GPS services). Systems interact with a Web service in a manner prescribed by its description using SOAP (Simple Object Access Protocol) messages, typically conveyed with HTTP (Hypertext Transfer Protocol) with XML (eXtensible Markup Language) serialization in conjunction with other Web-related standards [2]. However, Web services are not restricted to being SOAP-based. The two commonly utilized Web services forms are based on SOAP and REpresentational State Transfer (REST).

2.1.1 Web Services Design

The standard Web services which comply with the SOA (Service Oriented Architecture) principle of service design [35] are called SOAP-based Web services; SOAP is the protocol employed to handle the communication between the client interface and the service provider. However, Roy Fielding [36] proposed a new architecture style, under the name REST, that provides a new principle for building Web services. Web services built on top

of REST are known as REST-based Web services. Appendix A discusses the details and specifications of SOAP- and REST-based Web services.

SOAP-based vs. REST-based Web services

The question of which Web services types (SOAP-based or REST-based) is better is always debatable in the Web services communities. Some argue that REST-based Web services are taking over from the traditional SOAP-based Web services because of REST's simplicity, scalability, and high adoption in the community. However, SOAP-based Web services have competitive advantages in some areas. Security, service-level agreements, execution methods, and transportation-protocol availability (i.e., not only HTTP) are some of SOAP-based Web services' advantages over REST-based Web services. As a matter of fact, deciding between SOAP or REST Web services depends on the developers' preferences in addition to the domain and the environment of the Web services. Table 2-1 illustrates the variances between SOAP-based Web services and REST-based Web services from different perspectives [37, 38].

Table 2-1: Comparison between SOAP and REST Web Services

	SOAP-based	REST-based
Stands for	Simple Object Access Protocol	Representational State Transfer
Design Style	Standard protocol with well-defined rules and specifications	An architectural style includes guidelines and recommendations
Approach	Function-driven; Data available as services	Data-driven; Data available as resources
Data Model	Hidden	Exposed
Statefullness	Stateless by default and supports stateful operations	Stateless (no server-side sessions)
Security	WS-Security with SSL support, HTTP, and HTTPS; Built-in ACID compliance	Supports HTTPS and SSL; Lacks ACID compliance
Bandwidth	More resources and bandwidth	Fewer resources and bandwidth
Caching	API calls cannot be cached	API calls can be cached
Reliability	Reliable	Not reliable
Scalability	Not scalable	Scalable, HTTP hyperlinks
Interface	For each Web service	Web browser
Performance	Requires more bandwidth and computing power	Requires fewer resources
Payload Format	Only XML (SOAP)	Plain text, HTML, XML, JSON, YAML, and others
Transfer Protocol(s)	HTTP, SMTP, UDP, FTP, and others	Only HTTP

Service Discovery	UDDI, DIY	DIY
Service Description	Textual documentation, XML Schema, WSDL1.1, WSDL2.0	Textual documentation, WSDL2.0, WADL, Swagger, RAML, Hydra, and others
Usage	Financial services, Payment gateways, Enterprise apps, High-security apps, Distributed environment, Telecommunication	Public APIs for Web services, Mobile services, Social networks
Advantages	W3C Standardized, High security, Usually easier to consume, Extensibility, distributed computing	Scalability, Flexibility, Readability, Lightweight, Better performance, Browser friendly, Easier to build
Disadvantages	More complexity, Less flexibility, Worse performance, Difficult to setup, Extra complicated coding	Less security, Lack of standards, not appropriate for distributed environments

2.1.2 Web Service Description and Documentations

The purpose of describing Web services is to generate documentation in a machine-readable format or in a human-readable format that can be employed in the process of Web services discovery, selection, and invocation. For example, WSDL (Web Services Description Language) [39] is an XML format for Web service description language that describes SOAP-based Web services in a machine-processable and readable way.

WSDL uses an optional wsdl:document element [39] as a container for human-readable documentation, but Web services developers rarely use it. Generating a WSDL document is the standard way to describe SOAP-based Web services. Some developers found WSDL syntax specifications lack some of the requirements, and therefore, they started considering semantic representation to describe their Web services. On the other hand, REST-based Web services have no standard way for description, and different academic and industry initiatives have proposed several ways to describe REST-based Web services. In general, the description of Web services can be based on syntax, or semantic description languages approaches. Each approach has its own specifications, features, requirements, details, advantages, and disadvantages.

Syntax Description Approaches

The syntax of any natural language refers to the language's structure or grammar that indicates the rules to define whether the sentence is accurately constructed [40]. If a document is not structured correctly, it will be considered as having a syntax error. In general, syntax description approaches describe Web services at the syntactic level; for example, WSDL

syntax-structured documents represent Web services data. There are several syntax-level approaches to describe and document Web services, including WSDL [39], Web Application Description Language (WADL) [41], Swagger [42][43], and RESTful API Modeling Language (RAML) [44].

For SOAP-based Web services, the syntax-based description is done through the standard description language WSDL (WSDL1.1/WSDL2.0) [39], which describes a Web service based on four major elements: port, message, types, and binding [3]. A WSDL port defines which operations can be performed and the messages that are involved, while a WSDL message defines a typed definition for the data that is communicated. WSDL 2.0 is the newer version of WSDL with some enhancements and support for describing REST-based Web services. WSDL 2.0 enables REST integration by introducing HTTP binding specifications to enable both SOAP and HTTP bindings [45].

For REST-based Web services, there is no standard for describing the functionalities of Web services. Web Application Description Language (WADL) [41], which was submitted to W3C by Sun Microsystems, was one of the earliest efforts to develop a description language for REST-based Web services. However, it was not standardized by the consortium [46]. WADL provides a machine-processable XML description of HTTP-based Web applications where REST Web services fit. REST-based Web services are described with a set of resource elements. Each resource contains a method and parameter element to describe request parameters, response parameters, media types, and status codes [41]. The request element specifies input representations, required HTTP headers, and types. Furthermore, the response element specifies the service's response representation and error handling. Some Web services developers and providers adopted WADL initially; however, it did not get that much attention in the community due to the lack of standardization and an incredibly time-consuming process of generating the descriptions. Besides, neither WSDL nor WADL was human-readable and understandable, making them difficult to use.

Several Web services (i.e., Web APIs) description languages that have made improvements over WSDL and WADL are becoming more popular today, such as Swagger [42, 43], RAML (RESTful API Modeling Language) [44], and API Blueprint [47]. Swagger (i.e., OpenAPI Specification), which is one of the most adopted description languages, uses a simple JSON or YAML (Yet Ain't Markup Language) specification to describe

REST-based Web services in terms of the HTTP methods, media types, data types, and status code [43]. Swagger's main objective is to create a RESTful contract detailing all the API resources and operations in a human- and machine-readable format. Swagger's functionality is equivalent to the WSDL document for SOAP by providing an automatically generated description that makes it easier to discover and integrate with REST APIs.

The OpenAPI Specification (OAS) allows developers to describe a Web service by providing general information about the Web API, the available paths or resources for its operations, and the input and output for each operation in addition to other parts which can be included in the documentation. One of the most popular REST-based Web services description languages, after the OpenAPI Specification, is RAML, which is supported by Mulesoft. RAML is a YAML-based language for describing REST-based Web services in a way that is readable for both humans and machines [48]. The RAML specification [44] describes resources, methods, parameters, responses, media types, and other HTTP constructs.

Today, the most common syntax description approaches for Web services are WSDL and the OpenAPI Specification. WSDL is universal because it is the standard way to describe SOAP-based Web services, which makes it inevitable regardless of its limitations. OAS is the most common REST-based Web service description language due to its simplicity, supporting tools, and data formats in addition to organizational and community support.

Semantic Description Approaches

In general, semantics relates to the linguistic study of meaning in any language based on understanding and studying the relationships between signifiers such as words, symbols, phrases, and signs and what they stand for in reality [49]. From a computer programming language perspective, semantics defines the steps a computer follows when executing a program in a specific language. This can be done by describing and modeling the relationships between the input and the output of the program. Semantics can help in writing specific language compilers and provide a better way to understand what a program is doing based on the defined model. Describing Web services semantically can be done by annotation-based (bottom-up), and ontology-based (top-down) approaches [50].

Annotation-based approaches aim to add clarity to the Web service definition by adding specific extensions allowing a connection between the syntactic definition and the semantic annotation. Furthermore, annotation enables Web services to be machine-readable, which adds more ability to understand and interpret the Web service by machines. Annotated Web services are employed to automate the process of discovery, recommendation, and composition. Unlike the ontology-based approaches that model the entire service, a bottom-up approach will represent an extension of existing technologies and semantic annotations. In ontology-based approaches, an entire high-level ontology is defined to describe Web services in a formal language without dependencies on syntax-based Web services.

An ontology can be defined as a formal and explicit specification of a shared conceptualization [51]. Ontologies define agreed common terminologies, which provide concepts with relationships in a class hierarchy structure. The semantic descriptions of Web services provide a clear definition of their terms and address the shortfalls in the understanding of the semantic meaning of Web services data and messages. Considering a semantic description language allows integrating the Web services' non-functional properties, which adds more advantages compared to syntax description languages.

Web services have various semantic Web services description languages, including the Web Ontology Language, OWL-S [52]; Web Service Modeling Ontology (WSMO) [53]; and WSDL-S [54]. There is no standard for semantic Web services description; each language has its own vocabularies and specifications. One of the drawbacks of considering semantics to describe Web services is that Web services described with a specific semantic description language will not be able to communicate with others described with different semantic description languages.

2.2. Clustering Algorithms

Clustering is an unsupervised data mining technique that partition data into separate groups of instances according to their similarity [55]. In clustering, instances that are in the same cluster must be similar as much as possible, instances that are in the different clusters must be different as much as possible, and the measurement of similarity and dissimilarity must be precise with practical meaning [56]. Clustering algorithms are employed to construct

clusters based on distance (dissimilarity) among data points considering their feature attributes.

By clustering documents, we create groups of them based on their features in such a way that the documents belonging to the same groups are similar, and those belonging in different groups are dissimilar. The process of grouping documents is done automatically by clustering algorithms, and considering different clustering methods may lead to generating different clusters on the same (dataset) documents [57]. Various clustering algorithms are employed to group Web services based on similarity or distance functions. Typically, Web services are clustered considering their extracted features from their description documents, such as WSDL, WADL, or Swagger documents, into similar groups/clusters based on a predefined similarity function or functions.

Various traditional and modern clustering algorithms have been proposed [56]. We will discuss the three popular categories of these algorithms: partition-based clustering, hierarchical-based clustering, and affinity propagation-based clustering methods. Every clustering algorithm has its advantages and disadvantages.

2.2.1 Partition-based Clustering Methods

The general idea of partition-based clustering is to regard the center of the data points as the center of the corresponding cluster [56]. The K-means clustering algorithm [58] has been cited in the area of Web services clustering, with the number of clusters identified at the beginning. The algorithm gets the number of clusters as an input, and then finds the centers of the clusters with an iterative algorithm that assigns points to each center based on the distance of the points to the center of the clusters. K-medoids [58] is another partition algorithm that has been cited in the area of Web services clustering. It is similar to K-means in that it defines clustering as an optimization problem and tries to find the best centers for the cluster in an iterative model. While K-means clustering is sensitive to noise and outliers, K-medoids is much less sensitive to noise or outliers in the data.

The complexity of partition algorithms depends on the stopping criterion and the number of iterations. Furthermore, there is a need to know the number of clusters before beginning the clustering. It is evident that in many applications such as Web services clustering, the number of clusters is not available.

2.2.2 Hierarchical-based Clustering Methods

Hierarchical clustering methods [58] group the data points into the hierarchy or tree structure in order to cluster them. Strategies for hierarchical clustering can be agglomerative (bottom-up) or divisive (top-down). Hierarchical agglomerative clustering (HAC) treats every single document as a cluster and merges pairs of clusters based on similarities until all clusters are merged into a single cluster that contains all documents [59]. Divisive clustering is where all documents start as one cluster, and clusters are split recursively until individual documents are reached. HAC is utilized more than divisive clustering in the area of Web services clustering. The first step in hierarchical algorithms is to find a distance measure to determine when to split or merge the clusters. However, in many applications, such a measure is hard to find.

2.2.3 Affinity Propagation-based Clustering Methods

Affinity propagation (AP) clustering is a newer algorithm proposed by B. J. Frey et al. in 2007 [60]. AP takes a similarity between pairs of data points as an input where real-value messages are exchanged between data points to determine the exemplars, responsibility, and availability of the data points. The algorithm stops when a high-quality set of exemplars and the corresponding set of clusters emerge. As explained by [56], the general idea of AP is that all data points are regarded as potential cluster centers and the negative value of the distance between two data points as the affinity. When the sum of the affinities of one of the data points for others is more significant, the probability of this data point to be the cluster center is higher. AP provides a simple and clear algorithm that is insensitive to noise and outliers. However, it is not suitable for a large dataset due to its high time complexity [56]. Although AP shows much lower error than other clustering algorithms, it has not been employed to cluster Web services. We believe that AP can show better results in clustering Web services. Accordingly, we will propose and validate this idea in Chapter 6.

2.3. Similarity Measures

Similarity measures are defined in general as the distance or similarity between various data points. The strength of the relationship between two data points reflects the degree of

similarity. Equivalently, the measurement of separation or divergence between two data points reflects the degree of dissimilarity [57]. Calculating the similarity between two data points or documents depends on selecting a suitable distance function over the selected dataset. Selecting a suitable distance or similarity metric that can best calculate the similarity between two documents depends on the documents' representation [61]. For example, employing Bag of Words (BOW) to represent two text documents with the vector space model (VSM) requires a distance function (such as Cosine) to find the distance between the two vectors in order to find the similarity between the two documents. Usually, similarity measures for documents are either based on syntactic similarity or semantic similarity [62], where syntactic or semantic information needs to be considered in measuring the similarity between documents.

2.3.1 Syntactic Similarity Measures

The syntactic similarity refers to how two words are similar with respect to their function and role. The syntactic relatedness is calculated with the usage of a vector space model (VSM) [61] combined with the use of some similarity measures such as Cosine distance, Euclidean distance, Minkowski distance, and Jaccard distance. In the context of Web services, we deal with Web services documents, which contain different words (data objects). In order to find the similarity between two Web services, we calculate the syntactic similarity between two documents considering their representation. The representation of a document with term-based representation methods such as BOW combined with a weighting scheme does not consider the semantic relation between terms. Therefore, the distance or similarity between two documents is based on the syntactic level, and no semantics are considered in this process.

2.3.2 Semantic Similarity Measures

The semantic similarity refers to the meaning of words and measures how two words are semantically related. Semantic similarity considers the lexical relations of synonymy and hypernymy [63] to link the semantic meanings of the documents in order to see how similar they are to each other [64]. The semantic similarities can be measured with ontologies, considering the distance between terms. An ontology is an acyclic graph containing the

definition of relationships between terms. Ontologies can be classified into general-purpose ontologies and domain-specific ontologies. WordNet, a lexical reference system developed at Princeton University [65], is an example of a general-purpose ontology that is utilized to measure the semantic similarity. One way to measure semantic similarity with the usage of an ontology, such as WordNet, is the length of the shortest path between the two terms.

Semantic relatedness, semantic distance, and semantic similarity are three related terms in measuring semantic similarity. Semantic relatedness refers to having any kind of lexical or functional association in general. This makes semantic relatedness more general than semantic similarity, which is more specific and can be considered as a type of semantic relatedness [66]. For example, the terms “Professor” and “Student” are related terms, but not similar; all similar terms are related, but not all related terms are similar. Semantic distance is the inverse of semantic relatedness, where the more two terms are semantically related, the more semantically close they are [64].

2.4. Summary

We provided background about Web services technologies, including their design (SOAP-based and REST-based), and their description and documentation methods. We presented a notion of clustering algorithms and briefly presented three methods: partition-based clustering, hierarchical-based clustering, and affinity propagation-based clustering, linking them to the area of Web services discovery and recommendation. We also introduced similarity measures for clustering Web services, including syntactic and semantic similarity measures.

In the next chapter, we introduce the research design and methodology applied in this thesis, with the associated research steps and processes. We also discuss how we contribute to the knowledge base as part of this research methodology.

Chapter 3. Research Design and Methodology

This chapter describes the research design and the overall plan for this thesis, considering the Design Science Research Methodology (DSRM) [67, 68], adopted to guide the development of useful knowledge, including design theory and artifacts, throughout this thesis. The chapter also answers some questions on how this research fits the useful knowledge base and knowledge contribution framework of DSRM.

3.1. Research Design

Research design provides an overall plan for obtaining answers to research problems and questions. On the other hand, research methodology provides systematic activities and processes for research. To accomplish the research objectives identified in Section 1.2, open-ended research questions were developed with the purpose of exploring the area of Web services discovery and recommendation through a systematic literature review (SLR), allowing us to conceptualize the problem and focus on open gaps that need to be investigated. In a pragmatic philosophical paradigm, where the concern is with applications and what will work and solve the research problem, in addition to theory, the researcher has the freedom to choose the methods, techniques, and procedures and use all available approaches to understand the research problem and meet its needs and objectives [69]. In this problem-centered research, a mixed-methods research approach was employed.

The emphasis of the research questions is on what (descriptive) and how (prescriptive) Web services discovery and recommendation can help Web services requesters find the most suitable Web service based on their functional or non-functional requirements. Descriptive research seeks to describe the current status of the research problem by providing systematic information about the phenomenon [69, 70]. It aims to make observations, classifications, and contributions to the knowledge base. Therefore, a systematic literature review is conducted in Chapter 4 as descriptive research based on predefined research questions. The SLR provides answers to specific questions to identify the current status of

methods, techniques, algorithms, datasets, evaluation measures, research trends, and future directions for Web services discovery and recommendation.

In conducting the SLR, we discovered that there are different perspectives on Web services discovery systems. For example, one research project may focus on one characteristic of a Web services discovery system that is related to clustering Web services based on their description languages, while another may focus on the matchmaking and querying perspectives of Web services discovery. Therefore, we developed a conceptual model and a typology to understand Web services discovery systems based on identified characteristics. The proposed conceptual model and typology provide the means to conceptualize the Web services discovery system by showing its elements, relationships, and differences based on the identified characteristics. The designed conceptual model and typology of Web services discovery systems are both descriptive and prescriptive knowledge. The artifacts include the constructs (typology and conceptual vocabulary of Web services discovery systems) and the model (specified set of propositions showing relationships between defined constructs of Web services discovery systems).

The identified research trends and future directions as part of the descriptive research (i.e., the SLR) build the basis for the proposed multi-layer data mining Web services discovery architecture, including its methods and instantiations. One of the gaps identified is that most of the Web services discovery process focuses on only WSDL-based or semantic-based Web services, neglecting the fact that most Web services are described in plain text. Furthermore, WSDL-based Web services suffer from a limited number of words to match user queries or to cluster Web services based on their similarities. Related auxiliary text and knowledge from different sources are utilized to extend WSDL text and its representation to alleviate this limitation. The proposed architecture and its methods use different similarity measures to cluster Web services considering both WSDL and auxiliary text to support Web services discovery where natural language processing (NLP), text mining, and various clustering algorithms are employed.

3.2. Research Scope and Delimitations

The Web services community, including researchers and practitioners, has proposed several methods to facilitate the process of Web services discovery and recommendation. This

research focuses on investigating the knowledge base of developed theories and artifacts in the area of Web services discovery and recommendation with a specific focus on clustering and association rules techniques by running a systematic literature review. Furthermore, the research focuses on understanding the different elements and characteristics of building Web services discovery systems by developing a conceptual model and a typology of Web service discovery systems. Moreover, we focus on clustering of the collected Web services description files and documentations with the developed multi-layer data mining architecture to speed up the search process for discovery purposes. Finally, the designed and developed methods are evaluated using the collected Web services datasets based on different evaluation measures.

This research does not focus on building a complete system for Web services discovery; instead, we use NLP, data mining, and analytics techniques to cluster Web services based on proposed similarities measures and to analyze and evaluate the result of clustering. This then can be employed within a complete system of Web services discovery. The clustering is based on the functional requirements of Web services where non-functional requirements are excluded. The research is general where it does not focus on a specific type of Web services instead, it investigates considering data mining techniques to speed up the process of a search engine by considering different ways to cluster Web services based on their descriptions and documentations. The SLR deliberately focuses on investigating clustering and association rules techniques for Web services discovery and recommendation with the purpose of identifying developed methods and similarity measures to serve as a knowledge base to design theories and artifacts. A dataset of collected WSDL and text descriptions is utilized in evaluating the developed artifacts.

3.3. Methodology

Design Science Research Methodology (DSRM) [67, 68] is adopted as a research methodology that provides guidelines for conducting research and determining when, where, and how to apply the guidance in research. The output of any design science research is design science knowledge, which includes prescriptive knowledge and descriptive knowledge [71]. Prescriptive knowledge is about different types of artifacts designed to improve the

natural world, including constructs, models, methods, instantiations, frameworks, architectures, design principles, and design theories. Descriptive knowledge is about the descriptions of natural, artificial, and human-related phenomena composed of observations, classifications, and measurements, and the cataloging of them into an accessible form [71]. The descriptive knowledge provides the theoretical basis for designing useful and practical artifacts. Theories are an important contribution of any research which aim to describe, explain, and expand our understanding of the world and sometimes produce predictions and expectations of what will happen and give justifications to intervene and act in certain ways [72]. The theory resulting from this research can be considered as both an analysis and description theory (descriptive) and a design and action theory (prescriptive).

In this thesis, we utilized existing descriptive knowledge in the area of Web services discovery and recommendation, where clustering and association rules are used to develop the proposed artifacts. The developed artifacts include a conceptual model to understand Web services discovery systems based on identified characteristics, constructs, and design principles, as well as a multi-layer architecture for Web services discovery with methods and techniques to cluster Web services based on various similarities. We investigate the previous descriptive knowledge base by collecting knowledge from various sources, including published articles and papers, in developing the systematic literature review (SLR). In the SLR, we investigate known artifacts and design theories employed in the area of Web services discovery and recommendation where clustering and association rules were applied to solve the research problem in the past. Furthermore, we contribute to the descriptive knowledge base and theories in providing analysis, synthesis, and observation of the previously proposed methods. We provide a classification of methods and identify algorithms, similarity measures, and evaluation metrics. Moreover, we classify the datasets considered for evaluating clustering and association rules techniques for Web services discovery and recommendation, identify their location, method of collection, types, and nature. Web services discovery and recommendation trends and future directions are discussed as part of the SLR.

The research contributions of any design science research can fit the DSR Knowledge Contribution Framework and its four quadrants presented by [71]. This re-

search contribution can be positioned in the improvement quadrant as it develops new solutions for known problems resulting in research opportunities and knowledge contributions. The goal of the improvement quadrant is to create better solutions, which are in the form of products, processes, services, technologies, or ideas. In the improvement quadrant, contributions to the prescriptive knowledge base can be in the form of artifacts in one or more levels. Level 1 includes situated instantiations to evaluate the level of improvement. Level 2 is more general in the form of constructs, methods, models, and design principles, which are proposed as research improvements. Level 3 is a mid-range design theory that results from an improved understanding of the problem and solution spaces.

This research fits in the improvement quadrant of the DSR contribution framework where the problem maturity of Web services discovery and recommendation is high and specific solutions maturity is low. This shows research opportunities in the area and possible continuations to the knowledge base. The contributions of this research, according to the identified levels, are as follows:

- Level 2: An architecture for Web services discovery with various similarity measures to cluster Web services, a conceptual model to understand Web services discovery systems, a conceptual vocabulary, and design principles; and
- Level 1: The implementation of the proposed methods to cluster Web services based on different representations and similarities for evaluation purposes.

In addition to the contribution to the prescriptive knowledge base, we contributed to the descriptive knowledge by expanding our understanding of the previously proposed methods by providing observations, classifications, expectations, and predictions as part of the systematic literature review.

3.4. Research Processes and Framework

We employed a formalized DSRM process model presented by [73], which consists of six activities in a nominal sequence, as shown in Figure 3-1.

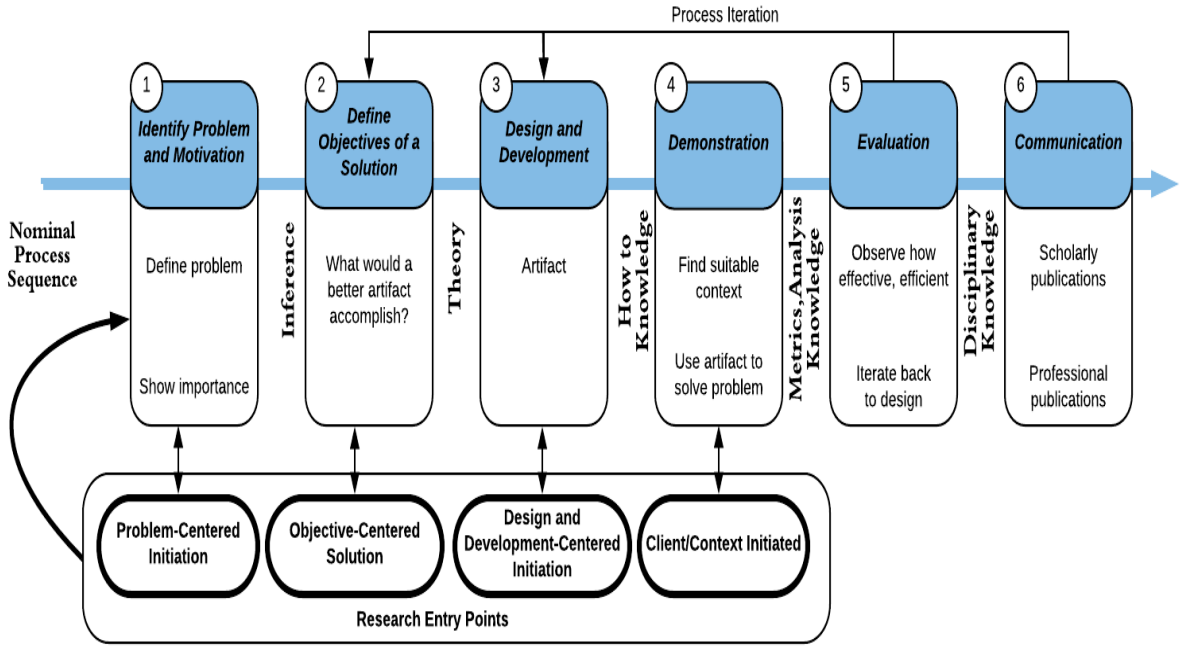


Figure 3-1: Design Science Research Methodology Process

In the first activity, the problem identification and motivation are where the value of the potential solution is justified. The second activity defines the solution objectives in which objectives are inferred rationally from the problem definition and specifications. The third activity is the design and development of an artifactual solution. This includes identification of an artifact's functionality, method, and architecture as well as the development of the actual artifact if needed. The fourth activity demonstrates the efficacy of the designed artifact in solving one or more instances of the research problem. This may involve the use of experimentation, proof, simulation, case studies, or another appropriate activity. The fifth activity evaluates and observes how the developed artifact will support a solution to the research problem. This activity includes comparing the solution objectives to the results from using the artifact in the demonstration activity. The nature of the research may dictate whether iteration to activity three is feasible or not. The sixth activity communicates problems, artifacts, importance, utility, and effectiveness to researchers and audiences through publications.

We applied a problem-centered approach of a DSRM process [73] and aligned the activities and methods with the guidelines of DSRM [67] as follows:

1. **Problem identification and motivation:** We identify the research problem, main requirements, challenges, motivation, and value of the proposed solution. The research problem, as outlined in Table 1-1, highlights the challenges of searching in a large number of Web services. It requires significant time and effort when service requesters are unable to identify the best Web services to meet their functional or non-functional requirements. The main requirement is to cluster Web services to minimize the search space considering Web services descriptions and specifications in the matchmaking process and recommend a Web service based on user preferences and requirements in the selection process. The challenges and motivations are discussed in Section 1.1.
2. **Define the objectives for a solution:** We explain how the new artifacts offer support and provide a solution to the problem. We analyze existing Web services discovery systems as well as recommendation approaches in the SLR to specify a design strategy for developing a multi-layer Web services discovery architecture which clusters Web services based on considering different word representation and embedding and syntactic and semantic similarity measures to minimize the search process by grouping similar Web services into one cluster. In addition, we provide a solid understanding of Web services discovery systems by identifying their various characteristics with the proposed conceptual model and typology.
3. **Design and development:** We design new artifacts that meet the objectives of this research. This involves determining the artifacts' desired functionality, methods, and architecture with which the actual artifacts are developed. In this research, the designed and developed artifacts are a conceptual model and a typology to understand Web services discovery systems based on identified characteristics, and an architecture for Web services clustering based on multi-layer data mining techniques and methods and their design theories.
4. **Demonstration:** To demonstrate how the proposed artifacts can solve one or more instances of the research problem, we employ an analytic and descriptive approach to demonstrate the utility of the conceptual model and the typology to understand Web services discovery systems. As for the proposed architecture for Web services clustering, we use an experimental approach to demonstrate how the proposed artifact can solve or improve the investigated problem. With selected Web services datasets for

clustering purposes, the experiment shows how the proposed methods and techniques can solve and support the process of Web services discovery. We identify the experiments, the datasets, and the experimental environment.

5. **Evaluation:** To observe how efficient and effective the proposed artifacts are in solving the research problem, we employ a descriptive evaluation method (informed argument) by which we use information from the related knowledge base and relevant research to show the utility and effectiveness of the proposed conceptual model in understanding Web services discovery systems and typology based on the identified characteristics. Evaluation metrics based on the experimental method are employed to evaluate the effectiveness and efficiency of the proposed multi-layer architecture in clustering Web service. We consider evaluation metrics utilized by other researchers for the same purpose, as indicated in the SLR for analytical purposes.
6. **Communication:** We communicate part of the research results to peers through conferences and journal publications. A list of publications is provided in Section 1.5.

The nature of this research dictates the need for iteration in activity 3 in the process of designing and developing the multi-layer data mining architecture. After evaluating the first developed architecture, we had to iterate back to activity 3 where we added new similarity measures and included the new state-of-the-art word embedding models within the proposed architecture.

3.5. Summary

In this chapter, we described the research design and methodology followed in conducting this thesis. We categorized our contributions to the advancement of knowledge into descriptive knowledge and prescriptive knowledge. Furthermore, we positioned our research into the DSR Knowledge Contribution Framework.

In the next chapter, we introduce the systematic literature review for Web services discovery and recommendation, which focuses on clustering and association rules techniques. The conducted SLR consists of three phases: the planning phase, the conducting phase, and the reporting and analysis phase. The SLR is an integral component of the descriptive knowledge contributions of this thesis.

Chapter 4. Web Services Discovery and Recommendation Approaches Using Clustering and Association Rules Techniques: A Systematic Literature Review

This chapter provides a systematic literature review (SLR) of Web services discovery and recommendation approaches, considering clustering and association rules techniques. It aims to identify, summarize, and systematically compare current clustering and association rules techniques for Web services discovery and recommendation by providing answers to a set of research questions. The SLR contributes to this research by providing descriptive knowledge that discusses the previous knowledge base. Prior recent studies published in this area of research are reviewed and synthesized, and then gaps are identified. The content of this chapter is published in the Springer Nature Journal of Computer Science [31].

4.1. Motivation

The motivation and reasons to conduct this SLR are as follows:

- The discovery problem is inherently difficult because of the large scale of Service Oriented Architecture (SOA) systems. This includes Web services, which are dynamic, changing, and uncertain in nature. The scope of services in SOA rapidly changes as services are added and removed or modified. Developers abandon older standards as new standards emerge. Web services discovery and recommendation are critical to the success of Web services because the services are useless if service requesters cannot discover them;
- The use of Web services in the industry is gaining attention. Thus, it is critical to identify techniques facilitating Web services discovery and recommendation to allow developers to efficiently use, discover, and select Web services;
- Data mining, text mining, and machine learning techniques provide solutions to facilitate the process of Web services discovery and recommendation when algorithms for

clustering, classification, and association rules have been applied. The focus of clustering and association rules is based on the identification of algorithms, similarity measures, datasets, and evaluation metrics. These favor the building of practical Web services discovery and recommendation systems. This will help future researchers to make sense of the Web services discovery and recommendation landscape in employing clustering and association rules;

- Over the past decade, several clustering and association rules techniques for Web services have been proposed to facilitate the process of Web services discovery and recommendation. Thus, there is a need to investigate the characteristics and details of these techniques with a systematic method and quality requirements; and
- Data mining and analytics is a fast-evolving field, and this SLR provides useful information and synthesis about the current state of Web services discovery and recommendation research considering clustering and association rules techniques, in particular with the lack of existing comprehensive SLR in this area.

4.2. Review Methodology

We follow general guidelines and procedures according to [74, 75]. The three phases of an SLR are the 1) planning; 2) conducting; and 3) reporting phases, as illustrated in Figure 4-1. The planning phase involves four activities: 1) identification of the need for an SLR; 2) definition of the search questions; 3) development of the review protocol; and 4) evaluation of the review protocol. Based on three research questions and search strings identified in the planning phase, six digital databases were searched. The results retrieved 4,581 papers, which included several formats (long, short, and poster papers) and duplicates. A rigorous two-stage scanning and filtering process was followed according to predefined criteria. Employing Covidence, an online Web-based tool to support the production of systematic reviews [76], resulted in 66 papers being selected based on the two screening stages. We conducted a manual search based on snowballing of the 66 selected papers, which led to identifying two additional papers after applying the selection criteria. The 68 primary selected papers went through a quality assessment, as suggested by [74, 77, 78]. We employed the quality assessment to weigh the relevance of studies as the results synthesize and support the validity of the selected papers in this review. The final activity identified

57 papers as final selections. We collected the required information to answer research questions as well as analyze and summarize the results.

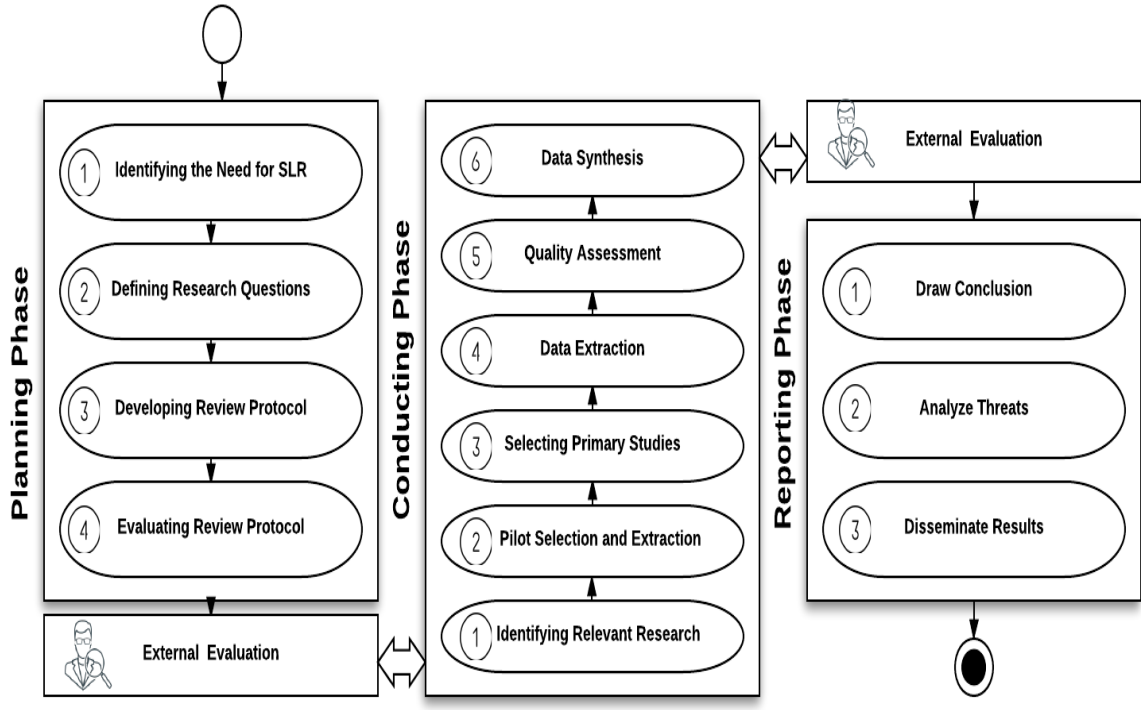


Figure 4-1: Review Methodology Phases

4.3. Phase 1: Planning Phase

The planning phase identifies research questions, search strategy, selection process, quality assessment, and data extraction. Moreover, we evaluate the review protocol as part of this phase.

4.3.1 Research Questions Definitions

We considered research questions from the viewpoints of population, intervention, comparison, outcome, and context (PICOC) as outlined in Table 4-1. The general goal and scope of the SLR were formulated through PICOC. We employed population and intervention in search terms and keywords. Research questions addressed in this SLR include:

- ***RQ1***: What techniques, methods, and algorithms are studied in clustering and association rules for Web service discovery and recommendation?

- ***RQ2***: What are the most common datasets cited to validate the proposed clustering and association rules approaches to facilitate Web services discovery and recommendation?
- ***RQ3***: What are the trends and future research directions related to the discovery and recommendation of Web services?

Table 4-1: Criteria to Define Scope, Goals, and Strings

Criteria	Describe / Reflect
Population	Web services, Web services discovery, Web services recommendation
Intervention	Clustering, Association rules
Comparison	Comparison of methods, algorithms, datasets, and evolution metrics
Outcome	Classification and comparison, trends and future research directions
Context	Systematic investigation to consolidate peer-reviewed and academic research

A summary of the research questions and their motivation is listed in Table 4-2. RQ1 identifies 1) Techniques employed in the investigated studies, including clustering-based and association rules-based Web services discovery and recommendation; 2) The proposed method to facilitate the process of Web service discovery and recommendation. Clustering and association rules techniques are employed in the proposed method and supported by other techniques, such as natural language processing (NLP) and information retrieval (IR). Methods include the proposed processes/framework to facilitate Web service discovery and recommendation, similarity measures and evaluation matrices to validate the work; and 3) The clustering and association rules algorithms utilized to support the object of the study. For example, in clustering, K-means, hierarchical, and DBSCAN algorithms are employed. In association rules, Progressive Size Working Set (PSWS) and Apriori algorithms are used.

Table 4-2: Research Questions and Motivation

RQ#	Research Question	Motivation
RQ1	What techniques, methods, and algorithms are studied in clustering and association rules for Web services discovery and recommendation?	- Identify and classify existing techniques. - Identify and compare proposed methods with their similarity measures and evaluation metrics. - Determine and compare algorithms.
RQ2	What are the most common datasets cited to validate the proposed clustering and association rules approaches to facilitate Web services discovery and recommendations?	- Identify discovery benchmark dataset collections and their locations. - Identify recommendation benchmark dataset collections and their locations. - Determine the type, purpose, and method of collection of designated datasets.
RQ3	What are the trends and future research directions related to the discovery and recommendation of Web services?	- Understand and disclose challenges. - Identify research gaps. - Identify trends and future directions.

4.3.2 Search Strategy

To avoid missing relevant papers, we employed generic search strings, which included a broad number of articles in its initial results. We identified the search terms and keywords by following five criteria identified by [74]. Based on the PICOC, we utilized search terms and keywords to construct search strings as follows:

$$(P_1 \text{ OR } P_2 \ldots \text{OR } P_n) \text{ AND } (I_1 \text{ OR } I_2 \ldots \text{OR } I_n)$$

$$P_n : \textit{population terms}, I_n : \textit{intervention terms}$$

Different spellings were considered when constructing the search strings by considering truncation as an asterisk (*) and Boolean operators AND and OR. When databases allowed, the advanced search option inserted the complete search string. To extend the scope of the results, we considered the population and intervention in the search strings. We included a comparison in the primary study selection process by filtering studies that did not contain methods, algorithms, datasets, and evaluation metrics. This compensated for the exclusion of the comparison in the search strings. The search strings were as follows:

("Web Service*" OR "Web Service* Discovery" OR "Web Service* Recommend*")
AND
(“Clustering" OR "Association Rule*")

We used an automatic search method to search selected, well-known digital libraries. We applied the search strings to search the six digital libraries listed in Table 4-3 with the number of initial retrieved papers from each digital library. Search strings were adjusted according to the requirements of the selected database search engines. Prior to running a full search, the search string’s effectiveness was examined as recommended by [74], considering a list of known publications [20, 79, 80]. The search, which was conducted 1st of February 2018, was limited to studies published between 2006 and January 2018. We chose 2006 as the starting year for this study because it was the year that major companies (i.e., Microsoft, IBM, and SAP) ended their UDDI business registry (UBR) project for Web services discovery [14] and solutions. We also conducted a manual search as recommended by [74].

Table 4-3:Digital Libraries Selected and Initially Retrieved Result

	Database Name	URLs	Initially Retrieved
1	ACM Digital Library	http://dl.acm.org/	637
2	Science Direct	http://www.sciencedirect.com/	62
3	Scopus	https://www.scopus.com/	2196
4	IEEE Xplore	http://ieeexplore.ieee.org/	342
5	Web of Science	https://webofknowledge.com/	71
6	Springer Link	http://link.springer.com/	1273
Total Number of papers			**4581**

4.3.3 Study Selection Process

We identified the selection process and the inclusion and exclusion criteria, as outlined in Table 4-4. A four-phase search and selection process was employed to search the digital libraries, filter results, and collect and screen relevant papers. In the first phase, an automatic search of the selected digital libraries was performed using search strings. In the second phase, all retrieved studies were merged after duplicates were removed. Two

screening stages were performed. Title and abstract screening was applied where the inclusion and exclusion criteria were checked against the title and abstract of the retrieved studies. Full-text screening was applied where the inclusion and exclusion criteria were checked against the full text of the initially selected studies. In the third phase, a manual search was performed by snowballing from reference lists of the primary selected studies as recommended by [74]. In the final phase, we identified the selected papers' quality based on quality assessment criteria (Section 4.3.4). The final selected papers were identified based on the quality (relevance) score. The overall searching and selection process is illustrated in Figure 4-2.

Table 4-4: Inclusion and Exclusion Criteria

Criteria		Rationale
Inclusion	I1: Studies in the form of academic peer-reviewed conference and journal articles.	Peer-reviewed papers guarantee a level of quality and a well-trusted volume of content.
	I2: Studies published between 2006 and January 2018.	Eleven years has been selected as the SLR period. This decision was made based on the start time (2006) that UBRs closed, and alternative Web service discovery and recommendation solutions emerged.
	I3: Studies with a proposed solution and evaluation/validation of Web services discovery and recommendation.	We are looking for methods, algorithms, datasets, and evaluation metrics using clustering and association rules in Web services discovery and recommendation.
Exclusion	E1: Studies in the form of abstracts, posters, short papers (less than six pages), technical reports, tutorial summaries, or books.	These studies do not usually provide the information needed for the purpose of this review.
	E2: Studies that are not in the English language.	Although there are different languages, we excluded non-English studies due to language limitations.
	E3: Studies where the keywords (Web services discovery or clustering or association rules or Web services recommendation) were not addressed explicitly in the article.	We are investigating studies which are related to the research questions. These keywords are needed for screening the primary selected studies.
	E4: Studies that do not include their methods, datasets, or validation procedures.	Studies will be eliminated if they do not consider the full-text scanning phase. This is because they will not relate to the research questions.

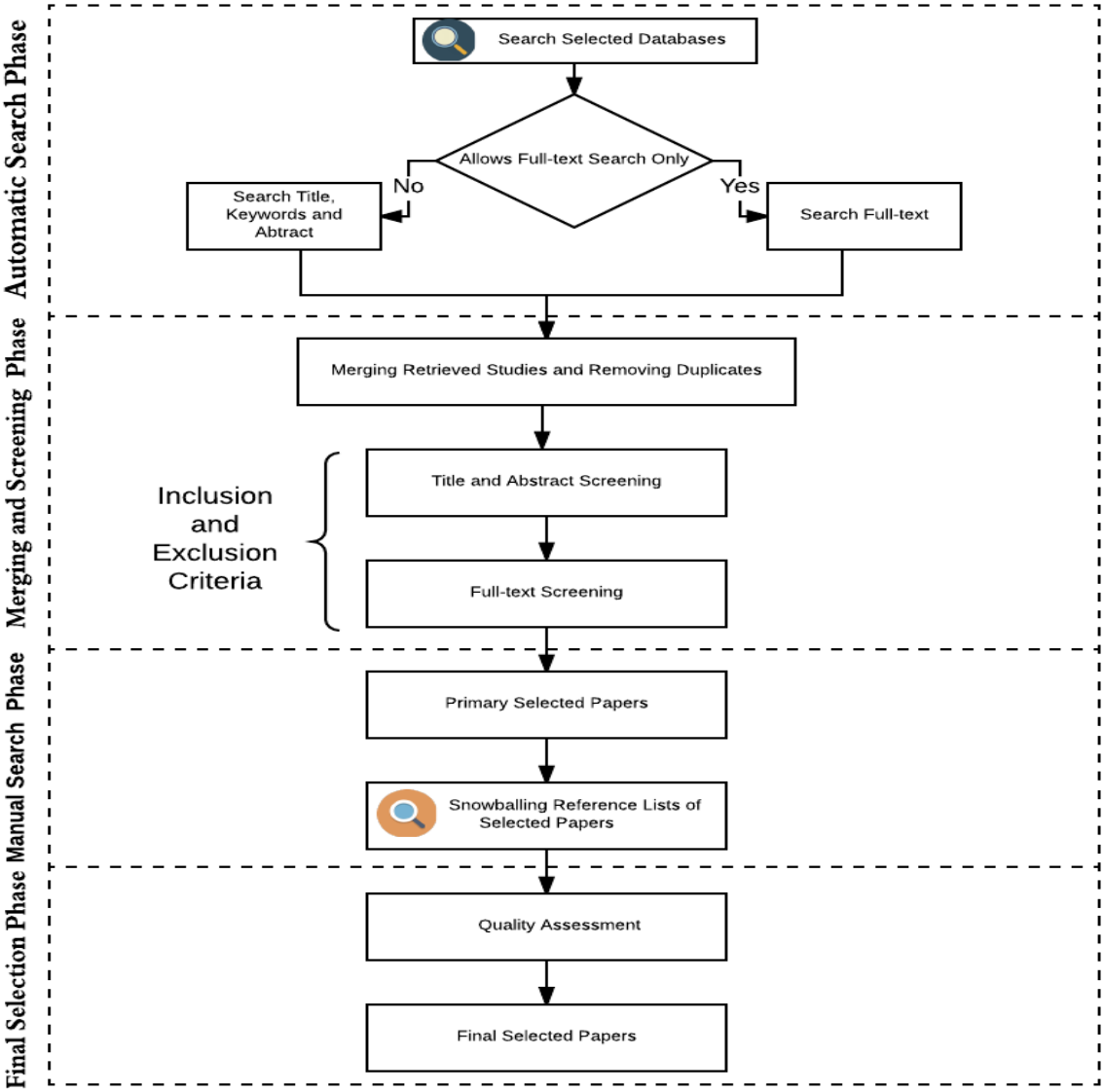


Figure 4-2: Search and Selection Process

4.3.4 Quality Assessment of the Primary Selected Studies

We employed a quality assessment method based on defined questions to assess the selected papers' relevance to minimize bias and maximize validity [74, 77]. The quality assessment was a means for weighting the importance and relevance of studies during the synthesis stage of this SLR. It also supported the validity of the selected papers in this review. A numerical quantification (quality score) is assigned to rank the selected papers based on the quality assessment questions. This was done to help with the selection of the most related studies. It was not utilized as an absolute measure of a paper's quality, nor to

compare the papers and authors' work against each other. It indicates the relevance of the selected papers to our research questions. Based on the recommendation of [77, 78], we divided the quality questions to assess general questions (GQ) and specific questions (SQ) as outlined in Table 4-5. The GQs focused on the quality of reporting, including the papers' rationales, aims, and context. The SQs concentrated on the technical rigor and credibility of the papers.

Table 4-5: Quality Assessment Questions

General Quality Assessment Questions (GQ)		Score		
		Yes 1	Partly 0.5	No 0
GQ1	(Motivation and Problem) Do the authors explain the aims and objectives of the primary study? Is there a strong and clear rationale for why the study is undertaken?			
GQ2	(Research Environment) Is the context in which the research was carried out adequately described?			
GQ3	Are the contributions of the study well-presented and inline with the result?			
GQ4	Are the research methodology and design clearly stated?			
Specific Quality Assessment Questions (SQ)		Score		
		Yes 1	Partly 0.5	No 0
SQ1	Is the study clearly focused on Web services discovery or Web services recommendation?			
SQ2	Are the techniques, methods, and algorithms in the study clearly described and linked to their usefulness? (RQ1)			
SQ3	Does it identify the dataset information (name, location, etc.) for validation? (RQ2)			
SQ4	Is there more than one evaluation metric used in the study for validation? (RQ2)			
SQ5	Are the limitations and future directions clearly stated in the study? (RQ3)			

To assess quality, each paper was evaluated against GQ and SQ quality assessment questions. The answers to the questions could be "Yes," "Partly," or "No." Numerical values were assigned to the answers (1 = "Yes," 0 = "No," and 0.5 = "Partly"). The final quality score for each primarily selected paper was calculated by adding up each question's scores. The maximum score was 9; scores of 7.5-9 represented high-quality (high-relevance) papers, scores less than 7.5 and greater than or equal to 5.5 were acceptable with average quality (relevance), and scores less than 5.5 were low quality (relevance) and

resulted in exclusion. Some of the primarily selected papers were recent publications. Therefore, they were not ranked based on the number of citations.

4.3.5 Data Extraction

The data extraction phase involved collecting data and information relevant to the RQs from the primarily selected papers. We designed a data extraction form, as outlined in Table 4-6. The test-retest process [74] was employed to check the consistency and accuracy of the extracted data on the original sources. This was performed during the evaluation stage.

Table 4-6: Data Extraction Form

#	*Data Extraction Category*	*Description*	*Purpose*
Study's General Description			
1	Identifier	Identifier number (DOI)	
2	Date	Data extraction date	
Study Description			
1	Title	Title of study	
2	Authors' Names	Name of study authors	
3	Country	Country of study publication (first author)	
4	Publication Year	Publication year	
5	Type	Conference proceedings, journal article	
6	Venue	Name of conference or journal	
7	Author Affiliation(s)	Affiliation of author(s) (first author)	
Study Content			
1	Objectives	Objectives of study	
2	Method	Method to support objectives	RQ1
3	Techniques	Clustering-discovery, clustering-recommendation, association rules-discovery, association rules-recommendation, or combined	RQ1
4	Algorithm	Name of algorithm in study	RQ1
5	Evaluation Metric	Name of evaluation metric	RQ1
6	Metric Purpose	Evaluation metric purpose	RQ1
7	Dataset	Name of dataset for validation	RQ2
8	WS in Dataset	Number of WS in dataset	RQ2
9	Se vs. Sy Dataset	Semantic or syntax datasets	RQ2
10	Dataset Purpose	Recommendation or discovery	RQ2
11	Future Directions	Future directions	RQ3
12	Limitations	Restriction of the work	RQ3
13	Challenges	What challenges in this study are associated with Web services discovery and recommendation?	

4.3.6 Review Protocol Evaluation

Based on the recommendation of [74, 75], we confirmed that search strings were appropriately derived from the research questions. Furthermore, the search strings' effectiveness was preliminarily examined with a list of known publications [20, 79, 80] before going to the full search. Search strings were constructed with the university librarian who had experience in working with the selected digital databases. We followed the test-retest approach to determine the final search strategy.

To evaluate the study selection process, we ran and tested two stages of screening. This included the title and abstract screening and full-text screening on a sample of retrieved papers with inclusion and exclusion criteria. As recommended by [74], we re-evaluated the screening stages to check the sample consistency. For evaluating quality assessment and data extraction, we performed a pilot study with a sample of the primarily selected papers to check the consistency and accuracy of the data. We confirmed that the extracted data addressed the research questions. Figure 4-3 illustrates the methods used in evaluating the review protocol. Furthermore, we externally evaluated the protocol before conducting the review. External experts including two professors were regularly contacted for assessments, feedback, and recommendations to refine the review protocol. We refined the review scope, improved the search strategy, and improved the inclusion and exclusion criteria based on this feedback.

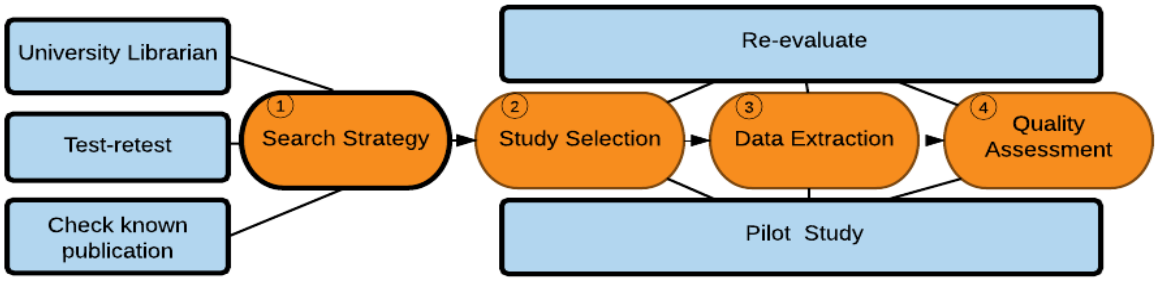


Figure 4-3: Evaluation Review Protocol

4.4. Phase 2: Conducting Phase

In this phase, we conducted the search process considering the selected digital libraries to identify the final selected papers for the SLR.

4.4.1 Identifying Relevant Research

Based on search strings defined in the planning phase, we searched selected digital libraries and retrieved results. Search strings were adapted and modified to fit the restrictions and requirements of the digital libraries. Some digital libraries supported searches for abstract, title, and keywords; other libraries supported full-text search. The search was conducted in February 2018. It was limited to studies published between 2006 and 2018. The search queries for each digital library are shown in Appendix B-1.

The number of retrieved papers from each of the selected digital databases is outlined in Table 4-3. The total number of initially retrieved papers was 4,581, including several formats and duplicate papers. A total of 482 papers were excluded due to duplications. A selection process was performed on the remaining 4,099 papers. The process proceeded with pilot selection and data extraction. All of the 4,099 papers from the digital library search process were transferred into Covidence [76] which is an online Web-based software for managing and streamlining systematic literature review. We utilized Covidence for facilitating the process of merging papers, removing duplicates, and conducting the two-stage screening.

4.4.2 Pilot Selection and Extraction

Before running the selection process on 4,099 papers, we ran a pilot study to test and evaluate selection, data extraction, and quality strategies. We followed the two stages of screening, including title and abstract screening and full-text screening with Covidence. The selection process stopped at ten papers. The data extraction and quality assessment strategies were then applied. Based on this pilot study, we re-evaluated the selection, data extraction, and quality assessment strategies and refined the review protocol. Figure 4-4 indicates the process of conducting the pilot study based on a 10-paper sample. We refined the review protocol based on the result of the pilot.

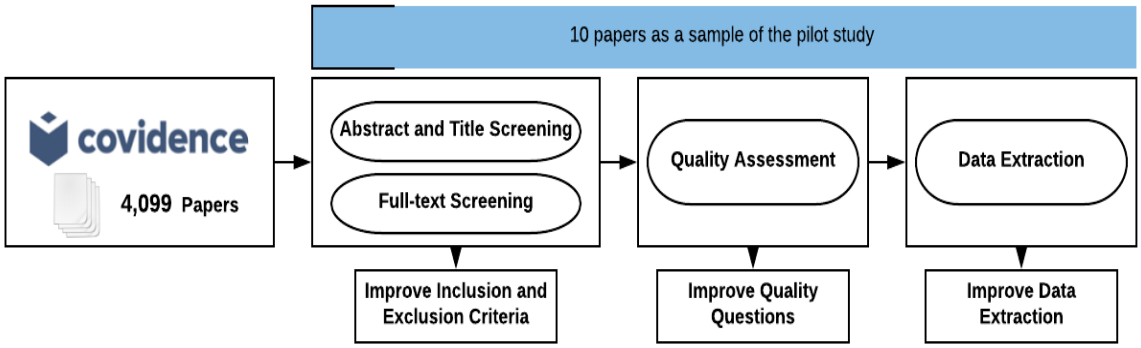


Figure 4-4: Conducting a Pilot Study

4.4.3 Selecting Primary Studies

Covidence was utilized to import the extracted metadata from the six selected digital libraries. It was employed to fetch initial studies' information and contents into one place. After deleting the 482 duplicates with Covidence, the selection process consisted of two stages to manually screen the remaining 4,099 papers as follow:

- ***Abstract and title screening***: Covidence was utilized to screen papers by looking at keywords, filtering keywords, and applying inclusion and exclusion criteria. Based on this stage of screening, 3,926 papers were excluded. We found that some of the retrieved papers did not address Web services discovery and recommendation. For example, several papers were related to biology, physics, and nature. Furthermore, some digital databases did not provide accurate results based on the chosen search strings. We excluded papers that did not explicitly address Web services discovery and recommendation. Furthermore, we excluded abstracts, posters, short papers (less than six pages), technical reports, tutorial summaries, and books.
- ***Full-text screening:*** During this stage, 173 papers were scanned, and 107 papers were excluded. Finally, 68 papers were selected based on the two screening stages.

The manual process, which was based on snowballing reference lists of the primarily selected papers, led to the identification of nine additional primary papers. After applying inclusion and exclusion criteria, seven of those papers were excluded, and two additional paper was included. Figure 4-5 depicts a flowchart to illustrate the primarily selected papers' selection process, including the manually selected papers going to the quality assessment step.

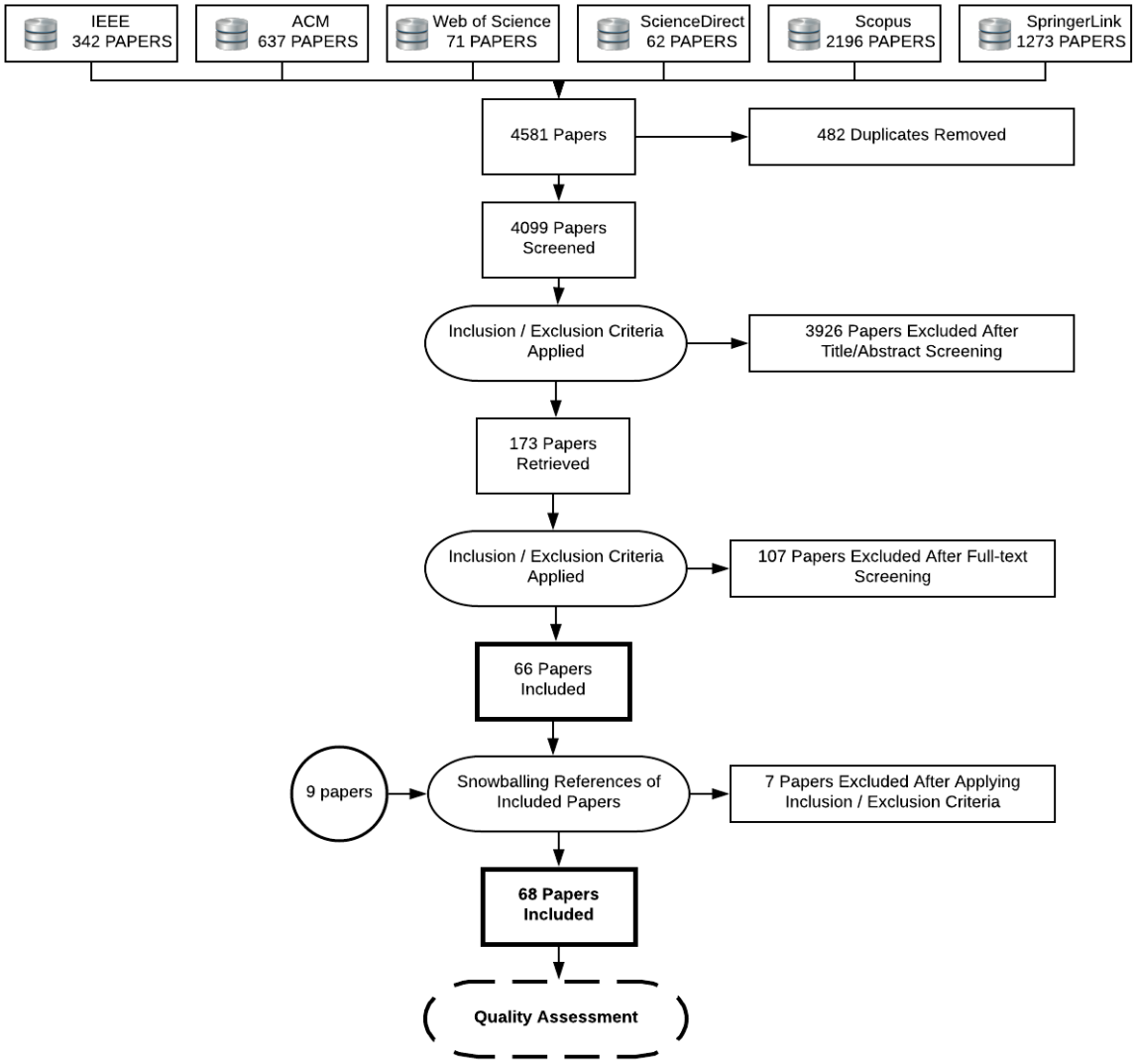


Figure 4-5: Overall Selection Process

4.5. Phase 3: Reporting and Analysis Phase

In this phase, we employed various factors to analyze the selected papers. We analyzed the content of the final papers to answer the research questions of this SLR.

4.5.1 Analyzing the Selected Papers

Quality of selected studies

The quality of the selected papers was identified with the quality assessment questions previously described. Figure 4-6 indicates the primarily selected papers that successfully went through the data extraction and quality assessment process. Green bubbles provide

the number of papers with high-quality scores (between 9 and 7.5). Blue bubbles provide the number of papers with average acceptable-quality scores (less than 7.5 and greater than or equal to 5.5). Red bubbles provide the number of papers with low-quality scores, which lead to exclusion (less than 5.5). The results suggest that the selected papers are of relatively average acceptable quality with 15 papers are in high quality, and 42 papers are in acceptable quality. Eleven papers are excluded as low quality. The list of the 57 final selected papers is illustrated in Table 4-7 after data extraction and quality assessment.

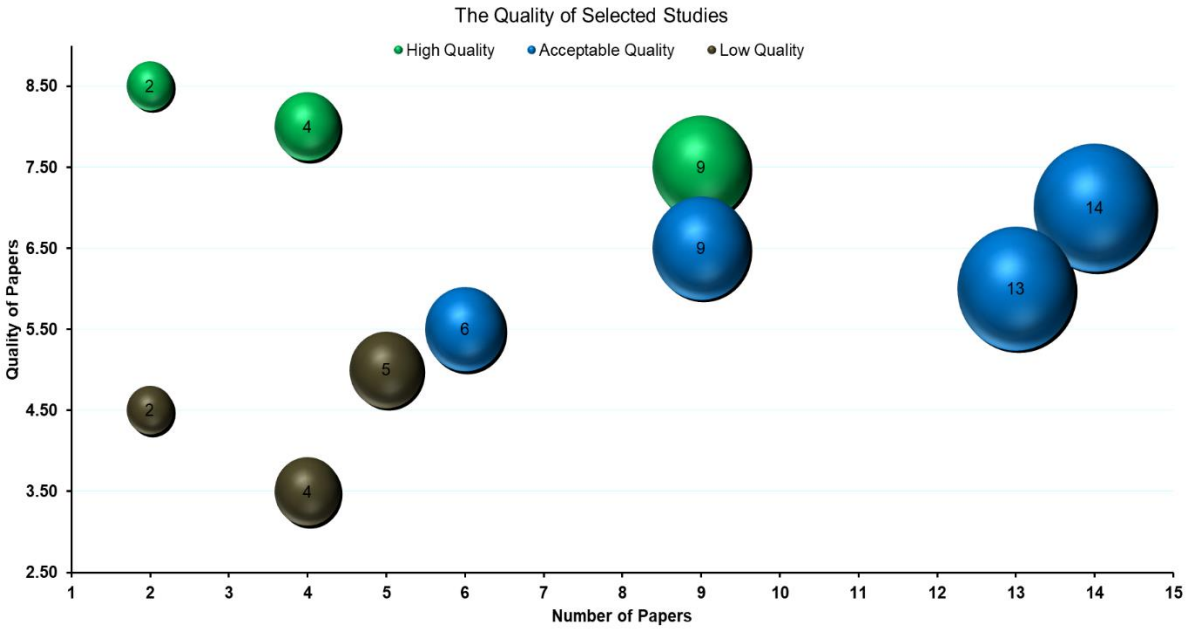


Figure 4-6: Quality of Selected Papers

Table 4-7: List of Final Selected Papers

ID	Title	Venues	Year	Citation	Quality Score
[FS1]	TAP: A personalized trust-aware QoS prediction approach for web service recommendation [81]	KNOSYS (J)	2016	0	8.5
[FS2]	Clustering Web services to facilitate service discovery [80]	KAIS (J)	2014	21	8.5
[FS3]	Modeling and exploiting tag relevance for Web service mining [82]	KAIS (J)	2014	9	8
[FS4]	Web Service Recommendation via Exploiting Location and QoS Information [25]	TPDS (J)	2014	26	8
[FS5]	Web service discovery among large service pools utilizing semantic similarity and clustering [83]	EIS (J)	2015	1	8
[FS6]	Characterization and search of web services through intensional knowledge [15]	JIIS (J)	2016	1	8
[FS7]	Automatic Tagging Web Services Using Machine Learning Techniques [84]	WI-IAT (C)	2014	4	7.5
[FS8]	Leveraging Formal Concept Analysis with Topic Correlation for Service Clustering and Discovery [85]	ICWS(C)	2014	12	7.5

[FS9]	Location-based Hierarchical Matrix Factorization for Web Service Recommendation [86]	ICWS(C)	2014	11	7.5
[FS10]	Collaborative personal profiling for web service ranking and recommendation [87]	ISF(J)	2015	9	7.5
[FS11]	CluCF: a clustering CF algorithm to address data sparsity problem [88]	SOCA (J)	2016	1	7.5
[FS12]	Transferring auxiliary knowledge to enhance heterogeneous web service clustering [89]	IJHPCN (J)	2016	0	7.5
[FS13]	Clustering facilitated web services discovery model based on supervised term weighting and adaptive metric learning [90]	IJWET (J)	2013	5	7.5
[FS14]	Recommending web service based on user relationships and preferences [91]	ICWS (C)	2013	7	7.5
[FS15]	Context Aware Post-filtering for Web Service Clustering [92]	SCC (C)	2014	1	7
[FS16]	Personalized Web Service Ranking via User Group Combining Association Rule [93]	ICWS (C)	2009	49	7
[FS17]	User-QoS-based Web Service Clustering for QoS Prediction [94]	ICWS (C)	2015	0	7
[FS18]	Service discovery acceleration with hierarchical clustering [79]	ISF(J)	2015	5	7
[FS19]	Co-clustering WSDL Documents to Bootstrap Service Discovery [95]	SOCA (C)	2014	2	7
[FS20]	Service Recommendation Using Customer Similarity and Service Usage Pattern [96]	ICWS (C)	2015	1	7
[FS21]	Time-aware Semantic Web Service Recommendation [97]	SCC (C)	2015	1	7
[FS22]	Collaborative filtering based hybrid approach for web service recommendations [98]	RJASET (J)	2014	0	7
[FS23]	Particle swarm optimization for clustering semantic web services [99]	ISPDC (C)	2011	5	7
[FS24]	Clustering WSDL documents to bootstrap the discovery of Web services [20]	ICWS (C)	2010	193	7
[FS25]	Web Services Discovery Based on Latent Semantic Approach [100]	ICWS (C)	2009	12	6.5
[FS26]	A Hierarchical Matrix Factorization Approach for Location-based Web Service QoS Prediction [24]	SOSE (C)	2014	0	6.5
[FS27]	Web Service Clustering using a Hybrid Term-Similarity Measure with Ontology Learning [101]	IJWSR (J)	2014	5	6.5
[FS28]	Web Service Clustering using Multidimensional Angles as Proximity Measures [102]	TOIT (J)	2009	85	6.5
[FS29]	Semantic-Based Clustering of Web Services [103]	JWE (J)	2015	0	6.5
[FS30]	QoS-aware service selection via collaborative QoS evaluation [104]	WWW (J)	2014	6	6.5
[FS31]	Taxonomic clustering and query matching for efficient service discovery [105]	ICWS (C)	2011	19	6.5
[FS32]	Leveraging Auxiliary Knowledge for Web Service Clustering [22]	CJE (J)	2016	0	6
[FS33]	Semantics-Based Automated Service Discovery [106]	TSC (J)	2012	42	6
[FS34]	Ontology learning method for Web services clustering [107]	ICCCT (C)	2012	0	6
[FS35]	Cluster-Based Web Service Recommendation [23]	SCC (C)	2016	0	6
[FS36]	Similarity analysis of service descriptions for efficient Web service discovery [21]	DSAA (C)	2014	1	6
[FS37]	Ontology Learning with Complex Data Type for Web Service Clustering [108]	CIDM (C)	2014	0	6
[FS38]	Web Service Clustering Using Relational Database Approach [109]	IJSEKE (J)	2015	1	6
[FS39]	Improving REST Service Discovery with Unsupervised Learning Techniques [18]	CISIS (C)	2015	1	6
[FS40]	A clustering-based QoS prediction approach for web service recommendation [26]	ISORCW (C)	2012	7	6
[FS41]	Web service clustering using text mining techniques [19]	IJAOSE (J)	2009	93	6
[FS42]	Efficiently finding web services using a clustering semantic approach [10]	CSSSIA (C)	2008	77	6
[FS43]	Research on Web service discovery with semantics and clustering [110]	ITAIC (C)	2011	20	5.5

[FS44]	A concept analysis approach for guiding users in service discovery [111]	SOCA (C)	2012	4	5.5
[FS45]	DaaS: Cloud-based mobile Web service discovery [112]	PMC (J)	2013	22	5.5
[FS46]	WS-HFS: A Heterogeneous Feature Selection Framework for Web Services Mining [113]	ICWS (C)	2015	1	5.5
[FS47]	Learning Sparse Functional Factors for Large-Scale Service Clustering [114]	ICWS (C)	2015	3	5.5
[FS48]	Hierarchical clustering based web service discovery [115]	ICISO (C)	2014	1	5.5
[FS49]	QoS Aware Service Clustering to Bootstrap the Web Service Selection [116]	SCC (C)	2017	0	6
[FS50]	A new QoS-aware web service recommendation system based on contextual feature recognition at server-side [117]	TNSM (J)	2017	0	7
[FS51]	Mashup service clustering based on an integration of service content and network via exploiting a two-level topic model [118]	ICWS (C)	2016	6	7
[FS52]	Domain-aware Mashup service clustering based on LDA topic model from multiple data sources [119]	IST(J)	2017	0	6.5
[FS53]	Leveraging Track Relationships for Web Service Recommendation [120]	ICEBE (C)	2016	1	6
[FS54]	A Web Services Discovery Approach Based on Mining Underlying Interface Semantics [121]	TKDE (J)	2017	2	7
[FS55]	Improving Web Service Clustering through a Novel Ontology Generation Method by Domain Specificity [122]	ICWS (C)	2017	0	7
[FS56]	WE-LDA: A Word Embeddings Augmented LDA Model for Web Services Clustering [123]	ICWS (C)	2017	0	7.5
[FS57]	Correlation-Aware Multi-Label Active Learning for Web Service Tag Recommendation [124]	ICWS (C)	2017	0	6.5

Publication Venues and Ranking

Types of publication venues with the number of selected papers are shown in Figure 4-7, showing the number of papers per venue with publication numbers equal to or higher than two. The venues, as well as their ranks and impact factors, include a list of 15 conferences and 23 journals as outlined in Appendix B-2. The ICWS and SCC conferences are A-ranked leading conferences with 14 and 4 papers, respectively. The conference ranking is based on the Computing Research and Education Association of Australasia (CORE) [125]. CORE 2014 assigned conference categories of A*, A, B, C, and Unranked. A* refers to a flagship conference, A refers to an excellent conference, B refers to a good conference, C refers to a conference meeting minimum standards, and Unranked refers to a conference without a ranking decision [125]. We selected two articles from KAIS and ISF journals, which have good impact factors. The journal rankings rely on the impact factor reported by Thomson Reuters Journal Citation Reports [126].

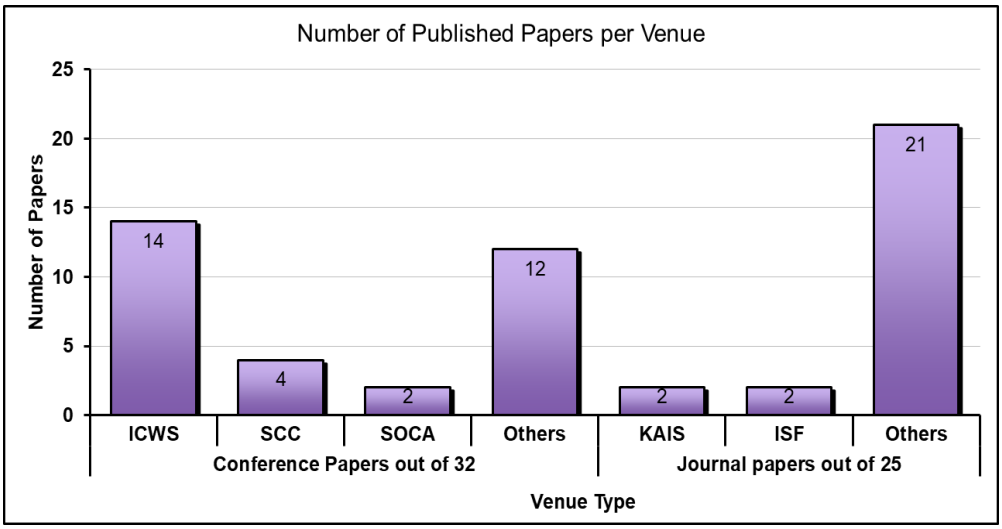


Figure 4-7: Number of Published Papers per Venues

Distribution of Papers over Years

The selected papers that met our criteria were published in the last decade. The papers show a growing interest in this topic within the last three years. This study's search ranges from 2006 to January 2018 based on the search protocol. Figure 4-8, which illustrates the papers' distribution, shows that a significant number of papers were published in 2014 and 2015.

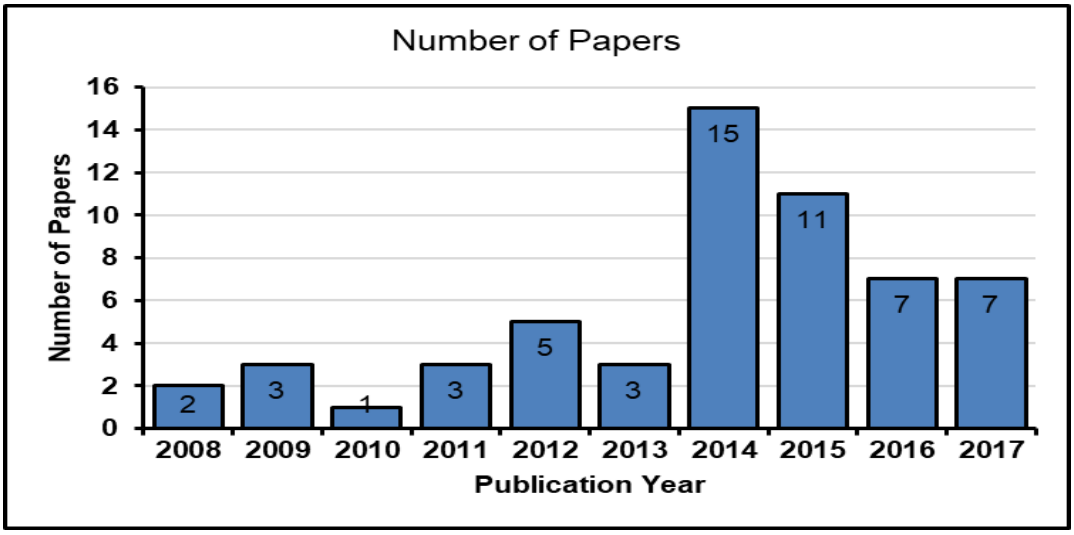


Figure 4-8: Distribution of Papers over Years

Distribution of Papers by Countries

Table 4-8 outlines the distribution of papers by country. It shows that while there is a general growth in the academic literature in the area of Web service discovery and recommendation, specific research groups and authors play a major role in this growth. China has the largest number of contributions with 27 selected papers (47% of the overall selected papers). Distribution is based on the first author's country; it does not formulate a theory on the geographical allocation of teams working on Web service discovery and recommendation at the time of the review. However, it shows a growing interest in the conducted study research area from different countries and teams

Table 4-8: Distribution of Papers by Countries

Country	Number of Papers	Percentage
USA	5	9%
UK	1	2%
Tunisia	1	2%
Sri Lanka	2	4%
Spain	1	2%
Romania	1	2%
Japan	5	9%
Italy	1	2%
India	2	4%
France	2	4%
China	27	47%
Canada	3	5%
Brazil	1	2%
Austria	1	2%
Australia	3	5%
Argentina	1	2%
Total	**57**	**100%**

4.5.2 Analyzing the Content of the Final Selected Papers

In this section, we present the results of the analysis based on the contents of the final selected papers and present synthesized data to answer the research questions. We synthesized to demonstrate the classification of techniques employed for Web service discovery and recommendation according to the final selected papers. This was followed by a summarization of the methods. Then, synthesized data on the algorithms, similarity measures, and evaluation metrics were discussed. Finally, we discussed the classification of datasets

utilized in Web service discovery and recommendation approaches, as well as future trends, directions, and research gaps.

RQ 1: Classification of Web services discovery and recommendation techniques

After careful analysis of the final selected papers' contents, the papers were classified into five groups given their utilized techniques to support the process of Web discovery and recommendation. Table 4-9 outlines the techniques employed in the final selected papers to facilitate the process of discovery and recommendation. Figure 4.9 illustrates the classification of papers according to the techniques.

- **Clustering-Discovery:** This technique refers to papers where Web services discovery was indicated as a research problem and where clustering techniques are employed on Web services datasets to facilitate the process of discovery;
- **Clustering-Recommendation:** This technique refers to papers where Web services recommendation was indicated as a research problem and where clustering techniques are employed on specific Web services datasets to facilitate the process of Web services recommendation;
- **Association-Discovery:** This technique refers to papers where Web services discovery was indicated as a research problem and where association rules techniques are employed on specialized Web services datasets to facilitate the process of Web services discovery;
- **Association-Recommendation:** This technique refers to papers where Web services recommendation was identified as a research problem and where association rules techniques were studied to facilitate the process of Web services recommendation; and
- **Combined Technique:** This technique refers to papers where both Web services recommendation and discovery identified a research problem and where clustering or association rules (or both) were studied to facilitate the process of Web services discovery and recommendation.

Table 4-9: Classification of Paper over Techniques

Number of Papers	Techniques	
30	Clustering-Discovery	53%
16	Clustering-Recommendation	28%
2	Association-Discovery	4%
3	Association-Recommendation	5%
6	Combined	11%
57	**Total**	**100%**

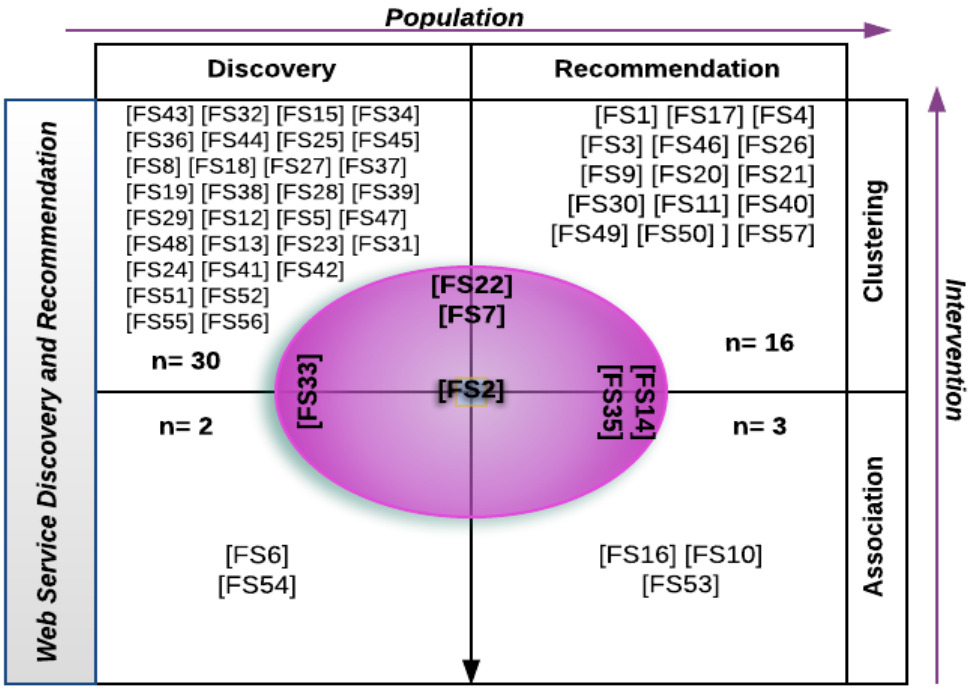


Figure 4-9: Classification of Papers According to the DM Techniques Employed

RQ 1: Methods for Web services discovery and recommendation

The methods proposed by researchers to facilitate the process of Web services discovery and recommendation can follow our techniques classification, as illustrated in Figure 4-9. Appendix B-3 outlines the proposed methods and their specifications and details. 53% of the selected papers employed clustering to minimize the search spaces of Web services by clustering similar Web services considering different similarity measures and clustering algorithms. 28% of the selected papers employed clustering to support the process of Web service recommendation by clustering similar service requesters or similar Web services to facilitate the process of selection and recommendation. Two selected papers employed association rules to support the process of Web service discovery. Three papers employed

association rules to support the process of Web service recommendation. Finally, 11% of the selected papers employed clustering or association rules to support the combined process of Web service discovery and recommendation

I. Methods based on clustering for Web services discovery

After arguing that registries-based Web services discovery was inefficient for good discovery, [FS43] proposed a method to utilize Web services' semantic representations to cluster similar Web services. They proposed a functional-based Web services clustering approach employing semantic measures, including service description similarity measurement based on WordNet, service name similarity analysis based on Levenshtein distance (LD) and WordNet, and input/output parameters based on domain ontology similarity considering the distance and the depth of positions in the concept taxonomy. A modified k-medoids clustering algorithm was utilized to cluster Web services described with OWL-S. The result demonstrated that the proposed method saves time by clustering compared to fully scanning and searching all Web services.

Motivated by the drawback of other similarity measures, [FS15] proposed context-aware similarity (CAS) to improve semantic similarity considering terms with up-to-date knowledge and fine-grained information. CAS was based on a post-filtering method to increase Web services clustering performance with Support Vector Machine (SVM) and domain context datasets from search engines. The SVM was trained to classify domains based on terms extracted from Google and Wikipedia. The hierarchical agglomerative algorithm (HAC) was employed as a clustering algorithm, and NLP was employed in a preprocessing step to extract service features.

Considering the hidden semantic pattern in complex terms, [FS34] proposed a method to increase the performance of Web services clustering with the ontology-learning method. Three WSDL features were extracted, and an ontology was generated for each feature. Similarities were captured based on domain ontology learning (logic-based reasoning) and edge count based on WordNet measures. Features were integrated, and clustering was performed based on the hierarchical agglomerative clustering algorithm.

To increase clustering performance by adding extra information from tags, [FS36] proposed crawler-based systems to gather Web services descriptions and cluster them

based on functionality. The proposed solution built a Web services repository and preprocessed WSDL for feature extraction. They also aimed to discover noun phrases from vectors for tagging Web services. Hierarchical agglomerative clustering was employed for clustering and WordNet for similarity.

[FS44] proposed an approach for Web services discovery by taking advantage of the service description to extract the concepts and semantic relations with WordNet and ConceptNet. They extracted concepts from the service description and clustered them by employing the K-means clustering algorithm. The method extracted concepts from the users' queries and guided service requesters to formulate queries to return services associated with the concepts.

To address the lack of semantics and scalability in Web services discovery, [FS25] utilized modified K-means for functional clustering and the cosine angle for the similarity between documents. The method includes two main phases to decompose service collection and match services semantically. An extensive service collection is partitioned into a set of smaller clusters. The Singular Value Decomposition (SVD) technique is applied to the cluster so that service matching against the query can be carried out at the concept level. SVD is employed to extract the semantics between the user query and the service description.

[FS45] proposed a discovery as a service (DaaS), a cloud-based context-aware service-discovery framework for mobile environments considering network characteristics, user preferences and context, and device profile. Web services were clustered into similar functional groups based on service descriptions in the Web services clustering component. The service matching did not need to match the user request against all service offerings. It employed data mining and text analytics to cluster Web services and the Normalized Google Distance (NGD) as a featureless distance measure between words. In [FS24], they provided a method to extract WSDL documents features based on five features and cluster Web services based on the similarity of the extracted features. The quality threshold clustering algorithm was employed to cluster Web services based on integrated similarities with weights.

To overcome the time and computational complexity of logic-based semantic matching by extracting semantic, [FS8] utilized HAC and NLP for non-logic-based Web

services discovery, including matching and ranking. Correlated Topic Model (CTM) is employed to extract topics from semantic service descriptions and model the correlations between the extracted topics. Based on the topic correlation, service descriptions are grouped into hierarchical clusters, and Formal Concept Analysis (FCA) formalism is employed to organize the constructed hierarchical clusters into concept lattices according to their topics.

To overcome the limitations of service repositories, especially with the growing size of their services, [FS18] utilized HAC to cluster Web services in the repository based on their functions considering distance based on semantic subsumption relations to organize services registered in the repositories into clusters for the purpose of improving the response time, recall and precision.

To improve Web services clustering by considering different similarities, [FS27] [FS37] utilized HAC for functional clustering, TFIDF, WordNet, and Ontology for term similarity measures in a multiphase Web services clustering method including a feature extraction phase employing NLP, ontology-learning phase, similarity calculation phase, feature-integration phase, and clustering phase.

[FS19] proposed the WCCluster method to bootstrap the process of Web services discovery by clustering the WSDL document and its words. They employed the co-clustering algorithm based on K-means and dealt with the discovery problem as a bipartite graph-partitioning problem.

To improve Web services clustering's efficiency, [FS38] proposed the relational database self-join operation (RDBJO) method to cluster Web services with ontology. Ontology technology was employed to compute the semantic level; the self-join operation in the RDB was utilized to calculate the service interface and capability tables.

To enhance search engine results with a list of similar services, [FS28] proposed an approach to creating clusters of related Web services to allow Web services consumers to find and relate specific services to a given query. The clusters were built specifically for the results of a search query. This limited the number of elements to a reasonable size and improved the result’s visibility. They used statistical cluster analysis and agglomerative hierarchal clustering. A prototype was built to support their solution.

To improve REST service discovery and explore the discovery of different Web services types, [FS39] proposed a method based Cluster Enhanced REST Service Registry (CE-RSR), which provided an IR-based discovery approach for REST services described via WADL. The method employed NLP preprocessing steps on WADL files, K-means clustering, and the well-known Apache Lucene framework for indexing.

To minimize discovery time and maximize the use of Web services for business process integration within organizations, [FS29] proposed a method to create semantically similar domains of Web services by applying partitioning around medoid (PAM) and hierarchical agglomerative clustering algorithms to cluster Web services described with OWL-S semantically.

To enhance the performance of Web services clustering and address the drawbacks of limited service data being available for clustering, [FS32] [FS12] utilized K-means clustering and NLP for functional-based Web services clustering after incorporating auxiliary long texts from Wikipedia with corresponding tags to learn from the set of data and improve clustering. Tag-aided dual author topical model (TD-ATM) on short text from PW was employed in [FS32], and [FS12] used a dual tag-aided latent dirichlet allocation (DT-LDA) method based on transferring learning from auxiliary long text data from Wikipedia to enrich the service data obtained from PW.

To minimize the discovery time among large service pools, [FS5] proposed a semantic Web services discovery method combining both functional and process similarity based on semantic similarity measures. Preclustering minimized the search space with the density-based clustering algorithm (DBSCAN). Different similarity measures were introduced to measure both Web services' functional similarity and process similarity.

[FS47] proposed a nonparametric Bayesian model-based latent functional factors (BN-LFF) method for large-scale Web services clustering to learn the number of LFF in a service space. This provided a way to represent Web services in the LFF space as well as a learning method to identify LFF and clustering with K-means. The method provided an efficient way to cluster Web services by adding a small amount of accurate information in the LFF space to improve clustering for the purpose of discovery.

[FS48] presented a method to improve semantic Web services discovery's accuracy and efficiency by combining semantic analysis and hierarchal clustering. Based on WSDL,

four features were extracted: 1) name; 2) description; 3) input; and 4) output. TFIDF and Cosine were employed for similarity measures, and the agglomerate hierarchy algorithm was utilized for clustering.

To overcome the drawback of keyword-based discovery systems, and motivated by the fact that most Web services are in non-semantic form, [FS13] proposed a functional clustering facilitated Web services discovery model (CFWSFinder) with WordNet and LSI to represent non-semantic Web services as a service feature vector. A modified kernel batch self-organizing map (KBSOM) neural network was employed for clustering; TFID-inverse category frequency (ICF), Cosine similarity, and large-margin metric learning (LMML) methods were utilized to measure similarity and matching. The proposed method included service representation, service clustering, and service matching.

Motivated by the fact that searching for the best cluster can be viewed as an optimization problem since the number of services involved in a clustering process is large, [FS23] utilized Particle Swarm Optimization (PSO) to functionally cluster Web services where they define the semantic similarity metrics between Web services based on the degree of match (DoM) based on ontology

To overcome the limitations of semantic similarity distance measures and their thresholds, [FS31] employed a technique based on a self-organizing clustering algorithm called taxonomic clustering for functionally organizing semantic Web services advertisements. A Semantic Genome Propagation Scheme (SGPS)-based ontology was utilized to measure the similarity between semantic concepts to semantically position Web services in the cluster space by searching the most specific parent (MSP) and the least specific children (LSC).

[FS41] proposed a method based on text mining to automatically gather, discover, extract, and integrate features from WSDL files and cluster them to form homogenous service communities. The authors extracted four features from a WSDL file with NLP techniques, then applied the tree-traversing ant (TTA) clustering algorithm [127] and NGD featureless similarity measurement and a featureless distance measure based on Wikipedia data (n° W).

By considering user queries to find Web services based on keyword-based and semantic extraction, [FS42] employed K-means clustering to eliminate and filter out irrelevant services based on user queries. Probabilistic Latent Semantic Analysis (PLSA) is utilized to capture semantics hidden behind the words in user queries and Web service descriptions. NLP, term frequency-inverse document frequency (TFIDF), Cosine, and Euclidean distance are employed for similarity measurements.

To accurately discover service requesters desired mashup services, [FS51] and [FS52] proposed a method for clustering mashup services based on exploring services document content and network. Data from PW is employed where a two-level topic model (LDA) is designed to mine the latent functional topics by incorporating relationships among mashup services and their content. The similarity between mashup services calculated considering different similarity measures where a combination of K-means and HAC are utilized to cluster similar mashup services.

To improve Web services clustering based on their functional properties, [FS55] proposed a method to cluster Web services based on ontology-generation of the extracted terms by focusing on the specific terms than general terms, where the specificity of terms are calculated before generating ontology. Ontology relationship and IR-based similarities are employed to find the similarities between Web services and cluster them based on HAC. This work utilized [FS34] approach and add their method of ontology-generation based on specific terms.

To improve semantic Web services discovery's efficiency and accuracy, [FS48] employed HAC for functional clustering, TFIDF, Cosine, and WordNet for similarity measures. Four WSDL features are extracted with NLP pre-processing steps. A topic-based clustering method proposed by [FS56] uses an augmented LDA model to improve Web services clustering based on functional properties. Motivated by the lack of text information, they train the latent topic information on Web services descriptions based on a trained model, where Web services with the same topic are clustered together. The training process is done based on clustering similar words from Web services documents into similar clusters.

II. Methods based on clustering for Web services recommendation

Most existing approaches ignore the data credibility problem. Therefore, they are vulnerable to the unreliable QoS data contributed by malicious users. [FS1] proposed a trust-aware approach (TAP) for reliable, personalized QoS prediction. The historical QoS information was collected from similar users or services to predict current users' unknown QoS. It focused on accurately predicting unknown QoS values and recommending optimal services for active users by Collaborative filtering (CF) approaches. It combined user clustering and service clustering to predict QoS and recommend a service. User reputation was calculated based on UserClustering and ServiceClustering. It identified similar users, trustworthy users, and similar services. Then, missing values were predicted by combining related services and similar trustworthy users. For clustering, a K-means algorithm was employed; Pearson correlation coefficient (PCC) was used for similarity. To overcome the challenge of data sparsity in QoS prediction based on Collaborative filtering (CF), [FS17] employed HAC to cluster services based on their physical environment, measured the similarity of users with PCC based on the clusters, and then applied CF.

To increase QoS prediction performance and provide a more personalized recommendation, [FS4] employed HAC to cluster users and services based on their locations and QoS data. The system works to predict Web services QoS values and recommend the best one for active users based on historical Web services QoS records with QoS-aware collaborative filtering. PCC is utilized as a similarity measure between users based on the QoS values of Web services they both invoked.

To handle the problem of limited tags attached to Web services, [FS3] proposed a hybrid Web Service Tag Relevance Measurement mechanism (WS-TRM) to measure the relevance of a tag to a Web service considering the semantic similarity and tag authority. WSDL features extracted with the tag are calculated with NGD and then integrated with Tag-authority-based Hyperlink-Induced Topic Search (HITS) similarity. WS-TRM is applied in tag recommendation for Web services.

Based on the use of heterogeneous features to improve recommendation, [FS46] employed K-means to cluster services in a Web Service Heterogonous Feature Selection (WS-HFS) method on heterogeneous data sources of Web services to perform Web services mining. The framework has two components: feature selection and service mining.

Features are selected and transformed into knowledge in the feature subspace, and a data source-based weight learning method is proposed.

Service requesters tend to invoke a few services, leading to many missing QoS values; that is, the user-service matrix might be sparse. [FS9] and [FS26] proposed a location-based hierarchical matrix factorization (HFM) method to predict missing QoS values. User-service global context and geographic information were utilized in clustering to build an HFM model-based collaborative filtering method to predict missing values.

[FS20] proposed a CF method to study user behavior from a service usage history perspective. Then, patterns were found among Web services in the repository to predict user preference on certain Web services. To solve the common sparsity problem, they utilized customer profiles and historical usage experience of service invocations. They mined service usage patterns to reflect potential dependencies among services that were invoked together to accomplish complex requirements. They employed K-means to cluster customers based on the assumption that customers had similar service usage history if their personal attributes were similar.

To prevent time-consuming and inaccurate recommendation for semantic Web services for a composite, [FS21] proposed a method of clustering and recommendation for OWL-S Web services in evolution (CRE). This was based on Web services similarities measurement and recommendation. TFIDF and ontology (WordNet) were employed for clustering; matrix factorization was used for recommendation.

Considering the historical QoS experiences of similar users to predict a user's QoS on a known Web service, [FS30] employed K-means clustering to cluster users and services in a global structure considering Relational Clustering-based Model (RCM) and a collaborative filtering-based scheme to evaluate the QoS of a priori unknown service providers. CF is utilized to predict the QoS of unknown Web services. PCC, WordNet, and Cosine are employed to measure similarities.

[FS11] proposed a clustering collaborative filtering (CluCF) method by employing users and service clusters considering time-aware similarity measures and a location factor to update clusters. Clustering added scalability in service recommendations based on considering small and similar clusters rather than a full dataset. It used QoS performance and

invocation time to improve prediction accuracy. PCC was considered to calculate similarities between service requesters and services; the K-means clustering algorithm was utilized to build clusters.

To enhance the QoS prediction accuracy via clustering, [FS40] proposed a landmark-based QoS prediction based on user-based clustering (UBC) and Web services-based clustering (WSBC) prediction algorithms for Web services. Considering real QoS data from a set of fixed landmarks from PlanetLab, it employed hierarchical clustering and PCC as a similarity between two users.

To increase the performance of Web services selection and recommendation, [FS49] proposed clustering Web services based on their functionalities considering Hybrid Term Similarity (HTS) [13] based semantic similarities. Then, cluster them based on their manually assigned QoS values with Spherical Associated Keyword Space (SASKS) algorithms.

By not only considering QoS metrics for Web services recommendation, [FS50] proposed a method to include Web services functional properties extracted from WSDL files in addition to QoS metric for Web service recommendation. Fuzzy C-means clustering algorithm is utilized to cluster user and services where the matrix factorization approach integrates the functional categories of the Web service.

[FS57] proposed an active learning approach for Web services tag recommendation where Support Vector Machine (SVM) is trained on labeled Web services to classify unlabelled Web services based on a previously labeled small set of Web services by a domain expert. By learning the correlations among tags by computing Jaccard and hierarchical clustering based tag correlations, they minimize the efforts of the domain expert with an active learning-binary classification on a service for each possible label to determine if the tag should be recommended to the service

III. Methods based on association rules for Web services discovery

To infer patterns from service descriptions by providing a summarized and integrated representation of Web services functionality, [FS6] proposed Web services DiscoveRy via intEnsionAl knowledge Mining (W-DREAM) method to extract a useful intensional representation of Web services repository contents. This will help application developers to learn new knowledge about the available service and make the right choice. Association

rules were used to extract useful knowledge from a set of datasets based on WSDL files and service invocations. W-DREAM provides an infrastructure to perform intentional service representation and querying to select Web services that best suit application developer needs.

Considering Web services interfaces when their parameters contain meaningful synonyms, abbreviations, and semantic can improve the discovery process, [FS54] propose an approach that mines the underlying semantics and conduct semantic extension of Web services interfaces. They mine the underlying semantics to create index libraries by employing association rules and then clustering interaction interface names and fragments under the supervision of co-occurrence probability.

IV. Methods based on association rules for Web services recommendation

To decide which service among the retrieved set of semantically equivalent Web service candidates is the best, [FS16] proposed a method for Web services recommendation considering collaborative filtering and association rules (Apriori algorithm). The authors studied user invocation behavior from the Web services composition perspective. The method employed association rules among Web services in the repository to predict user preference on certain Web services. It considered user usage patterns and generated a profile to provide a personalized ranking for the retrieved set of Web services.

To improve Web services' ranking and recommendation, [FS10] proposed a method considering consumer usage history to develop personal profiling. Association rules mining was utilized to understand the correlation between service requesters and construct group and individual profiling. A group knowledge-based recommendation system employed collaborative filtering for ranking and recommendation.

By exploring service requesters relations, their history of consumed services, and the quality of services, [FS53] proposed a Web service relationships track (WSR-Track) approach where service ecosystems are represented in heterogeneous multigraph. Nodes represent service requesters and services, and Edges represent trust/usage relationships between service requesters and services. By incorporating history and QoS data and users, services data association rules are generated, and the Web services recommendation process provides adequate Web service for service requesters based on tracked relation.

V. Methods combining clustering and association rules for Web services discovery and recommendation

The methods described below employed clustering for Web services discovery and recommendation. The method proposed by [FS7] [FS22] supported both matching (discovery) and selection (recommendation).

Since clustering can boost the power of Web services search engines and generate tags to improve search accuracy of tag-based service recommendation, [FS7] proposed a method for automatic tagging for Web services considering machine learning. NLP extracted service features; services were clustered with carrot search clustering and the K-means clustering algorithm. The method extracted relevant tags from a WSDL file and utilized co-occurrence to generate tags from a collection of untagged service-description files. Tags provided additional information. This information was employed in clustering Web services in recommending systems.

Given the fact that the availability of QoS is important in Web services recommendation and to predict the QoS of non-invoked Web services, [FS22] proposed a hybrid approach considering content-based and collaborative filtering to predict QoS values of non-invoked Web services and to recommend top-k Web services to users. It used K-means to cluster similar Web services with semantic (latent semantic indexing) and non-semantic (keyword) approaches. Slope one collaborative filtering was utilized to predict QoS for similar Web services for similar users. Web services were ranked based on QoS value, and top-k ranked Web services were recommended to active users.

The methods, which combined clustering and association for Web services recommendation include [FS35] [FS14]:

[FS35] proposed a method to recommend Web services for invoking services with a proposed cluster-based service recommendation approach. Hybrid term similarity (HTS) was presented in [FS34], and association rules mining was also employed. Two factors used for clustering included the hierarchical agglomerative clustering algorithm and the association factor based on association rules.

To reduce the dimensionality of a sparse matrix and solve the cold-start problem in Web service recommendation, [FS14] proposed the user relationship and preferences clustering and recommendation (URPC-Rec) method. This combined user history behaviors and personal interests to make personal service recommendation for new users. The method

combined a clustering algorithm and a recommendation algorithm to find an association between service requesters and services. User interest tags, social network relationships, and information were used. WordNet was utilized to measure the similarity in the recommendation process.

For method combined clustering and association rules for discovery: [FS33] proposed an integrated method for semantic Web services discovery. The proposed method relied on semantic Web services categorization and semantic Web services selection. It utilized WSDL descriptions and added semantics based on WordNet and Suggested Upper Merged Ontology (SUMO) mapping. It used a hierarchical agglomerative clustering algorithm to categorize Web services. The selection process relied on latent semantic indexing (LSI) and discovered the association between permeants in a cluster.

For method combined both clustering and association rules for Web service discovery and recommendation: [FS2] proposed Web services clustering and tagging to improve Web services discovery. It focused on clustering non-semantic Web services because these services are more popular and widely supported by the industry circle. They employed NLP to extract the features from WSDL documents and added tags as another feature. The K-means clustering algorithm was utilized to cluster Web services. A Web services tag recommendation strategy (WSTRec) was proposed to overcome the limitation of noise and uneven distribution of tags. Association rules were used to extract associations with training and testing datasets.

RQ 1: Algorithms for Web services discovery and recommendation

The clustering and association rules algorithms employed by the researchers to facilitate the process of Web services discovery and recommendation are outlined in Table 4-10. K-means is the most-cited clustering algorithm (42% of the final selected papers). This was followed by the HAC algorithm (36%). K-means clustering was studied with modifications in [FS32], [FS7], [FS44], [FS25], [FS1], [FS3], [FS46], [FS26], [FS9], [FS19], [FS20], [FS39], [FS2], [FS30], [FS11], [FS12], [FS47], [FS22] and [FS42]. Hierarchical agglomerative clustering was cited with modifications and adjustments in [FS15], [FS33], [FS34], [FS35], [FS36], [FS8], [FS17], [FS4], [FS18], [FS27], [FS37], [FS28], [FS29], [FS21],

[FS48] and [FS40]. Bio-inspired clustering algorithms were employed to cluster Web services in [FS13] with a neural network, [FS23] utilized particle swarm optimization, and [FS41] used the tree-traversing ant algorithm. The quality threshold clustering algorithm is another partition-based clustering algorithm employed in [FS45] and [FS24]. Density-based clustering (DBSCAN) was cited in [FS5]. The self-organizing-based clustering algorithm was employed in [FS31]. A relational database (RDB), which is a self-join clustering algorithm, was employed in [FS38]. Among the final selected papers, clustering algorithms dominated and were more popular when comparing the association rules algorithms. For association rules algorithms, the well-known Apriori algorithm (43%) was employed in [FS16], [FS10], and [FS6] to find the association between different variables to meet the research objectives

Table 4-10: Clustering and Association Rules Algorithms

Clustering Algorithms		
K-Medoids Clustering	2	[FS43] [FS29]
K-Means Clustering	21	[FS32] [FS7] [FS44] [FS25] [FS1] [FS3] [FS46] [FS26] [FS9] [FS19] [FS20] [FS39] [FS2] [FS30] [FS11] [FS12] [FS47] [FS22] [FS42] [FS32] [FS7] [FS44] [FS25] [FS1] [FS17] [FS3] [FS46] [FS26] [FS9] [FS19] [FS20] [FS39] [FS2] [FS30] [FS11] [FS12] [FS47] [FS22] [FS42] [FS51] [FS56]
Hierarchical Agglomerative Clustering Algorithm (HAC)	20	[FS15] [FS33] [FS34] [FS35] [FS36] [FS8] [FS17] [FS4] [FS18] [FS27] [FS37] [FS28] [FS29] [FS21] [FS48] [FS40] [FS51] [FS54] [FS55] [FS57]
Density-Based Spatial Clustering of Applications with Noise (DBSCAN)	1	[FS5]
Quality Threshold (QT) Clustering Algorithm	2	[FS45] [FS24]
Neural Network Clustering	1	[FS13]
PSO Clustering Algorithm	1	[FS23]
Self-Organizing Based Clustering Algorithm	1	[FS31]
Self-Join Clustering Algorithm	1	[FS38]
Tree-Traversing Ant Algorithm	1	[FS41]
Fuzzy C-Means	2	[FS50] [FS52]

Association Rules Algorithms		
Hyperclique Patterns	1	[FS33]
Apriori Algorithm	4	[FS16] [FS10] [FS6] [FS53]
GARC: Gain Based Association Rule	1	[FS35]
Defined Correlation	2	[FS2] [FS14]

RQ 1: Similarity measures for Web services discovery and recommendation

Similarity measures cited in the final selected papers for Web services discovery and recommendation are listed in Table 4-11. Similarity measures can be classified into:

- ***Semantic-based similarity measurements:*** These measurements include WordNet, NGD, Wikipedia, and ontology. WordNet [65] is a lexical database that groups English words into sets of synonyms and records various semantic relations between these synonym sets. Eighteen percent of the final selected papers utilized WordNet, including [FS43] [FS15] [FS34] [FS35] [FS36] [FS44] [FS27] [FS21] [FS30] [FS48] [FS18] [FS37] and [FS14], as a similarity measure based on the structured knowledge base where the edge-based method was mostly cited in these papers. Normalized Google Distance [128] is computed by the Google Web search engine to measure semantic similarity and relatedness between words. NGD was cited in [FS45] [FS3] [FS2] [FS24] and [FS41]. Wikipedia was employed as a semantic similarity measure in [FS41]. Ontology logic-based reasoning was utilized in matchmaking and semantic similarity measurement for Web services in [FS43] [FS15] [FS34] [FS35] [FS27] [FS38] [FS29] [FS18] [FS31] [FS5] and [FS23].
- ***Syntactic-based similarity measurements:*** These measurements include IR and NLP techniques to measure similarities in Web services discovery and recommendation. Term frequency-inverse document frequency (TFIDF) is a text mining method employed mostly for feature extraction. It measures the importance of words in a document. TFIDF was cited in [FS43] [FS15] [FS35] [FS27] [FS37] [FS19] [FS39] [FS2] [FS21] [FS48] [FS13] [FS22] [FS25] and [FS42]. This is approximately 20% of the final selected papers. Cosine similarity was utilized to measure the similarity between two documents on the vector space by calculating the cosine of their angle. Cosine similarity was cited in [FS7] [FS33] [FS25] [FS20] [FS30] [FS48] [FS28] [FS13] [FS21] and [FS42].

- ***Distance-based similarity measurements:*** These measurements were employed to measure the similarity between Web services, including Levenshtein distance (LD), Euclidean distance, Mahalanobis distance, and Jaccard coefficient. LD is utilized to measure the similarity between two strings. The distance is the number of deletions, insertions, or substitutions required to transform one string into another [129] as in [FS43]. The Euclidean distance was employed to measure the distance between two vectors in [FS20] [FS42] and [FS28]. Mahalanobis distance was cited in [FS13], and Jaccard coefficient was cited in [FS2]. Pearson correlation coefficient was utilized mostly to find the correlation and similarity for Web services recommendation. PCC measures the linear correlations between two variables. It was cited in [FS16] [FS1] [FS17] [FS4] [FS20] [FS10] [FS30] [FS11] [FS22] and [FS40].
- ***Hybrid-based similarity measurements:*** These measurements are employed by combining semantic similarity measurements, syntactic-based similarity measurements, or distance-based similarity measurements. For example, [FS43] combined WordNet, NGD, and ontology as semantic similarity measures, TFIDF as an IR-based syntactic similarity measurement, and LD as a distance-based similarity measure. Hybrid-based similarity measures are the most studied measurements as they usually provide a better solution by integrating different similarities.

Table 4-11:Similarity Measures

Similarity Measures	#used	Papers
WordNet (path, res, wup, etc)	15	[FS43] [FS15] [FS34] [FS35] [FS36] [FS44] [FS27] [FS21] [FS30] [FS48] [FS14] [FS18] [FS55] [FS49]
Cosine Similarity	10	[FS7] [FS33] [FS25] [FS20] [FS30] [FS48] [FS13] [FS42] [FS28] [FS21]
Pearson Correlation Coefficient(PCC)	11	[FS16] [FS1] [FS17] [FS4] [FS20] [FS10] [FS30] [FS11] [FS22] [FS40] [FS50]
Normalized Google Distance	7	[FS45] [FS3] [FS2] [FS24] [FS41] [FS50] [FS55]
TFIDF-based (dot product)	16	[FS43] [FS15] [FS35] [FS27] [FS37] [FS19] [FS39] [FS2] [FS21] [FS48] [FS22] [FS42] [FS13] [FS18] [FS25] [FS49] [FS50] [FS55]
Ontology-based	15	[FS43][FS15] [FS34] [FS35] [FS27] [FS38] [FS29] [FS5] [FS23] [FS18] [FS31] [FS18] [FS8] [FS49] [FS55]
Levenshtein Distance (LD)	1	[FS43]
n ° of Wikipedia	1	[FS41]

Euclidean Distance	4	[FS20] [FS28] [FS49] [FS56]
Jaccard Coefficient	3	[FS2] [FS51] [FS56]
Mahalanobis Distance	1	[FS13]

Based on the algorithms and the similarity measures employed in the final selected papers, Appendix B.4 presents the most cited algorithms, linking them to the similarity measures. Furthermore, it shows gaps where the proposed methods did not investigate. For example, it indicates that a combination of HAC with the semantic similarity measure WordNet is one of the most cited in the proposed methods. It also indicates that HAC has not been employed with the NGD semantic similarity measure.

RQ 1: Evaluation metrics utilized for Web services discovery and recommendation

The evaluation metrics utilized in the final selected papers to validate and evaluate the proposed Web services discovery and recommendation approaches are outlined in Table 4-12. Knowing the evaluation metrics employed to evaluate the proposed methods helped researchers compare their evaluation results to other evaluation metrics. Evaluation metrics had individual measurement purposes depending on their usage context. Recall, a measure of correctness, completeness, and quantity, measured the fraction of relevant services matched or retrieved and was cited in [FS43] [FS15] [FS7] [FS34] [FS36] [FS44] [FS25] [FS45] [FS3] [FS46] [FS18] [FS27] [FS37] [FS19] [FS38] [FS20] [FS39] [FS29] [FS2] [FS5] [FS48] [FS22] [FS31] [FS24] [FS6] and [FS42]. Precision, a measure of exactness and quality, measured the fraction of retrieved or matched services that are relevant and correct and was cited in [FS15] [FS7] [FS34] [FS35] [FS36] [FS44] [FS25] [FS45] [FS3] [FS46] [FS18] [FS27] [FS37] [FS19] [FS20] [FS39] [FS10] [FS29] [FS2] [FS5] [FS48] [FS22] [FS31] [FS24] [FS6] and [FS42]. Entropy as a measure of diversity was cited in [FS32] [FS27] [FS21] and [FS12] to measure how various semantic classes were distributed within each cluster. F-measure as a measure considering both precision and recall to compute the score was cited in [FS32] [FS15] [FS33] [FS34] [FS8] [FS27] [FS37] [FS20] [FS39] [FS29] [FS5] [FS22] and [FS14] where higher F-measure values indicated the higher quality of the clusters formed. Precision@N , a precision measurement, identifies the fraction of top-N services that are relevant; it was cited in [FS16] [FS44] [FS8] [FS10] [FS2] and [FS14]. Likewise, Recall@N measures the fraction of relevant services

in the top-N. It measures recall by considering the top-N relevant services and was cited in [FS14] to measure the accuracy of top-N recommendation predictions of the proposed solution.

Mean Absolute Error (MAE) and Normalized Mean Absolute Error (NMAE) are widely used evaluation metrics to measure prediction accuracy. In the selected papers, they were always employed in the context of Web services recommendation. MAE is an effectiveness evaluation metric indicating the average absolute deviation of the predicted value and the actual value. It was cited in [FS1] [FS4] [FS26] [FS9] [FS21] [FS30] [FS11] [FS22] and [FS40] to measure prediction quality in recommendation systems. Correspondingly, the NMAE metric was cited in [FS1] [FS17] [FS9] [FS11] and [FS40] to measure the prediction accuracy where smaller values indicate higher prediction quality. Purity is a transparent evaluation metric utilized in [FS32] [FS27] [FS21] [FS12] and [FS46] to determine the quality of the resulting clusters. Time refers to the amount of time it takes to perform the proposed solution's task. For example, in the context of clustering Web services to enhance the discovery process, time refers to the time consumed to cluster the specific number of Web services. Time as an evaluation metric was employed in [FS43] [FS18] [FS38] and [FS48]. Accuracy as a prediction-correctness measure was employed in [FS7] [FS25] [FS38] [FS47] [FS13] and [FS42]. In the clustering context, accuracy is employed to measure efficiency by discovering the one-to-one relationship between clustering and the original category. Silhouette value is utilized in clustering to measure how similar a service is to the other services in its own cluster compared to services in other clusters and was cited in [FS29] and [FS47].

In [FS13], neural networks and self-organizing map (SOM) were employed for Web services discovery and quantization error (QE), topographic error (TE), and neuron utilization (NU) as evaluation metrics for their proposed solution. QE is used to measure the average distance between each input vector and its winner. TE describes how the SOM preserves the topology of the studied dataset. NU measures the percentage of neurons that are the winner of one or more input vectors in the map. The Dunn index is a metric measure of cluster compactness and separation that was cited in [FS23] to have the services in the same cluster situated closer to each other than to services from the other clusters. Root mean square error (RMSE) was employed to measure the differences between the values

predicted by the proposed model, and the values observed to measure the error in [FS17] and [FS22]. Median relative error (MRE) was employed to get the median recognition of all the relative error (ER) values in [FS17] and [FS40]. [FS2] employed success at rank K (S@k) to evaluate Web services' tags by taking the percentage of good and perfect tags in the top-K recommended tags averaged over the judged Web services. For evaluating their method in clustering semantic Web services, [FS23] utilized intra-cluster variance (ICV), and average item-cluster similarity (AICS). The ICV metric indicated the level of similarity between the services in a cluster and the AICS metric was employed to verify that the services in the same cluster are semantically similar. Normalized discounted cumulative gain (NDCGn) was cited in [FS8] in IR to evaluate the ranking results, and usefulness [FS46] utilized normalized mutual information (NMI) to assess the clustering performance in the way of information entropy theory.

Table 4-12: Evaluation Metrics

Evaluation Metrics	#cited	Papers
Recall	31	[FS43] [FS15] [FS7] [FS34] [FS36] [FS44] [FS25] [FS45] [FS3] [FS46] [FS18] [FS27] [FS37] [FS19] [FS38] [FS20] [FS39] [FS2] [FS5] [FS48] [FS22] [FS31] [FS24] [FS6] [FS42] [FS49] [FS51] [FS53] [FS54] [FS55] [FS56]
Precision	32	[FS15] [FS7] [FS34] [FS35] [FS36] [FS44] [FS25] [FS45] [FS3] [FS46] [FS18] [FS27] [FS37] [FS19] [FS20] [FS39] [FS10] [FS29] [FS2] [FS5] [FS48] [FS22] [FS31] [FS24] [FS6] [FS42] [FS49] [FS51] [FS53] [FS54] [FS55] [FS56]
Entropy	7	[FS32] [FS27] [FS21] [FS12] [FS51] [FS55] [FS56]
F-Measure	16	[FS32] [FS15] [FS33] [FS34] [FS8] [FS27] [FS37] [FS20] [FS39] [FS29] [FS5] [FS22] [FS14] [FS49] [FS55] [FS57]
Precision@N	6	[FS16] [FS44] [FS8] [FS10] [FS2] [FS14]
Recall@N	1	[FS14]
Mean Absolute Error (MAE)	11	[FS1] [FS4] [FS26] [FS9] [FS21] [FS30] [FS11] [FS22] [FS40] [FS50] [FS52]
Normalized Mean Absolute Error (NMAE)	5	[FS1] [FS17] [FS9] [FS11] [FS40]
Purity	7	[FS32] [FS46] [FS27] [FS21] [FS12] [FS51] [FS56]
Time	4	[FS43] [FS18] [FS38] [FS48]
Accuracy	6	[FS7] [FS25] [FS38] [FS47] [FS13] [FS42]
Silhouette	2	[FS29] [FS47]
Quantisation Error (QE)	1	[FS13]
Dunn Index	1	[FS23]
Root Mean Square Error (RMSE)	4	[FS17] [FS22] [FS50] [FS52]
S@K	1	[FS2]

Topographic Error (TE)	1	[FS13]
Intra-Cluster Variance (ICV)	1	[FS23]
Normalised Discounted Cumulative Gain (NDCGn)	1	[FS8]
Median Relative Error (MRE)	2	[FS17] [FS40]
Neuron Utilisation (NU)	1	[FS13]
Average Item-Cluster Similarity (AICS)	1	[FS23]
Normalized Mutual Information (NMI)	1	[FS46]

RQ 2: Datasets employed for Web services discovery and recommendation

This section presents a synthesis of the datasets and published benchmarks utilized in the area of Web services discovery and recommendation. The final selected papers cited various datasets to validate and evaluate their proposed Web services discovery or recommendation solutions. These datasets vary based on the method of collection. Some researchers gathered and captured their datasets from famous Web services repositories. Other researchers tested their solution based on publicly published benchmarks. We categorized Web services datasets based on the method of the collection into self-gathered and published benchmarks. Self-gathered datasets refer to Web services datasets collected, captured, and crawled from different Web services repositories over the internet for self-use of validation. Some researchers share their collected Web services datasets. For self-gathering of Web services datasets, researchers utilized several websites to capture and crawl Web services. These included Seekda[7], PW[8], WebserviceX[9], WebServiceList[10], xMethods[11], Salcentral, and Titan Search Engine. However, some of the websites are no longer available, including Seekda, WebServiceList, Salcentral, and xMethods.

The existence of published benchmarks helps researchers validate their solutions without the need to collect Web services. In the final selected papers, Web services published benchmarks have been used more than self-gathered Web services. Table 4-13 outlines and details eight Web services published benchmarks. The current location of the

[7] http://www.seekda.com
[8] https://www.programmableweb.com
[9] http://www.webservicex.net
[10] http://www.webservicelist.com
[11] http://www.xmethods.com/

employed published benchmarks is also indicated in the table. [FS25] and [FS42] cited a WSDL-based published benchmark named WS2003 for discovery purposes. WS2003 [130] was used to annotate Web services based on machine learning with WSDL files gathered from Salcentral and XMethods. [FS8] and [FS23] cited a semantic-based published benchmark named SAWSDL-TC for Web services discovery. [FS3], [FS46], and [FS2] cited a WSDL-based dataset annotated with user tags named STag 1.0 for discovery purposes. [FS18], [FS37], [FS38], [FS29], [FS5], [FS47], [FS48], [FS31], and [FS6] cited a semantic-based published benchmark named OWLS-TC for Web services discovery. [FS25] cited an IR-based test collection called MED to evaluate the performance of the proposed solution. WS2003, SAWSDL-TC, STag 1.0, OWLS-TC, and MED are published benchmarks and test collections that have been employed in the final selected papers for discovery purposes. For Web services recommendation, [FS1], [FS17], [FS4], [FS26], [FS9], [FS30], [FS11], [FS14], and [FS40] employed a Web services QoS dataset named WS-DREAM. [FS10] and [FS14] adopted the famous movies recommendation dataset MovieLens, and [FS22] utilized the quality of Web service (QWS) dataset for Web services recommendation. Figure 4-10 illustrates the usage of published benchmarks in the final selected papers.

Table 4-13: Published Benchmarks Details

	Name	Collection Datasets Benchmark	Type	Purpose	Papers
1	WS2003	http://www.andreas-hess.info/projects/annotator/ws2003.html	WSDL	Discovery	[FS25] [FS42]
2	WS-DREAM	http://wsdream.github.io/	QoS	Recommendation	[FS1] [FS17] [FS4] [FS26] [FS9] [FS30] [FS11] [FS14] [FS40]
3	SAWSDL-TC	http://projects.semwebcentral.org/projects/sawsdl-tc/	SAWSDL	Discovery	[FS8] [FS23]
4	STag 1.0	http://www.zjujason.com/data.html	WSDL,Tags	Discovery	[FS3] [FS46] [FS2]
5	OWLS-TC	http://projects.semwebcentral.org/projects/owls-tc/	OWLS	Discovery	[FS18] [FS37] [FS38] [FS2] [FS5] [FS47] [FS48] [FS31] [FS6]
6	MovieLens	https://grouplens.org/datasets/movielens/	Movies	Recommendation	[FS10] [FS14]
7	QWS	http://www.uoguelph.ca/~qmahmoud/qws/	WSDL QoS	Recommendation	[FS22]
8	MED	http://web.eecs.utk.edu/research/lsi/	IR test collection	Discovery	[FS25]

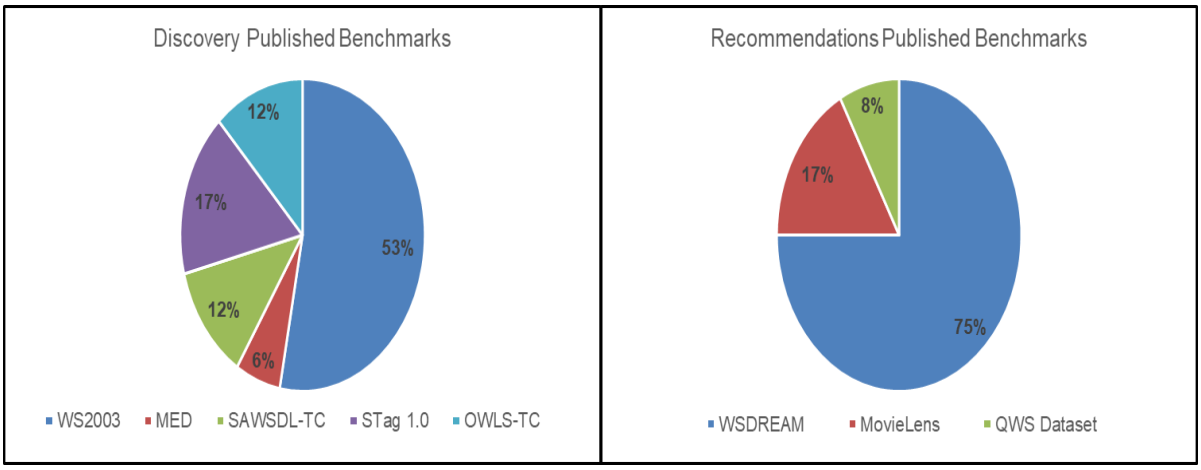


Figure 4-10: Published Benchmarks Usage

For Web services collection, Appendix B.5 outlines the methods of collection, the types of datasets, and the nature of datasets cited in the final selected papers. Fifty-six percent of the final selected papers employed known published benchmarks to validate their proposed solutions. On the other hand, 44% of the final selected papers employed self-gathering to collect Web services datasets for validation purposes. The type of Web services datasets studied in the final selected papers can be syntax, semantic, or both. Based on the description language (DL), researchers tend to use syntax-based Web services more than semantic-based to validate and test their solutions. The nature of Web services datasets in the final selected papers can be datasets from real-world Web services, synthetic datasets for validation purposes only, or a combination of both real and synthetic. Real-world Web services datasets are utilized more by researchers compared to other types.

RQ 3: Web services discovery and recommendation trends and future directions

The last decade witnessed the growth of Web services as providers developed and published an enormous number of Web-based services. This made it a challenge to discover services in a broad and diverse Web services space. The discovery problem is inherently difficult because of the large scale of SOA systems, including Web services, mobile services, cloud services, and cloud computing. These services are dynamic, changing, and uncertain in nature. The scope of services in SOA is changing rapidly as new services are added, and old ones are deleted or modified. New standards emerge as old ones are

abandoned by developers.Web services discovery and recommendation methods and techniques proposed by researchers have been applied separately rather than in a parallel fashion. Web services discovery (matching) and Web services recommendation (selection and ranking) need to be integrated to better understand user requirements and locate the right Web services. Previous approaches separate Web services discovery from Web services recommendation. We argue that the cause of this is the difficulty of finding the right Web services datasets to apply both the discovery and recommendation. However, matching and selecting the right Web services should be considered together and in parallel in this process. Another gap is the use of one type of similarity measure to find similar Web services in the process of Web service clustering or discovery. The limitations of syntactic-based similarity measures can be overcome or minimized if another similarity measure is employed, such as distance, semantic, or context similarity.

Clustering techniques can be utilized to reduce the search space when looking for the right Web services. However, it can be time-consuming to cluster Web services based on different similarities. The new parallel and distributed data processing tools and techniques, including Apache Hadoop, can be employed to group Web services and increase or enhance clustering speed. One example is its use in the K-means parallel clustering algorithm to deal with Web services discovery and prediction problems in the presence of big data. Another gap is found in achieving a domain-specific Web services clustering while considering service requesters and service context. Web services clustering's performance can be improved with additional features and Web services information. An example, tags associated with Web services can be utilized to improve the performance of Web services clustering. Most of the proposed approaches investigate offline Web services clustering. Real-time or online Web services clustering provides a better solution and ensures current Web services information. QoS parameters are an important aspect of the process of Web services ranking, selection, and recommendation. This is not considered in most Web services discovery works. Detecting malicious and outlier users with inaccurate QoS information and noise QoS improves prediction. Finding a complete solution for Web services discovery and recommendation starting from the user query to matching and retrieving the requested Web services and finally selecting and ranking Web services is rare to find in an environment where most proposed solutions enhanced only one part of the

overall process. Finding a complete Web services discovery and recommendation system in more detail is a gap. Semantic-based Web service approaches need to provide service requesters with step-by-step semantic guidance for locating the target Web service components to minimize the complexity.

We have identified three areas of trends and future directions for Web services discovery and recommendation. Table 4-14 outlines the three areas and the challenges to be addressed.

- ***Comprehensive and Neutral Web Services Discovery and Recommendation****:* the discovery process of Web services focuses on WSDL-based or semantic-based Web services, neglecting the fact that most Web services are described in plain text. Web services discovery approaches should extend their methods to be comprehensive and neutral to work with any Web services description. Most Web services (Web APIs) are described in plain text. Therefore, NLP, text and data mining, IR, collaborative tagging, and service information extraction techniques can be employed to identify useful patterns, cluster and group related Web services, and extract useful semantics.
- ***Social Network Web Services Discovery and Recommendation****:* Web services discovery and recommendation can be enhanced with social network data to detect hidden relationships between services and service requesters to generate recommendations. Information and interactions can improve the process of Web services discovery and recommendation by utilizing social network data through user activities. For example, user experiences, ratings, questions, and feedback posted on the social network regarding Web services can better understand user requirements. Social networks and network analysis tools and techniques can boost and improve Web services' discovery and recommendation.
- ***Big Data Analytics-driven Web Services Discovery and Recommendation****:* There are limitations in the proposed Web services discovery and recommendation approaches. These can be overcome by considering big data analytics tools and techniques. Clustering Web services approaches can benefit from running the parallel clustering algorithm to speed up the process of clustering. Furthermore, online Web services discovery and recommendation provide a new research direction to take advantage of the development of algorithms and models for processing data online

in big data ecosystems. These techniques provide a way to develop more scalable and adaptable discovery and recommendation solutions on a large scale and in a changing environment.

Table 4-14: Future Research and. Challenges Addressed

	Suggestions for Future Research	Challenges Addressed
1	**Comprehensive and Neutral Approaches** Employing NLP, IR, Text Mining, Tagging, Data Mining, Information Extraction	- Semantics extraction and extension. - Minimize search space (clustering). - Add extra data/words to Web services.
2	**Social Network Data and Network Analysis Techniques**	- Improve clustering and recommendation with additional information about service users. - Detect correlation and similarities between service users. - Detect user behavior patterns. - Enrich user context.
3	**Big Data Analytics Tools and Techniques**	- Increase clustering speed. - Online service clustering and processing. - Online Web services discovery including crawling and processing to delete duplicates of Web service files. - Online QoS prediction.

4.6. Validation

This SLR provides a classification of clustering and association rules techniques for Web services discovery and recommendation. It identifies methods, algorithms, similarity measures, and evaluation metrics. The review was carried out to provide the current state-of-the-art information in the area of clustering and association rules for Web services discovery and recommendation based on systematic procedures. The focus was to find answers to predefined research questions from the final selected papers. We identified 57 final selected papers with specific quality requirements. We extracted data to answer the research questions and believe that significant work remains to improve the current state of research in the area of clustering and association rules for Web services discovery and recommendation. However, validity is a primary concern in such empirical studies. We

discuss the limitations regarding construct, internal, and external validities [131] in validating this SLR.

Construct validity looks at whether the implementation of SLR matches the initial objectives. We identify the search and selection processes as possible limitations. Search terms and keywords were derived from the population and intervention utilized in forming the research questions. They were tested against a well-known list of research studies. However, the completeness and thoroughness of the terms are not assured. To mitigate the risk, we snowballed the references of the list of selected studies. The search identified two studies in the Chinese language, which were excluded. Additionally, 10 studies were excluded because we were unable to access the full text. This may present a threat to construct validity. Although well-known digital libraries were utilized to search for the selected studies, other digital libraries may contain relevant studies that have not been taken into consideration.

Internal validity is the extent to which the design and conduct of the study can prevent systematic error and work as a prerequisite for external validity [74]. Some concerns about data extraction are worth highlighting here. In the process of data extraction from the final selected studies, we rely on our interpretation and analysis when necessary data were not clearly stated. Some of the required data in the process of extraction were missing in the final selected studies. This may add a threat to the internal validity. This SLR was conducted in 2018 which may add a threat to internal validity since it has not been updated.

External validity is the extent to which the effects observed in the study are applicable outside the study [74]. It is about the generalizability and applicability of the study outcomes. The research questions and quality assessment reduced the risk of generalizability of the outcomes. Furthermore, the SLR is focused on being controlled by a focus problem related to clustering and association rules techniques for Web service discovery and recommendation. This was at a predefined time (2006-2018), and the problem was not previously investigated. When considering this information, we consider the outcome of the SLR to be generalizable and applicable.

4.7. Summary

We presented descriptive knowledge by conducting a systematic literature review to answer different questions. We provide our observation and classification of the investigated studies based on this SLR. Furthermore, we identified different methods, algorithms, similarity measures, and datasets. More importantly, we discuss and present the gaps and future research directions as part of this SLR.

Web services discovery systems have different perspectives and considerations according to the methods and approaches employed to support the process of discovery. Chapter 5 discusses our proposed conceptual model and typology of Web services discovery systems identifying its five characteristics.

Chapter 5. Conceptual Model and Typology of Web Services Discovery Systems

This chapter describes the proposed conceptual model and the typology for Web services discovery systems. The conceptual model provides a high-level representation of Web services discovery systems with regard to their different elements, tasks, and relationships. The proposed typology provides a thorough understanding of Web services discovery systems based on five identified characteristics, which include storage and location characteristics, formalization characteristics, automation characteristics, matchmaking characteristics, and selection characteristics. The proposed artifacts help Web services discovery developers in designing a meta-Web services discovery system by considering its different components and characteristics. Furthermore, the artifacts provide a better understanding for both end-users and software developers about interacting with Web services discovery systems. The details of the proposed conceptual model and typology with its different components are discussed below. We utilize the typology to compare Web services discovery methods and architectures from the extant literature by providing comparisons and presenting discussions, linking them to the five proposed characteristics. The content of this chapter is published in [14, 27].

5.1. Problem Identification

The goal of Web services discovery systems is to identify appropriate Web services resources through a combination of functional and non-functional requirements relevant to the service requester. Various frameworks, methods, and architectures for Web services discovery have been proposed in the literature [132, 133]. While some center on high-level characteristics and offer guidelines for Web services discovery mechanisms [27, 134–138], other proposed architectures recommend particular technological features for the implementation of Web services discovery systems [139–141]. These diverse perspectives and discourses illustrate the need for synthesis at a conceptual level to offer a comprehensive, streamlined understanding of Web services discovery systems. This situation establishes

the motivation to provide a conceptual model as a high-level plan for how Web services discovery systems work and how their elements fit together. Furthermore, the diversity of Web services discovery systems' methods and techniques imposes the need for a typology to provide conceptual and descriptive groups capturing how Web services discovery mechanisms are similar, related, or sometimes convergent based on different features.

There is a significant increase in the number and use of Web services on the internet. Therefore, Web services architects and software developers must implement productive Web services discovery mechanisms to better support the identification of appropriate Web services through a combination of different discovery system characteristics. Web services discovery systems require a set of components to provide functionalities based on their characteristics to ensure success. There are different characteristics that are needed to be considered to have a full, rich, and comprehensive Web services discovery system.

5.2. Objectives of the Developed Artifacts

The proposed artifacts, including a conceptual model and typology, with its characteristics, provide an abstract level of the requirements of Web services discovery systems. For each characteristic, we categorize different approaches to support and operate that function as part of the Web services discovery systems. The artifacts' main objective is to enable software developers to build a comprehensive Web services discovery system by considering our five identified characteristics. Another objective is to enable end-users to better understand how Web services discovery systems work and what are the related elements and components.

The main challenge in developing this solution is the diversity of Web services discovery systems, methods, and techniques from the extended literature, which makes it difficult to investigate them all. However, the lack of solutions that provide a satisfactory explanation and description on the conceptual level of Web services discovery systems motivates us to design and develop such artifacts. The proposed artifacts help with the process of developing Web services discovery systems by providing and investigating their different elements and components. Furthermore, they show how each element of the Web services discovery system interacts and communicates with the others. The observed characteristics play a significant role in identifying what the main focus of the discovery system

is and whether the defined characteristics have been investigated in developing a Web services discovery system.

5.3. Conceptual Model for Web Services Discovery Systems

Conceptual models support different tasks in the context of designing applications and technical solutions as well as organizational and managerial aspects of information systems [142]. The purpose of constructing conceptual models is to express, clarify, and simplify the meaning and relationships of concepts and terms and their functionalities and structure in a specific domain. Constructing a conceptual model is an essential aspect of the development process for supporting communication and diminishing different interpretations of these terms and concepts [143–145]. Without a proper conceptual model, the information architecture of the developed system will reflect a set of features with no clear connection to each other. These features may each solve a problem, but the end-users would need to work hard to understand and complete the required task they want to do. This is a consequence of different components of the system not being grouped together in a way that supports end-user expectations of what they need to complete the task. We need to connect the dots between the system's different features by developing a conceptual model showing both the technical perspective and user requirements. Therefore, the proposed conceptual model for Web services discovery systems provides a representation comprising concepts, elements, interrelationships, functionalities, and structure. This will help end-users, including software developers, business representatives, stakeholders, or even naïve users, to understand the different elements and components of the Web services discovery systems.

It is critical when developing a conceptual model to consider that all end-users and stakeholders correctly and fully understand the conceptual model, communicate about the model contents, and its requirements [145] to improve the understandability of the conceptual modeling and its success [142]. Therefore, the proposed conceptual model for Web services discovery systems, as illustrated in Figure 5-1, is discussed in detail by referring to its elements, relationships, functionalities, and characteristics.

At the conceptual level, Web services discovery systems can be divided into a front-end and a back-end. The front-end deals with the discovery systems' end-users based on how they interact with the system and how their requirements are analyzed and specified.

The discovery interface design plays a significant role in gathering the end-users' functional and non-functional requirements. Furthermore, discovery interfaces can extract detailed user data and information, such as the user context and profile, that can match users' requests. The back-end shows how the discovery system actually works and hides the details from the end-users. The matching algorithms employed in the discovery system match the end-users' requirements against the list of Web services, which are advertised and published in the Web services repository and storage. The following sections explain the elements of the conceptual model, their relationships, and their functionalities.

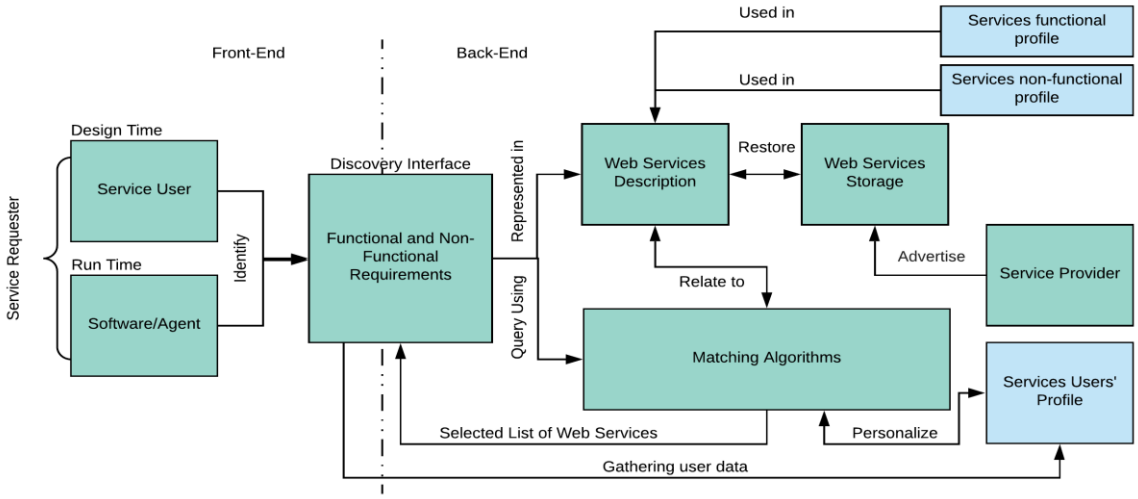


Figure 5-1: The Conceptual Model of Web Services Discovery Systems

5.3.1 Service Requester

The process of Web services discovery is invoked by a service requester who is looking for a Web service. Service requesters (i.e., service consumers) are the end-users for the discovery system who are looking for a Web service that meets their requirements. The discovery process can be invoked by a human service requester at design time or by software at run time. This situation indicates the need for a discovery interface that is able to deal with both humans—User Interface (UI) design—and machines for both manual and automated invocation. Humans use a manual invocation of the system by selecting and identifying their functional and non-functional requirements on the discovery interface. On the other hand, software/agent users use automatic invocation by identifying the requirements via a specialized machine-based discovery interface. The end-users do not need to

understand the complexity of the back-end of the Web services discovery systems. However, they expect to get the best list of Web services based on their functional or non-functional specified parameters.

5.3.2 Discovery Interface and Users' Requirements

The discovery interface is the first data-gathering step from the user side and the first point of contact where the system collects more information and data about the users, their context, and any further details. The discovery interface needs to be customized to fit both human and software users who identify and specify their selection based on functional and non-functional requirements. There are different methods and ways to let the end-users select and identify their requirements, which could be based on keyword search or filtering and selecting pre-identified requirements (i.e., parameters). This requires a mechanism enabling discovery systems to understand users' requirements and selection preferences. The matching algorithm applies users' selection preferences to filter the retrieved list of services based on their requirements. The functional and/or non-functional properties of Web services are represented in a unique format of Web services description languages. Eventually, the discovery interface is responsible for providing the end-users with the retrieved list of Web services that meet their requirements.

5.3.3 Web Services Description

Considering a specialized Web services description language to represent both functional and/or non-functional properties requires a specialized tool and community support to form such a language for discovery purposes. This requires language formalization specifications to form description languages capable of representing both functional and non-functional properties of Web services. The services' functional and non-functional profiles contain data about services' functions, QoS, context, and other details [146]. Creating services' profiles help in grouping services based on their functional or non-functional properties before service requesters make their requests, which facilitates and speeds up the matching algorithm's work. Furthermore, Web services can be grouped based on their similarities into smaller clusters based on their functional or non-functional properties. Web services description files are retrieved from the Web services storage or repositories.

5.3.4 Web Services Storage

Web services descriptions are published and stored on Web services repositories and data storage, which can have different architectures to support the discovery process. Most of the time, service providers publish their Web services where their description and related data are stored in Web services registries for query purposes. Web services registries have a centralized architecture, such as UDDI, consisting of different repositories that synchronize periodically [147]. Storing Web services descriptions based on a centralized architecture suffers some limitations that affect the scalability, flexibility, and robustness of the discovery mechanism [148]. This indicates the need for Web services storing mechanisms that provide a more scalable, flexible, and robust discovery process, and provides a better understanding of how Web services discovery system should be conceived based on the location and storing of Web services description files.

5.3.5 Matching Algorithm

All the previous components are useless if they are not utilized by the matching algorithm, which plays a significant role in the discovery processes. The matching algorithm can utilize all the previous components' data to optimize the matching process. Based on the end-user query, a list of Web services that match the user's functional and non-functional requirements is retrieved. Furthermore, the matching algorithm can utilize any information from the services' user profile to personalize the list of retrieved Web services. The matching algorithm may use artificial intelligence, text and data mining, NLP, similarity measures, and various techniques to match the end-user request. Most of the time, this process requires complicated mathematical computations and the use of different similarity measures to find and match the best suitable list of Web services.

5.3.6 Service Providers

Service providers are Web services (i.e., Web APIs) developers who develop and design the functionalities of Web services around their business model and competencies. Web services providers publish and advertise their services in different locations, including public directories, Web services portals, and Web API marketplaces in various description languages and formats [32]. Web services providers control where to advertise their Web

services and what description languages (i.e., formalization) to use to describe their Web services.

Based on the conceptual model's components and elements for Web services discovery systems, we identified five characteristics for building a comprehensive and wide-ranging Web services discovery system. The five characteristics include storage and location characteristics, formalization characteristics, matchmaking characteristics, automation characteristics, and selection characteristics. These are observed from the vast literature, and they are reflected in the proposed conceptual model, as illustrated in Figure 5-2. For example, the automation characteristics are observed from the component related to the service requestor. The five characteristics form the basis for our proposed typology of Web services discovery systems.

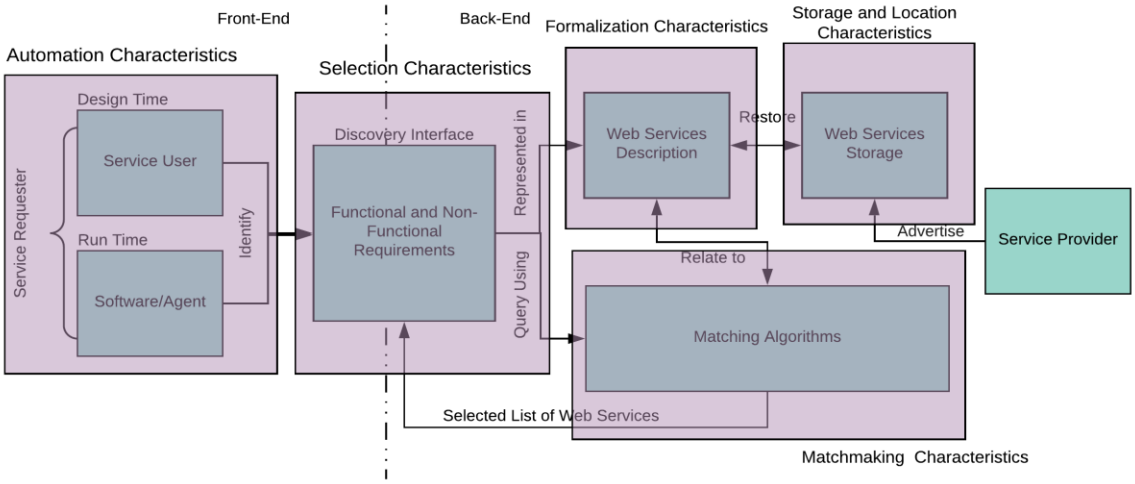


Figure 5-2: The Conceptual Model Reflecting the Five Characteristics

5.4. Typology of Web Services Discovery Systems

The proposed typology of Web services discovery systems is shown in Figure 5-3. The typology is based on five characteristics to configure a meta-Web services discovery system based on the descriptions and definitions of the Web services discovery system components, including

1) Storage and location characteristics: These refer to the architecture where Web services descriptions are stored to conceive the discovery process, including centralized and distributed architecture.

2) Formalization characteristics: These refer to the form of the markup/description languages employed to describe Web services, including syntax and semantic description languages.
3) Matchmaking characteristics: These refer to the matching algorithms employed to match the user request (query) and Web services descriptions, including syntactic, semantic, context, and hybrid-based matching.
4) Automation characteristics: These refer to the mechanism employed to invoke the Web services discovery process, which can be either automatic during run time or manual during design time.
5) Selection characteristics: These relate to the selected properties and parameters utilized by the discovery system, including functional and non-functional properties.

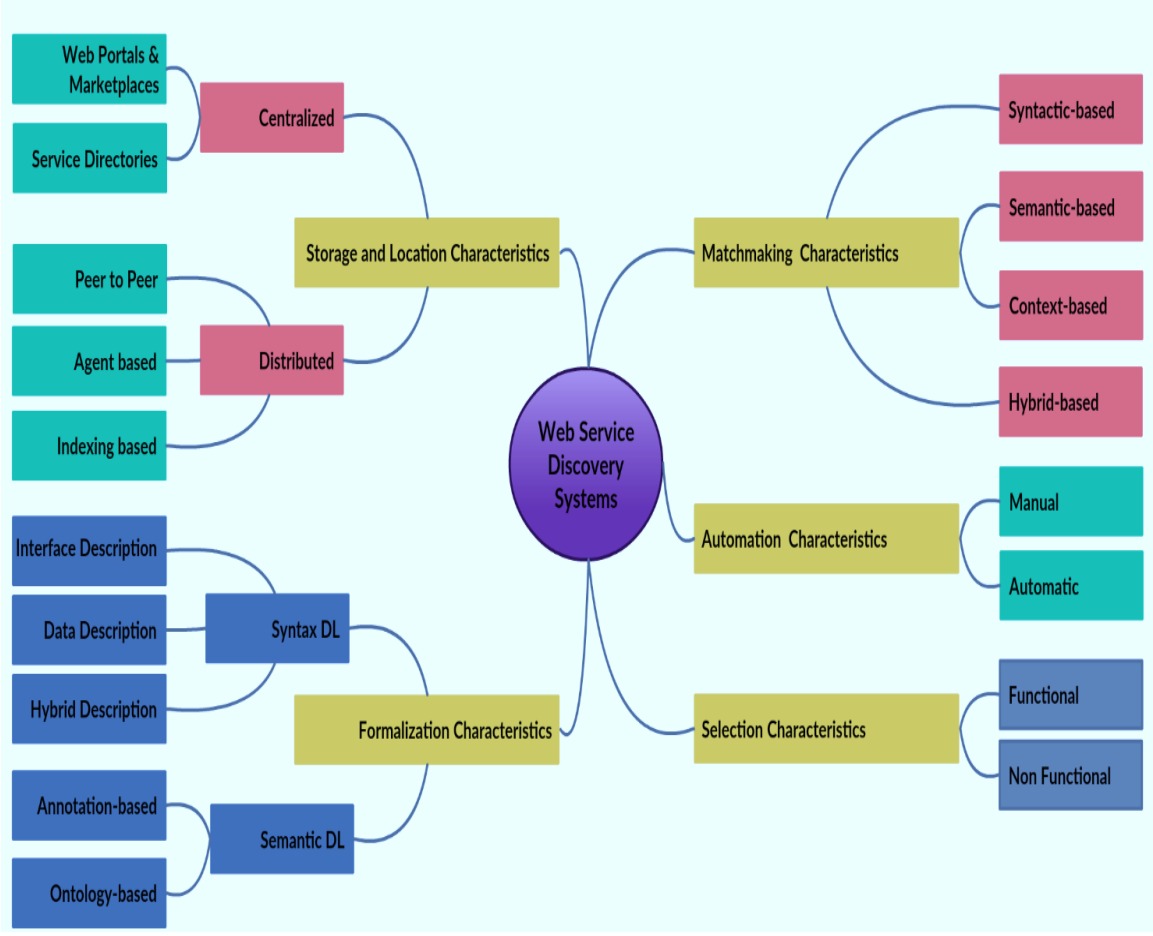


Figure 5-3: Typology of Web Services Discovery Systems

5.4.1 Storage and Location Characteristics

Storage and location characteristics refer to the architecture of the discovery systems where Web services descriptions are located and stored for access and matchmaking. The architecture can be a centralized architecture or a distributed architecture.

A centralized architecture uses a single central directory or global service directories (repository or registry). This architecture includes the UDDI registry, which is the standard representation of this category [149]. Other directories include Web services portals or marketplaces, such as webservicelist.com, mashape.com, or PW. Usually, a centralized architecture is supported by a middle agent (a matchmaker or a broker) [150]. A matchmaker determines if the request matches the advertisement. However, it does not interfere with interactions between a matched request and a service provider, which is controlled by a broker [151]. Easy access to advertised Web services descriptions is provided with centralized storage where an extra approach to collect service descriptions is not needed [149, 152].

There are limitations to centralized storage architectures. The use of UDDI registries, Web portals, or marketplace registration is voluntary for service providers. This limitation can be problematic if service providers do not publish their services [152]. Moreover, centralized storage architectures can suffer from a single point of failure and scalability problems. As the number of internet Web services increases, it becomes impractical to rely on centralized approaches. Consequently, distributed approaches can be utilized to overcome some of these limitations.

Distributed Web services discovery systems rely on Web services information storage and location mechanisms that are disseminated over different locations. The distributed storage architectures can be classified into one of three architecture types, based on their access methods and information, including peer-to-peer (P2P), agent-based, and indexing-based.

P2P provides an infrastructure for routing and data location. Each peer works as a node to provide service access [149]. P2P Web services discovery systems provide distributed, scalable, and self-organizing systems by distributing the load and data among different nodes. This architecture helps to overcome the limitation of centralized approaches

through improved fault tolerance and search efficiency [133]. In P2P Web services discovery systems, peers can access other resources and services without going through intermediate entities [149] as they were built considering P2P technologies and ontologies to publish and search for Web services descriptions [147].

P2P Web services discovery systems can employ a structured, unstructured, or hybrid P2P network topology [150]. Structured P2P systems do not have a central directory but rather a more controlled network topology structure and a distributed service index of peers. In unstructured P2P systems, peers have no index, no control over the network topology, and no inherent restrictions on types of Web services discovery they can be used for. Hybrid P2P systems combine structured and unstructured mechanisms [150]. However, P2P discovery systems tend to be complicated and challenging to implement. Moreover, sharing Web services information between peers can result in trust issues [152].

An agent-based architecture relies on the use of software agents for an intelligent and efficient Web services discovery mechanism. In a multiagent environment, an agent-based mechanism facilitates and supports the process of Web services discovery by processing user requests and identifying Web services to satisfy user requirements [152]. Any agent can be a service requester or service provider. Interactions among heterogeneous agents are required for the success of the service discovery mechanism [153]. In [154], DAML agents for service discovery (DASD) is an agent-based approach in which Web services requesters and providers discover each other through the intermediary action of matchmaking service.

An indexing-based architecture is a Web crawler or a search engine, which is employed to gather and collect Web services descriptions [155]. Index-based approaches are more efficient in a real Web services search environment, wherein a variety of heterogeneous Web services are published in different and geographically scattered service registers or Web portals. Search engines offer an easy way to publish and discover Web services on the internet [156]. A search engine (e.g., Google) can use crawlers to automatically find Web services and generate and maintain indexes that can be utilized in the Web services discovery process. Furthermore, multiple entities can create their own indexes, and free-market forces can determine one index's popularity over another. However, an index-based Web services discovery system cannot fully understand service requester

functional and non-functional requirements since it depends only on the indexed data and information of the Web services unless other techniques are incorporated.

5.4.2 Formalization Characteristics

Formalization characteristics relate to the markup languages or other languages employed in the description of Web services for discovery purposes. These include syntax and semantic description languages. Syntax-based description languages are further categorized into interface descriptions, data descriptions, or hybrid descriptions. Semantic-based description languages can be classified into annotation-based (bottom-up) or ontology-based (top-down) approaches.

For syntax description languages, an interface description provides a general Web services structure, including operations, parameters, and abstract data types. WSDL is the W3C standard language for describing the SOAP-based Web services interface [54]. There are no standard specifications for describing these functionalities in relation to REST-based Web services. In most cases, the developers of REST services document the functionality of services in public HTML pages or Web portals or with API documentation tools such as Swagger [43]. However, there are some syntax-based description languages for REST service interfaces adopted by developers, including the Web application description language (WADL), RESTful service description language (RSDL), and the 2.0 version of WSDL. The data description provides the messaging and payload format of Web services, such as JSON and YAML. The hybrid description combines both the interface and the data description approaches. An example of such a hybrid description is Apache Avro, which provides a framework for data serialization and remote procedure calls with JSON-based schemas [157].

Semantic description languages are used to overcome the limitations of the syntax Web services description languages, which include not being able to model the complete vocabularies and relationships between Web services functionalities (the semantics), as well as the potential misrepresentation of user intentions. Semantic description languages allow the addition of annotations and knowledge representations to Web services, which can enable them to be machine-readable and machine-understandable [4]. There are two approaches to describe Web services semantically.

The first approach is annotation-based (bottom-up) to adopt an incremental method by including semantic annotations in existing Web services standards. For SOAP-based Web services, WSDL-S and semantic annotations for WSDL (SAWSDL) are two examples of bottom-up approaches for annotating syntax description languages to represent semantics. Moreover, semantic Web services description languages can be used to describe REST-based Web services. With a bottom-up approach, semantic annotations for REST (SA-REST) [158] is a framework for semantically annotating RESTful service APIs.

The second approach is the ontology-based (top-down) approach to use high-level ontology as a base framework to describe Web services. Various ontology frameworks have been introduced for the top-down semantic description of Web services, including OWL-S and WSMO. SEmantic REstful DAta Services (SEREDASj) [159] is another framework based on JSON to describe REST services.

5.4.3 Matchmaking Characteristics

Matchmaking characteristics, which represent the core of Web services discovery systems, relate to the matching algorithms employed to match service requester requirements with available Web services descriptions based on finding and defining similarities. A Web services matchmaker is a program, method, or framework designed to implement the matching algorithm of the discovery task based on different measures. From matchmaking characteristics, Web services discovery systems can be categorized into syntactic-based, semantic-based, context-based, and hybrid-based systems.

Syntactic-based Web services discovery systems use keyword-based matching. A service requester can initiate the search process for the required Web services based on a keyword that is matched against Web services descriptions. Syntactic approaches include keywords and category-based matching and use different techniques and technologies such as NLP, IR, controlled vocabularies, or a directory of synonyms. The UDDI registry is syntactic in its nature and supports keyword- and category-based matching [133, 152]. Syntax description languages can be utilized to formalize Web services in pure syntactic-based discovery systems. Syntactic-based matching approaches are simple and widely employed techniques. Furthermore, they provide a straightforward and quick way to filter

large quantities of Web services. There are limitations of syntactic-based approaches, including the inability to model the semantics of Web services and the potential to misrepresent user intentions [160]. Sometimes, syntactic approaches are utilized as the first step of a more complex and complicated matchmaking method.

Semantic-based Web services discovery systems are based on manually or automatically annotating Web services description documents. These play an essential role in the process of automating the Web services discovery process. For Web services with syntax formalization, semantic Web services discovery approaches use similarity matchmaking based on semantic similarity measures (measuring the likeness of their meaning or semantic). This is based on considering different similarity measures that can measure the semantic similarities and relatedness between words extracted from the Web services description. For Web services with semantic formalization, semantic service-discovery approaches use similarity matchmaking based on ontology as well as service annotation using a variety of matching algorithms. The quality and accuracy of semantic Web services discovery systems depend on the maturity level and accuracy of the matching algorithms. Based on the types of reasoning and measuring techniques employed in the matching process, these systems can be categorized into logic-based, non-logic-based, and hybrid-based matching classes [150]:

1) Logic-based matchmakers: These use logical inferencing of Web services semantic annotations to determine semantic relations and the degree of matching (DOM) between Web services. Most semantic Web services matchmakers perform deductive reasoning based on the semantic annotations of Web services. By considering appropriate logic-based reasoning, matchmakers can interpret both functional and non-functional semantics [132]. In the process of logic-based matching, different ontologies from service providers and service requesters are matched at design time or run time. The DOM can be determined by measuring the logical relevance between service semantic parts using logic-rules or by combining logical relevance measurements with algorithm-based measurements [132]. Depending on the DOM strategy applied in the service profile, four degrees of logical relevance matching are employed, including exact, plugin, subsume, and disjoint. Logic-reasoning approaches, when compared to syntactic-based approaches, provide better matching

results due to the reasoning allowed and the mathematical basis. However, logic-based approaches tend to be more complex, time-consuming, and computationally intensive.

2) Non-logic-based semantic matchmakers: These approaches do not use logic-based deductive reasoning to determine the DOM between a pair of Web services descriptions. Instead, the DOM is determined using different similarity measure approaches, including syntactic similarity matching, structured graph matching, pattern discovery techniques, machine learning, and text and data mining techniques (i.e., clustering, classification, association rules, etc.). Non-logic approaches can overcome limitations of logic-based approaches by providing less complicated and lighter computational methods for matching Web services. The iMatcher1 [134] and DSD matchmaker [161] are first attempts for non-logic matchmakers.
3) Hybrid-based semantic matchmakers: These combine both logic-based reasoning and non-logic-based approaches in the process of matchmaking. This may increase the complexity of the matching process. Nevertheless, if the logic-based approach fails, the non-logic-based approach may succeed. Moreover, experimental evaluation of hybrid approaches showed that the DOM significantly improved.

Semantic-based matchmaking can overcome the limitations of syntactic-based approaches. However, the matching algorithms employed in semantic matching are more complex than syntactic-based approaches, resulting in increased complexity and degraded real-time performance. Furthermore, considering a variety of ontologies to describe Web services semantically can increase the heterogeneity between semantic Web services and the required mediation support for interoperability [152].

Context-based Web services discovery systems are employed to interpret user requests with added information. The ability of the matchmaker to incorporate context is needed for request adaptability and personalization. According to [162], the three most important aspects of context are where you are, who you are with, and what resources are nearby. However, the context may extend beyond location and can include lighting, noise level, network connectivity, communication cost, and social status. The context associated with Web services discovery can be defined as any information that implicitly or explicitly supports a user request [160]. Service requesters in the matchmaking process can provide an explicit context, whereas agents or systems gather the implicit context in automatic or

semiautomatic ways. User locations, social profiles, and usage histories can support the Web services discovery process by providing extra information during the matchmaking process.

Hybrid-based Web services discovery systems involve a combination of syntactic-based matching, semantic-based matching, and context-based matching. Their combination helps to form multistage matching. For example, [163] proposed WSMO-MX, a hybrid semantic matchmaker, to combine logic-based semantic matching with syntactic-based matching.

5.4.4 Automation Characteristics

Automation characteristics refer to the approach that discovery systems follow to invoke the discovery process. From automation characteristics, Web services discovery systems can be categorized into manual and automatic discovery approaches. In manual approaches, a (human) requester uses the discovery system during design time to find and select Web services meeting predetermined functional and non-functional requirements. In automatic approaches, the requestor is an agent or software that uses a discovery system at the task's design time or run time [149]. The process of Web services discovery needs to be performed in a dynamic and automated way without affecting the quality of service.

The requirements and constraints for the two approaches are typically different. For example, interface requirements intended for human interaction with the manual approach are different from machine interaction requirements with the automatic approach [164]. The service requester may not trust an agent to look for the right service during run time. Therefore, they may prefer a manual approach at design time. Web services discovery systems' developers are always looking to develop and design fully automated discovery systems, but this requirement is still a challenge.

5.4.5 Selection Characteristics

Selection characteristics refer to property and parameter selection approaches utilized by discovery systems to select the most appropriate Web services during the matching stage. This refers to the parameters and properties within the Web services description that are

selected in the matchmaking process. From the selection characteristics, Web services discovery systems can be categorized into functional and non-functional discovery approaches.

The functionalities of Web services describe the types of services, how they work, and what they serve [149, 152]. Functional properties can be captured from the service profile, which describes the signature of the service based on its IOPEs [149] in the case of semantic Web services (e.g., WSDL-S). IOPE includes input (I) and output (O) parameters, preconditions (P), and effects (E) as part of the semantic description of the Web services. In IO matching, only the input and output elements are considered for functional selection properties. In PE matching, only preconditions and effects are used as matching parameters. In the same way, all parameters are considered in the process of matching in case of IOPE matching. Web services semantic formalization description languages may have different names for the functional description parameters. For example, WSDL-S uses IOPE, which is equivalent to the IOPR (Results) of OWL-S [52, 54]. For syntax-formalization-based Web services, the functional properties can be captured from the description of the document and its elements. For example, WSDL functional properties can be captured from the operation, message, and documentation elements within the WSDL description file. The selection of the functional properties depends on how and what functional parameters/elements will be selected to feed into the matching algorithm.

Non-functional properties (NFP) represent an important factor in Web services by measuring QoS. QoS parameters are considered in the process of selecting and matching user requirements by the matching algorithm. QoS can include availability, privacy, reputation, price, response time, usability, standards compliance, scalability, security, server location, reliability, and jurisdiction [132, 165]. The matching algorithms use selection approaches to match user requirements. However, different matching algorithms can apply different selection approaches. Each semantic formalization of Web services, such as OWL-S or WSMO, can provide a specific representation for a set of parameters. These can be employed in the process of matching; however, the parameters can differ between description languages.

5.5. Evaluation and Comparison of Studies

As part of the Design Science Research Methodology (DSRM), evaluation consists of two activities, which include demonstrating the artifacts and evaluating the artifacts [68]. Any appropriate activity can be employed to demonstrate how the developed artifacts can solve one or more instances of the problem, which requires effective knowledge of how to use the artifacts [73]. To demonstrate the usefulness of the developed artifacts, we show how the developed conceptual model and typology are reflected in and relate to Web services discovery systems' development from the literature and how they benefit the development of new Web services discovery systems in the future. To do this, we investigated sixteen randomly selected studies to show and compare how different proposed Web services discovery systems relate to the proposed conceptual model and typology, as in Table 5-1. For each selected study, we identify and determine how their proposed Web services discovery method, technique, or architecture fits our proposed artifacts. The components and elements of the proposed conceptual model were sometimes discussed and referred to in the selected studies in developing their Web services discovery methods. However, no apparent connection between the components on the conceptual level was discussed. We identify all the components and elements (constructs) and show their relationship (conceptual model) and how they interact with each other. All the proposed Web services discovery systems in the selected studies investigate and show how to improve one or more aspects of the five proposed characteristics as part of our proposed typology of the Web services discovery systems. We observed these characteristics and showed the need to develop a meta-Web services discovery system by identifying the characteristics and connecting them to the conceptual model. Furthermore, the proposed typology based on the identified characteristics helps in choosing and selecting what approaches to satisfy each characteristic. For example, a syntactic approach can be utilized to satisfy the matchmaking characteristic of the Web services discovery typology. Table 5-1 shows and compares approaches from the selected studies from the literature by relating to the developed artifacts.

Table 5-1:Comparative Study of Web Services Discovery Approaches Reflecting Five Characteristics

Characteristics / *Approaches*	*Storage and Location Characteristics*	*Formalization Characteristics*	*Matchmaking Characteristics*	*Automation Characteristics*	*Selection Characteristics*
Bernstein et al.[134]	Central	Semantic DL Ontology OWL-S	Semantic Non-logic	Manual	Functional
Palathingal et al. [135]	Distributed Agent-based	Syntax DL WSDL	Syntactic Keywords TFIDF	Manual	Functional IO-match
Paolucci et al. [166]	Distributed Agent-based	Semantic DL Ontology DAML-S	Semantic Logic	Automated	Functional IO-match
Chiat et al.[167]	Central	Semantic DL Ontology DAML-S	Semantic Logic	Manual	Functional
Fenza et al.[136]	Central	Semantic DL Ontology OWL-S	Hybrid	Manual	Functional IOPR-match
Jaeger et al. [168]	Central	Semantic DL Ontology OWL-S	Semantic Logic	Manual	Functional IO-match, non-functional
Keller et al.[137]	Central	Semantic DL Ontology WSMO	Semantic Logic	Automated	Functional
Vu et al. [169]	Distributed P2P	Semantic DL Ontology WSMO	Semantic Logic	Manual	Functional, non-functional
Klusch et al. [170]	Central	Semantic DL Ontology OWLS	Hybrid	Automated	Functional IO-match
Kiefer et al. [138]	Central	Semantic DL Ontology OWL-S	Hybrid	Manual	Functional
Sycara et al. [11]	Central	Semantic DL Ontology DAML-S	Semantic Logic	Manual	Functional & non-functional
Klusch et al. [163]	Central	Semantic DL Ontology WSMO	Hybrid	Manual	Functional IOPE-match
Klusch et al. [171]	Central	Semantic DL SAWSDL	Hybrid	Automated	Functional
Skoutas et al. [172]	Central/distributed	Semantic DL Ontology OWL-S	Hybrid	Manual	Functional IO-match
Meditskos el al. [173]	Central	Semantic DL Ontology OWL-S	Hybrid	Manual	Functional & non-functional
Plebani el al. [174]	Central	Semantic DL SAWSDL	Non-logic	Manual	Functional

Furthermore, to demonstrate how the proposed artifacts can help end-users and software developers in better understanding how Web services discovery systems work and enable the development of a comprehensive Web services discovery system, we highlight the identified five characteristics of the Web services discovery system for each selected paper,

as illustrated in Table 5-1. For example, [134] studied a central approach for storage and location characteristics, OWL-S for formalization characteristics, semantic non-logic for matchmaking characteristics, a manual approach for automation characteristics, and functional properties for selection characteristics. This shows how the proposed artifacts can relate and connect to previous proposed Web services discovery systems considering what components and characteristics were involved. Future Web services discovery developers can use the proposed components, elements, and characteristics to design and develop discovery systems, methods, and architectures by considering and referring to the proposed artifacts, as we will show in Chapter 6.

To evaluate the utility of the developed artifacts, we employed a descriptive evaluation method based on DSR [67], where informed arguments or scenarios can be used. In an informed argument, information from the knowledge base, such as the relevant research, can be used to build a convincing argument to show the utility of the artifacts. In the descriptive evolution method with scenarios, detailed scenarios around the developed artifacts are constructed to demonstrate their utilities. In the previous demonstration activity based on Table 5-1, we cited relevant research from the literature to reflect and relate to the proposed artifacts. Sixteen selected papers were utilized to show how the proposed artifacts were related to and reflected in the developing Web services discovery systems.

We use informed argument to support our claim that the proposed conceptual model and typology help end-users in better understand how Web services discovery systems work and in building a comprehensive system by considering the five identified characteristics. We argue that in order to build a Web services discovery system, the components as identified in the conceptual model should exist and be discussed and investigated. However, the level of investigation (to find a possible improved or new solution) is varied for each component. For instance, one study focuses on the development of a Web services discovery system by proposing a solution at the service requester end (i.e., automation characteristics) to invoke the discovery preprocess based on specific requirements and SLA. Therefore, all other components will be part of their overall solution or architecture, but only one component is rigorously investigated where their solution is applied. The supporting evidence is that the studies [163, 170, 171] proposed their solutions to target the

matching algorithm components by improving the matchmaking processes (i.e., matchmaking characteristics). However, all the other components were involved with minimum details, nevertheless.

On the other hand, [135] proposed a solution to target the Web service storage (i.e., storage and location characteristics), and all the other components were involved and discussed with minimum details. If there is no apparent connection and related interactions between the proposed components, studies that investigated Web services discovery systems will not relate and refer to the other components. This indicates that the components of the proposed conceptual model of Web services discovery systems are connected and related in a matter that contributions from the literature can target a specific component where new or improved solutions can apply.

The proposed five components represent and show where contributions from the literature have been proposed as new solutions or improvements to the process of Web services discovery. These contributions can vary based on the targeted components where each component represents (holds) a set of characteristics for the development of Web services discovery systems. These characteristics (defined and proposed) are employed to build our typology of Web services discovery systems. We argue that the typology helps in building a comprehensive Web services discovery system by referring to the defined characteristics. The supporting evidence can be constructed by running inductive reasoning based on the observation during reviewing the related literature (as in Table 5-1) and in our SLR. This indicates that in each observed characteristic, as part of the typology, there are different approaches to implement or construct a Web services discovery system. For instance, part of the architecture of Web services discovery systems can be implemented based on matchmaking characteristics. We observe that syntactic-based, semantic-based, context-based, and hybrid-based characteristics have been employed to build discovery systems. However, how and what parts of the syntactic, semantic, context, and hybrid matching algorithms specifications are employed vary from one study to another based on their core contribution. These observations lead to a more generalized conclusion by proposing a typology of Web services discovery systems. The typology provides a generalized view on how, where, and what characteristics are used to build Web services discovery systems.

As stated earlier, the components of the conceptual model and the characteristics need to be identified in order to build a Web services discovery system. Researchers can focus on one component and investigate its characteristics to provide a solution. As such, in Chapter 6, we provide our proposed solution targeting the matching algorithm component of the conceptual model. We propose a multi-layer Web services discovery architecture based on considering syntactic and semantic measures to cluster Web services to facilitate the matchmaking process. We utilized the identified matchmaking characteristics to propose this solution, where text mining and NLP approaches are applied. As part of the DSR descriptive evaluation, this serves as our detailed scenario to build a Web services discovery system based on the proposed artifacts.

5.6. Discussions and Conclusions

Based on our investigation, we found that some of the studies focused on one of the characteristics by proposing their methods and techniques to improve the process of discovery on that specific characteristic and ignoring others. On the other hand, a few studies focused on more than one characteristic in providing their methods to improve the process of discovery. For example, in the proposed method of Klusch et al. [170], the authors focused mainly on the matchmaking characteristic of the Web services discovery system. However, they relate to formalization and selection characteristics in the matching process. The conclusion is that all the investigated papers incorporated and included the five characteristics in developing their Web services discovery systems. However, they tend to focus more on one or two characteristics instead of providing more details about how other characteristics are configured and structured.

Most of the literature on Web services discovery focuses on SOAP- rather than REST-based Web services. With the increasing number of REST-based Web services [46], the process of discovery has become more challenging, and it represents an open research area. It may be fruitful to focus on Web APIs' discovery rather than specific types of Web services. Providing a solution that caters to different types of Web APIs is critical to the success of the discovery processes in the realm of REST Web services, as is a discovery solution that can apply to a different format of Web services description despite the type of Web services or what kind of formalization characteristics are used. As noted in Table

5-1, relatively few approaches have adopted WSDL-S in comparison to OWL-S, WSMO, and SAWSDL standards. Further research should be pursued to elaborate on the potential of WSDL-S, which has been purported to provide more capabilities for matching parameters than competing standards such as SAWSDL. Similarly, WSMO provides a mediator-centric approach to solve the heterogeneity problem, but little research has focused on this aspect.

Index-based discovery systems have become more popular over other alternatives such as registry and P2P systems, but only a few studies have been published regarding the creation of such systems. Furthermore, logic-based and hybrid-based matchmakers' scalability is another issue, as the processes have become more complex and time-consuming with the growing number of Web services. With the growing number of APIs, the demand for a standard description language (for REST) and API discovery systems is likely to increase, and additional research should be conducted to help organizations develop their API discovery services. The most important is to find a general solution for Web services discovery that can always work despite the type of Web services and their descriptions. We believe that applying text and data mining, machine learning, NLP, and AI can provide a general solution for Web services discovery regardless of the type of Web services.

5.7. Summary

The number of Web services being published on the Web is increasing at an ever-increasing rate as many providers seek to grant their value chain partners access to their computing capabilities and transactional resources. The Web APIs being used for this purpose may be developed and implemented with different formats and standards. The ability to find the "right" Web service that meets a service requester's functional and non-functional requirements depends on the adeptness of Web services discovery systems. Furthermore, the success or failure to find an appropriate Web service is contingent upon how well the Web services is described, and how well matching algorithms and other related components are able to use and employ the Web services description specifications and details to meet user expectations.

In this chapter, drawing upon various proposed Web services architectures and frameworks, we proposed a conceptual model that identifies different Web services discovery systems' components and elements, their functionalities, and relationships. The identified elements include service requester, discovery interface, users' requirements, Web services description, Web services storage, matching algorithm, and service providers. Based on the conceptual model, we proposed a typology of Web services discovery systems based on five identified characteristics, namely, 1) location and storage, 2) formalization, 3) matchmaking, 4) automation, and 5) selection. Subsequently, we utilized the typology to compare various Web services discovery approaches discussed in the extant literature. We demonstrated how the proposed conceptual model and the identified characteristics could be utilized to build a comprehensive Web services discovery system. Furthermore, the conceptual model provides a thorough understanding of how Web services discovery systems work and relate to their different components.

In the next chapter, we present the proposed multi-layer Web services discovery architecture, its methods, and functionalities while employing clustering with various algorithms, attributes representation and embedding models, and similarity measures to minimize the search space before matchmaking. Furthermore, we show how we use the proposed conceptual model and the identified characteristics to develop and design the proposed solution.

Chapter 6. Multi-Layer Web Services Discovery Using Combined Word Embedding, Similarity Measures, and Clustering Techniques

In this chapter, we expand on the proposed conceptual model to design and develop the proposed multi-layer Web services discovery architecture based on clustering Web services considering various text representation models, similarity measures, and clustering algorithms. First, we show how the conceptual model and the typology are utilized in the development of the proposed Web services discovery solution by identifying its components and characteristics. Second, we present the multi-layer architecture with various text representations and similarity models of computation-based Web services clustering. Referring to the architecture's layers, we identify the methods and functionalities employed for the process of Web services clustering.

6.1. Introduction

The large number of Web services over the internet with different formats and functionalities motivates us to develop a Web services discovery approach based on clustering similar Web services with syntactic- and semantic-based similarities to minimize the search space. By clustering related Web services, service matchmakers do not need to match user queries against all the service offerings; instead, the matchmaker can match the user queries against Web services clusters. We propose the use of text and data mining methods while considering various word representations and embedding models, and similarity measures to find the similarity between Web services.

6.2. The Proposed Multi-layer Architecture to Cluster Web Services

In the general Web services discovery architecture, illustrated in Figure 6-1, we identify the components and characteristics by relating them to our proposed conceptual model and

typology. The storage and location characteristics of Web services discovery systems are demonstrated by a centralized approach in Web services storage. The formalization characteristics of Web services discovery systems are demonstrated through the ability to use both syntax and semantic Web services descriptions. This configuration emphasizes the idea and the need for a Web services discovery system that works with all types of Web services description languages. The proposed solution to cluster Web services based on employing different word representation models, similarity measures, and clustering algorithms is part of the matchmaking characteristics.

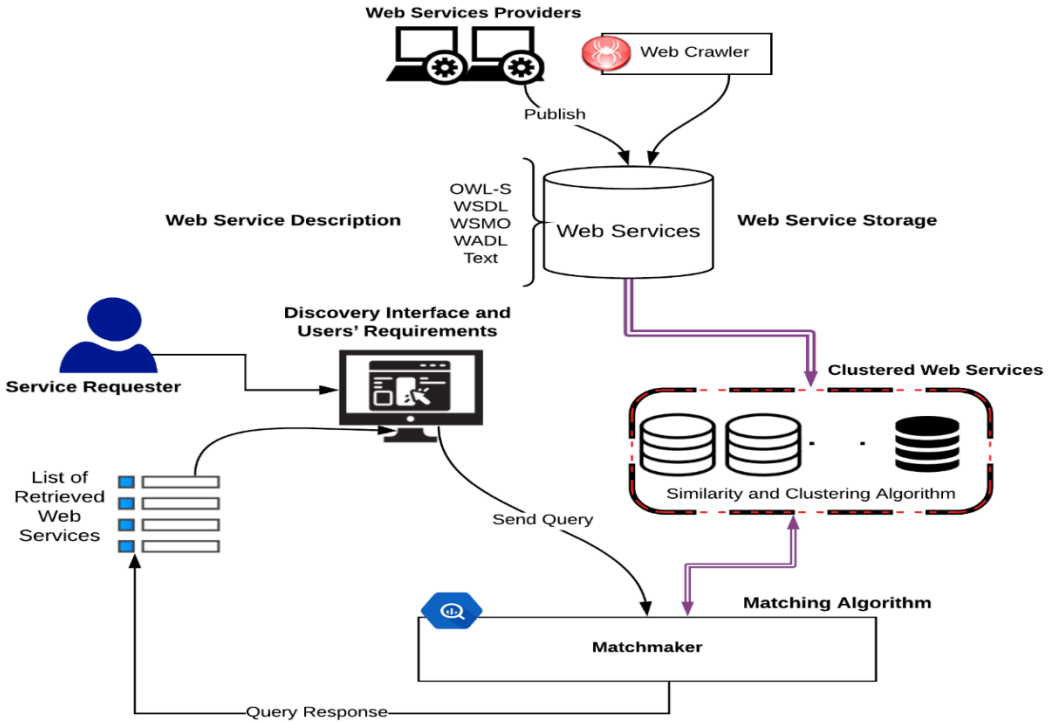


Figure 6-1: The General Architecture of Web Services Discovery

The matchmaker, supported by a matching algorithm, responds to the service requester's query with a list of Web services by referring to the clustered Web services, which is where we propose our solution. The selection characteristics of Web services discovery systems are demonstrated by allowing the user to express the functional and non-functional requirements with the discovery interface. The automation characteristics are demonstrated by allowing the service requester to manually invoke the discovery process.

The proposed clustering-based multi-layer Web services discovery architecture is demonstrated in Figure 6-2. Based on the future directions and gap analysis as part of the systematic literature review presented in Chapter 4, we propose a multi-layer data mining architecture to cluster Web services based on five layers. It is multi-layer because it consists of various layers, including representation and embedding models and similarity measures within each layer. The proposed solution uses natural language processing (NLP), text and data mining models, and various similarity measures and clustering techniques to cluster Web services. It should be noted that the focus of the proposed architecture is only on clustering Web services based on their functional requirements, but not on matching query with the final results (i.e. query is not part of the proposed architecture). The proposed architecture includes a stack of five layers as follows:

1. **Web Services Description and Data Preprocessing**: At this layer, NLP techniques are applied to select and extract terms (features) from Web services description files. This includes different steps starting with parsing selected features from the Web services description files, tokenizing, removing stopwords, and lemmatizing words. The feature selection depends on the type of Web services description documents;
2. **Word Representation, Embedding, and Transformation:** At this layer, two types of NLP models are considered to transform the extracted terms into a computer understandable and processable form. This includes Bag of Words (BOW) with Term Frequency-Inverse Document Frequency (TFIDF) as a weighted scheme, and three word-embedding models. These models include two pre-trained models and one self-trained model;
3. **Syntactic Similarity:** Four syntactic similarity measures are implemented at this layer to find the similarity between Web services. This includes Cosine, Euclidean, Minkowski, and Word Mover's distance;
4. **Semantic Similarity:** At this layer, two WordNet-based similarity measures are employed to find similarities between Web services, including path and WUP similarities. Normalized Google Distance (NGD), also called Normalized Web Distance (NWD), is employed to find the semantic similarity between Web services; and
5. **Web Services Clustering**: At this layer, affinity propagation (AP), K-means, and hierarchical agglomerative clustering (HAC) are studied and implemented to cluster

Web services into some number of clusters based on their functionalities. Search engines or matchmakers use this step as a precursor by categorizing Web services based on their functionality in Web services sources.

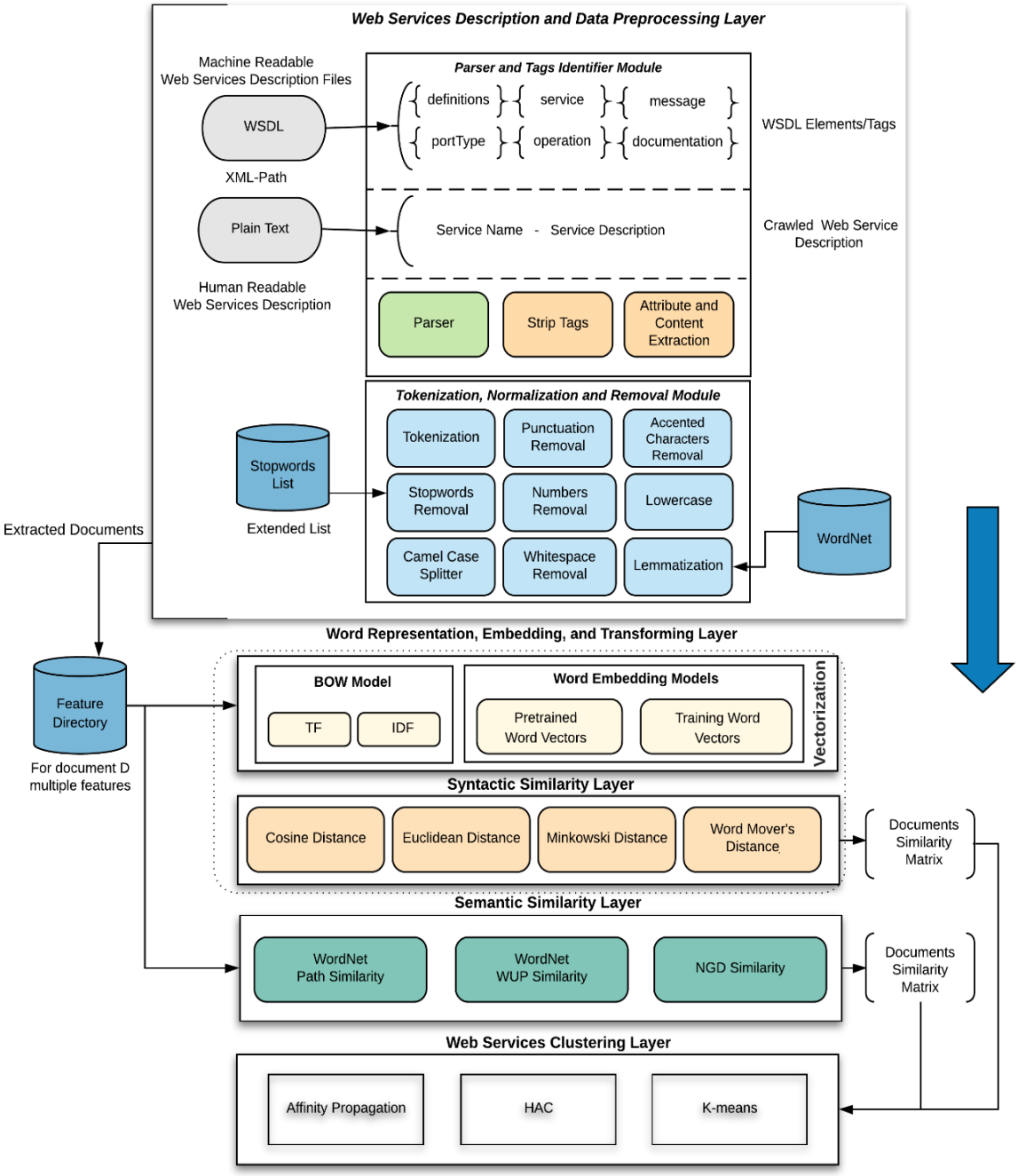


Figure 6-2: The Proposed Clustering-based Multi-Layer Web Services Discovery Architecture

6.3. Layer-1: Web Services Description and Data Preprocessing

The first layer of the proposed architecture is to prepare Web services description files for the mining process. Web services have different types of formalization based on the markup language or other description languages utilized to describe Web services, which can be syntax- and semantic-based description languages [14]. This layer consists of two modules, which are the parser and tags identifier module and the tokenization, normalization, and removal module. The parser and tags identifier module relies on the type of the Web services description language. We identify the features (i.e., tags or terms), including their attributes and content to be selected and extracted. The tokenization, normalization, and removal module applies several operations on the extracted text to make it ready for the mining process. Figure 6-3 illustrates the four main steps for Web services data preprocessing. The collected Web services data include WSDL, which is a machine-readable format, and a textual, human-readable Web services description. The proposed solution uses text mining and various similarity calculations to cluster Web services. This fact makes the solution applicable to any type of Web services description, such as WADL or OWL-S. The key is to parse and identify, from the Web services files, the related data which best help in the process of Web services clustering. This layer is to identify and select the most important and useful features (terms), extract them out of any Web services description files, and then apply NLP preprocessing techniques.

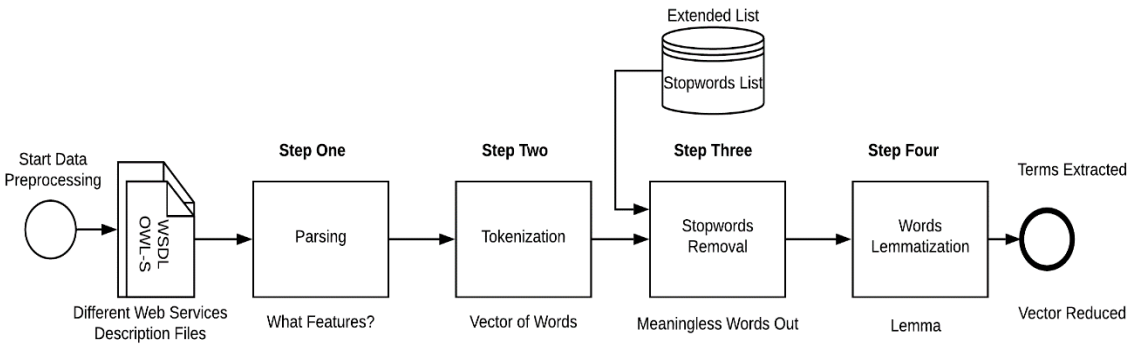


Figure 6-3: Data Preprocessing Steps

6.3.1 Step one: Parsing

The parsing step focuses on identifying what data and features are to be extracted from the Web services description file. As presented in Section 5.4.2, there are varieties of Web services description languages such as WSDL, WSDL-S, OWL-S, and WSMO. A parser is employed to extract the required features and terms from the Web services description where the extracted features are different based on different considerations and strategies. For WSDL, a variety of XML-based parsers are available and provide such functions, such as Beautiful Soup [175], which is a Python XML-based parser utilized by us to extract the WSDL elements that contain meaningful and useful information. The first step depends on identifying useful information in Web services descriptions that can be parsed and extracted to be used in the mining processes. The collected Web services description dataset is based on WSDL. However, this is an implementation detail, and any format of Web services description languages or documentations can be selected for this purpose. We identify the following WSDL elements, their attributes, and the contents to be extracted, as illustrated in Table 6-1.

Table 6-1: Extracted WSDL's Elements

WSDL Element	Extracted	Example
Definition	Name	`<definitions name="MortgagePayments.cmortgagepayments" targetNamespace="http://sal006.salnetwork.com:83/a1vbcode/MortgagePayments/CMortgagePayments.xml" xmlns:tns="http://sal006.salnetwork.com:83/a1vbcode/MortgagePayments/CMortgagePayments.xml" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns="http://schemas.xmlsoap.org/wsdl/"> </definitions>`
Message	Name, Part	`<message name="TotalRepaidRequest"> <part name="LoanAmount" type="xsd:double"/> <part name="Rate" type="xsd:double"/> <part name="NumberOfYears" type="xsd:long"/> </message>`
PortType	Name, Operation	`<portType name="MortgagePayments"> <operation name="MonthlyPayments"> <input message="tns:MonthlyPaymentsRequest"/> <output message="tns:MonthlyPaymentsResponse"/> </operation> </portType>`
Service	Name	`<service name="MortgagePayments"> <port name="MortgagePayments.cmortgagepaymentsPort" binding="tns:MortgagePayments.cmortgagepaymentsbinding"> <soap:address location="http://sal006.salnetwork.com:82/bin/finance.cgi"/>`

		</port> </service>
Documentation	Content	<operation name="ManufacturerKitchen"> <documentation>A ManufacturerKitchenAndHousewaresBox requires a kitchen and housewares manufacturer name as input. An example is "Calphalon". </documentation> <input message="s0:ManufacturerKitchHttpPostIn" /> <output message="s0:ManufacturerKitchHttpPostOut" /> </operation>

The Definition tag is the root element for all WSDL documents and specifies different attributes such as name and targetNamespace [39]. The name attribute is selected, which defines the name of the Web service. The Message tag describes the data exchanged between the Web service provider and consumer as input and output messages. Both the attributes' names are extracted here within the message and part tags. The portType element combines and describes multiple messages and operations. The name attributes for both the portType and operation tags are extracted. The service element defines the ports supported by the Web service and provides other details. The name attribute is extracted from the service element, which provides the name of the service. The documentation element is a container for human-readable documentation, which is allowed in any WSDL element [176]. The content of all documentation tags in the WSDL file are extracted. There are other WSDL elements that we did not consider in this step because of their limited useful information (noise), which is not needed and does not add value for the feature extraction process. As the documentation element is not always utilized in WSDL and is usually ignored [31], the resulting extracted features are short with limited text. Therefore, a scraped Web service description in a human-readable format is employed as auxiliary support text. The text file contains plain text about the service name and service description.

In general, the process of identifying useful data from the Web services description (i.e., feature selection and extraction) is vital for the success of the mining process. WSDL is employed in this case for feature selection, where different WSDL elements are selected. In the same way, other Web services description languages can be utilized for this purpose. Depending on the Web services description language or documentation considered, the useful data (features) are identified based on user judgment of the usefulness of the feature

in identifying the similarities between Web services. The same process can be applied to WSDL-S, OWL-S, WADL, and WSMO.

6.3.2 Step two: Tokenization

Tokenization is the process of splitting the text into a string (sequence of characters) and subunits, called tokens, for identifying the boundaries and breaks of the words [177]. For extracting tokens from the Web services description, we include the sequence of the uppercase letter and lowercase letters (e.g., GetAddress to "get" "address") based on camel case splitting and the sequence of words with symbols in between (e.g., money-amount-transfer to "money" , "amount" , "transfer") based on punctuation removal. All words are transformed to lowercase, and numbers, whitespace, and accent marks are removed. The result of this step is a vector of words extracted from the Web services description, including meaningful parts (e.g., words) and discarding other meaningless chunks (e.g., whitespace).

6.3.3 Step three: Stopwords Removal

Stopwords removal is considered to improve the performance of the clustering by eliminating some words such as 'the', 'is', 'at', 'which', and 'on' for dimensionality reduction. Most common words in text documents are articles, pronouns, and prepositions, which do not give meaning, provide very little semantic information, and are rarely useful. The list of stopwords can be created based on sorting the terms in Web services document collection by frequency of occurrence and entitling the number of high-frequency terms as stopwords [178]. On the other hand, there are different publicly available generic and domain-specific lists of stopwords that can be utilized. It is crucial to add and delete words from the list of stopwords, depending on the text to be analyzed.

We employed the publicly available NLTK [179] stopwords list to eliminate very frequent and rarely useful words from Web services extracted text. However, we updated this list by adding and removing words based on the WSDL specification and its elements. For example, words such as "http", "get", "post", "soap", "cfc", "service", "webservice", and "port" have been added to the list of stopwords. The extended stopwords list contains 213 words. Table 6-2 illustrates part of the updated stopwords list based on selected WSDL

elements. After tokenization, each word from the extracted text from the Web services description data is checked against the stopwords list, and those that match are eliminated from further processing.

Table 6-2: Updated List of Stopwords

Elements	Stopwords
Definition	xmlns, def, xml, xsd
Message	message, msg, soap, in, out, part, para, parameter, cfc,type
PortType	porttype, http, post, get, port, op, response, request
Service	service, webservice, servicename, serv
Documentation	csv, wsdl, from, now, ..

6.3.4 Step Four: Words Lemmatization

Words are reduced into their root form, also known as base or dictionary form or lemma, which is the portion of a word after removing its prefixes and suffixes, to save time and memory space. In natural language processing (NLP), both stemming and lemmatizing can be utilized for this purpose, which represents a special case of normalization [180]. The main difference between the two processes is that stemming is based on rules which trim word beginnings and endings. In contrast, lemmatization uses more complex morphological analysis and dictionaries. A stemmed word might not be an actual word (nonexistent words), whereas a lemma is an actual word. For example, considering NLTK [179] to get the base form of the word "caring", lemmatization with WordNet [65], it identifies the base form correctly as "care", whereas stemming with the Porter stemmer algorithm [181] would cut off the "ing" part and convert the word to "car". We choose to apply WordNet lemmatization on all the extracted words instead of Porter stemmer. As we will be considering different similarity measures that require a dictionary and hierarchy-based measures, lemmatization is more suitable. The reason is that we can find and locate a lemma in a dictionary; however, the root stem may not be located.

This layer is about selecting and extracting features (terms) from the Web services description files. The process of selecting features is based on their usefulness and meaningfulness in differentiating between the various Web services. This is important in the process of measuring the similarities between Web services since similar features indicate

similar or related Web services. The extracted and preprocessed terms are stored in the feature directory for future similarity and clustering analysis. The feature directory contains all the extracted words from the Web services document collection, which are stored in a way to facilitate future text mining and analysis tasks. We can run a simple or more complicated analysis with different techniques and models.

6.4. Layer-2: Word Representation, Embedding, and Transforming

Before applying any algorithms or learning techniques, we need to convert the unstructured text extracted from the Web services documents into a format acceptable by those algorithms. We usually convert these documents to vectors of words, which is a numeric array whose values are specific weights for each word that could either be its frequency, its occurrence, or various other representations. After applying the data preprocessing on Web services description files, the Web services extracted text needs to be represented in numerical values to make them mathematically computable and machine understandable. For a given Web service documents (WSd), $WSd = \{ wsd_i, i = 1, 2, 3, ..., n\}$ where wsd_i represents a Web service document and n is the number of the Web service documents. The vector space model (VSM) [182] is employed to represent each wsd_i of WSd as a numerical value or point ns_i in the numerical space (NS). In the VSM, each Web service document wsd_i is represented as a vector of term/feature weights $wsd_i = \{ t_{1i}, t_{2i}, t_{3i}, ..., t_{ji}\}$ where t_{ji} represents a real number indicating the weight and importance of the term j in the Web service document i. The process of vectorization converts textual information into numbers.

The Bag of Words (BOW) model is employed to generate these real numbers where it uses all words in a Web service document as an index of the document vector. BOW provides the simplest form of text representation in numbers but does have its drawbacks and limitations. However, under the BOW, there are different weighting schema providing different text representation results to overcome some of the model's limitations [183]. BOW with the TFIDF term weighting schema is utilized in part of this layer to represent the extracted Web services features from the feature directory.

Word embedding is the new state-of-the-art approach in NLP. It is based on learning low-dimensional vectors from text corpora employing neural networks [184]. Various

word embedding models have been proposed and have been shown to provide valuable prior knowledge in word representations thanks to their generalization power [185]. Word embeddings are "dense, distributed, fixed-length word vectors, built using word co-occurrence statistics as per the distributional hypothesis"[185]. Word embedding provides a dense vector representation of a word based on its context. Word embedding models focus on representing the idea of a word (looking to the context) by a vector. On the other hand, BOW with TFIDF focuses on representing a word (looking to the frequency) as a vector. We employed both BOW with TFIDF and word embedding to represent the words extracted from the Web services documents.

6.4.1 Term Frequency (TF) and Inverse Document Frequency (IDF)

The TFIDF model is one way to compute the nonbinary weights in BOW to represent Web services documents as a vector document by identifying the importance of terms in text representation. It uses real values to capture the term distribution among Web services documents in the collection in order to assign a weight to each term in every member Web services document. The TFIDF perception is that the more times a term occurs in a Web service document, the more important this term is to this Web service document. Consequently, the assigned weight increases corresponding to the number of times (frequency of occurrence) this term shows in the document. On the contrary, a term that shows in many Web service documents in the collection will be penalized by assigning a lower weight to it [183]. The following are the steps to calculate BOW with TFIDF:

Step one: Term Frequency (TF)

TF measures the number of times a term occurs in a Web service document. The frequency of terms in a Web service document can be calculated with normalized TF:

$$\boldsymbol{TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}} \quad \textbf{(6.1)}$$

where $n_{i,j}$ represents the number of occurrences of a term (t_i) in a Web service document (wsd_j), and the denominator is the number of terms in wsd_j.

Step two: Inverse Document Frequency (IDF)

IDF measures the importance of a term by considering all documents, lowering the weight of the frequent terms, and raising the weight of the rare terms. To calculate IDF for a Web service vector of words, we use:

$$\boldsymbol{IDF_i = \log \frac{|Wsd|}{|\{j: t_i \in wsd_j\}|}} \quad \textbf{(6.2)}$$

where $|Wsd|$ is the total number of Web services documents in the collection. $|\{j: t_i \in wsd_j\}|$ represents the number of Web service documents where the term t_i occurs.

Step three: Term Frequency-Inverse Document Frequency (TFIDF)

The TFIDF for a term (t_i) in an extracted word vector of a document (wsd_j) is calculated with:

$$\boldsymbol{TFIDF_{i,j} = TF_{i,j} * IDF_i} \quad \textbf{(6.3)}$$

where $TF_{i,j}$ is the term frequency and IDF_i is the uniqueness of the term in all documents. After calculating the TFIDF for all terms in the Web services documents, it can be represented by an m by n matrix $A \in R^{m \times n}$ where n represents the number of unique terms occurring in the Web service documents, and m represents the number of Web service documents in the collection. In matrix A, each Web service document is represented by a row vector a_j where j= 1,2,3,….,m, and each term is represented by a column vector. The entries of the matrix A are represented by a_{ji}, where i=1,2,3,…,n. As illustrated in Figure 6-4, the document-term matrix is utilized to find the similarity between different Web service documents. For example, considering syntactic similarity measures between the vectors of wsd_1 and wsd_2 can identify how they are similar and if they share common terms.

Since measuring the similarity between Web services documents with TFIDF depends on the word overlap [186], the probability of having common words decreases, especially in our case where the length of the extracted text is relatively short. Some of the limitations of TFIDF are ignoring synonyms, any semantic relatedness, and the correlation between words in the process of text representation [187]. However, BOW with the TFIDF term weighting schema is still one of the most frequently cited text representations [183].

To minimize these limitations, we propose considering word embedding for representation learning as an additional and supportive step in finding the similarities between Web services.

	T_1	T_2	T_3	..	T_i
wsd_1	$tfidf_{1,1}$	$Tfidf_{1,2}$	$Tfidf_{1,3}$	……..	$Tfidf_{1,i}$
wsd_2	$Tfidf_{2,1}$	$Tfidf_{2,2}$	$Tfidf_{2,3}$	……..	$Tfidf_{2,i}$
wsd_3	$Tfidf_{3,1}$	$Tfidf_{3,2}$	$Tfidf_{3,3}$	……..	$Tfidf_{3,i}$
..	……..	……..	……..	……..	……..
wsd_j	$Tfidf_{j,1}$	$Tfidf_{j,2}$	$Tfidf_{j,3}$	……..	$Tfidf_{j,i}$

Figure 6-4: Document-Term Matrix for Web Services Documents

6.4.2 Word Embedding - Representation by Learning

In NLP, word embedding is the process of converting words/text into numerical representations to create feature vectors [184]. The embedding technique can be based on the frequency of occurrence, which is the case with BOW and TFIDF. Furthermore, the embedding technique can be based on prediction (learning), which is the case with Word2Vec, FastText, and GloVe [185]. In the literature, the term word embedding usually refers to the prediction-based type of embedding. Word embedding models provide a dense vector representation of words that capture some of the words' meaning based on context. Based on the generated vectors, words are placed in such a way that words having similar meanings appear together, and dissimilar words are placed far away from each other. By considering a well-trained set of words, the vector places similar words close to each other in the vector space, allowing calculation of a similarity score for any pair of words.

We incorporated word embedding as part of the proposed solution to represent extracted words from the Web services documents for several reasons. First, word embedding models are capable of representing each extracted word as a dense vector instead of one number, as is the case with the BOW and TFIDF model. Second, the word embedding representation allows us to add the word's semantic meaning by considering the word's context. Third, we use word embedding for text enrichment as we have a limited amount of extracted text from the Web service documents. We use two approaches: a pre-

trained word embedding model and a word embedding model that we trained based on a specific domain based on corpora we created and collected.

Pre-trained word embedding

Pre-trained word embedding models are a set of word vectors that have been created and trained, usually on a general-purpose corpus such as Wikipedia [188], Google News [189], and English Gigaword [190]. These models leverage massive text corpora and are trained and built with billions of various words, allowing them to capture word meanings robustly. One drawback of the pre-trained models is that if we are working on an application or a specific domain, the result of the model will not be optimal due to the generality of the word embedding.

We employed two pre-trained word embedding models, as illustrated in Table 6-3. The first one is based on training the Word2Vec-based skip-gram model on text from the English Wikipedia. The corpus has 3.5 billion tokens and knows 249,212 English words. The preprocessing step was applied to the corpus before training, which includes splitting into sentences, tokenizing, lemmatizing, part of speech (POS) tagging, and removing stop-words. The second word embedding model is based on training a GloVe model on the Common Crawl corpus [191]. The result of the learning process, which is the representation of words (aka word embedding) depends on the corpus utilized in the learning process. For example, the representation of the word "unit" is not going to be the same for the two models. In the first model, the word "unit" and other words need the POS tagging to find a representation match. Therefore, additional steps are required by adding the POS tagging for each word according to the model format. In the second model, the POS tags were not considered in the learning process. Therefore, the word "unit" and other words can be matched with no need for further configurations. The word embedding for the word "unit" based on the two models is demonstrated in Figure C-1 and Figure C-2.

Table 6-3:Selected Word Embedding Pre-trained Models

Name	Algorithm	Vector Size	Window	Vocabulary Size
English Wikipedia	Word2Vec	300	3	249212
ENC3 Common Crawl	GloVe	300	10	2000000

Self-trained word embedding

We trained our word embedding model with the Word2Vec skip-gram model based on a collected corpus in a domain-specific area related to Web services, which fit this problem-specific domain. The corpus was collected from the description of Web services from different Web services repositories and portals such as PW and GitHub. The Web services documents include both a human-readable description and a machine-readable description of Web services. The preprocessing step was applied to the corpus before training, including tokenizing, lemmatizing, removing stopwords and numbers. Table 6-4 illustrates the details of the trained model.

Table 6-4: Word2Vec Self-trained Model Details

Name	Algorithm	Vector Size	Window	Vocabulary size
Web Services Documents	Word2Vec skip-gram	50	3	5563

The process of training the Word2Vec model involves the following steps:

1. Preprocessing and convert text into a suitable format for training: the collected Web services corpus includes 918 Web services documents that are utilized for training purposes. The documents need to go through preprocessing steps, which include tokenization, accented mark, stopwords, punctuation, and white space removal. Then, lowercase and lemmatization steps are applied. Training a Word2Vec model requires that every Web service document is represented as a list, and every list contained a list of tokens/words of that document. This requirement leads to convert all text into a list of lists for each token in the collected corpus. We noticed that a short number of tokens/words are identified for each document, which is one of the challenges in dealing with Web services documents.
2. Choosing the right Word2Vec models: the choice of the training algorithm is a task-specific decision [192]. The Word2Vec algorithm has two models for representing words as a dense vector, including the CBOW (Continuous Bag of Words) and skip-gram models [193]. Both models use shallow neural networks to map words to the target variable, considering the context. In the CBOW architecture, we can give the

context of the word as an input, and it will return the target word as an output. On the other side, the skip-gram architecture takes the current word as an input and predicts the context words [193]. We selected the skip-gram architecture to train our model based on the fact that we have a limited amount of text for training purposes. According to [184, 192, 193], the process of learning is faster for CBOW than skip-gram, which takes a long time. Skip-gram works better for a small amount of training data with its ability to have better representation for rare words. Since we have a small amount of text for training purposes, skip-gram is a better choice to train our model based on the collected corpus despite the time and resources restriction.

3. Setting up the learning hyperparameter configurations: hyperparameter values control the learning process, where different problems or tasks have different optimal hyperparameter configurations [192]. We adjusted the following hyperparameters:
   - Size: each word or token is represented as a dense vector, and the size of the vector is significant in the learning process. Since we have limited data, the size of the vector should set to a smaller value. This is because there are only a few unique neighbors for a given word. We set the size of the vector as 50.
   - Window: this is related to the maximum distance between the target word and its surrounding words (neighbors). The value of the window indicates the number of neighbors to the left and right of the target word. As such, considering our limited data, a smaller window gives words that are more related [192]. We selected the value three as the window size for training the model.
   - Minimum count: this is related to the frequency count of words where uncommon words are usually unimportant. Since our data has a small amount of text, we set the minimum count as one so that all words are considered in training the model.

The word embedding depends on the learning process, and the corpus consumed for model learning. The representation of the word “unit” based on this model is demonstrated in Figure C-3 with 50 as the size of the vector.

Transformation of Web Services Documents in Word Embedding

Word embedding algorithms create a dense vector of the desired size representing each word within the utilized corpus. For example, considering the general-purpose English

Wikipedia pre-trained model resulted in each word from the Web service document being represented in a dense vector of the size 300. A Web service document for a specific Web service has different words, and to represent this Web service, a very high dimension vector is produced. Therefore, the average of the word embedding is taken and employed to represent a Web service document. The resulting dense vector represents the position of the Web service in the vector space, giving a document-word embedding (i.e., document embedding). The distance between Web services' dense vectors is calculated to find the similarity between Web services. In the process of representing Web services based on word embedding, each Web service is represented and transformed into a dense vector based on the representation of each word in the Web service document and then taking the average based on the number of words. The result is document-word embedding (i.e., document embedding).

Demonstration of Web Services Representation, Embedding, and Transformation

This section demonstrates how to represent Web service documents after the Web services description and data preprocessing layer with the BOW with TFIDF weighting schema and word embedding models. We randomly selected eight Web services documents, which include their WSDL files (machine-readable) and their auxiliary text (human-readable) files to demonstrate the process of Web services representation, embedding, and transformation.

TFIDF depends on the frequency of words that occurred in the Web services documents. If a word exists in the document vector, a TFIDF weight will be assigned. Otherwise, 0 will be assigned, showing that this word does not exist in the document. Figure 6-5 illustrates the TFIDF calculation for the eight Web services documents (columns) and nine words (rows) out of 117 unique words. This shows that there are some similarities and relatedness between the Web service documents based on the weight of the words' occurrences. We employed three models to represent Web service documents, including two pre-trained models and one self-trained model. Considering the self-trained model, Figure 6-6 shows the document embedding of eight selected Web service documents, each represented as a dense vector (column) with 50 as the vector size.

	Angle Unit	Area Unit	Transform	Computer Unit	Advanced Travel Information	Schedule	Flight Status	SBG Get Air Fare Quote Service
added	0.000000	0.000000	0.000000	0.000000	0.096059	0.000000	0.000000	0.000000
advanced	0.000000	0.000000	0.000000	0.000000	0.288176	0.000000	0.000000	0.000000
aggregating	0.000000	0.000000	0.000000	0.000000	0.096059	0.000000	0.000000	0.000000
air	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.512724
airline	0.000000	0.000000	0.000000	0.000000	0.080505	0.000000	0.000000	0.368317
airport	0.000000	0.000000	0.000000	0.000000	0.000000	0.685570	0.409961	0.000000
alphabet	0.000000	0.000000	0.000000	0.000000	0.000000	0.272675	0.000000	0.000000
angle	0.778107	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
area	0.000000	0.767149	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Figure 6-5: Document-Term Matrix-based TFIDF Model

	Angle Unit	Area Unit	Transform	Computer Unit	Advanced Travel Information	Schedule	Flight Status	SBG Get Air Fare Quote Service
0	-0.957674	-0.898110	-0.536482	-1.431837	-0.328319	-0.593404	-0.500246	-0.331654
1	0.341187	0.438397	0.475139	0.390757	0.542016	1.059792	1.180433	0.400980
2	-0.450321	-0.341609	-0.383356	-0.402388	-0.382995	-0.044081	-0.159614	-0.332551
3	0.296082	0.034966	-0.099906	0.191460	0.068095	0.045282	0.057029	0.391283
4	0.254418	0.088073	-0.059662	0.265359	-0.229283	-0.380342	-0.447391	-0.349348
5	1.000099	0.737616	0.247602	0.984196	0.164038	0.261254	0.307611	0.513396
6	-0.070222	-0.179206	-0.453377	-0.064383	-0.115866	0.170516	0.292161	0.062143
7	-0.138558	-0.122640	0.078960	-0.179288	0.202571	0.017609	0.225157	0.104745
8	0.025075	-0.053588	-0.194660	0.168614	-0.132342	-0.104968	-0.103972	-0.135740
9	-0.350361	-0.502248	-0.222804	-0.262485	-0.282051	-0.627881	-0.662743	-0.272480
10	-0.778538	-0.761747	-0.106618	-0.551865	-0.042513	-0.214949	-0.237187	-0.016303

Figure 6-6: Documents Embedding based Self-Trained Model

6.5. Layer-3: Syntactic Similarity

The distance between two Web services documents' vectors can be utilized to show how similar or dissimilar the two documents are. For syntactic similarity, we use Cosine distance to measure the similarity between Web services (vector of words) in the vector space model. Furthermore, Euclidean distance and Minkowski distance are employed to measure the distance between Web services by utilizing the generated Web services' vectors.

The Word Mover's Distance (WMD), which is a specific distance metric for word embedding, is also utilized. All the models presented in layer two to represent Web service documents are utilized to find the similarity/distance between Web services. To find the similarities between Web services, we process the TFDIF document-term matrix and document embedding matrix. The outputs of this layer are different document similarity matrices based on four different distance measures.

6.5.1 First Distance Measure: Cosine Distance

Cosine distance measures the similarity or distance between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors projected in a multidimensional space and determines whether two vectors are pointing in roughly the same direction [194]. The word vectors extracted from Web services documents are converted into numbers with weights, making them ready to apply the similarity measure, and the Cosine similarity can be calculated. The vector representations of Web services documents are utilized in measuring the Cosine similarity and distance. To measure the Cosine similarity between Web services documents, we use:

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum_{i=1}^{n} x_i \, y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \;\; \sum_{i=1}^{n} y_i^2}} \quad \textbf{(6.4)}$$

$\vec{x}, \vec{y}$ refer to the word vectors. Applying equation 6.4 resulted in the Cosine distance between all documents, as in Figure 6-7. The result is a document-document similarity matrix showing Cosine distance between all Web services documents.

CosSim(ws1, ws2) = Dot product (ws1, ws2) / (||ws1|| * ||ws2||)

Dot product (ws1, ws2) = ws1[0] * ws2 [0] + ws1[1] * ws2 [1] + - - - + ws1[n] * ws2 [n]

||ws1|| = square root ($ws1[0]^{2}$ + $ws1[1]^{2}$ + $ws1[2]^{2}$ + - - - - + $ws1[n]^{2}$)

||ws2|| = square root ($ws2[0]^{2}$ + $ws2[1]^{2}$ + $ws2[2]^{2}$ + - - - - + $ws2[n]^{2}$)

n = index based on number of terms

Figure 6-7: Cosine Distance for Web Services

6.5.2 Second Distance Measure: Euclidean Distance

Euclidean distance is a measure of the actual straight line distance between two points or vectors in Euclidean space [194]. The word vectors extracted from the Web service documents are converted into numbers with weights, which make them ready to apply the similarity measure, and the Euclidean similarity can be calculated. The vector representation,

including both the document-term matrix and the document embedding matrix of Web service documents are used in measuring the Euclidean similarity and distance. To measure the Euclidean distance between Web services documents, we use:

$$\boldsymbol{d(\vec{x},\vec{y}) = \|\vec{x} - \vec{y}\| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}} \qquad \textbf{(6.5)}$$

$\vec{x}, \vec{y}$ refer to the word vectors, and n refers to the number of attributes. Euclidean distance follows the three conditions of non-negativity, the identity of indiscernible, and symmetry. To find the distance or the similarity between two Web services documents based on the Euclidean metric, we use the generated word vectors, as illustrated in Figure 6-8. The result is a document-document distance matrix showing Euclidean distance between Web services documents.

$$EuclDis(ws1, ws2) = \sqrt{(ws1[0] - ws2\,[0])^2 + (ws1[1] - ws2\,[1])^2 + \cdots + (ws1[n] - ws2\,[n])^2}$$

EuclSim(ws1, ws2) = 1 - EuclDis(ws1, ws2)

n = index based on number of terms

Figure 6-8: Euclidean Distance and Similarity

6.5.3 Third Distance Measure: Minkowski Distance

Minkowski distance is a generalization of the Euclidean and Manhattan distances for calculating distance similarity between two points or vectors in the normed vector space [194]. The vector representation of Web services documents is employed in measuring the Minkowski similarity and distance. To measure the distance between Web services documents, we employ:

$$\boldsymbol{d(\vec{x},\vec{y}) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p} = \sqrt[p]{|x_1 - y_1|^p + |x_2 - y_2|^p + \cdots + |x_n - y_n|^p}} \qquad \textbf{(6.6)}$$

$\vec{x}, \vec{y}$ refer to the word vectors, and n refers to the number of attributes. p is a real number and represents the order of the norm where $p = 1$ is the Manhattan distance. To find the distance between two Web services documents based on the Minkowski metric, we use the generated word vectors, as illustrated in Figure 6-9. The result is a document-document distance matrix showing the Minkowski distance between Web service documents.

$$MinkDis(ws1, ws2) = \sqrt{|ws1[0] - ws2[0]| + |ws1[1] - ws2[1]| + \cdots + |ws1[n] - ws2[n]|}$$

$$MinkSim(ws1, ws2) = 1 - MinklDis(ws1, ws2)$$

$$n = index\ based\ on\ number\ of\ terms$$

Figure 6-9: Minkowski Distance and Similarity

6.5.4 Fourth Distance Measure: Word Mover's Distance

Word Mover's Distance (WMD) is proposed by [195] as a distance function between text documents based on word embedding, which shows that the distance between embedded word vectors is semantically related. WMD measures the dissimilarity between two documents based on the minimum amount of distance that an embedding word in one document needs to travel to reach another embedding word in the other document. WMD works with and leverages word embedding such as Word2Vec and GloVe, where word embedding is generated based on large data sets. The word embedding in each Web service document is utilized to find the distance between other word embedding belonging to other Web services documents. Based on the provided word embedding, WMD works by generating a normalized Bag of Words (nBow) and calculating word travel cost, which is the distance between words based on Euclidean distance. Finally, document distance is calculated as the minimum (weighted) cumulative cost required to move all words from one document to another [195].

6.5.5 Syntactic Similarities Demonstration, Analysis, and Discussion

The vectorization process is done by applying the BOW with TFIDF weighting scheme and three word-embedding models. Four distance measures are utilized to find the similarities between Web service documents (vectors), including the Cosine distance, Euclidean distance, Minkowski distance, and Word Mover's Distance. We explain and demonstrate how the four distance measures are employed to find the similarities between Web services documents. For demonstration purposes, eight randomly selected Web services documents were used to generate the similarity or distance matrix. Two similar Web services will have a high similarity score close to 1 and a small distance measurement; two different Web services will have a low similarity score close to 0 and a large distance measurement.

We manually evaluate the similarity of the eight Web services documents to reflect how the four measures perform in measuring relatedness between the Web services documents. Similar Web services are put together with a suggested name, which works as a ground truth. The Convertor services cluster contains Angle Unit, Area Unit, Computer Unit, and Transform Web services. The Airlines Services cluster contains Schedule, Flight Status, and SBG Get Air Fare Quote Web services. The Travel Services cluster contains the Advanced Travel Information Web service. Our methodology here is to provide an analysis of the similarity metrics by comparing the two most similar and the two most dissimilar of the Web services documents.

The first measure is Cosine distance measure the similarities between the eight Web services documents. In the TFIDF-based similarity matrix, as in Figure 6-10, the two most similar Web services are Flight Status with Schedule and Computer Unit with Angle Unit, which agrees with our manual evaluation. Two of the most dissimilar Web services are Flight Status with Transform and Area Unit with Transform, which does not entirely agree with our manual evaluation. In the Word2Vec-based similarity matrix, as in Figure 6-11, the two most similar Web services are Flight Status with Schedule and Transform with Advanced Travel Information. Two of the most dissimilar Web services are Schedule with Angle Unit and Flight Status with Angle Unit, which agrees with our manual evaluation.

	Angle Unit	Area Unit	Transform	Computer Unit	Advanced Travel Information	Schedule	Flight Status	SBG Get Air Fare Quote Service
Angle Unit	1.000000	0.380361	0.000000	0.386107	0.002732	0.002103	0.002725	0.000000
Area Unit	0.380361	1.000000	0.000000	0.380670	0.002693	0.002074	0.002687	0.000000
Transform	0.000000	0.000000	1.000000	0.001415	0.019205	0.000000	0.000000	0.002929
Computer Unit	0.386107	0.380670	0.001415	1.000000	0.002734	0.002105	0.002727	0.000000
Advanced Travel Information	0.002732	0.002693	0.019205	0.002734	1.000000	0.006323	0.062384	0.039535
Schedule	0.002103	0.002074	0.000000	0.002105	0.006323	1.000000	0.681938	0.000000
Flight Status	0.002725	0.002687	0.000000	0.002727	0.062384	0.681938	1.000000	0.000000
SBG Get Air Fare Quote Service	0.000000	0.000000	0.002929	0.000000	0.039535	0.000000	0.000000	1.000000

Figure 6-10: Cosine Similarity Matrix based on the TFIDF Model

	Angle Unit	Area Unit	Transform	Computer Unit	Advanced Travel Information	Schedule	Flight Status	SBG Get Air Fare Quote Service
Angle Unit	1.000000	0.640053	0.493834	0.570412	0.467457	0.327930	0.347038	0.388168
Area Unit	0.640053	1.000000	0.385397	0.659006	0.468954	0.374169	0.367962	0.369127
Transform	0.493834	0.385397	1.000000	0.588419	0.698499	0.475987	0.460145	0.410476
Computer Unit	0.570412	0.659006	0.588419	1.000000	0.580081	0.388717	0.398959	0.382490
Advanced Travel Information	0.467457	0.468954	0.698499	0.580081	1.000000	0.615482	0.696788	0.632357
Schedule	0.327930	0.374169	0.475987	0.388717	0.615482	1.000000	0.902118	0.530059
Flight Status	0.347038	0.367962	0.460145	0.398959	0.696788	0.902118	1.000000	0.591360
SBG Get Air Fare Quote Service	0.388168	0.369127	0.410476	0.382490	0.632357	0.530059	0.591360	1.000000

Figure 6-11: Cosine Similarity Matrix based on the Pre-trained Word2Vec Model

The second measure uses Euclidean distance to measure the similarities between the selected eight Web services documents. The TFIDF based distance matrix, as in Figure C-4, agreed with the ground truth: the two most similar Web services are Flight Status with Schedule and Computer Unit with Angle Unit. As for the two most dissimilar Web services, there are different Web services with high distance score. Therefore, it is not feasible to identify the two most dissimilar. However, we can spot that Schedule and Flight Status are dissimilar to Angle Unit, Area Unit, Computer Unit, and Transform, which agrees with our manual evaluation. In the Word2Vec based distance matrix, as in Figure C-5, the two most similar Web services are Flight Status with Schedule and Advanced Travel Information with Transform. The most dissimilar Web services are Schedule with Angle Unit, and SBG Get Air Fare Quote Service with Angle Unit.

The fourth measure is WMD measure the similarities between Web services based on word embedding. Three models are employed to generate three distance matrices based on WMD for the eight selected Web services. In the distance matrix based on the pre-trained Word2Vec model, as in Figure 6-12, two of the most similar Web services are Flight Status with Schedule and Area Unit with Computer Unit, which agrees with our manual evaluation. The two most dissimilar Web services are Schedule with Angle Unit and Transform with Area Unit. In the distance matrix based on the pre-trained GloVe model, as in Figure C-6, like the Word2Vec model, two of the most similar Web services are Flight Status with Schedule and Area Unit with Computer Unit, which agrees with our manual evaluation. The two most dissimilar Web services are SBG Get Air Fare Quote Service with Angle Unit and Schedule with Angle Unit. In the distance matrix based on the self-trained Word2Vec model, as in Figure C-7, two of the most similar Web services are Flight Status with Schedule and Area Unit with Angle Unit, which agrees with our

manual evaluation. The two most dissimilar Web services are Flight Status with Angle Unit and Flight Status with Area Unit, which also agrees with our manual evaluation.

	Angle Unit	Area Unit	Transform	Computer Unit	Advanced Travel Information	Schedule	Flight Status	SBG Get Air Fare Quote Service
Angle Unit	0.000000	0.750659	1.247789	0.728724	1.249137	1.281621	1.273041	1.267069
Area Unit	0.750659	0.000000	1.282456	0.605802	1.243213	1.263541	1.271047	1.274109
Transform	1.247789	1.282456	0.000000	1.203030	1.151607	1.234489	1.251466	1.244644
Computer Unit	0.728724	0.605802	1.203030	0.000000	1.212234	1.258629	1.259671	1.269558
Advanced Travel Information	1.249137	1.243213	1.151607	1.212234	0.000000	1.189325	1.125983	1.129202
Schedule	1.281621	1.263541	1.234489	1.258629	1.189325	0.000000	0.481433	1.203642
Flight Status	1.273041	1.271047	1.251466	1.259671	1.125983	0.481433	0.000000	1.186388
SBG Get Air Fare Quote Service	1.267069	1.274109	1.244644	1.269558	1.129202	1.203642	1.186388	0.000000

Figure 6-12: WMD Distance Matrix based on the Pre-trained Word2Vec Model

To conclude, employing the four measures for syntactic similarities as part of the proposed multi-layer architecture can find the similarities between Web services, which are utilized in clustering Web services. The performance of finding the similarities between Web services can vary according to the measures. Considering the eight selected Web services, all the measures correctly identified six out of eight Web services as they belong to similar clusters compared to our manual evaluation. To provide a better solution for clustering Web services based on the generated similarities, we will use three clustering algorithms in layer five of the architecture.

6.6. Layer-4: Semantic Similarity

Words extracted from the Web services documents are consumed in this layer without transforming the text into numbers with the VSM. In the previously employed syntactic similarity measures, the extracted words from Web services documents needed to be transformed into a vector of numbers with the BOW with TFIDF model and the word embedding models to find the similarities between Web services. In this layer, we are considering knowledge-based approaches based on WordNet [65] and Normalized Google Distance (NGD) [128, 196] to find the similarities between Web services. The following sections explain the functionalities and the measures employed as part of our proposed multi-layer architecture for semantic similarity calculations.

6.6.1 First Measure: WordNet Similarity

WordNet is an extensive lexical database of English, which includes nouns, verbs, adjectives, and adverbs. They are grouped into sets of cognitive synonyms called synsets, each expressing a distinct concept and employed to find the relations between different synsets [197]. WordNet-based similarities depend on the lexical semantics, sense relations, and additional information to identify the similarities between words. Each synset in WordNet is labeled with three parts, including the word, the part of speech (POS), and the index of the word's sense. For example, the synset "service.n.01" indicates the first meaning of "service" as a noun. WordNet's structure makes it a useful tool for computational linguistics and natural language processing (NLP). To measure the similarities between Web services, we utilize two WordNet-based similarities, which are include the path and Wu-Palmer [198] similarities.

WordNet Path Similarity

The path similarity measures the similarity of two word senses based on the shortest path that connects the senses in the WordNet hypernym and hyponym taxonomy. Two words in the Web service vector of words are similar to each other if they are near each other in the WordNet hierarchy. The returned score denotes how similar and related two word senses are, ranging from 0 to 1, where 0 is least similar and 1 is identical. The score is calculated based on the number of edges from one-word sense to another word sense. To measure the path similarity between two words, we need to get their synsets and use:

$$\mathbf{PathSim}(s_1, s_2) = \frac{1}{Edgedistance+1} \quad (6.7)$$

WordNet Wu-Palmer Similarity

Wu-Palmer (WUP) semantic similarity measures the similarity and relatedness between two word senses based on the path length between their senses located in the taxonomy along with the depth of their Least Common Subsumer (LCS), which is the closest ancestor to both senses. The returning score denotes how similar and related two-word senses are, ranging from 0 to 1, where 0 is least similar and 1 is identical. To measure the WUP similarity between two words, we need to get their synsets and use:

$$\mathbf{WUPSim}(s_1, s_2) = \frac{2 * depth(LCS)}{depth(s_1) + depth(s_2)} \quad (6.8)$$

The WordNet path and WUP metrics are employed to measure the semantic similarities between Web services documents. The process of generating the Web service documents similarity matrix is not straightforward as several processes are needed since the WordNet similarity measures are based on the words' similarity. Figure 6-13 illustrates the pseudo-code for generating the semantic similarity matrix between Web services documents with bothWordNet's path and WUP.

Input: Web services Documents
Output: Semantic Similarity Matrix between Web services

FOR each Web service document:
    FOR words in a Web service document:
        Tokenize each word
        Get NLTK POS tagging
        Convert NLTK's POS tag into WordNet tags
        Get the first synset from WordNet
    END FOR
END FOR
Output: list of synsets for all words in Web service documents
To generate the pairwise similarity matrix between Web service documents.
FOR each Web service documents:
To calculate normalized similarity score between pair of Web service documents using the list of synset based on Path and WUP similarity metric.
    FOR each synset in Web service document 1:
        Get similarity score to synset in Web service document 2
        Sum all largest score
        Divide the sum by the number of synsets
    END FOR
    Output 1: similarity score document 1 to document 2
    FOR each synset in Web service document 2
        Get similarity score to synset in Web service document 1
        Sum all largest score
        Divide the sum by the number of synsets
    END FOR
    Output 2: similarity score document 2 to document 1
    Take the sum of Output 1 and Output 2, then divide by 2 for averaging.
    Output 3: normalized similarity score between a pair of Web services.
END FOR
Output: pairwise semantic similarity matrix between Web service documents generated based on Output 3.

Figure 6-13: Pseudocode of Generating Similarity Matrix between Web Services Documents

6.6.2 Second Measure: Normalized Google Distance (NGD)

Normalized Google Distance (NGD) is a semantic similarity proposed by [128, 196] which measures the semantic distance between words based on the number of search hits returned

by search engines for a given set of words. Words with the same, similar, or related meanings in the natural language sense have a tendency to be close in units of NGD. On the other side, words with dissimilar meanings have a tendency to be farther apart. We utilized NGD to measure the similarity between words in the Web services documents to come up with words similarity matrix between two Web services. Based on the word similarity matrix, the similarity score between the two Web services is calculated as in Section 6.6.1. NGD between two words w_i, w_j calculated with:

$$NGD(w_i, w_j) = \frac{G(w_i,w_j) - min\left(G(w_i), G(w_j)\right)}{max\left(G(w_i), G(w_j)\right)} = \frac{max\{\log f(w_i), \log f(w_j)\} - \log f(w_i, w_j)}{\log N - min\{\log f(w_i), \log f(w_j)\}} \quad (6.9)$$

Where G refers to Google search engine and $f(w_i)$ is the number of hits returned for the first word w_i, $f(w_j)$ is the number of hits returned for the second word w_j, $f(w_i,w_j)$ is the number of hits returned for both w_i and w_j. N is the total number of pages indexed by the search engine. If NGD(w_i, w_j) = 0 then the two words are viewed as alike as possible, but if NGD(w_i, w_j) => 1 then the two words are very different. The similarity between two words in Web service documents based on NGD is:

$$NGDSim(w_i, w_j) = 1 - NGD(w_i, w_j) \quad (6.10)$$

The similarity between two Web services ws1 and ws2 calculated by measuring the NGDSim , as in equation 6.10, between their words to construct the words similarity matrix, which indicates each word similarity in the ws1 and ws2 documents. The process of normalizing and calculating the similarity score between a pair of Web service documents follows the same steps as WordNet similarity. Search engines such as Google or Bing work for NGD (i.e., Normalized Web Distance) as the frequency count extractor, and the corpus is the Web. In this metric, we use the page counts returned by the search engine as the frequency based on each word and both words together as in equation 6-9. The semantic relatedness of a word consists of the set of Web pages counted and returned by the query.

6.6.3 Semantic Similarities, Demonstration, Analysis, and Discussion

In this layer, a knowledge-based approach and statistical search engine co-occurrence approach are employed to find the similarities between Web services documents. We measure

the similarities and dissimilarities between Web services documents with WordNet's path, WordNet's WUP, and NGD (search engine) metrics. We explain and demonstrate how the metrics are employed to find the similarities between Web services based on the eight selected Web services, following the same methodology and our manual evaluation of the eight selected Web services.

In the path similarity matrix, as in Figure 6-14, the two most similar Web services are Flight Status with Schedule and Computer Unit with Angle Unit, which agrees with our manual evaluation by putting them in the same group. The two most dissimilar Web service are Schedule with SBG Get Air Fare Quote Service and Flight Status with SBG Get Air Fare Quote Service, which does not agree with our manual evaluation.

	Angle Unit	Area Unit	Transform	Computer Unit	Advanced Travel Information	Schedule	Flight Status	SBG Get Air Fare Quote Service
Angle Unit	1.000000	0.569397	0.191624	0.588842	0.223676	0.186272	0.200443	0.148285
Area Unit	0.569397	1.000000	0.142773	0.652522	0.195241	0.186566	0.174735	0.154522
Transform	0.191624	0.142773	1.000000	0.148978	0.251905	0.197246	0.200856	0.170265
Computer Unit	0.588842	0.652522	0.148978	1.000000	0.197341	0.187282	0.174155	0.144964
Advanced Travel Information	0.223676	0.195241	0.251905	0.197341	1.000000	0.284178	0.347654	0.343723
Schedule	0.186272	0.186566	0.197246	0.187282	0.284178	1.000000	0.861383	0.132192
Flight Status	0.200443	0.174735	0.200856	0.174155	0.347654	0.861383	1.000000	0.129416
SBG Get Air Fare Quote Service	0.148285	0.154522	0.170265	0.144964	0.343723	0.132192	0.129416	1.000000

Figure 6-14: Path Similarity Matrix based on WordNet

In the WUP similarity matrix, as in Figure C-8, the two most similar Web services are Flight Status with Schedule and Computer Unit with Area Unit, which agrees with our manual evaluation. On the other hand, the most dissimilar Web services are Computer Unit with SBG Get Air Fare Quote Service and Flight Status with SBG Get Air Fare Quote Service, which does not agree with our manual evaluation.

In the second measure, we employed NGD (i.e., search engine similarity) to measure the similarities between the selected eight Web services documents. In the NGD matrix, as in Figure 6-15, the two most similar Web services are Flight Status with Schedule and Angle Unit with Area Unit, which agrees with our manual evaluation. On the other hand, the most dissimilar Web services are Angle Unit with SBG Get Air Fare Quote Service and Schedule with Angle Unit, which also agrees with our manual evaluation.

	Angle Unit	Area Unit	Transform	Computer Unit	Advanced Travel Information	Schedule	Flight Status	SBG Get Air Fare Quote Service
Angle Unit	0.000000	0.530587	1.188876	0.544186	1.155142	1.214239	1.203872	1.214039
Area Unit	0.530587	0.000000	1.168548	0.547801	1.131361	1.157583	1.171786	1.202945
Transform	1.188876	1.168548	0.000000	1.107387	1.095181	1.169556	1.182077	1.211952
Computer Unit	0.544186	0.547801	1.107387	0.000000	1.122079	1.183508	1.181921	1.200072
Advanced Travel Information	1.155142	1.131361	1.095181	1.122079	0.000000	1.086442	1.054210	1.064334
Schedule	1.214239	1.157583	1.169556	1.183508	1.086442	0.000000	0.452807	1.094781
Flight Status	1.203872	1.171786	1.182077	1.181921	1.054210	0.452807	0.000000	1.055826
SBG Get Air Fare Quote Service	1.214039	1.202945	1.211952	1.200072	1.064334	1.094781	1.055826	0.000000

Figure 6-15: NGD Matrix between Web Services

To conclude, employing both WordNet and NGD for semantic similarities as part of the proposed multi-layer architecture can find the similarities between Web services, which helps in clustering Web services into different clusters. Based on the eight selected Web services, we noticed that NGD performed better than WordNet according to our manual evaluation by referring to the two most similar and dissimilar Web services. To provide a better solution to cluster Web services based on the generated semantic similarities, we will use clustering algorithms in the fifth layer of the architecture.

6.7. Layer-5: Web Services Clustering

Three clustering methods are studied and implemented to cluster Web services based on the observed similarities and dissimilarity matrixes. The first method uses the affinity propagation (AP) clustering algorithm [60]. The second method uses a partition-based clustering method where K-means clustering is employed to cluster Web services. The third method uses a hierarchical-based clustering method where hierarchical agglomerative clustering (HAC) is employed to cluster Web services.

6.7.1 First Method: Affinity Propagation

The affinity propagation (AP) clustering algorithm can find the number of clusters simultaneously and automatically based on the similarity matrix (S) received as an input. Web services are treated as data points, and sending messages between all Web services can find the exemplars, which are the most significant Web services to their clusters. These messages are stored in responsibility and availability matrices. Three matrices are needed to cluster Web services based on AP. The first (input) matrix is the Web services generated

similarity matrix (S) where similar Web services have larger values. The second (initialized) matrix is the responsibility matrix $r(w_i,w_j)$ where the similarities between two Web services are utilized to calculate the responsibility for Web service w_j to be the cluster center (exemplar) for another Web service w_i. The third (initialized) matrix is the availability matrix $a(w_i,w_j)$ to calculate how appropriate it is for a Web service w_i to select the Web service w_j as its cluster center. The number of clusters is influenced by the preference values $S(w_i,w_j)$ and messaging procedures defined in the $r(w_i,w_j)$ and $a(w_i,w_j)$ calculations [60]. Both responsibility and availability matrices are initialized to all zeroes and then updated with the input from the similarity matrix such as:

$$\boldsymbol{r(w_i, w_j) \leftarrow S(w_i, w_j) - \max_{w'_j \neq w_j} \{a(w_i, w'_j) + S(w_i, w'_j)\}} \quad \textbf{(6.11)}$$

$$\boldsymbol{a(w_i, w_j) \leftarrow \min\left\{0, r(w_j, w_j) + \sum_{w'_i \notin \{w_i, w_j\}} \max(0, r(w'_i, w_j))\right\}} \quad \textbf{(6.12)}$$

$$\boldsymbol{a(w_j, w_j) \leftarrow \sum_{w'_i \neq w_j} \max(0, r(w'_i, w_j))} \quad \textbf{(6.13)}$$

Based on the generated availability and responsibility matrices, different iterations are performed until convergence or until a predetermined number of iterations is reached. Web services which represent the examples and other related Web services are placed in the same cluster.

6.7.2 Second Method: Hierarchical-based Clustering

The agglomerative clustering algorithm is utilized for clustering Web services based on the observed distance matrix. Hierarchical agglomerative clustering (HAC) is a bottom-up clustering method that starts assigning each Web service to its own cluster (singleton cluster), then iteratively finds the most similar pair of clusters and merges them into a single cluster until the stop conditions are met or until all clusters have been merged into a single cluster that contains all Web services documents. HAC clustering can be visualized as a dendrogram as in Figure 6-16, where a horizontal line represents each merge between Web services. The y-coordinate of the horizontal line represents the similarity between the Web services' clusters that were merged where the average linkage is employed.

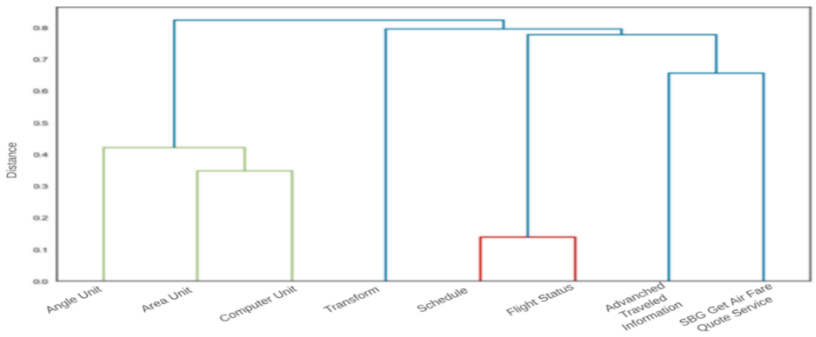


Figure 6-16: Hierarchical Agglomerative Clustering of Web Services Visualized by Dendrogram

6.7.3 Third Method: Partition-based Clustering

K-means is one of the partition-based clustering algorithms utilized to organize Web services documents into clusters. The first step is the initialization, which starts with identifying the number of Web services clusters (k) to find. The second step is choosing k random points as the initial clusters' centers. The third step is to assign Web services to their nearest cluster center. The fourth step is to update the cluster centers by replacing them with the mean of coordinates of all Web services assigned to that cluster. Then, the third and fourth steps are repeated until the clusters converge. The final clustering result depends on the first and second steps, where we need to select the best number of clusters to find and to select the initial centroids. For selecting the optimal number of clusters to find, we implemented the elbow method, which fits the K-means model with a range of k values. We used distortion score, which is the sum of squared distance from each Web services to its assigned center, as a metric, as shown in Figure 6-17. However, in our case we have the number of clusters k as part of the ground truth. We employed K-means++ to randomize the initialization of the centroids to overcome the centroid location selection limitation.

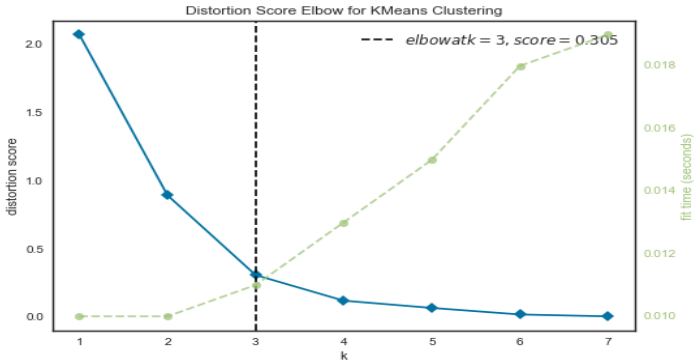


Figure 6-17: K-means Elbow Method

6.7.4 Clusters Analysis, Demonstration, and Discussion

In this chapter, we discussed and demonstrated the four layers of the proposed multi-layer Web services discovery architecture based on clustering Web services to speed up and facilitate the discovery process. Three clustering algorithms are utilized in the fifth layer, including AP, K-means, and HAC to cluster Web services based on the observed similarity or dissimilarity matrices.

Based on the proposed architecture, the clustering performance depends on different factors. The first is the word representations and embedding models for Web services documents transformations. Section 6.4 discussed the four models, including BOW with TFIDF, pre-trained Word2Vec skip-gram, pre-trained GloVe, and self-trained Word2Vec skip-gram. The second factor is the syntactic similarity measures employed to measure the relatedness and similarity between Web services discussed in Section 6.5. The final factor is the semantic similarity measures, including the WordNet ontology knowledge-based and NGD statistical-based search engine methods to find the similarities between Web services discussed in Section 6.6. Based on the observed and resulting similarity or dissimilarity matrices, we cluster Web services considering the three employed clustering algorithms.

We compare and discuss the performance of clustering algorithms to cluster Web services by referring to the number of clusters and the number of agreements and disagreements with ground truth. We report the accuracy, precision, recall, and F-measure based on each similarity measure employed, as shown in Table 6-5. Accuracy is the number of correct predictions divided by the total number of predictions. Precision is a measure of exactness showing how many Web services were predicted correctly out of the ones that were predicted as belonging to a given cluster. Furthermore, recall is a measure of completeness showing how many Web services were predicted correctly out of the ones that should have been predicted as belonging to a given cluster. In addition, F-measure is a harmonic means of precision and recall, which provides a better indicator of the clustering algorithm performance. The details of the evaluation methodology are discussed in Section 7.4.

By comparing the four models based on syntactic similarity, word embedding models perform better in clustering Web services, as shown in Table 6-5. AP based on the BOW with TFIDF model provided the exact number of clusters as the ground truth. However,

the performance of clustering is low for all the syntactic similarity measures when compared to the other models. Among the three word embedding models, AP was able to cluster Web services with higher performance with the pre-trained Word2Vec model. Euclidean, Minkowski, and WMD measures provide better performance than Cosine. Despite that AP predicted two clusters instead of three clusters as in the ground truth, it was able to get the relation between the Airlines Services cluster and Travel Services cluster. This is due to the similar and common words between the two clusters in addition to the close connection between these Web services. AP clustering based on pre-trained GloVe word embedding performs well when the WMD measure is employed. When considering the self-trained word embedding model, the performance of AP clustering is not as good as the other two models. Not all the semantic similarity measures based on AP perform as expected. Considering WordNet path similarity and NGD to cluster Web services did not perform well compared to the other measures. On the other hand, the WordNet WUP measure performs as well as the pre-trained Word2Vec word embedding model.

Table 6-5: Clustering Evaluation of Selected Web Services

Clustering Algorithms	**Affinity Propagation**				**Hierarchical Agglomerative Clustering**				**K-Means**			
Syntactic Similarity	***Cosine***	***Euclidean***	***Minkowski***	***WMD***	***Cosine***	***Euclidean***	***Minkowski***	***WMD***	***Cosine***	***Euclidean***	***Minkowski***	***WMD***
BOW with TFIDF												
#Clusters	3	3	3		3	3	3		3	3	3	
Agreement	5	5	5		6	6	4		6	6	6	
Disagreement	3	3	3		2	2	4		2	2	2	
Accuracy	0.63	0.63	0.63		0.75	0.75	0.5		0.75	0.75	**0.75**	
Precision	0.75	0.75	0.75		0.78	0.78	0.38		**0.92**	**0.92**	**0.81**	
Recall	0.62	0.62	0.62		0.75	0.75	0.5		0.75	0.75	**0.75**	
F-measure	0.68	0.68	0.68		0.75	0.75	0.42		**0.79**	**0.79**	**0.76**	
Pre-trained Word2Vec												
#Clusters	3	2	2	2	3	3	3	3	3	3	3	3
Agreement	6	7	7	7	5	5	5	6	7	6	5	7
Disagreement	2	1	1	1	3	3	3	2	1	2	3	1
Accuracy	0.75	**0.875**	**0.875**	0.875	0.625	0.625	0.625	0.75	**0.878**	0.75	0.625	0.875
Precision	0.92	0.78	0.78	0.78	0.91	0.69	0.69	0.92	**0.94**	**0.92**	0.78	**0.94**
Recall	0.75	**0.88**	**0.88**	0.82	0.62	0.62	0.62	0.75	**0.88**	0.75	0.62	**0.88**
F-measure	0.79	**0.82**	**0.82**	0.82	0.67	0.64	0.64	0.79	**0.89**	0.77	0.65	**0.89**
Pre-trained GloVe												
#Clusters	3	3	3	2	3	3	3	3	3	3	3	3
Agreement	6	6	6	7	6	6	6	6	6	5	5	7
Disagreement	2	2	2	1	2	2	2	2	2	3	3	1
Accuracy	0.75	0.75	0.75	0.875	0.75	0.75	0.75	0.75	0.75	0.625	0.625	0.875
Precision	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.92	0.78	0.78	0.78	**0.94**
Recall	0.75	0.75	0.75	0.88	0.75	0.75	0.75	0.75	0.75	0.62	0.62	0.88
F-measure	0.75	0.75	0.75	0.82	0.75	0.75	0.75	0.79	0.75	0.65	0.65	**0.89**
Self-Trained												

#Clusters	2	2	3	2	3	3	3	3	3	3	3	3
Agreement	6	6	6	6	5	5	6	6	5	7	6	6
Disagreement	2	2	2	2	3	3	2	2	3	1	2	2
Accuracy	**0.75**	0.75	0.75	0.75	0.625	0.625	0.75	0.75	0.625	**0.875**	0.75	0.75
Precision	**0.72**	0.72	0.92	0.72	0.69	0.69	0.92	0.92	0.69	**0.94**	0.92	0.92
Recall	**0.75**	0.75	0.75	0.75	0.62	0.62	0.75	0.75	0.62	**0.88**	0.75	0.75
F-measure	**0.71**	0.71	0.79	0.71	0.64	0.64	0.79	0.79	0.64	**0.89**	0.79	0.79
Semantic Similarity	**Path**	**WUP**	**NGD**		**Path**	**WUP**	**NGD**		**Path**	**WUP**	**NGD**	
#Clusters	3	2	3		3	3	3		3	3	3	
Agreement	6	7	6		6	5	6		6	5	6	
Disagreement	2	1	2		2	3	2		2	3	2	
Accuracy	0.75	**0.875**	0.75		0.75	0.625	0.75		0.75	0.625	0.75	
Precision	0.92	**0.88**	0.75		0.78	0.69	0.78		0.92	0.75	**0.92**	
Recall	0.75	**0.88**	0.75		0.75	0.62	0.75		0.75	0.62	0.75	
F-measure	0.75	**0.82**	0.75		0.75	0.64	0.75		**0.79**	0.68	**0.79**	

Considering HAC based on BOW with TFIDF performs better and sometimes the same compared to all word embedding models when Cosine and Euclidean measures are employed. However, word embedding models provide better performance when the WMD measure is employed. Among the three word embedding models, HAC was able to cluster Web services with higher performance with the pre-trained GloVe model. In general, employing Cosine, Euclidean, and WMD measures provide better performance than Minkowski. For semantic similarity measures, the WordNet path similarity measures and NGD perform the same in clustering Web services based on HAC while WordNet WUP similarity reports lower performance.

Considering K-means on BOW with TFIDF provides stable performance in all the similarity measures. The pre-trained Word2Vec model provided better performance in clustering Web services with both Cosine and WMD measures. The Word2Vec models perform better than the GloVe model. Among the syntactic similarities, WMD reports a better and more stable performance. For semantic similarity measures, both WordNet path similarity and NGD report a similar result, and WordNet WUP similarity reports lower performance.

Based on the overall performance, AP clustering performs better in clustering Web services and discovers hidden relations between Web services. On the other side, K-means provided a good and comparative performance that replicates the ground truth. Usually, the developed model will never be able to predict the exact ground truth due to noise in the model. The reality is that no model will give a hundred percent accuracy; nevertheless, we want the model to be as close as possible to the ground truth. The case here is that K-means

was able to predict Web services clusters that, to some level, align with the ground truth based on the number of clusters. AP was able to predict a smaller number of clusters and discover hidden relatedness between Web services beyond the ground truth. The performance of HAC is not as good as K-means and AP.

6.8. Summary

In this chapter, we proposed the multi-layer data mining architecture for Web services discovery to cluster Web services. The architecture is composed of 5 layers of Web services description and data preprocessing, word representation, embedding and transformation, syntactic similarity, semantic similarity, and Web services clustering layers. We considered eight randomly selected Web services for demonstration purposes as part of the DSRM activities. We studied and implemented three clustering algorithms to cluster Web services based on various similarity and dissimilarity matrices. Using a ground truth evaluation, we reported and discussed the performance of the clustering algorithms for clustering the eight selected Web services.

In the next chapter, we will run several experiments using a collected set of Web services description data. Various evaluation metrics will be utilized to evaluate Web services clustering's performance, followed by an analysis of the experimental results.

Chapter 7. Performance Evaluation and Analysis

This chapter presents the evaluation and analysis of the proposed multi-layer data mining Web services discovery architecture based on clustering Web services considering various models and similarity measures. The related research questions that we aim to consider in the evaluation process are: (a) can the use of different word representation, embedding, and transformation models help with increasing the clustering performance; (b) which one of the syntactic and semantic similarity measures provides better clustering performance; and (c) how do the models perform on the three clustering algorithms with different word representation and similarity considerations. To answer these questions, we first identify the evaluation criteria for measuring Web services clustering performance. We also specify the Web service datasets, ground truth, and experimental environment. We then run several experiments, measure and compare the performance of the models.

7.1. Web Services Clustering Performance Evaluation

Most of the studies in the area of Web services discovery systems rely on different evaluation methods to determine the effectiveness of their proposed solutions. As identifying the appropriate evaluation method is one of the requirements for a more effective assessment and evaluation of the proposed solution, we follow and use the evaluation methods and metrics by similar research as specified in our SLR (see Chapter 4) to measure the performance of Web services clustering. Our SLR identified various performance metrics that have been employed by researchers to measure the performance of Web services clustering in the proposed discovery and recommendation solutions.

To measure the performance of Web services discovery systems where clustering is part of the solution, an offline evaluation is usually employed to measure how the proposed solution facilitates Web services discovery. With its different models and methods, the proposed architecture facilitates the process of Web services discovery by focusing on clustering Web services based on different similarity measures, algorithms, and considerations prior to the matchmaking steps. We use offline experiments to cluster Web services

considering a collected dataset that contains Web services description files to measure the clustering performance based on the clustering algorithms.

To evaluate the performance, we consider a strategy that we developed due to the variety of evaluation metrics in the area of Web services clustering [31]. Our evaluation strategy is to find the agreements and disagreements between the ground truth and the Web services clusters resulting from our proposed architecture. This requires a ground truth that specifies each Web service classification based on their similarity, primarily employing human labeling for this task. This strategy is the most employed for evaluating Web services clustering performance where accuracy, recall, precision, and F-measure evaluation metrics are used. We need to find the best cluster-to-class association between the resulting clusters and the ground truth classes in this strategy.

The proposed architecture contains various models, similarity measures, and three clustering algorithms. We measure Web services clustering performance based on the different models and similarity measures by considering the selected clustering algorithm. For instance, we measure the performance of AP clustering based on 1) BOW with TFIDF, 2) pre-trained Word2Vec, 3) pre-trained GloVe, and self-trained Word2Vec models with Cosine distance as compared with the other syntactic and semantic similarity measures. We report all of the performance metrics based on the use of AP, HAC, and K-means clustering. Finally, we discuss the clustering algorithms' overall performance to identify the best algorithm for clustering Web services.

7.2. Datasets

One of the critical challenges in clustering Web services considering the proposed architecture is to find publicly available Web services description files that provide balanced, less sparse data. As part of the SLR in Chapter 4, we identified Web services collection methods and different Web services description files, with their location, that were cited in previous research. As the proposed architecture is configured to work with Web services description files based on text and WSDL, we scraped Web service documents, including text and WSDL files from PW and GitHub. WSDL files represent a machine-readable format, and text represents a human-readable format. In general, the Web services description format can be in any format and not restricted to WSDL and text. As stated in layer one of

the proposed architecture, the parser and tags identifier module identifies the required text (features) to be extracted from the Web services descriptions for the mining process.

The initially collected Web services dataset contains 459 human-readable files and 459 machine-readable (WSDL) files. We reviewed the Web service documents to identify any duplications and invalid WSDL formats. The overall collected Web services dataset was employed to train the Word2Vec model as part of the self-trained word embedding model used in layer two. We selected 102 Web service documents to evaluate the performance of clustering based on the proposed architecture. We manually labeled the 102 Web services into different domains and classes based on their functionality to serve as ground truth. The generated ground truth was utilized to measure Web services clustering performance by comparing the ground truth with the resulting clusters (agreements and disagreements) using external evaluation methods.

Note: The collected Web services dataset and the trained Word2Vec model can be found at https://github.com/waealobidallah/WS-Dataset-Models.

7.3. Experimental Environment

The experiments were conducted on Microsoft Windows 10 with an Intel Core i7-10510U 1.8GHz CPU and 16GB of RAM. Python was employed as the programming language. Various Python-based data- and text-mining libraries were utilized in the implementation of the proposed architecture. Some of the libraries utilized include NLTK [179], Gensim [199], and scikit-learn [200]. Modifications of some of the built-in functions were needed to control the input and the output of each layer of the proposed architecture.

7.4. Evaluation Metrics

Two evaluation strategies are employed in the literature: internal and external. For external evaluation, metrics are used to measure the agreement between the previously known clustering structure (ground truth) and results from the clustering procedures. For internal evaluation, metrics are utilized to measure the goodness of clusters without the need of external information such as ground truth. We followed the external evaluation strategy by employing accuracy, precision, recall, and F-measure based on the developed ground truth.

Accuracy measures the overall performance of a model and is calculated as the number of correct predictions (agreements) divided by the total number of predictions, as shown in equation 7.1. However, accuracy alone is not always a good performance indicator, especially in the case of an imbalanced dataset.

$$\boldsymbol{Accuracy}\left(\boldsymbol{C_i, CL_j}\right) = \frac{\boldsymbol{number\ of\ members\ of\ C_i\ in\ CL_j}}{\boldsymbol{total\ number\ of\ members}} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+TN+FP+FN}} \quad \textbf{(7.1)}$$

where C_i represents the class, and CL_j is the cluster.

Precision measures exactness and quality by showing how many Web services were predicted correctly out of the ones that were predicted as belonging to a given cluster. Precision takes the number of Web services that were correctly predicted by the model to be in a given cluster and divides that by the total number of Web services that were predicted, correctly and incorrectly, to belong to the cluster, as shown in equation 7.2.

$$\boldsymbol{Precision}\left(\boldsymbol{C_i, CL_j}\right) = \frac{\boldsymbol{number\ of\ members\ of\ C_i\ in\ CL_j}}{\boldsymbol{number\ of\ members\ of\ CL_j}} = \frac{\boldsymbol{Successfully\ (CL_j)}}{\boldsymbol{Successfully\ (CL_j)+Misplaced(CL_j)}} \quad \textbf{(7.2)}$$

Recall measures the completeness, correctness, and quantity by showing how many Web services were predicted correctly out of the ones that should have been predicted to belong to a given cluster. Recall takes the number of Web services that were correctly predicted for a given cluster and divides that by the number of Web services that were predicted correctly to belong to this cluster and Web services that were incorrectly predicted as not belonging to this cluster, as shown in equation 7.3.

$$\boldsymbol{Recall}\left(\boldsymbol{C_i, CL_j}\right) = \frac{\boldsymbol{number\ of\ members\ of\ C_i\ in\ CL_j}}{\boldsymbol{number\ of\ members\ of\ C_i}} = \frac{\boldsymbol{Successfully\ (CL_j)}}{\boldsymbol{Successfully\ (CL_j)+Missed(CL_j)}} \quad \textbf{(7.3)}$$

F-measure is a harmonic mean of precision and recall, as shown in equation 7.4, which provides a better indicator of the clustering algorithm performance by considering both misplaced and missed Web services.

$$\boldsymbol{F-measure}\left(\boldsymbol{C_i, CL_j}\right) = \boldsymbol{2}.\frac{\boldsymbol{Precision}\left(\boldsymbol{C_i,CL_j}\right)\times\ \boldsymbol{Recall}\left(\boldsymbol{C_i,CL_j}\right)}{\boldsymbol{Precision}\left(\boldsymbol{C_i,CL_j}\right)+\boldsymbol{Recall}\left(\boldsymbol{C_i,CL_j}\right)} \quad \textbf{(7.4)}$$

7.5. Experiments and Discussions

We ran five experiments to demonstrate how the proposed artifacts of Web services clustering can provide improvement by employing various word representations models, and syntactic and semantic similarity measures in the process of Web services discovery.

- In the first experiment, we cluster Web services based on AP, K-means, and HAC clustering algorithms with BOW with TFIDF as the word representation, and Cosine, Euclidean, and Minkowski as syntactic similarity measures.
- In the second experiment, we cluster Web services based on AP, K-means, and HAC clustering algorithms with pre-trained Word2Vec as the word representation and embedding model, and Cosine, Euclidean, Minkowski, and Word Mover as syntactic similarity measures.
- In the third experiment, we cluster Web services with the pre-trained GloVe as the word representation and embedding model, and Cosine, Euclidean, Minkowski, and Word Mover as syntactic similarity measures.
- In the fourth experiment, we use self-trained Word2Vec as the word representation and embedding model, and Cosine, Euclidean, Minkowski, and Word Mover as syntactic similarity measures.
- In the fifth experiment, we cluster Web services considering the WordNet path, WordNet WUP, and NGD as semantic similarity measures.

In each set of experiments, the evaluation metrics are employed to measure the performance of the clustering based on the collected dataset. As a quality assurance measure, we run and test the experiments using the same script for each layer instead of running the scripts all at once.

7.5.1 Experiment I: BOW with TFIDF Model

This experiment aims to measure clustering performance when the BOW with TFIDF model is employed in the word representation, embedding, and transformation layer. We measured the clustering performance, considering the syntactic similarity measures and the clustering algorithms. The result indicates that, among the clustering algorithms, AP provided better performance in clustering Web services by showing more agreements with the ground truth, as illustrated in Figure 7-1. With Cosine, Euclidean, and Minkowski, almost

similar accuracy and F-measure scores were reported for AP clustering. Both K-means and HAC performed poorly in clustering Web services when compared to the ground truth. K-means performed better than HAC based on two of the three syntactic similarity measures.

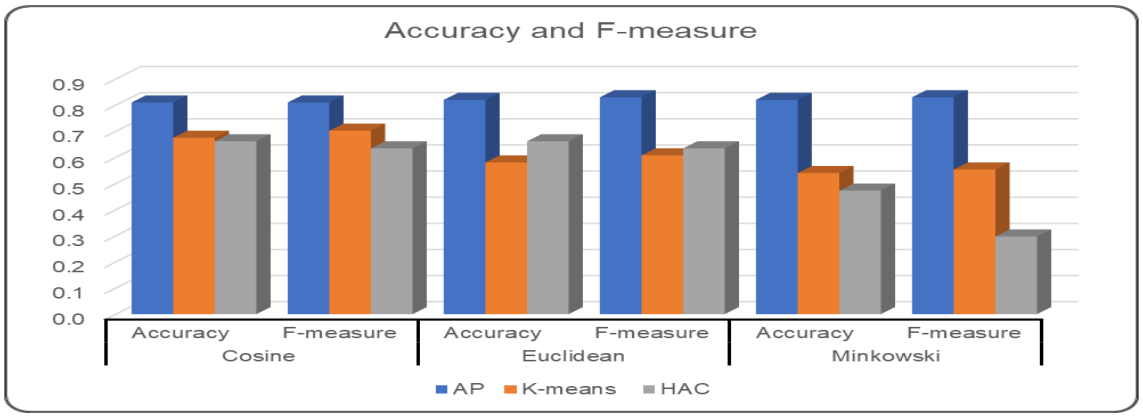


Figure 7-1: Accuracy and F-measure for Experiment I (BOW with TFIDF Model)

7.5.2 Experiment II: Pre-trained Word2Vec Model

This experiment was conducted to measure the performance of clustering when the pre-trained Word2Vec model is employed. Four syntactic similarity measures are employed to cluster Web services with AP, K-means, and HAC. The result indicated that AP performed better in clustering Web services based on the syntactic similarities methods. Furthermore, considering a specialized similarity measure for dense vector representations such as the Word Mover's Distance (WMD), Web service clustering performance improved compared to Cosine, Euclidean, and Minkowski similarity measures. In Figure 7-2, we report the accuracy and F-measure performance metrics.

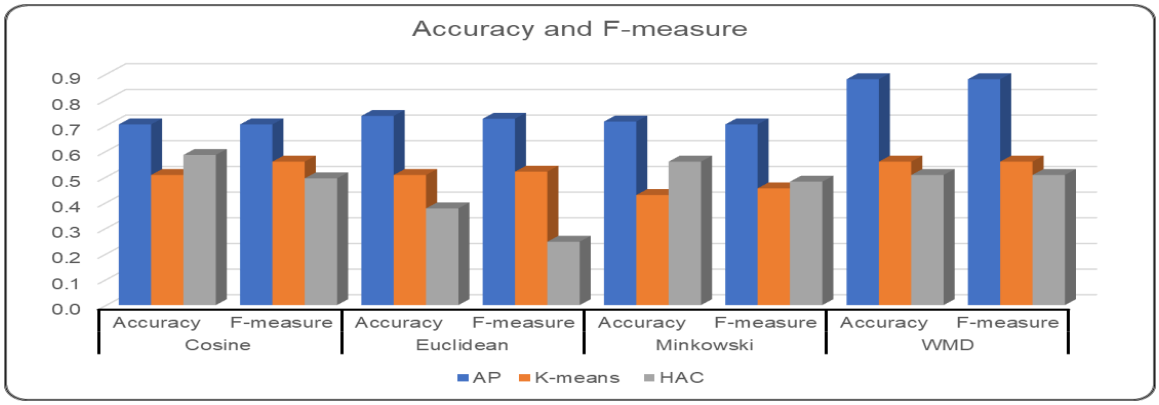


Figure 7-2: Accuracy and F-measure for experiment II (Pre-trained Word2Vec model)

7.5.3 Experiment III: Pre-trained GloVe Model

This experiment measured the clustering performance when the pre-trained GloVe model is employed. The four syntactic similarity measures were employed to cluster Web services with AP, K-means, and HAC. In this experiment, AP performed better in clustering Web services based on the syntactic similarities measures. By considering WMD, Web service clustering performance improved compared to Cosine, Euclidean, and Minkowski similarity measures. We report accuracy and F-measure metrics for the performance evaluation in Figure 7-3.

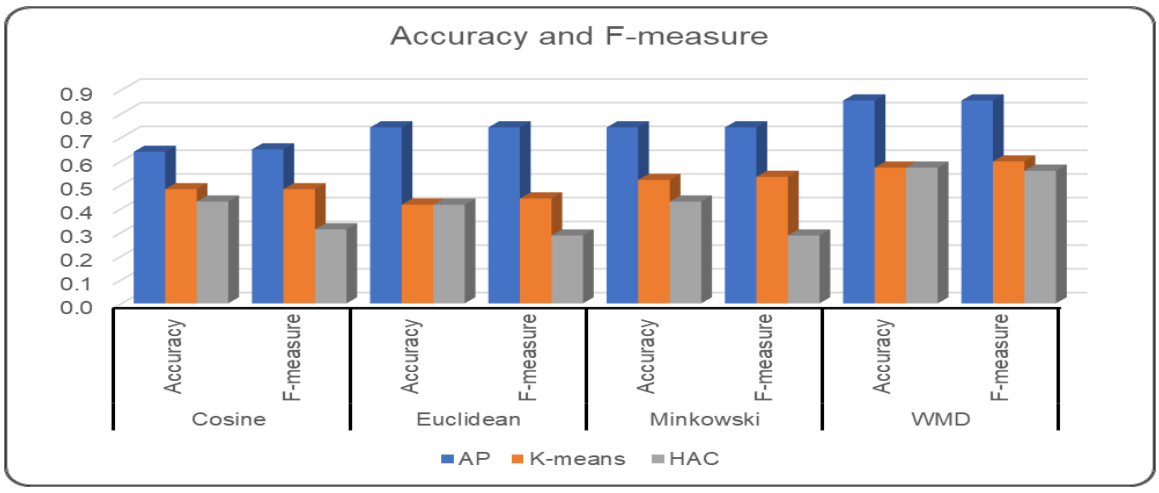


Figure 7-3: Accuracy and F-measure for Experiment III (Pre-trained GloVe Model)

7.5.4 Experiment IV: Self-trained Word2Vec Model

We conducted this experiment to measure Web services clustering performance when the self-trained Word2Vec model is employed. In this experiment, four syntactic similarity measures were employed to cluster Web services by employing AP, K-means, and HAC clustering algorithms. Based on the reported performance metric, as illustrated in Figure 7-4, AP performed better in clustering Web services based on the syntactic similarity measures. Furthermore, Web services clustering performance improved compared to the other similarity measures when WMD is employed. K-means performed better than HAC across all syntactic similarity measures.

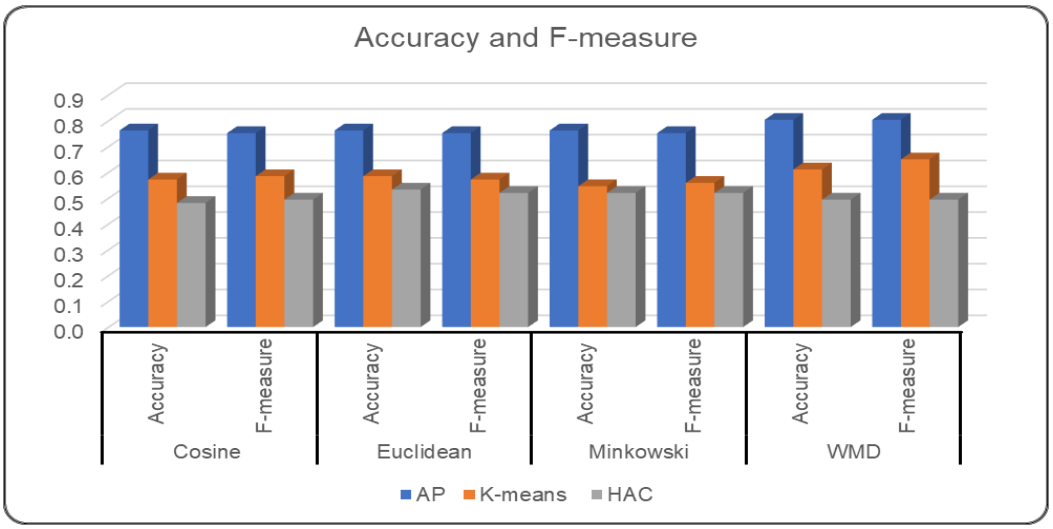


Figure 7-4: Accuracy and F-measure for Experiment IV (Self-trained Word2Vec Model)

7.5.5 Experiment V: WordNet and NGD

The goal of this experiment is to measure the performance of clustering when semantic similarity measures are employed. WordNet's Path and WUP and NGD are employed as semantic similarity measures to cluster Web services with AP, K-means, and HAC. The result indicated that AP performs better in clustering Web services based on the employed semantic similarities. WordNet Path similarity provides a better similarity measure to cluster Web services among the three semantic similarity measures. In Figure 7-5, we reported accuracy and F-measure to evaluate the clustering performance.

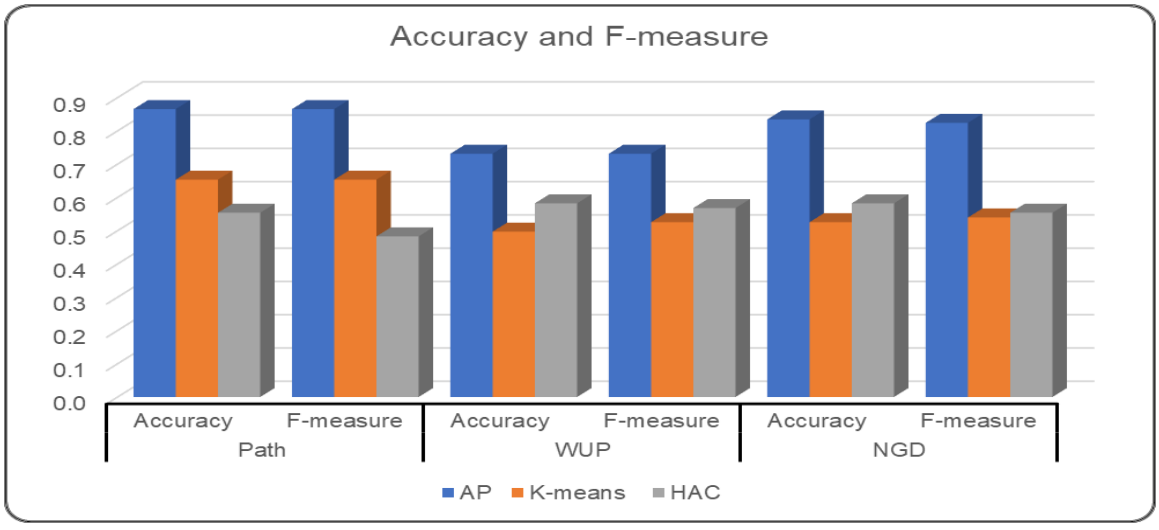


Figure 7-5: Accuracy and F-measure for Experiment V (WordNet and NGD)

7.5.6 Summary of the Experiments and Conclusions

Based on the conducted experiments, we demonstrated that the clustering of Web services is best aligned with the ground truth when the AP clustering algorithm is employed, showing high accuracy and F-measure. Table 7-1 provides more details about clustering performance among the employed word representation, embedding, and transformation models; syntactic and semantic similarity measures; and clustering algorithms.

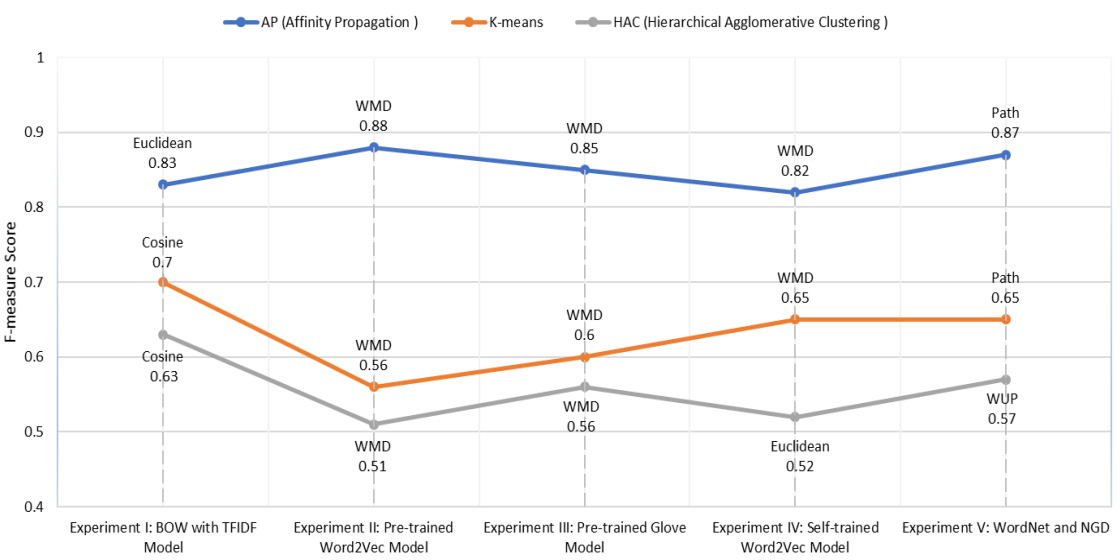


Figure 7-6: The Best F-measure Scores of the Five Experiments using Different Clustering Techniques and Similarity Measures. WMD (Word Mover's Distance), WUP(Wu-Palmer Semantic Similarity)

Referring to experiment I in Figure 7-6, AP with the Euclidean similarity measure reported the best performance considering the various syntactic similarity measures (bold font, moving from left to right). Referring to experiment II, AP with the WMD reported the best performance with high accuracy, precision (minimized false positive), recall (minimized false negative), and F-measure. In experiment III and experiment IV as well, considering the WMD with AP improved the clustering performance. In experiment V, employing WordNet's path similarity reported the best performance for AP clustering. Word embedding models reported the best result in clustering Web services when specialized similarity were employed, such as WMD.

While the AP clustering algorithm has not been studied in previous research to cluster Web services documents, as indicated in [31], we achieved good performance in clustering Web services compared to other clustering algorithms such as K-means and

HAC. Based on the same datasets and configurations as part of the proposed multi-layer data mining architecture, we also reported K-means' and HAC's performance in clustering Web service documents.

Table 7-1: Performance Evaluation for Clustering Web services(WUP(Wu-Palmer Semantic Similarity), WMD(Word Mover's Distance), NGD(Normalized Google Distance))

Clustering Algorithms	**Affinity Propagation**				**Hierarchical Agglomerative Clustering**				**K-Means**			
Syntactic Similarity	***Cosine***	***Euclidean***	***Minkowski***	***WMD***	***Cosine***	***Euclidean***	***Minkowski***	***WMD***	***Cosine***	***Euclidean***	***Minkowski***	***WMD***
Experiment I: BOW with TFIDF Model												
Accuracy	**0.81**	**0.82**	**0.82**		0.66	0.66	0.47		0.68	0.58	0.54	
Precision	0.83	**0.84**	0.84		0.81	0.81	**0.89**		**0.86**	0.77	0.65	
Recall	**0.81**	**0.82**	**0.82**		0.66	0.66	0.46		0.68	0.58	0.54	
F-measure	**0.81**	**0.83**	**0.83**		0.63	0.63	0.30		0.70	0.61	0.55	
Experiment II: Pre-trained Word2Vec Model												
Accuracy	**0.70**	**0.74**	**0.72**	**0.88**	0.59	0.38	0.56	0.51	0.51	0.51	0.43	0.56
Precision	**0.76**	**0.78**	**0.76**	**0.91**	0.70	0.68	0.70	0.61	0.73	0.70	0.57	0.73
Recall	**0.70**	**0.74**	**0.72**	**0.88**	0.59	0.38	0.56	0.51	0.51	0.51	0.43	0.56
F-measure	**0.70**	**0.73**	**0.70**	**0.88**	0.49	0.25	0.48	0.51	0.56	0.52	0.46	0.56
Experiment III: Pre-trained Glove Model												
Accuracy	**0.64**	**0.74**	**0.74**	**0.85**	0.48	0.42	0.52	0.57	0.43	0.42	0.43	0.57
Precision	**0.73**	0.79	0.79	**0.88**	0.59	0.53	0.62	0.77	**0.81**	**0.81**	0.81	0.61
Recall	**0.64**	**0.74**	**0.74**	**0.85**	0.48	0.42	0.52	0.57	0.43	0.42	0.43	0.56
F-measure	**0.65**	**0.74**	**0.74**	**0.85**	0.48	0.44	0.53	0.60	0.31	0.29	0.29	0.56
Experiment IV: Self-trained Word2Vec Model												
Accuracy	**0.76**	**0.76**	**0.76**	**0.80**	0.48	0.53	0.52	0.49	0.57	0.59	0.55	0.61
Precision	**0.80**	**0.82**	**0.81**	**0.82**	0.60	0.61	0.61	0.60	0.73	0.68	0.72	0.81
Recall	**0.76**	**0.76**	**0.76**	**0.80**	0.48	0.53	0.52	0.49	0.57	0.59	0.55	0.61
F-measure	**0.75**	**0.75**	**0.75**	**0.80**	0.49	0.52	0.52	0.49	0.59	0.57	0.56	0.65
Experiment V: WordNet and NGD												
Semantic Similarity	**Path**	**WUP**	**NGD**		**Path**	**WUP**	**NGD**		**Path**	**WUP**	**NGD**	
Accuracy	**0.87**	**0.73**	**0.83**		0.55	0.58	0.58		0.65	0.50	0.53	
Precision	**0.89**	**0.78**	**0.84**		0.55	0.67	0.75		0.84	0.67	0.71	
Recall	**0.85**	**0.72**	**0.82**		0.55	0.58	0.58		0.65	0.50	0.53	
F-measure	**0.87**	**0.73**	**0.82**		0.48	0.57	0.55		0.65	0.53	0.54	

7.6. Summary

This chapter evaluates the performance of the proposed multi-layer data mining architecture for Web services discovery based on clustering Web services. We employed external evaluation methods, which require the existence of ground truth. We reported accuracy, recall, precision, and F-measure of the clustering performance based on the five different experiments. The result indicates that clusters built based on the word embedding models perform better than clusters built based on Bag of Words with TFIDF models when the right similarity measure and corpus are employed. Among the three word embedding models, the pre-trained Word2Vec model reported higher performance in clustering Web services. The WMD based on the word embedding model increases clustering performance compared to other syntactic measures. Among the three semantic similarity measures, WordNet path similarity reported higher clustering performance. AP performed better in clustering Web services and discovered hidden relations between previously assigned clusters among the clustering algorithms.

In the next chapter, we conclude the dissertation by summarizing the research, highlighting the contributions, and elaborating on the future work.

Chapter 8. Conclusions and Future Work

Considering the importance of Web services in E-business, where different businesses share their capabilities and functions to enhance their business models, software developers look for Web services that meet their functional and non-functional requirements. In this thesis, with the intention of facilitating the process of Web services discovery and recommendation, using a combination of theory and experimental approaches, we addressed the problem of Web services discovery and recommendation where searching for the right Web service takes a long time due to the ample search space. Data mining, text mining, and machine learning techniques provide solutions to facilitate the process of Web services discovery and recommendation when algorithms for clustering, classification, and association rules are applied to minimize the search space.

Over the past decades, several clustering and association rules techniques have been proposed to facilitate the process of Web services discovery and recommendation. Thus, there is a need to investigate the methods, characteristics, and details of these techniques with a systematic method and quality requirements. Furthermore, the lack of an existing comprehensive literature review motivated us to conduct a systematic literature review. We systematically identified, summarized, and compared existing clustering and association rules techniques for Web services discovery and recommendation by providing answers to a set of research questions. We explored various algorithms, similarity measures, datasets, and evaluation metrics applied in Web services discovery and recommendation approaches. We identified that the most employed clustering and association rules algorithms are the K-means and the Apriori algorithms, respectively. Semantic measures considering WordNet are the most employed among the similarity measures. Among the published benchmarks, OWLS-TC is the most cited benchmark for discovery purposes, and WS-DREAM is the most cited for recommendation purposes. We discussed the trends and future directions in three areas.

The diversity of Web services discovery systems, methods, and techniques from the vast literature makes it interesting and enticing to investigate all of them. Furthermore,

the lack of solutions that provide an adequate explanation and description on the conceptual level of Web services discovery systems motivated us to design and develop a conceptual model and typology of Web services discovery systems. The proposed conceptual model helps with the process of developing Web services discovery systems by providing and investigating their different components, functionalities, and interrelationships to provide a better understanding of Web services discovery systems. The proposed typology is based on five identified characteristics: location and storage, formalization, matchmaking, automation, and selection. Subsequently, we utilized the typology to compare various Web services discovery approaches discussed in the extant literature. We demonstrated how the proposed conceptual model and the identified characteristics could be utilized to build a comprehensive Web services discovery system. We concluded that centralized approaches as part of storage and location characteristics dominate despite their limitations; semantic enabling of Web services as part of formalization characteristics are the most investigated, especially OWL-S and WSMO; hybrid systems with syntactic and semantic matching as part of the matchmaking characteristics have become more prevalent in recent years; manual discovery approaches from the automation characteristics are more prevalent; and, finally, considering both functional and non-functional parameters from the selection characteristics provide a better solution if the non-functional parameters are available.

As stated, data mining, text mining, and machine learning can provide solutions to facilitate the process of Web services discovery. We proposed a multi-layer data mining architecture for Web services discovery to cluster Web services based on five layers: Web services description and data preprocessing; word representation, embedding, and transformation; syntactic similarity; semantic similarity; and Web services clustering. We utilized BOW with TFIDF and three word-embedding models for the word representation, embedding, and transformation layers. Four syntactic similarity measures (Cosine, Euclidean, Minkowski, and Word's Mover) and three semantic similarity measures (WordNet Path, WordNet WUP, and NGD) are employed to estimate the similarity between Web services. Finally, three clustering algorithms (AP, K-means, and HAC) were studied to cluster Web services based on their similarities. We run five different experiments considering the different employed layers. According to the agreements and disagreements with the ground truth, the clustering performance was reported based on evaluation metrics,

including accuracy, precision, recall, and F-measure. The results indicated that clusters built based on the word embedding models perform better than clusters built based on Bag of Words with TFIDF models. Among the three word embedding models, the pre-trained Word2Vec model reported higher performance in clustering Web services. Considering the WMD based on word embedding models increased clustering performance compared to other syntactic measures. Among the three semantic similarity measures, WordNet path similarity reports higher clustering performance. AP performed better in clustering Web services and discovers hidden relations between previously assigned clusters among the clustering algorithms.

As for the implications for Web services developers and practitioners, Web services need to be described based on the new and popular description languages to be discovered effectively. Swagger [42][43], RAML [44], and API Blueprint [47] have made significant improvements over WSDL or WADL. Swagger (i.e., OpenAPI Specification) is one of the most adopted description languages, which uses a simple JSON or YAML to describe REST-based Web services concerning the HTTP methods, media types, data types, and status code [43].

Today, the most common syntax description approaches for Web services are WSDL and OpenAPI Specification. WSDL is universal because it is the standard way to describe SOAP-based Web services, which makes it inevitable regardless of its limitations. OAS is the most common REST-based Web service description language among other description languages (other specifications) due to its simplicity, supporting tools, data formats in addition to organizations and community support.

8.1. Summary of Contributions

We contributed to the descriptive and prescriptive knowledge base of Web services discovery and recommendation. The contributions of the thesis can be summarized as follows:

a) We conducted a systematic literature review (SLR) of clustering and association rules techniques for Web services discovery and recommendation. This review was published in [31]. It aims to identify and compare existing techniques, methods, and algorithms applied in the process of Web services discovery and recommendation.

Furthermore, we identified the benchmark data collections for validating the proposed methods. We also discussed the syntactic and semantic similarities for each proposed method. We introduced a classification of Web services discovery and recommendation techniques from the literature where clustering and association rules are employed in their studies. The most cited algorithms related to the various similarity measures are specified and discussed. We also categorized the validation and evaluation metrics for Web services discovery and recommendation based on the selected studies. Furthermore, the trends and future research directions are discussed as part of the SLR. Future researchers can utilize the results of the synthesis in this study to learn about research gaps, different algorithms, similarity measures, and datasets employed in Web services discovery and recommendation. This published SLR contributed to the descriptive knowledge, as discussed in Section 3.3.

b) We structured a conceptual model and a typology of Web services discovery approaches based on five identified characteristics, including storage and location, formalization, matchmaking, automation, and selection characteristics. This study was published in [14, 27]. The conceptual model identifies the different elements and components of Web services discovery systems. Our proposed typology is derived from the observation of the wide-ranging literature about Web services discovery systems. The typology provides a big picture of where researchers conducted their investigation to improve the process of Web services discovery. Furthermore, it provided the basis of building a meta-Web services discovery system by incorporating the different characteristics. We conducted a comparison of different Web services discovery mechanisms based on the proposed typology by referring to the identified characteristics.

c) We designed and verified a multi-layer data mining architecture for Web services discovery based on clustering Web services to speed up the matchmaking process where we focused on the matchmaking characteristics of the proposed typology (the manuscript is being submitted). The proposed architecture consists of five layers: Web service description and data preprocessing; representation, embedding, and

transformation; syntactic similarity; semantic similarity; and clustering. The implementation and performance evaluation are discussed using appropriate evaluation measures. The aim is to cluster Web services based on different similarities to minimize the search space and measure the performance. The results indicate that considering the word embedding models combined with Affinitive Propagation (AP) clustering improved clustering performance compared to the K-means and HAC clustering algorithms, which were the most cited in the literature.

d) We proposed exploiting the Affinity propagation (AP) clustering algorithm to cluster Web services based on various syntactic and semantic similarity measures, which reported higher performance than the most cited clustering algorithms (baseline) in the literature for clustering Web services.

8.2. Limitations and Future Work

The scope of Web services discovery and recommendation is enormous and rich with opportunities to improve and expand the proposed solution in a real-world environment. One potential enhancement is to develop a fully functional system for Web services discovery and recommendation. The five identified characteristics of Web services discovery systems can be utilized in developing the discovery systems. One of the limitations of the proposed solutions, as discussed in Section 3.2, focuses on applying data mining solutions to cluster Web services instead of building a Web services discovery system. Although we discussed the area of Web service recommendation as part of the conducted SLR, we did not provide a data mining solution to cluster Web services to improve the recommendation process, which is an opportunity for future research.

In the proposed multi-layer data mining architecture, we employed three word embedding models, including two pre-trained and one self-trained models. However, the self-trained Word2Vec model's training process was based on extracted text from a limited number of Web services compared to the two pre-trained models. Training the Word2Vec model based on extensive Web services extracted text would improve the performance of finding the similarities between Web services. However, the performance of clustering based on the self-trained word embedding model is comparative to the other models.

We clustered Web services based on various similarities where each Web service should belong to one cluster. However, a Web service can belong to more than one cluster. The possibility to find a clustering algorithm or a method that can place a Web service into more than one cluster is an opportunity for future work. Finding the correlation and relationship between Web services based on previous Web services composition is another opportunity for future work.

We evaluated Web services clustering performance based on external evaluation metrics, which requires the existence of ground truth. We collected Web services documents and employed human labeling to build the ground truth. Although the collected dataset is based on WSDL and auxiliary text, the proposed solution can work with any format of Web services description language or documentation that needs to be evaluated and tested for external validity.

The proposed multi-layers architecture focused on clustering Web services based on finding the similarity between Web services where querying can be used on top of the clustered Web services. Querying of the generated clusters was not included in the proposed solution which worth investigating in future work. Furthermore, using Web services' non-functional requirements in building a Web services recommendation system is a potential improvement in future work.

References

[1] M. Crasso, A. Zunino, and M. Campo, "Easy web service discovery: A query-by-example approach," *Sci. Comput. Program.*, vol. 71, no. 2, pp. 144–164, Apr. 2008.

[2] H. Haas and A. Brown, "Web Services Glossary," *World Wide Web Consortium*, 2004. http://www.w3.org/TR/ws-gloss/ (accessed Oct. 03, 2017).

[3] L. D. Ngan, A. Goh, and C. H. Tru, "A Survey of Web Service Discovery Systems," *Int. J. Inf. Technol. Web Eng.*, vol. 2, no. 2, pp. 65–80, 2007.

[4] M. Klusch, "Service Discovery," *Encycl. Soc. Netw. Anal. Min.*, pp. 1707–1717, 2014.

[5] J. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. Tsakalidis, "Web Service Discovery Mechanisms: Looking for a Needle in a Haystack?," in *International Workshop on Web Engineering*, 2004, vol. 5, no. 3, pp. 265–290.

[6] M. Zhang, X. Liu, R. Zhang, and H. Sun, "A Web Service Recommendation Approach Based on QoS Prediction Using Fuzzy Clustering," in *2012 IEEE Ninth International Conference on Services Computing*, Jun. 2012, pp. 138–145.

[7] K. Hashmi, A. Alhosban, Z. Malik, B. Medjahed, and S. Benbernou, "Automated Negotiation Among Web services," in *Web Services Foundations*, New York, NY, NY: Springer New York, 2014, pp. 451–482.

[8] L. Clement, A. Hately, C. Von Riegen, and T. Rogers, "UDDI Version 3.0.2," *OASIS UDDI Spec Technical Committee Draft*, 2004. http://www.uddi.org/pubs/uddi_v3.htm (accessed Nov. 02, 2016).

[9] H. Lausen and T. Haselwanter, "Finding Web Services," in *Proc. of the 1st European Semantic Technology Conference (ESTC)*, 2007, pp. 1–7.

[10] J. Ma, Y. Zhang, and J. He, "Efficiently finding web services using a clustering semantic approach," in *Proceedings of the 2008 international workshop on Context enabled source and service selection, integration and adaptation organized with the 17th International World Wide Web Conference (WWW 2008) - CSSSIA '08*, 2008, vol. 292, pp. 1–8.

[11] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan, "Automated discovery, interaction and composition of Semantic Web services," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 1, no. 1, pp. 27–46, Dec. 2003.

[12] C. Ma, M. Song, K. Xu, and X. Zhang, "Web Service discovery research and implementation based on semantic search engine," in *2010 IEEE 2nd Symposium on Web Society*, Aug. 2010, pp. 672–677.

[13] M. K. Nair and V. Gopalakrishna, "Look Before You Leap: A Survey of Web Service Discovery," *Int. J. Comput. Appl.*, vol. 7, no. 5, pp. 22–30, Sep. 2010.

[14] W. J. Obidallah, U. Ruhi, and B. Raahemi, “Current Landscape of Web Service Discovery: A Typology Based on Five Characteristics,” in *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, Oct. 2016, pp. 678–683.

[15] D. Bianchini, P. Garza, and E. Quintarelli, “Characterization and search of web services through intensional knowledge,” *J. Intell. Inf. Syst.*, vol. 47, no. 3, pp. 375–401, Dec. 2016.

[16] B. K. Holley, *Rethinking the way business is done*. IBM Academy of Technology, 2014.

[17] W.-D. Zhu *et al.*, *Exposing and Managing Enterprise Services with IBM API Management*. IBM Redbooks, 2014.

[18] J. M. Rodriguez, A. Zunino, C. Mateos, F. O. Segura, and E. Rodriguez, “Improving REST Service Discovery with Unsupervised Learning Techniques,” in *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, Jul. 2015, pp. 97–104.

[19] W. Liu and W. Wong, “Web service clustering using text mining techniques,” *Int. J. Agent-Oriented Softw. Eng.*, vol. 3, no. 1, p. 6, 2009.

[20] K. Elgazzar, A. E. Hassan, and P. Martin, “Clustering WSDL documents to bootstrap the discovery of Web services,” in *ICWS 2010 - 2010 IEEE 8th International Conference on Web Services*, Jul. 2010, pp. 147–154.

[21] Sowmya Kamath S. and Ananthanarayana V.S., “Similarity analysis of service descriptions for efficient Web service discovery,” in *2014 International Conference on Data Science and Advanced Analytics (DSAA)*, Oct. 2014, pp. 142–148.

[22] G. Tian, W. Jian, K. He, and C. Sun, “Leveraging Auxiliary Knowledge for Web Service Clustering,” *Chinese J. Electron.*, vol. 25, no. 5, pp. 858–865, Sep. 2016.

[23] B. Kumara, I. Paik, T. H. A. S. . Siriweera, and K. R. C. Koswatte, “Cluster-Based Web Service Recommendation,” in *2016 IEEE International Conference on Services Computing (SCC)*, Jun. 2016, pp. 348–355.

[24] P. He, J. Zhu, J. Xu, and M. R. Lyu, “A Hierarchical Matrix Factorization Approach for Location-Based Web Service QoS Prediction,” in *2014 IEEE 8th International Symposium on Service Oriented System Engineering*, Apr. 2014, pp. 290–295.

[25] X. Chen, Z. Zheng, Q. Yu, and M. R. Lyu, “Web Service Recommendation via Exploiting Location and QoS Information,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1913–1924, Jul. 2014.

[26] J. Zhu, Y. Kang, Z. Zheng, and M. R. Lyu, “A Clustering-Based QoS Prediction Approach for Web Service Recommendation,” in *2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, Apr. 2012, pp. 93–98.

[27] W. J. Obidallah and B. Raahemi, “A Taxonomy to Characterize Web Service Discovery Approaches, Looking at Five Perspectives,” in *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, Mar. 2016, pp. 458–459.

[28] W. Obidallah, “Business Process and Service Change Management in Service Oriented Virtual Organizations,” Université d’Ottawa/University of Ottawa, 2013.

[29] W. J. Obidallah and B. Raahemi, “Managing Changes in Service Oriented Virtual Organizations,” *J. Electron. Commer. Organ.*, vol. 15, no. 1, pp. 59–83, Jan. 2017.

[30] W. J. Obidallah, B. Raahemi, S. M. Amin Kamali, and M. H. Danesh, “Service oriented virtual organizations: A service change management perspective,” in *2013 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, May 2013, pp. 1–6.

[31] W. J. Obidallah, B. Raahemi, and U. Ruhi, “Clustering and Association Rules for Web Service Discovery and Recommendation: A Systematic Literature Review,” *SN Comput. Sci.*, vol. 1, no. 1, p. 27, Jan. 2020.

[32] W. J. Obidallah and B. Raahemi, “A survey on web service discovery approaches,” in *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing - ICC ’17*, 2017, pp. 1–8.

[33] F. Firozbakht, W. J. Obidallah, and B. Raahemi, “Cloud computing service discovery framework for IaaS and PaaS models,” in *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing - ICC ’17*, 2017, pp. 1–6.

[34] M. M. Mohammadi, B. Raahemi, F. Cheraghchi, W. J. Obidallah, and E. Bigdeli, “Big data analytics using hadoop,” in *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*, 2014, pp. 323–325.

[35] T. Erl, *SOA principles of service design*. Prentice Hall: The Prentice Hall Service-Oriented Computing Series, 2007.

[36] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” Doctoral dissertation, University of California, Irvine, 2000.

[37] P. K. Potti, S. Ahuja, K. Umapathy, and Z. Prodanoff, “Comparing Performance of Web Service Interaction Styles: SOAP vs. REST,” in *Proceedings of the Conference on Information Systems Applied Research ISSN*, 2012, vol. 2167, p. 1508.

[38] K. Elgazzar, “Discovery , Personalization and Resource Provisioning of Mobile Services, Ph.D Thesis,” *Queen’s Univ. Kingston, Ontario, Canada*, no. August, 2013.

[39] W3C, “Web Service Definition Language (WSDL) 1.1,” *W3C Working Group*, 2001. .

[40] T. Winograd, “Understanding natural language,” *Cogn. Psychol.*, 1972.

[41] M. J. Hadley, “Web Application Description Language,” *W3C Member Submission*, 2009. http://www.w3.org/Submission/wadl/ (accessed Oct. 05, 2017).

[42] “Swagger RESTful API Documentation Specification.” https://docs.swagger.io/spec.html (accessed Mar. 13, 2019).

[43] Swagger.io, “OpenAPI Specification | Swagger,” 2018. https://swagger.io/specification/ (accessed Mar. 13, 2019).

[44] RAML Workgroup, “RAML Version 1.0: RESTful API Modeling Language,” *GitHub*, 2017. https://github.com/raml-org/raml-spec/blob/master/versions/raml-10/raml-10.md/.

[45] L. Mandel, “Describe rest web services with wsdl 2.0,” *Rational Software Developer, IBM*, 2008. https://www.ibm.com/developerworks/library/ws-restwsdl/ (accessed Oct. 03, 2017).

[46] H. Jayathilaka, C. Krintz, and R. Wolski, “Service-Driven Computing with APIs,” in *Handbook of Research on Architectural Trends in Service-Driven Computing*, IGI Global, 2014, pp. 355–379.

[47] Apiblueprint.org, “API Blueprint,” 2013. https://apiblueprint.org/documentation/ (accessed Mar. 19, 2019).

[48] Raml.org, “RAML.” https://raml.org/about-raml (accessed Mar. 21, 2019).

[49] D. Geeraerts, *Theories of Lexical Semantics*. 2010.

[50] T. Slimani, “Semantic Description of Web Services,” *arXiv Prepr. arXiv1310.7367*, vol. 10, no. 1, pp. 368–377, 2013.

[51] W3.org, “Semantic Web - W3C,” *W3.org*, 2015. http://www.w3.org/standards/semanticweb/ (accessed Sep. 01, 2017).

[52] W3C, D. Martin, M. Burstein, and J. Hobbs, “OWL-S: Semantic markup for web services,” *W3C Member ...*, 2004. http://www.w3.org/Submission/OWL-S/ (accessed Sep. 01, 2017).

[53] W3C, “Web Service Modeling Ontology (WSMO),” *W3C Member ...*, 2005. https://www.w3.org/Submission/WSMO/ (accessed Sep. 01, 2017).

[54] R. Akkiraju *et al.*, “Web Service Semantics - WSDL-S,” in *W3C Workshop on Frameworks for Semantics in Web Services*, 2005, pp. 1–42.

[55] S. Äyrämö and T. Kärkkäinen, *Introduction to Partitioning-Based Clustering Methods with a Robust Example*, vol. 35, no. C. 2006.

[56] D. Xu and Y. Tian, “A Comprehensive Survey of Clustering Algorithms,” *Ann. Data Sci.*, vol. 2, no. 2, pp. 165–193, Jun. 2015.

[57] J. Han, M. Kamber, and J. Pei, “Cluster Analysis,” in *Data Mining*, J. Han, M. Kamber, and J. B. T.-D. M. (Third E. Pei, Eds. Boston: Elsevier, 2012, pp. 443–495.

[58] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967, vol. 1, no. 14, pp. 281–297.

[59] B. S. Everitt, S. Landau, M. Leese, and D. Stahl, “Hierarchical Clustering,” in *Cluster Analysis*, 2011, pp. 71–110.

[60] B. J. Frey and D. Dueck, “Clustering by Passing Messages Between Data Points,” *Science (80-.).*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.

[61] E. Asghari and M. KeyvanPour, “XML document clustering: techniques and challenges,” *Artif. Intell. Rev.*, vol. 43, no. 3, pp. 417–436, Mar. 2015.

[62] A. Islam and D. Inkpen, “Semantic text similarity using corpus-based word similarity and string similarity,” *ACM Trans. Knowl. Discov. Data*, vol. 2, no. 2, pp. 1–25, 2008.

[63] D. Sarkar, *Text Analytics with Python*. 2019.

[64] J. Gracia and E. Mena, “Web-Based Measure of Semantic Relatedness,” in *Web Information Systems Engineering - WISE 2008*, vol. 5175 LNCS, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 136–150.

[65] G. a. Miller, “WordNet: a lexical database for English,” *Commun. ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995.

[66] T. Slimani, “Description and Evaluation of Semantic Similarity Measures Approaches,” *Int. J. Comput. Appl.*, vol. 80, no. 10, pp. 25–33, 2013.

[67] A. Hevner, S. March, J. Park, and S. Ram, “Design science in information systems research,” *Jouranl Manag. Inf. Syst. Q.*, vol. 28, no. 1, pp. 75–105, Mar. 2004, Accessed: Jun. 26, 2013.

[68] A. Hevner and S. Chatterjee, *Design Research in Information Systems*, vol. 22, no. 1. Boston, MA: Springer US, 2010.

[69] J. W. Creswell, *Research Design: Qualitative, Quantitative and Mixed Methods Approaches*, 4th ed. Sage Publications, 2014.

[70] V. Vaishnavi, B. Kuechler, and S. Patter, “Design Science Research in Information Systems,” 2017.

[71] S. Gregor and A. R. Hevner, “Positioning and Presenting Design Science Research for Maximum Impact,” *MIS Q.*, vol. 37, no. 2, pp. 337–355, Feb. 2013.

[72] Gregor, “The Nature of Theory in Information Systems,” *MIS Q.*, vol. 30, no. 3, p. 611, 2006.

[73] K. Peffers, T. Tuunanen, M. a. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007.

[74] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature reviews in Software Engineering,” 2007.

[75] B. Kitchenham, “Procedures for performing systematic reviews,” *Keele, UK, Keele Univ.*, vol. 33, no. TR/SE-0401, p. 28, 2004.

[76] C. Mavergames, “Covidence (Systematic Review Software),” 2013. https://www.covidence.org/ (accessed Dec. 07, 2016).

[77] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering - A systematic literature review,” *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009.

[78] T. Dybå and T. Dingsøyr, “Empirical studies of agile software development: A systematic review,” *Inf. Softw. Technol.*, vol. 50, no. 9–10, pp. 833–859, Aug. 2008.

[79] Z. Cong, A. Fernandez, H. Billhardt, and M. Lujak, “Service discovery acceleration

with hierarchical clustering," *Inf. Syst. Front.*, vol. 17, no. 4, pp. 799–808, Aug. 2015.

[80] J. Wu, L. Chen, Z. Zheng, M. R. . M. R. Lyu, and Z. Wu, "Clustering Web services to facilitate service discovery," *Knowl. Inf. Syst.*, vol. 38, no. 1, pp. 207–229, Jan. 2014.

[81] K. Su, B. Xiao, B. Liu, H. Zhang, and Z. Zhang, "TAP: A personalized trust-aware QoS prediction approach for web service recommendation," *Knowledge-Based Syst.*, vol. 115, pp. 55–65, Jan. 2016.

[82] L. Chen, J. Wu, Z. Zheng, M. R. . M. R. Lyu, and Z. Wu, "Modeling and exploiting tag relevance for Web service mining," *Knowl. Inf. Syst.*, vol. 39, no. 1, pp. 153–173, Apr. 2014.

[83] F. Chen, M. Li, H. Wu, and L. Xie, "Web service discovery among large service pools utilising semantic similarity and clustering," *Enterp. Inf. Syst.*, vol. 11, no. 3, pp. 452–469, Mar. 2017.

[84] M. Lin and D. W. Cheung, "Automatic Tagging Web Services Using Machine Learning Techniques," in *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Aug. 2014, vol. 2, pp. 258–265.

[85] M. Aznag, M. Quafafou, and Z. Jarir, "Leveraging Formal Concept Analysis with Topic Correlation for Service Clustering and Discovery," in *2014 IEEE International Conference on Web Services*, Jun. 2014, pp. 153–160.

[86] P. He, J. Zhu, Z. Zheng, J. Xu, and M. R. Lyu, "Location-Based Hierarchical Matrix Factorization for Web Service Recommendation," in *2014 IEEE International Conference on Web Services*, Jun. 2014, pp. 297–304.

[87] W. Rong, B. Peng, Y. Ouyang, K. Liu, and Z. Xiong, "Collaborative personal profiling for web service ranking and recommendation," *Inf. Syst. Front.*, vol. 17, no. 6, pp. 1265–1282, Dec. 2015.

[88] C. Yu and L. Huang, "CluCF: a clustering CF algorithm to address data sparsity problem," *Serv. Oriented Comput. Appl.*, vol. 11, no. 1, pp. 33–45, Mar. 2017.

[89] G. Tian, C. Sun, K. He, and X. Ji, "Transferring auxiliary knowledge to enhance heterogeneous web service clustering," *Int. J. High Perform. Comput. Netw.*, vol. 9, no. 1/2, p. 160, 2016.

[90] L. Chen, G. Yang, W. Zhu, Y. Zhang, and Z. Yang, "Clustering facilitated web services discovery model based on supervised term weighting and adaptive metric learning," *Int. J. Web Eng. Technol.*, vol. 8, no. 1, p. 58, 2013.

[91] M. Gong, Z. Xu, L. Xu, Y. Li, and L. Chen, "Recommending Web Service Based on User Relationships and Preferences," in *2013 IEEE 20th International Conference on Web Services*, Jun. 2013, pp. 380–386.

[92] B. Kumara, I. Paik, H. Ohashi, W. Chen, and K. R. C. Koswatte, "Context Aware Post-filtering for Web Service Clustering," in *2014 IEEE International Conference on Services Computing*, Jun. 2014, pp. 440–447.

[93] W. Rong, K. Liu, L. Liang, W. Rong, K. Liu, and L. Liang, "Personalized Web Service Ranking via User Group Combining Association Rule," in *2009 IEEE International Conference on Web Services*, Jul. 2009, pp. 445–452.

[94] F. Chen, S. Yuan, and B. Mu, "User-QoS-Based Web Service Clustering for QoS Prediction," in *2015 IEEE International Conference on Web Services*, Jun. 2015, pp. 583–590.

[95] T. Liang, L. Chen, H. Ying, and J. Wu, "Co-Clustering WSDL Documents to Bootstrap Service Discovery," in *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, Nov. 2014, pp. 215–222.

[96] R. Liu, X. Xu, and Z. Wang, "Service Recommendation Using Customer Similarity and Service Usage Pattern," in *2015 IEEE International Conference on Web Services*, Jun. 2015, pp. 408–415.

[97] Y. Lei, Z. Jiantao, Z. Junxing, W. Fengqi, and W. Juan, "Time-Aware Semantic Web Service Recommendation," in *2015 IEEE International Conference on Services Computing*, Jun. 2015, no. PG-664-671, pp. 664–671.

[98] G. Vadivelou and E. Ilavarasan, "Collaborative filtering based hybrid approach for web service recommendations," *Res. J. Appl. Sci. Eng. Technol.*, vol. 8, no. 5, pp. 615–622, Aug. 2014.

[99] A. Nagy, C. Oprisa, I. Salomie, C. B. Pop, V. R. Chifu, and M. Dinsoreanu, "Particle Swarm Optimization for Clustering Semantic Web Services," in *2011 10th International Symposium on Parallel and Distributed Computing*, Jul. 2011, pp. 170–177.

[100] J. Ma, Y. Zhang, and J. He, "Web Services Discovery Based on Latent Semantic Approach," in *2008 IEEE International Conference on Web Services*, Sep. 2008, pp. 740–747.

[101] B. Kumara, I. Paik, W. Chen, and K. H. Ryu, "Web Service Clustering using a Hybrid Term-Similarity Measure with Ontology Learning," *Int. J. Web Serv. Res.*, vol. 11, no. 2, pp. 24–45, Apr. 2014.

[102] C. Platzer, F. Rosenberg, and S. Dustdar, "Web service clustering using multidimensional angles as proximity measures," *ACM Trans. Internet Technol.*, vol. 9, no. 3, pp. 1–26, Jul. 2009.

[103] L. D. J. Silva, D. B. Claro, and D. C. Pavao Lopes, "Semantic-based clustering of web services," *J. Web Eng.*, vol. 14, no. 3–4, pp. 325–345, Jul. 2015.

[104] Q. Yu, "QoS-aware service selection via collaborative QoS evaluation," *World Wide Web*, vol. 17, no. 1, pp. 33–57, Jan. 2014.

[105] S. Dasgupta, S. Bhat, and Y. Lee, "Taxonomic Clustering and Query Matching for Efficient Service Discovery," in *2011 IEEE International Conference on Web Services*, Jul. 2011, pp. 363–370.

[106] A. V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-Based Automated Service Discovery," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 260–275, Apr. 2012.

[107] B. Kumara, I. Paik, and G. Lee, "Ontology learning method for Web services clustering," in *Proceedings - 2012 7th International Conference on Computing and Convergence Technology (ICCIT, ICEI and ICACT), ICCCT 2012*, 2012, pp. 705–710.

[108] B. Kumara, I. Paik, K. R. C. Koswatte, and W. Chen, "Ontology learning with complex data type for Web service clustering," in *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CIDM 2014: 2014 IEEE Symposium on Computational Intelligence and Data Mining, Proceedings*, Dec. 2014, pp. 129–136.

[109] J. Liu, F. Liu, X. Li, K. He, Y. Ma, and J. Wang, "Web Service Clustering Using Relational Database Approach," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 25, no. 08, pp. 1365–1393, Oct. 2015.

[110] T. Wen, G. Sheng, Y. Li, and Q. Guo, "Research on Web service discovery with semantics and clustering," in *2011 6th IEEE Joint International Information Technology and Artificial Intelligence Conference*, Aug. 2011, vol. 1, pp. 62–67.

[111] B. Upadhyaya, F. Khomh, Ying Zou, A. Lau, and J. Ng, "A concept analysis approach for guiding users in service discovery," in *2012 Fifth IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, Dec. 2012, pp. 1–8.

[112] K. Elgazzar, H. S. Hassanein, and P. Martin, "DaaS: Cloud-based mobile Web service discovery," *Pervasive Mob. Comput.*, vol. 13, pp. 67–84, Aug. 2014.

[113] L. Chen, Q. Yu, P. S. Yu, and J. Wu, "WS-HFS: A Heterogeneous Feature Selection Framework for Web Services Mining," in *2015 IEEE International Conference on Web Services*, Jun. 2015, pp. 193–200.

[114] Q. Yu, H. Wang, and L. Chen, "Learning Sparse Functional Factors for Large-Scale Service Clustering," in *2015 IEEE International Conference on Web Services*, Jun. 2015, pp. 201–208.

[115] H. Gao, S. Wang, L. Sun, and F. Nian, "Hierarchical Clustering Based Web Service Discovery," in *IFIP Advances in Information and Communication Technology*, vol. 426, 2014, pp. 281–291.

[116] B. Kumara, I. Paik, T. H. A. S. Siriweera, and K. R. C. Koswatte, "QoS Aware Service Clustering to Bootstrap the Web Service Selection," in *2017 IEEE International Conference on Services Computing (SCC)*, Jun. 2017, pp. 233–240.

[117] S. Li, J. Wen, F. Luo, M. Gao, J. Zeng, and Z. Y. Dong, "A New QoS-Aware Web Service Recommendation System Based on Contextual Feature Recognition at Server-Side," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 2, pp. 332–342, Jun. 2017.

[118] B. Cao *et al.*, "Mashup service clustering based on an integration of service content and network via exploiting a two-level topic model," in *Proceedings - 2016 IEEE International Conference on Web Services, ICWS 2016*, Jun. 2016, pp. 212–219.

[119] B. Cao, X. (Frank) Liu, J. Liu, and M. Tang, "Domain-aware Mashup service

clustering based on LDA topic model from multiple data sources," *Inf. Softw. Technol.*, vol. 90, pp. 40–54, Oct. 2017.

[120] F. Slaimi, S. Sellami, O. Boucelma, and A. Ben Hassine, "Leveraging Track Relationships for Web Service Recommendation," in *2016 IEEE 13th International Conference on e-Business Engineering (ICEBE)*, Nov. 2016, pp. 220–225.

[121] B. Cheng, S. Zhao, C. Li, and J. Chen, "A Web Services Discovery Approach Based on Mining Underlying Interface Semantics," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 950–962, May 2017.

[122] R. Rupasingha, I. Paik, and B. Kumara, "Improving Web Service Clustering through a Novel Ontology Generation Method by Domain Specificity," in *2017 IEEE International Conference on Web Services (ICWS)*, Jun. 2017, pp. 744–751.

[123] M. Shi, J. Liu, D. Zhou, M. Tang, and B. Cao, "WE-LDA: A Word Embeddings Augmented LDA Model for Web Services Clustering," *2017 IEEE Int. Conf. Web Serv.*, pp. 9–16, 2017.

[124] W. Shi, X. Liu, and Q. Yu, "Correlation-Aware Multi-Label Active Learning for Web Service Tag Recommendation," in *2017 IEEE International Conference on Web Services (ICWS)*, Jun. 2017, pp. 229–236.

[125] CORE, "The Computing Research and Education Association of Australasia, CORE," 2014. http://core.edu.au/home (accessed Mar. 17, 2017).

[126] Clarivate Analytics, "2015 Journal Citation Reports," 2017. https://jcr.incites.thomsonreuters.com (accessed Mar. 17, 2017).

[127] W. Wong, W. Liu, and M. Bennamoun, "Tree-Traversing Ant Algorithm for term clustering based on featureless similarities," *Data Min. Knowl. Discov.*, vol. 15, no. 3, pp. 349–381, Oct. 2007.

[128] R. L. Cilibrasi and P. M. B. Vitanyi, "The Google Similarity Distance," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 370–383, Mar. 2007.

[129] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys. Dokl.*, vol. 10, no. 8, pp. 707–710, 1966.

[130] A. Heß and N. Kushmerick, "Machine learning for annotating semantic web services," 2004.

[131] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*, vol. 15. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[132] M. Klusch, *CASCOM: Intelligent Service Coordination in the Semantic Web*, vol. 1. Basel: Birkhäuser Basel, 2008.

[133] M. Crasso, A. Zunino, and M. Campo, "A Survey of Approaches to Web Service Discovery in Service-Oriented Architectures," *J. Database Manag.*, vol. 22, no. 1, pp. 102–132, 2011.

[134] A. Bernstein and C. Kiefer, "iRDQL-Imprecise RDQL Queries Using Similarity Joins," *Proc. K-CAP 2005 Work. Ontol. Manag. Searching, Sel. Rank.*

Segmentation, 2005.

[135] P. Palathingal and S. Chandra, "Agent approach for service discovery and utilization," in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, 2004, vol. 00, no. C, p. 9 pp.

[136] G. Fenza, V. Loia, and S. Senatore, "A hybrid approach to semantic web services matchmaking," *Int. J. Approx. Reason.*, vol. 48, no. 3, pp. 808–828, Aug. 2008.

[137] U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel, "Automatic Location of Services," in *European Semantic Web Conference*, 2005, pp. 1–16.

[138] C. Kiefer, A. Bernstein, H. J. Lee, M. Klein, and M. Stocker, "Semantic Process Retrieval with iSPARQL," in *The Semantic Web: Research and Applications*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 609–623.

[139] Y. Li, F. Zou, Z. Wu, and F. Ma, "PWSD: A Scalable Web Service Discovery Architecture Based on Peer-to-Peer Overlay Network," in *Asia-Pacific Web Conference*, 2004, pp. 291–300.

[140] W. Hoschek, "The web service discovery architecture," in *Supercomputing, ACM/IEEE 2002 Conference*, 2002, p. 38.

[141] B. Sapkota, D. Roman, S. R. Kruk, and D. Fensel, "Distributed Web Service Discovery Architecture," in *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, 2006, no. Fp6 004617, pp. 136–136.

[142] C. Houy, P. Fettke, and P. Loos, "Understanding Understandability of Conceptual Models – What Are We Actually Talking about?," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, pp. 64–77.

[143] Y. Wand and R. Weber, "Research commentary: Information systems and conceptual modeling - A research agenda," *Inf. Syst. Res.*, 2002.

[144] J. Krogstie, A. Lothe Opdahl, and S. Brinkkemper, *Conceptual Modelling in Information Systems Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[145] D. L. Moody, "Cognitive Load Effects on End User Understanding of Conceptual Models: An Experimental Analysis," 2004.

[146] Z. Yuan and M. YaoFeng, "A web service personal discovery based on QoS," *J. Softw. Eng.*, vol. 9, no. 1, pp. 169–178, 2015.

[147] C. Schmidt and M. Parashar, "A Peer-to-Peer Approach to Web Service Discovery," *World Wide Web*, vol. 7, no. 2, pp. 211–229, Jun. 2004.

[148] W. J. Obidallah, U. Ruhi, and B. Raahemi, "Current Landscape of Web Service Discovery: A Typology Based on Five Characteristics," in *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, Oct. 2016, pp. 678–683.

[149] J. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. Tsakalidis, "Contemporary Web Service Discovery Mechanisms," *J. Web Eng.*, vol. 5, no. 3, pp. 265–290, 2006.

[150] M. Klusch, “Semantic Web Service Coordination,” in *CASCOM: Intelligent Service Coordination in the Semantic Web*, Basel: Birkhäuser Basel, 2008, pp. 59–104.

[151] I. Toma, K. Iqbal, M. Moran, D. D. Roman, T. Strang, and D. Fensel, “An evaluation of discovery approaches in Grid and Web services environments,” in *Proceedings of the Net. ObjectDays 2005*, 2005, vol. 69, pp. 233–247.

[152] D. A. D’Mello and V. S. Ananthanarayana, “A Review of Dynamic Web Service Description and Discovery Techniques,” in *2010 First International Conference on Integrated Intelligent Computing*, Aug. 2010, pp. 246–251.

[153] K. Sycara, M. Paolucci, J. Soudry, and N. Srinivasan, “Dynamic discovery and coordination of agent-based semantic web services,” *IEEE Internet Comput.*, vol. 8, no. 3, pp. 66–73, May 2004.

[154] A. Ankolekar *et al.*, “DAML-S: Semantic markup for web services,” in *Proceedings of the First International Conference on Semantic Web Working*, 2001, vol. 75, pp. 411–430.

[155] C. Wu and E. Chang, “Searching Services ‘on the Web’: A Public Web Services Discovery Approach,” in *2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*, Dec. 2007, pp. 321–328.

[156] H. Song, D. Cheng, A. Messer, and S. Kalasapur, “Web Service Discovery Using General-Purpose Search Engines,” in *IEEE International Conference on Web Services (ICWS 2007)*, Jul. 2007, no. Icws, pp. 265–271.

[157] “Apache Avro,” 2012. http://avro.apache.org/ (accessed May 02, 2016).

[158] A. Ranabahu, A. Ranabahu, and A. Sheth, “SA-REST: Semantic Annotation of Web Resources.” https://www.w3.org/Submission/SA-REST/ (accessed May 31, 2020).

[159] M. Lanthaler and C. Gutl, “A semantic description language for RESTful data services to combat Semaphobia,” *... (DEST), 2011 Proc. 5th ...*, vol. 5, no. June, pp. 47–53, 2011.

[160] W. Rong and K. Liu, “A Survey of Context Aware Web Service Discovery: From User’s Perspective,” in *2010 Fifth IEEE International Symposium on Service Oriented System Engineering*, Jun. 2010, pp. 15–22.

[161] M. Klein and B. König-Ries, “Coupled signature and specification matching for automatic service binding,” in *Proceedings of European Conference of Web Services (ECOES)*, vol. 3250, L.-J. (LJ) Zhang and M. Jeckle, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 183–197.

[162] B. Schilit, N. Adams, and R. Want, “Context-Aware Computing Applications,” in *1994 First Workshop on Mobile Computing Systems and Applications*, Dec. 1994, pp. 85–90.

[163] M. Klusch and F. Kaufer, “WSMO-MX: A hybrid Semantic Web service matchmaker,” *Web Intell. Agent Syst. An Int. J.*, vol. 7, no. 1, pp. 23–42, 2009.

[164] W3C Working Group Note *et al.*, “Web Services Architecture,” *W3C Working Group Note*, 2004. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

(accessed Sep. 22, 2015).

[165] H. Becha and D. Amyot, "Non-Functional Properties in Service Oriented Architecture – A Consumer's Perspective," *J. Softw.*, vol. 7, no. 3, pp. 575–587, Mar. 2012.

[166] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic Matching of Web Services Capabilities," in *International Semantic Web Conference ISWC 2002: The Semantic Web*, vol. 2342, I. Horrocks and J. Hendler, Eds. Springer Berlin Heidelberg, 2002, pp. 333–347.

[167] Lee Choon Chiat, Linpeng Huang, and Jinkui Xie, "Matchmaking for semantic web services," in *IEEE International Conference onServices Computing, 2004. (SCC 2004). Proceedings. 2004*, 2004, pp. 455–458.

[168] M. C. Jaeger, G. Rojec-Goldmann, C. Liebetruth, G. Mühl, and K. Geihs, "Ranked Matching for Service Descriptions Using OWL-S," in *Kommunikation in Verteilten Systemen (KiVS)*, vol. 2, Berlin/Heidelberg: Springer-Verlag, 2005, pp. 91–102.

[169] L.-H. Vu, M. Hauswirth, and K. Aberer, "Towards P2P-Based Semantic Web Service Discovery with QoS Support," in *Business Process Management Workshops*, vol. 3812, no. 507483, C. J. Bussler and A. Haller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 18–31.

[170] M. Klusch, B. Fries, and K. Sycara, "Automated semantic web service discovery with OWLS-MX," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems - AAMAS '06*, 2006, p. 915.

[171] M. Klusch and P. Kapahnke, "Semantic web service selection with SAWSDL-MX," in *Proceedings of the Second International Conference on Service Matchmaking and Resource Retrieval in the Semantic Web-Volume 416*, 2008, pp. 2–16.

[172] D. Skoutas, D. Sacharidis, V. Kantere, and T. Sellis, "Efficient Semantic Web Service Discovery in Centralized and P2P Environments," in *Proceedings of the 7th International Conference on The Semantic Web (ISWC '08)*, 2008, pp. 583–598.

[173] G. Meditskos and N. Bassiliades, "Structural and Role-Oriented Web Service Discovery with Taxonomies in OWL-S," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 2, pp. 278–290, Feb. 2010.

[174] P. Plebani and B. Pernici, "URBE: Web Service Retrieval Based on Similarity Evaluation," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 11, pp. 1629–1642, Nov. 2009.

[175] L. Richardson, "Beautiful Soup Documentation." https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (accessed Apr. 26, 2020).

[176] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana, "Web Service Definition Language (WSDL) 1.1," *W3C*. https://www.w3.org/TR/wsdl.html#_documentation (accessed Apr. 29, 2020).

[177] G. Grefenstette, "Tokenization," no. October, 1999, pp. 117–133.

[178] E. Rasmussen, “Stoplists BT - Encyclopedia of Database Systems,” L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 2794–2796.

[179] S. Bird, “NLTK: The Natural Language Toolkit,” in *Proceedings of the COLING/ACL on Interactive presentation sessions* -, 2006, no. July, pp. 69–72.

[180] Stanford-University, “Stemming and lemmatization.” https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html (accessed May 05, 2020).

[181] M. F. Porter, “An algorithm for suffix stripping,” *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[182] G. Salton, A. Wong, and C. S. Yang, “A vector space model for automatic indexing,” *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975.

[183] J. Yan, “Text Representation,” in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 3069–3072.

[184] J. Camacho-Collados and M. T. Pilehvar, “From word to sense embeddings: A survey on vector representations of meaning,” *J. Artif. Intell. Res.*, vol. 63, pp. 743–788, 2018.

[185] F. Almeida and G. Xexéo, “Word Embeddings: A Survey,” no. 1991, Jan. 2019.

[186] C. De Boom, S. Van Canneyt, T. Demeester, and B. Dhoedt, “Representation learning for very short texts using weighted word embedding aggregation,” *Pattern Recognit. Lett.*, vol. 80, pp. 150–156, Sep. 2016.

[187] V. N. Gudivada, D. L. Rao, and A. R. Gudivada, “Information Retrieval: Concepts, Models, and Systems,” in *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*, 1st ed., vol. 38, V. N. Gudivada and C. R. B. T.-H. of S. Rao, Eds. Elsevier, 2018, pp. 331–401.

[188] “Wikimedia Downloads.” https://dumps.wikimedia.org/ (accessed Jun. 27, 2020).

[189] “Google Code Archive - Long-term storage for Google Code Project Hosting.” https://code.google.com/archive/p/word2vec/ (accessed Jun. 27, 2020).

[190] “English Gigaword Fifth Edition - Linguistic Data Consortium.” https://catalog.ldc.upenn.edu/LDC2011T07 (accessed Jun. 27, 2020).

[191] “Common Crawl.” https://commoncrawl.org/ (accessed Jun. 28, 2020).

[192] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[193] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” *arXiv Prepr.*, pp. 1–12, Jan. 2013.

[194] J. Han, M. Kamber, and J. Pei, “Getting to Know Your Data,” in *Data Mining*, Elsevier, 2012, pp. 39–82.

[195] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, “From Word Embeddings to Document Distances,” in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, 2015, pp. 957–966.

[196] P. M. B. Vitányi, R. L. Cilibrasi, and P. M. B. Vitanyi, “Normalized web distance and word similarity,” *Handb. Nat. Lang. Process. Second Ed.*, pp. 293–314, May 2009.

[197] Princeton-University, “WordNet-A Lexical Database for English,” 2014. http://wordnet.princeton.edu/ (accessed Dec. 31, 2014).

[198] Z. Wu and M. Palmer, “Verbs semantics and lexical selection,” in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics* -, 1994, pp. 133–138.

[199] “gensim: Topic modelling for humans.” https://radimrehurek.com/gensim/ (accessed Jun. 29, 2020).

[200] F. Pedregosa *et al.*, “Scikit-Learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, no. null, pp. 2825–2830, Nov. 2011.

[201] D. A. Chappell and T. Jewell, *Java Web Services*. O’Reilly Media, 2002.

[202] E. Newcomer, *Understanding Web Services: XML, Wsdl, Soap, and UDDI*. Addison-Wesley Professional, 2002.

[203] M. Papazoglou, *Web services: principles and technology*. Pearson Education, 2008.

[204] D. Mukhopadhyay and A. Chougule, “A Survey on Web Service Discovery Approaches,” in *Advances in Computer Science, Engineering & Applications SE - 99*, vol. 166, D. C. Wyld, J. Zizka, and D. Nagamalai, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1001–1012.

[205] M. Papazoglou, *Web Services & SOA: Principles and Technology*. 2012.

[206] C. Pautasso, O. Zimmermann, and F. Leymann, “Restful web services vs. ‘big’’ web services,” in *Proceeding of the 17th international conference on World Wide Web - WWW ’08*, 2008, p. 805.

[207] L. D. Ngane, A. Goh, and C. H. Tru, “A Survey of Web Service Discovery Systems,” G. I. Alkhatib and D. C. Rine, Eds. IGI Global, 2008.

[208] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, “Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI,” *IEEE Internet Comput.*, vol. 6, no. 2, pp. 86–93, Mar. 2002.

[209] K. Wagh and R. Thool, “A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host,” *J. Inf. Eng. Appl.*, vol. 2, no. 5, pp. 12–16, 2012.

[210] R. Mizouni, M. A. Serhani, R. Dssouli, A. Benharref, and I. Taleb, “Performance Evaluation of Mobile Web Services,” in *2011 IEEE Ninth European Conference on Web Services*, Sep. 2011, pp. 184–191.

Appendix A. Related Topics in Web Services

This appendix presents additional details on the topics related to SOAP- and REST-based Web services.

A.1. SOAP-based Web Services

In SOAP-based Web services, the service provider and service consumer establish a common understanding of the Web services syntax and operation, allowing an RPC-like interaction with remote systems. A unique interface for each Web service is described with WSDL. The WSDL is an XML-based document to describe a Web service interface by describing the data types and the structure of the Web services and allowing clients to automatically understand how to communicate with the Web service [201, 202]. WSDL 1.1 and WSDL 2.0 are both describe SOAP-based Web services. WSDL 2.0 is more straightforward, usable, and readable for users than WSDL 1.1. Furthermore, WSDL 2.0 provides several improvements, including support for interoperation, language clarification, and simplifications [203]. However, WSDL 1.1 is the most consumed and most popular and is more accessible than WSDL 2.0. More details about WSDL 1.1 and WSDL 2.0 elements are illustrated in Table A-1.

Table A-1:WSDL 1.1 and WSDL 2.0 Elements

Element		Description
WSDL 1.1	WSDL 2.0	
<definitions>	<description>	Root element for all WSDL documents. Specifies document name, target namespace attribute and namespace.
<types>	<types>	XML schema to define the data types between the client and the server.
<message>	No support	The message element describes data being exchanged as input and output messages for operations. Replaced by Types in WSDL 2.0 to specify a variable number of inputs and outputs.
<portType>	<interface>	Describe operations and messages. Renamed to interface in WSDL 2.0 to support interface inheritance.

<binding>	<binding>	Provides details about the protocol to transmit the operations of the wire, which can be HTTP GET, HTTP POST, or SOAP.
<port>	<endpoint>	Specifies a single address for a binding. Renamed to endpoint in WSDL 2.0

WSDL descriptions are usually published in a public UDDI registry. UDDI provides a mechanism for registering and discovering Web services. It provides the service providers with the ability to publish their Web services description by providing specifications of their Web services and how to communicate with them using XML [204]. Applications and developer tools that use Web services standards such as SOAP/XML and WSDL can use UDDI for registration purposes. UDDI provides a global, independent platform and an open framework that makes it easy for enterprises and services providers to conduct business, find trading partners, and interoperate with their partners over the internet [205]. Furthermore, it enables service requesters to discover information about the Web services offering, find Web services description provided by services developers, and find technical information about the Web services interfaces and definitions.

The UDDI specifications allow its users to perform two types of exchanges, which include inquiring about and publishing Web services. The publishing API allows Web services developers to add, update, and delete data about a Web service in the UDDI registry. The inquiry API allows potential Web services service requesters to search the UDDI registry for information about a Web service based on various classifications [205]. UDDI provides Web services with a method to be discovered by service consumers based on keywords matching over the provided inquiry API. A Web service's real value is when a Web service user can quickly and smoothly locate, discover, and communicate with a Web service anywhere [202].

Public UDDIs, started by industry-leading companies allowed Web services providers to publish and register their Web services in the registry. As a service requester, we can search the UDDI registry to look for a Web service that meets our requirements. UDDI technology has been employed for this purpose for many years, stably and maturely. However, UDDI failed to get widespread acceptance by industries [206] due to its limitations, which are due to it keyword-based matching based on WSDL that has no semantic understanding. This limitation makes UDDI not efficient as a massive number of Web services

can be returned that match the queried keyword, making it challenging to come up with the best Web service based on the user request [204, 207].

An XML messaging protocol such as SOAP or XML-RPC, is employed to construct Web services communications. SOAP is an attempt to codify the usage of existing internet technologies to standardize distributed communication over the Web [205]. SOAP is the standard messaging protocol part of the RPC mechanism to invoke a Web service and get the result asynchronously. It is a network application protocol to transfer messages between service different parties and instances described by WSDL interfaces. It provides a common format for XML document transportation over HTTP, SMTP, and FTP [201, 202]. By having a standard messaging format, different clients using different languages can invoke the same Web service with SOAP. For example, .Net components can invoke a Web service component written in Java with SOAP, which standardizes the communications. A SOAP document contains a top-level XML element called the envelope [206], which contains a header and body. The header is a container for message layer infrastructure information [206, 208]. The body contains the payload of the message where XML Schema is utilized to structure the message.

Figure A-1 illustrates the three components that provide the building blocks for service description (in WSDL), discovery (in UDDI), and communication (in SOAP) for SOAP-based Web services. Web service clients can use the inquiry API to look for Web services that are published by service providers in the UDDI registry. Web service providers can use the publishing API to share and register the Web services description document, endpoints, and other details. After finding the right Web service, the service client can invoke the required service where SOAP is the communication and messaging protocol.

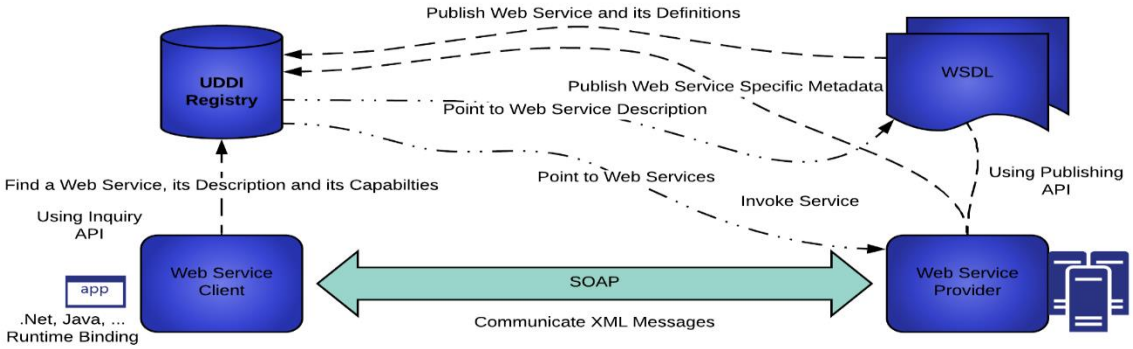


Figure A-1: SOAP-based Web Services Components

SOAP-based Web services were heavily promoted by major software companies that provided robust and automated solutions with support for almost all development tools. The adoption of SOAP-based Web services is dropping as REST-based Web services are taking over by providing different advantages over the SOAP-based Web services, mainly in the mobile domain of Web services.

A.2. REST-based Web Services

REST is an architecture style for building loosely coupled applications based on a set of recommendations where HTTP is employed for data transmission [36]. Services that are built by developers who follow the REST architecture style recommendations are called RESTful Web services. In Web services developers' communities, other developed Web services that do not follow the recommendations and the principles of REST are called RESTless.

The simplicity, flexibility, and scalability of REST Web services made it more popular among Web services requesters and developers. Although the RESTful Web services are tightly coupled with the HTTP protocol, studies show that they outperformed SOAP Web services within resource-constrained environments and became more widespread among service providers [206, 209, 210]. Google, Facebook, Yahoo, and Amazon have adopted REST services in their Web services offerings. RESTful-designed Web services expose their functionalities as resources with a unique URI for each resource. Users can access the preferred resources with the associated URI. REST-based Web services are better for Web-based APIs, especially in mobile domains, as they make data available as a resource with a lightweight payload. However, choosing between SOAP-based Web services and REST-based Web services depends on different factors and requirements. For example, SOAP-based Web services provide better security measures than REST-based Web services.

Appendix B. Additional Data and Stats Produced for Systematic Literature Review

This appendix presents the details of the search queries for each used search engine. Furthermore, the tables and stats produced during the systematic literature review are presented here.

B.1. Search Queries for Each Digital Library

This section presents the search queries employed for each digital library while conducting the SLR phase:

- **ACM Digital Library:** Does not support abstract, title and keywords search.
  "query": { ("Web Service*" "Web Service* Discovery" "Web Service* Recommend*" "Clustering" "Association Rule*") }
  "filter": {"publicationYear":{ "gte":2006 }},
  {owners.owner=HOSTED}
- **IEEE Digital Library:** Does not support abstract, title and keywords search.
  "Web Service*" OR "Web Service* Discovery" OR "Web Service* Recommend*") AND ("Clustering" OR "Association Rule*") and refined by Year: 2006-2018
- **Springer Link:** Does not support abstract, title and keywords search.
  '("Web Service*" OR "Web Service* Discovery" OR "Web Service* Recommend*") AND ("Clustering" OR "Association Rule*")'
  within English Computer Science Article 2006 2018
- **ISI Web of Science:** supports abstract, title and keywords search.
  TOPIC: (("Web Service*" OR "Web Service* Discovery" OR "Web Service* Recommend*") AND ("Clustering" OR "Association Rule*"))
  Timespan=2006-2018
- **Science Direct:** supports abstract, title and keywords search.
  pub-date>2006 and TITLE-ABS-KEY ("Web Service?" OR "Web Service? Discovery" OR "Web Service? Recommend*") AND TITLE-ABS-KEY ("Clustering" OR "Association Rule?")
- **Scopus:** supports abstract, title and keywords search.
  TITLE-ABS-KEY (("Web Service*" OR "Web Service* Discovery" OR "Web Service* Recommend*") AND ("Clustering" OR "Association Rule*")) AND PUBYEAR > 2006 AND (LIMIT-TO (LANGUAGE , "English"))

B.2. List of Conferences and Journals

The list of conferences and journals cited in the SLR is presented Table B-1.

Table B-1: SLR List of Conferences and Journals

Type	Venues	Abbrevation	Rank /IMPF
Conferences	IEEE Joint International Information Technology and Artificial Intelligence Conference	ITAIC	_
	IEEE International Conference on Web Services	ICWS	A
	IEEE International Conference on Services Computing	SCC	A
	IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)	WI-IAT	C
	International Conference on Computing and Convergence Technology	ICCCT	_
	IEEE International Conference on High Performance Computing and Communications	HPCC	B
	IEEE International Conference on Data Science and Advanced Analytics	DSAA	C
	IEEE International Conference on Service-Oriented Computing and Applications	SOCA	C
	IEEE International Conference on Information Reuse & Integration	IRI	C
	International Conference Networking in Education and Research	RoEduNet	_
	IEEE International Symposium on Service Oriented System Engineering	SOSE	_
	IEEE Symposium on Computational Intelligence and Data Mining	CIDM	C
	International Conference on Complex, Intelligent, and Software Intensive Systems	CISIS	C
	International Conference on Informatics and Semiotics in Organisations	ICISO	_
	IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops	ISORCW	_
	International Symposium Parallel and Distributed Computing	ISPDC	C
	IEEE International Conference on e-Business Engineering	ICEBE	B
	International workshop on Context enabled source and service selection, integration and adaptation	CSSSIA	_
Journals	Chinese Journal of Electronics	CJE	0.319
	IEEE Transactions on Services Computing	TSC	2.365
	Pervasive and Mobile Computing	PMC	1.719
	Knowledge-Based Systems	KNOSYS	3.325
	Journal of Research and Practice in Information Technology	JRPIT	0.222
	Knowledge and Information Systems	KAIS	1.702
	IEEE Transactions on Parallel and Distributed Systems	TPDS	2.661
	Information Systems Frontiers	ISF	1.45
	International Journal of Web Services Research	IJWSR	0.257
	Journal of Information Science and Engineering	JISE	0.392
	International Journal of Software Engineering and Knowledge Engineering	IJSEKE	0.24
	ACM Transactions on Internet Technology	TOIT	0.705
	Journal of Web Engineering	JWE	0.622
	World Wide Web: Internet and Web Information Systems	WWW	1.539
	Service Oriented Computing and Applications	SOCA	_
	International Journal of High Performance Computing and Networking	IJHPCN	_
	International Journal of Internet Protocol Technology	IJIPT	_
	Enterprise Information Systems	EIS	2.269
	Research Journal of Applied Sciences, Engineering and Technology	RJASET	_
	International Journal of Engineering and Technology	IJET	_
	Journal of Networks	JNW	_

	International Journal of Web Engineering and Technology	IJWET	_
	Information Technology Journal	ITJ	_
	International Journal of Agent-Oriented Software Engineering	IJAOSE	_
	Journal of Intelligent Information Systems	JIIS	1
	IEEE Transactions on Network and Service Management	TNSM	3.134
	Information and Software Technology	IST	2.694
	IEEE Transactions on Knowledge and Data Engineering	TKDE	3.438

B.3. Methods for Web Services Discovery and Recommendation with Details

The methods including techniques, algorithms, features, cited description language and employed selection characteristic (functional and non-functional) investigated and studied by the final selected papers are illustrated in Table B-2.

Table B-2: Final Selected Studies' Methods for Web Services Discovery and Recommendation

Techniques	Algorithm	FS#	Year	Features	population	Functional	Non-Functional
Clustering-Discovery	K-medoids	[FS43]	2011	WordNet, Levenshtein Distance, Ontology	OWL-S	+	-
	K-means	[FS32]	2016	NLP, Tags, TD-ATM, PW, Wikipedia	short text	+	-
	HAC	[FS15]	2014	NLP, WordNet, TFIDF, Ontology, CAS,	WSDL	+	-
	HAC	[FS34]	2012	NLP, Ontology, WordNet	WSDL	+	-
	HAC	[FS36]	2014	NLP, WordNet, Tags, crawler-based	WSDL	+	-
	K-means	[FS44]	2012	NLP, WordNet, ConceptNet	WSDL/WADL	+	-
	K-means	[FS25]	2009	Cosine, TFIDF, service collection, SVD	WSDL	+	-
	quality threshold	[FS45]	2013	NLP, NGD, context-aware, mobile environments	WSDL	+	-
	quality threshold	[FS24]	2010	NLP, NGD	WSDL	+	-
	HAC	[FS8]	2014	NLP, Ontology, CTM, FCA, Apache Lucene	WSDL	+	-
	HAC	[FS18]	2015	TFIDF, WordNet, Ontology	OWL-S	+	-
	HAC	[FS27]	2014	Ontology Learning, TFIDF, WordNet, Ontology	WSDL	+	-
	HAC	[FS37]	2014	Ontology Learning, TFIDF, WordNet, Ontology	WSDL	+	-
	K-means	[FS19]	2014	NLP, TFIDF, co-clustering, bipartite graph	WSDL	+	-
	Self-Join RDB Clustering	[FS38]	2015	Relational databased self-join operation, ontology	OWL-S	+	-
	HAC	[FS28]	2009	Euclidean distance, Cosine, search engine	WSDL	+	-
	K-means	[FS39]	2015	NLP, TFIDF, Apache lucene, CE-RSR	WADL	+	-

	K-medoids, HAC	[FS29]	2015	Ontology, PAM, business process integration	OWL-S	+	-
	K-means	[FS12]	2016	NLP, Tags, DT-LDA, PW, Wikipedia	short text	+	-
	DBSCAN	[FS5]	2016	Ontology, TFIDF, matching, ranking	OWL-S	+	-
	K-means	[FS47]	2015	BN-LFF, add information in LFF space	WSDL	+	-
	HAC	[FS48]	2014	NLP, TFIDF, Cosine, WordNet	WSDL	+	-
	neural network	[FS13]	2013	WordNet, LSI, Cosine, TFIDF, Mahalanobis distance	WSDL	+	-
	PSO	[FS23]	2011	Ontology, DoM, optimization problem	SAWSDL	+	-
	self-organizing based clustering	[FS31]	2011	Ontology, taxonomic clustering, SGPS	OWL-S	+	-
	Tree-Traversing Ant (TTA) clustering	[FS41]	2012	NLP, NGD, n° W, search engine	WSDL	+	-
	K-means	[FS42]	2008	NLP, TFIDF, Cosine, Euclidean distance, PLSA	WSDL	+	-
	K-means HAC	[FS51] [FS52]	2016 2017	NLP, Mashup services, LDA topic model, Tags, topic clustering	Text	+	-
	HAC	[FS55]	2017	NLP, ontology-generation, specific terms	OWL-S	+	-
	K-means++	[FS56]	2017	Word vectors, LDA, topic clustering	WSDL	+	-
Clustering-Recommendation	K-means	[FS1]	2016	QoS credibility, CF, PCC, user clustering, service clustering	QoS	-	+
	HAC	[FS17]	2015	Data sparsity, clustering QoS prediction, PCC, user similarity	QoS	-	+
	HAC	[FS4]	2014	personalized recommendation, CF, users location, PCC	QoS	-	+
	K-means	[FS3]	2014	NLP, tag recommendation, NGD, tag relevance	WSDL	+	-
	K-means	[FS9][FS26]	2014	QoS recommendation, matrix factorization, location based	QoS	-	+
	K-means	[FS20]	2015	PCC, Cosine, Euclidean distance, CF, customers clustering	WS	+	-
	HAC	[FS21]	2015	WordNet, Cosine, TFIDF, matrix factorization	OWL-S	+	-
	K-means	[FS30]	2014	RCM, Cosine, CF, PCC, WordNet, QoS prediction	QoS,WS	+	+
	K-means	[FS11]	2016	CluCF, PCC, data sparsity, location user clustering	QoS	-	+
	K-means	[FS46]	2015	mining Web services, heterogonous feature selection	WSDL,Tags	+	-
	HAC	[FS40]	2012	Real time QoS, PCC, user clustering, Qos predication	QoS	-	+
	HAC	[FS49]	2017	Hybrid Term Similarity, Euclidean distance	WSDL, OWL-S	+	+
	Fuzzy C-Means	[FS50]	2017	PCC, matrix factorization, NGD	QoS, WSDL	+	+
	HAC	[FS57]	2017	Tag recommendation, SVM, Jaccard, active learning	Text	+	-
Association-Discovery	Apriori algorithm	[FS6]	2014	intensional knowledge mining, intensional querying system	WSDL	+	-
	Apriori algorithm	[FS54]	2017	Web Service Model Extraction, interface underlying semantic mining, co-occurrence	WSDL/WADL	+	-
Association-Recommendation	Apriori algorithm	[FS16]	2009	CF, PCC, service composition, ranking	Movielens	-	-
	Apriori algorithm	[FS10]	2015	CF, PCC, usage history, user profiling	Movielens	-	-

	Apriori algorithm	[FS53]	2016	Graph representation, relationship tracking,	WS-PW	-	+
Combined	K-means	[FS7]	2014	NLP, Carrot search, Tags recommendation, Cosine	WSDL	+	-
	K-means	[FS22]	2014	NLP, TFIDF, Cosine, PCC, CF, QoS predication	WSDL, QoS	+	+
	HAC, GARC	[FS35]	2016	TFIDF, WordNet, Ontology, Association factor	WSDL, OWL-S	+	+
	Defined	[FS14]	2013	tags, social network relationships, WordNet, algorithm	QoS	-	+
	HAC, Hyperclique patterns	[FS33]	2012	NLP, WordNet, SUMO, LSI, Cosine	WSDL	+	-
	K-means, Defined AR	[FS2]	2014	NLP, NGD, Tags, TFIDF, Jaccard coefficient	WSDL	+	-

B.4. Similarity Measures vs. Algorithms

Syntactic and semantic similarity measures relating to clustering and association rules algorithms considering the final selected papers are illustrated in Table B-3

Table B-3: Similarity Measures vs. Algorithms

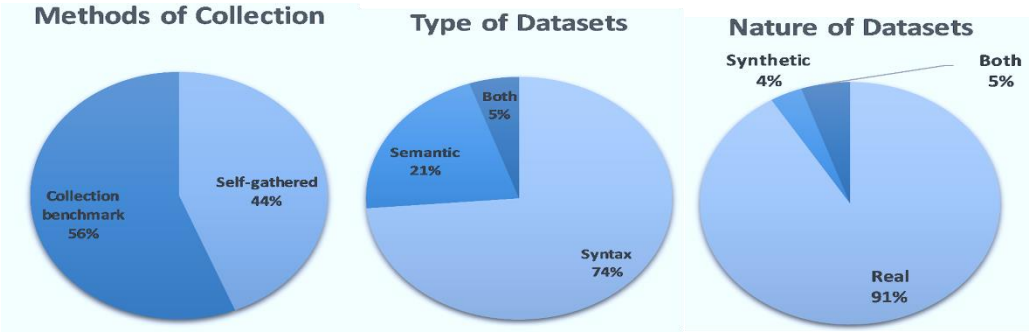
		Algorithms													
		k-medoids	**K-means**	**HAC**	**DBSCAN**	**QT**	**neural network clustering**	**PSO clustering algorithm**	**self-organizing clustering**	**Self-Join RDB Clustering**	**tree-traversing ant**	**Hyperclique patterns**	**Apriori algorithm**	**GARC**	**defined correlation**	**Fuzzy c-mean**
Similarity Measures	**WordNet**	[FS43]	[FS44] [FS30]	[FS15] [FS34] [FS35] [FS36] [FS27] [FS21] [FS48] [FS18] [FS55]										[FS35]	[FS14]	
	Cosine similarity		[FS7] [FS25] [FS20] [FS30] [FS42]	[FS33] [FS28] [FS48]			[FS13]					[FS33]				
	PCC		[FS1] [FS17] [FS20] [FS30] [FS11] [FS22]	[FS42] [FS4] [FS40]									[FS16] [FS10]			[FS50]
	NGD		[FS3] [FS2]	[FS55]		[FS24] [FS45]					[FS41]				[FS2]	[FS50]
	TFIDF	[FS43]	[FS19] [FS39] [FS2] [FS22] [FS42]	[FS15] [FS27] [FS37] [FS21] [FS48] [FS55]			[FS13]							[FS35]	[FS2]	[FS50]

Ontology	[FS43] [FS29]		[FS15] [FS34] [FS35] [FS27] [FS29] [FS18] [FS55]	[FS5]			[FS23]	[FS31]	[FS38]				[FS35]		
LD	[FS43]														
n ° of Wikipe-dia										[FS41]					
Euclidean dis-tance		[FS20] [FS56]	[FS28]												
jaccard coeffi-cient		[FS2] [FS51]	[FS51] [FS57]											[FS2]	
Mahalanobis distance						[FS13]									

B.5. Cited Datasets in the Final Selected Papers

The cited datasets considering the method of collections, types and nature of datasets based on the final selected papers are illustrated in Table B-4.

Table B-4: Cited Datasets in the Final Selected Papers

Final Selected Papers		
		
Methods of Collection		
Published Benchmarks	32	[FS7] [FS16] [FS25] [FS1] [FS8] [FS17] [FS3] [FS46] [FS4] [FS18] [FS26] [FS9] [FS37] [FS38] [FS10] [FS29] [FS2] [FS30] [FS11] [FS5] [FS47] [FS48] [FS22] [FS14] [FS40] [FS23] [FS6] [FS42] [FS49] [FS50] [FS52] [FS55]
Self-Gathered	25	[FS43] [FS32] [FS15] [FS33] [FS34] [FS35] [FS36] [FS44] [FS45] [FS27] [FS20] [FS28] [FS39] [FS19] [FS21] [FS12] [FS13] [FS31] [FS24] [FS41] [FS51] [FS53] [FS54] [FS56] [FS57]
Type of Datasets		
Syntax	42	[FS32] [FS15] [FS7] [FS33] [FS34] [FS16] [FS36] [FS44] [FS25] [FS45] [FS1] [FS17] [FS3] [FS46] [FS4] [FS26] [FS9] [FS27] [FS19] [FS20] [FS28] [FS39] [FS10] [FS2] [FS30] [FS11] [FS12] [FS47][FS22] [FS13] [FS14] [FS40] [FS24] [FS41] [FS42]
Semantic	12	[FS43] [FS8] [FS18] [FS38] [FS29] [FS21] [FS5] [FS48] [FS23] [FS31] [FS6] [FS55]
Both	3	[FS35] [FS37] [FS49]
Nature of Datasets		
Real	52	[FS32] [FS15] [FS7] [FS33] [FS34] [FS35] [FS36] [FS44] [FS25] [FS45] [FS1] [FS8] [FS17] [FS3] [FS46] [FS4] [FS18] [FS26] [FS9] [FS27] [FS37] [FS19] [FS38] [FS20] [FS28] [FS39] [FS29] [FS2] [FS21] [FS11] [FS12] [FS5] [FS47] [FS48] [FS22] [FS13] [FS14] [FS40] [FS23] [FS31] [FS24] [FS41] [FS6] [FS42] [FS50] [FS51] [FS52] [FS53] [FS54] [FS55] [FS56] [FS57]
Synthetic	2	[FS43] [FS16]
Both	3	[FS10] [FS30] [FS49]

Appendix C. Word Embedding Models

This appendix includes the representation of the word "unit" as a dense vector for each of the word embedding models, including the pre-trained Word2Vec skip-gram model, the pre-trained GloVe model, and the self-trained Word2Vec model. Furthermore, distance and similarity matrices built on different word representation and embedding models are presented.

C.1. Representation of Words based on Word Embedding Models

The word 'unit' represented based on the three word embedding is illustrated in Figure C-1, Figure C-2 and Figure C-3.

```
1 model1['unit_NOUN']
```

executed in 13ms, finished 13:05:45 2020-07-17

```
array([ 9.70490798e-02,  9.43083018e-02, -9.54553112e-02,  2.15089228e-02,
        5.13491333e-02, -5.02264462e-02,  2.37302423e-01, -6.75557330e-02,
       -7.07064793e-02,  7.81763792e-02,  8.96642581e-02,  1.34201601e-01,
       -9.64834243e-02,  2.50221580e-01,  3.61892313e-01,  2.07465202e-01,
        6.59910496e-03, -3.38062555e-01, -3.32866400e-01,  1.35102063e-01,
        8.08769464e-02, -3.87578197e-02,  5.42281754e-02, -1.12210371e-01,
        1.80358276e-01, -3.28130163e-02, -6.70738444e-02,  1.17699588e-02,
       -3.32311869e-01, -7.38676041e-02,  3.38359945e-03, -3.20915729e-01,
       -3.08224827e-01,  1.31829590e-01, -8.27066004e-02, -1.88393846e-01,
       -1.88418537e-01, -4.94246334e-01,  1.17577696e-02, -4.48354691e-01,
       -6.67556524e-02, -5.68489507e-02, -1.55068710e-01, -2.42539451e-01,
       -2.45374769e-01, -3.25225703e-02,  1.88731149e-01,  2.32443623e-02,
        2.84736902e-01, -9.02696475e-02, -3.82072069e-02, -1.82331100e-01,
       -1.70726940e-01,  5.81693947e-01, -1.40425175e-01, -5.32541990e-01,
       -2.17522606e-02,  1.22547604e-01,  1.03679575e-01,  3.57627235e-02,
       -4.84358400e-01,  2.47585863e-01,  5.69253005e-02,  2.06978366e-01,
        9.93692577e-02, -1.96113750e-01,  2.76026279e-02, -2.76740566e-02,
       -3.93853426e-01,  1.43301651e-01, -1.23076521e-01, -3.44702527e-02,
       -1.86399389e-02, -3.04455727e-01, -2.00088769e-01,  9.86017007e-03,
```

Figure C-1: "Unit" Word Embedding from the Pre-trained Word2Vec Model

```
1 model2['unit']
```

executed in 20ms, finished 13:07:06 2020-07-17

```
array([-1.982410e-01, -3.026140e-01, -2.466410e-01,  2.513800e-01,
        7.237350e-01, -4.947700e-02,  3.765860e-01, -1.895450e-01,
        1.452650e-01, -3.234160e-01,  3.420000e-02,  1.054910e-01,
       -1.844050e-01, -4.882900e-01, -3.732550e-01,  4.900350e-01,
       -2.967000e-03,  3.378700e-02, -5.046000e-01, -2.032110e-01,
        2.719190e-01,  1.288050e-01,  9.567160e-01,  4.400270e-01,
       -1.334300e-02, -8.670000e-04,  5.085740e-01, -3.380100e-02,
        3.615100e-02,  5.880210e-01, -5.049520e-01, -3.543370e-01,
       -5.946720e-01,  2.728360e-01, -2.739170e-01, -6.160390e-01,
       -2.304000e-01, -3.589480e-01, -4.888390e-01,  1.853450e-01,
       -2.714790e-01,  2.662790e-01,  6.142560e-01,  6.458000e-02,
        2.066360e-01, -1.711890e-01, -3.821150e-01, -2.395230e-01,
        3.462470e-01,  1.392400e-02, -5.220510e-01, -7.190220e-01,
        1.215180e-01,  4.918430e-01,  2.143870e-01, -1.775210e-01,
       -1.001900e-02,  2.337300e-02, -2.241180e-01,  3.464760e-01,
        2.801960e-01,  4.107530e-01,  1.614360e-01, -1.380410e-01,
        1.676950e-01, -4.280020e-01, -8.250700e-01,  1.135570e-01,
        1.778150e-01,  1.671530e-01,  4.205000e-01,  3.592900e-02,
        4.928210e-01, -3.151800e-02,  2.379680e-01,  7.326850e-01,
```

Figure C-2: "Unit" Word Embedding from the Pre-trained GloVe Model

```
1 trained_model.wv['unit']
```

executed in 20ms, finished 19:43:32 2020-07-24

```
array([-1.3905725e+00,  3.5216406e-01, -5.0447983e-01,  6.3211864e-01,
        3.4596404e-01,  1.4560655e+00, -8.2868062e-02, -3.6012420e-01,
        1.3983168e-01, -4.3400306e-01, -8.7092096e-01, -6.9388855e-01,
       -8.1086910e-01, -3.1267238e-01,  1.1449205e+00, -5.3423411e-01,
        1.8320778e-02, -1.0450894e+00,  1.9505690e-01,  1.6374694e+00,
        5.0110100e-03, -5.1211381e-01, -1.3888002e-03, -1.3940622e+00,
        9.6946079e-01, -3.3564633e-01,  5.1228422e-01, -1.1562469e+00,
       -2.9662457e-01,  4.2863613e-01, -6.5605104e-01,  3.3084702e-01,
       -1.0380832e-01, -3.3198214e-01,  1.5659845e-01,  1.1244955e+00,
        2.0878108e-01, -2.6626924e-01,  6.0519885e-02, -8.6116183e-01,
        5.0908327e-01,  3.1574053e-01, -1.1910938e+00,  6.2723860e-02,
        3.1075427e-01,  4.6630722e-01, -1.0781899e-01,  3.3335024e-01,
        1.4579655e+00, -1.9695614e-01], dtype=float32)
```

Figure C-3: "Unit" Word Embedding from the Self-trained Model

C.2. Web Services Distance and Similarity Matrices

Web services similarity and distance matrices based on the various syntactic and semantic similarity measures and considering the employed word representation and embedding models are presented in this section.

	Angle Unit	Area Unit	Transform	Computer Unit	Advanced Travel Information	Schedule	Flight Status	SBG Get Air Fare Quote Service
Angle Unit	0.000000	1.312679	1.414214	1.214127	1.414214	1.414214	1.414214	1.414214
Area Unit	1.312679	0.000000	1.414214	1.305556	1.414214	1.414214	1.414214	1.414214
Transform	1.414214	1.414214	0.000000	1.376404	1.343317	1.414214	1.414214	1.384713
Computer Unit	1.214127	1.305556	1.376404	0.000000	1.414214	1.414214	1.414214	1.414214
Advanced Travel Information	1.414214	1.414214	1.343317	1.414214	0.000000	1.363792	1.337474	1.355007
Schedule	1.414214	1.414214	1.414214	1.414214	1.363792	0.000000	1.097608	1.414214
Flight Status	1.414214	1.414214	1.414214	1.414214	1.337474	1.097608	0.000000	1.414214
SBG Get Air Fare Quote Service	1.414214	1.414214	1.384713	1.414214	1.355007	1.414214	1.414214	0.000000

Figure C-4: Euclidean Distance Matrix based on the TFIDF Model

	Angle Unit	Area Unit	Transform	Computer Unit	Advanced Travel Information	Schedule	Flight Status	SBG Get Air Fare Quote Service
Angle Unit	0.000000	2.013247	2.347744	2.263176	2.257073	2.822002	2.662478	2.748320
Area Unit	2.013247	0.000000	2.453401	1.932727	2.101540	2.605395	2.488815	2.677902
Transform	2.347744	2.453401	0.000000	2.090112	1.567079	2.346221	2.255037	2.550119
Computer Unit	2.263176	1.932727	2.090112	0.000000	1.993008	2.659937	2.521151	2.730831
Advanced Travel Information	2.257073	2.101540	1.567079	1.993008	0.000000	1.906998	1.572194	1.942204
Schedule	2.822002	2.605395	2.346221	2.659937	1.906998	0.000000	1.034818	2.375893
Flight Status	2.662478	2.488815	2.255037	2.521151	1.572194	1.034818	0.000000	2.131564
SBG Get Air Fare Quote Service	2.748320	2.677902	2.550119	2.730831	1.942204	2.375893	2.131564	0.000000

Figure C-5: Euclidean Distance Matrix based on the Pre-trained Word2Vec Model

	Angle Unit	Area Unit	Transform	Computer Unit	Advanced Travel Information	Schedule	Flight Status	SBG Get Air Fare Quote Service
Angle Unit	0.000000	0.515538	1.109015	0.519274	1.069281	1.134835	1.111273	1.147967
Area Unit	0.515538	0.000000	1.076693	0.511805	1.042811	1.064671	1.075253	1.120871
Transform	1.109015	1.076693	0.000000	1.029974	1.021854	1.085300	1.093832	1.134103
Computer Unit	0.519274	0.511805	1.029974	0.000000	1.036297	1.108733	1.091006	1.128592
Advanced Travel Information	1.069281	1.042811	1.021854	1.036297	0.000000	1.016879	0.983570	1.006743
Schedule	1.134835	1.064671	1.085300	1.108733	1.016879	0.000000	0.436817	1.044445
Flight Status	1.111273	1.075253	1.093832	1.091006	0.983570	0.436817	0.000000	1.009156
SBG Get Air Fare Quote Service	1.147967	1.120871	1.134103	1.128592	1.006743	1.044445	1.009156	0.000000

Figure C-6: WMD Distance Matrix based on the Pre-trained GloVe Model

	Angle Unit	Area Unit	Transform	Computer Unit	Advanced Travel Information	Schedule	Flight Status	SBG Get Air Fare Quote Service
Angle Unit	0.000000	0.334807	1.047892	0.374349	1.020409	1.074825	1.111518	1.066190
Area Unit	0.334807	0.000000	1.031723	0.448741	0.973656	1.034305	1.081779	1.038490
Transform	1.047892	1.031723	0.000000	0.991419	0.846657	1.000248	1.012155	1.042300
Computer Unit	0.374349	0.448741	0.991419	0.000000	0.983285	1.059601	1.105798	1.030547
Advanced Travel Information	1.020409	0.973656	0.846657	0.983285	0.000000	0.860013	0.860642	0.852658
Schedule	1.074825	1.034305	1.000248	1.059601	0.860013	0.000000	0.266463	0.964469
Flight Status	1.111518	1.081779	1.012155	1.105798	0.860642	0.266463	0.000000	0.997041
SBG Get Air Fare Quote Service	1.066190	1.038490	1.042300	1.030547	0.852658	0.964469	0.997041	0.000000

Figure C-7: WMD Distance Matrix based on the Self-trained Word2Vec Model

	Angle Unit	Area Unit	Transform	Computer Unit	Advanced Travel Information	Schedule	Flight Status	SBG Get Air Fare Quote Service
Angle Unit	1.000000	0.755274	0.544027	0.787571	0.587569	0.551059	0.554189	0.469883
Area Unit	0.755274	1.000000	0.457099	0.847114	0.469770	0.501615	0.463936	0.462023
Transform	0.544027	0.457099	1.000000	0.465214	0.600585	0.633771	0.608120	0.512021
Computer Unit	0.787571	0.847114	0.465214	1.000000	0.510385	0.527329	0.484491	0.427794
Advanced Travel Information	0.587569	0.469770	0.600585	0.510385	1.000000	0.619050	0.627647	0.540717
Schedule	0.551059	0.501615	0.633771	0.527329	0.619050	1.000000	0.931522	0.469373
Flight Status	0.554189	0.463936	0.608120	0.484491	0.627647	0.931522	1.000000	0.456369
SBG Get Air Fare Quote Service	0.469883	0.462023	0.512021	0.427794	0.540717	0.469373	0.456369	1.000000

Figure C-8: WUP Similarity based on WordNet