INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA



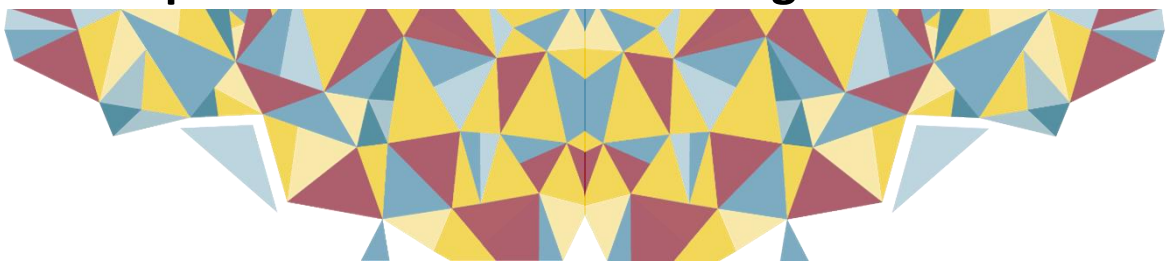

Plataformas de serviços web de aprendizagem automática com registo, pesquisa e composição semântica

RUI GONÇALVES DE CARVALHO
Setembro de 2024

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Platforms for registration, discovery, and semantic composition of machine learning web services



Rui Gonçalves de Carvalho

Dissertation for obtaining the Master's Degree in Computer Engineering, Area of Specialization in Information and Knowledge Systems

Supervisor: Professor Zita Vale

Co-supervisor: Gabriel Santos, Ph.D.

Porto, September 2024

Abstract

Power and Energy Systems have undergone significant changes in the past two decades. While these changes have advanced the industry, they have also introduced uncertainty and fluctuations in the power and energy systems, weakening their security and reliability. To address these challenges, the sector has undergone substantial restructuring, resulting in the emergence of new participants and the introduction of novel market and negotiation models. Consequently, consumers have assumed a more active role within the power and energy industry. Simultaneously, advanced methodologies, such as data mining and agent-based simulation, are being employed to optimize market outcomes. In this context, individual agents can be seen as service providers in Service-Oriented Architectures, with Multi-Agent Systems being capable of solving more complex tasks. Regarding the implementation of these strategies, a semantic approach to Multi-Agent Systems in the context of smart grids can unburden the load of designing and composing atomic services. However, executing complex tasks using web services often requires composing several simple atomic services through streams, which can be a time-consuming task since it is necessary to find and link the outputs and inputs of the services, whose descriptions are often not very clear. Concerning this issue, over time, technologies associated with the Semantic Web have been progressively applied. These technologies enable the description of services in a semantic and comprehensible manner, facilitating understanding for both users and machines. By developing a system for semantic services registration which features registration and search and composition tools, accompanied by a system for publishing scripts as web services, the stated problems can be addressed. Considering the need to frequently add new services to the database by users with varying levels of expertise, it became clear that integrating natural language processing (NLP) and machine learning (ML) techniques would allow users to find semantic Web services tailored to their needs through keyword searches. As such, this work describes the development of the systems as well as the development of the NLP and ML modules. In the development of these modules, an exploratory analysis of ML and NLP techniques was conducted allowing to achieve the expected classification, search and composition results. These exploratory procedures and corresponding results are also described and discussed.

Keywords: machine learning, multiagent-systems, power and energy systems, semantic-web, web-services.

Resumo

Os sistemas energia elétrica passaram por mudanças significativas nas últimas duas décadas. Embora estas mudanças tenham feito avançar a indústria, resultaram também em incerteza e variabilidade, enfraquecendo a segurança e confiabilidade. Para resolver estes problemas, o setor passou por uma reestruturação significativa, verificando-se a emergência de participantes e a introdução de novos modelos de mercado e de negociação. Consequentemente, por um lado, os consumidores assumiram um papel mais ativo na indústria energética. Por outro, diferentes abordagens de prospeção de dados e simulação baseada em agentes são utilizadas no sentido de obter os melhores resultados de mercado possíveis. Neste âmbito, os agentes podem ser vistos como provedores de serviços em Arquiteturas Orientadas a Serviços, permitindo a resolução de tarefas mais complexas por Sistemas Multiagentes. Em relação à implementação destas estratégias, uma abordagem semântica para Sistemas Multiagentes no contexto de redes inteligentes pode reduzir o esforço na definição e composição de serviços atómicos. Contudo, a execução de tarefas complexas usando serviços web requer frequentemente a composição de vários serviços atómicos simples através de fluxos. Esta pode ser uma tarefa demorada, já que é necessário encontrar e vincular inputs e outputs de serviços, cujas descrições muitas vezes não são muito claras. Relativamente a esta questão, ao longo do tempo, as tecnologias associadas à Web Semântica foram sendo progressivamente aplicadas a, permitindo a descrição de serviços de forma semântica e compreensível tanto para utilizadores e como para máquinas. Neste sentido, os problemas supracitados podem ser resolvidos com o desenvolvimento de um sistema de registo de serviços semânticos que apresente ferramentas de registo, pesquisa e composição, acompanhado por um sistema de publicação de scripts como serviços web. Adicionalmente, tendo em conta, a necessidade de adicionar frequentemente novos serviços ao sistema de registo de serviços semânticos por utilizadores com diversos níveis de conhecimento, tornou-se claro que a integração de técnicas de processamento de linguagem natural (PNL) e aprendizagem de máquina (ML) permitiria aos utilizadores encontrar serviços semânticos adaptados às suas necessidades através de pesquisas assentes em palavras-chave. Como tal, este trabalho descreve o desenvolvimento dos sistemas, bem como o desenvolvimento dos módulos PNL e ML. No desenvolvimento destes módulos foram realizadas análises exploratórias que permitiram alcançar os resultados esperados de classificação, pesquisa e composição. Estes procedimentos exploratórios e correspondentes resultados são igualmente descritos e discutidos neste documento.

Palavras-chave: aprendizagem de máquina, serviços web, sistemas de energia, sistemas multiagente, web semântica.

Acknowledgements

I would like to take this moment to express gratitude to everyone who has crossed paths with me throughout this journey and contributed to nurturing my curiosity and passion for learning. Each interaction, whether small or profound, has played an important role in shaping my perspective and fueling my enthusiasm for knowledge.

A special note of appreciation goes to my Advisor, Professor Zita Vale, and my co-Advisor, Dr. Gabriel Santos. Your expertise and mentorship not only helped steer this project in the right direction but also fostered a sense of confidence in my own abilities. I am grateful for all the time, patience, and insightful advice and revisions.

To my GECAD co-workers I also leave an acknowledgement of gratitude for the words of support and exchange of ideas. I would like to especially acknowledge Brígida Teixeira, for the thoughtful insights and constructive feedback.

Finally, I would like to thank my family and friends, whose unwavering love, encouragement, and support have been the foundation upon which I have been able to build and pursue my academic and personal goals. Through every challenge and triumph, your belief in me has been my greatest source of strength. Words cannot fully express the depth of my gratitude for all that you have done to make this journey possible.

To all, I extend my deepest gratitude.

Table of Contents

List of Figures

List of Tables

Acronyms

AAMAS	*International Conference on Autonomous Agents and Multiagent Systems*
ABM	*Agent Based Modeling*
AI	*Artificial Intelligence*
API	*Application Programming Interface*
BERT	*Bidirectional Encoder Representations from Transformers*
CBOW	*Continuous Bag of Words*
CF	*Collaborative Filtering*
CPLSA	*Clustered Probabilistic Latent Semantic Analysis Approach*
CPN	*Colored Petri Nets*
DF	*Directory Facilitator*
DM	*Data mining*
DoM	*Degree of Match*
FIPA	*Foundation for Intelligent Physical Agents*
FURPS	*Functionality, Usability, Reliability, Performance, and Security*
HTTP	*Hyper Text Transfer Protocol*
ICD	*nter-cluster distance*
IDeS	*Intelligent Decision Support*
IDF	*Inverse Document Frequency*
IG	*Information gain*
IOPE	*Inputs, Outputs, Pre-conditions and Effects*
IOT	*Internet of Things*
LDA	*Latent Dirichlet allocation*
LDAP	*Lightweight Directory Access Protocol*
LLM	*Large Language Models*
MAS	*Multi agent system*
ML	*Machine learning*
MLP	*Multi-layer Perceptron*
NCD	*New Concept Development*
NFP	*Non-Functional Properties*
NLP	*Natural Language Processing*
OpD	*Operations Discovery algorithm*
OSM	*Output Similarity Model*

OWL	*Web Ontology Language*
P2P	*Peer to peer*
PLSA	*Probabilistic Latent Semantic Analysis approach*
QoS	*Quality of Service*
RDF	*Resource Description Framework*
RDFS	*Resource Description Framework Schema*
SDAE	*Stacked denoising autoencoders*
SIMCR	*Similarity measure integrating multiple conceptual relationships*
SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SSC	*Semantic Services Catalog*
SVM	*Support Vector Machine*
SWTM	*Bi-directional sentence-word topic model*
TF	*Term Frequency*
TF-IDF	*Term Frequency - Inverse Document Frequency*
TF-IDF	*Term frequency – Inverse*
TSM	*Total Similarity Model*
UDDI	*Universal Description, Discovery and Integration*
URI	*Uniform Resource Identifiers*
VSM	*Vector Space Model*
W3C	*World Wide Web Consortium*
WE-LDA	*Word-Embeddings Latent Dirichlet allocation*
WS	*Web Service*
WSD	*Word Sense Disambiguation*
WSDL	*Web Services Description Language*
XGBoost	*Extreme Gradient Boosting*
XML	*Extensive Markup Language*

1 Introduction

This chapter is dedicated to the clarification of this thesis motivation, describing the context in which the objectives and research questions emerge, as well as the main contributions associated with the development of this work.

1.1 Motivation

This thesis builds upon the work presented by (Santos et al., 2021). In this context, power and energy systems have undergone various changes in the past two decades which have led to uncertainty and variation in the system, which has weakened its security and reliability. As a result, a restructuring of the sector has taken place, with new participants coming into play and new market and negotiation models emerging (Niu et al., 2018). Consumers have started to become more active in this area, and different approaches to data mining and agent-based simulation are being used to try and get the best possible outcomes from the market. With this objective in mind, different workflows are being combined to create a variety of different services. In Service-Oriented Architectures, individual agents can be seen as service providers, with Multi-Agent Systems (MAS) acknowledged to be capable of solving more complex tasks (Klusch et al., 2016). Both agents and MAS are capable of providing services, as can be observed in decision support agent-based systems (Pinto et al., 2015). As such, with the rise of self-contained services, the challenge of efficiently discovering relevant services has become critical (Tokmak et al., 2024). As the number of services performing similar tasks grows, users face difficulty in selecting the most appropriate one. Ensuring the fast and accurate delivery of relevant services to users is vital for maintaining an efficient infrastructure, as undiscovered services add unnecessary costs to the ecosystem (Tokmak et al., 2024). Considering a scenario where a system is expected to effortlessly integrate web services and agent-based services, adding semantic properties to agents can greatly improve discoverability given a set of specific requirements (Klusch et al., 2016). In this type of system, it is possible to identify and suggest potential solutions for specific scenarios using queries, such as discovering which machine learning (ML) algorithms can be applied, using the information about their attributes and the anticipated outcomes (Santos et al., 2021). Thus, a semantic approach to MAS in the context of smart grids can unburden the load of designing and composing atomic services. In this context, executing complex tasks using web services often requires composing several simple atomic services through streams. The development of these streams is a time-consuming task, since it is necessary to find and link the outputs and inputs of the services, whose descriptions are often not very clear (Santos et al., 2021). Acknowledging similar difficulties, (Gamha, 2023), emphasizes that these tasks require significant effort, especially as systems evolve towards advanced solutions capable of discovering web services or web APIs. In this regard, the author explores the relationship between service discovery and composition, highlighting that the discovery process is a foundational task. For successful service composition, appropriate services need to be identified at every major step of the process (Gamha, 2023). In this context,

technologies associated with the Semantic Web have been progressively applied to improved web service discovery and composition. These technologies enable the description of services in a semantic and comprehensible manner, facilitating understanding for both users and machines.

In the web ecosystem, intelligent agents are often seen as consumers of these services, but not as their facilitators, especially if integrated into MAS. It is relatively frequent to have MAS whose tasks depend on the execution of previous ones, made available not only by several MAS, but also by web services (Santos et al., 2021). However, there is no system that allows encapsulating its activities as if they were services, to allow for their composition in more complex simulations. Additionally, the publication of machine learning algorithms as web services, developed in several open access programming languages (e.g., Python, R, Prolog), with the objective of being available to different MAS and/or APIs is also a time consuming and thorough work. This process of algorithm publishing can also be streamlined and optimized through automation, reducing the probability of error.

Thus, two development opportunities arise from this background. The first opportunity concerns the search of web services having in mind the eventual necessity of a web service composition depending on the user's request. The second opportunity is focused on the publication of algorithms as web services. The identification of these opportunities prompts the idea of two systems working together. One that allows services registration and search, establishing a catalog for semantic service. And another that provides a framework for the publication of algorithms as web services as well as their registration in the aforementioned catalog.

The first system was proposed and implemented as shown in (Santos et al., 2021), presents the Semantic Services Catalog (SSC) for MAS societies as a solution for services registration and search. SSC is a platform for registration and search where tools can register to make service available or look for a certain kind of service to achieve their objectives. By offering a service where each registered tool describes its capabilities, the kind of service it offers, where it can be reached, what ontologies and languages are supported, and what are the expected inputs and outputs, it facilitates communication between diverse systems. Although it establishes a base for semantic services registration and discovery, it is lacking on the ability to perform an intelligent search where semantic meaning of the search terms is considered and the ability to suggest semantic services compositions in cases where a single service cannot provide the intended results.

1.2 Objectives and Research Questions

In light of the context outlined in the previous section, and taking into consideration the challenges identified, the following research questions have been formulated to guide the development and focus of this work:

- How to discover and compose web services that are registered in the SSC?

- How to streamline the publication of algorithms as web services?

The first question can be addressed by restructuring and upgrading the existing SSC allowing web service and MAS registration, the semantic search of services, administration tools, and service compositions suggestions. The second question can be answered through a system for web services configuration and publication that allows the semantic configuration and registration of the respective service on the system proposed in the response of the first question.

As such, the objectives established regarding the identified context and problems include the restructuring and upgrade of the existing SSC proposed in (Santos et al., 2021) adding semantic services intelligent search and composition functionalities

The objectives also include the implementation of a system for semantic services configuration and publication. The algorithms used by the service are developed in different open access programming languages (e.g., Python, R, Prolog). This platform should allow semantically configuring and registering the respective service on the SSC.

To this end, a set of more specific objectives has been formulated:

- Critical analysis of existing techniques and technologies, determining and justifying which ones will be taken into account for project development.
- Integration of intelligent search and composition in the SSC.
- Design and implementation of the Intelligent Decision Support (IDeS) Framework for algorithm configuration and publishing.
- Data collection and selection.
- Construction and treatment of relevant case studies.
- Analysis of results.

1.3 Main Contributions

The main contributions of this thesis materialized through the SSC and IDeS framework. This work consists in the development of the IDeS framework for the configuration and publication of semantic web services, and in the addition of an intelligent search and composition mechanism to the SSC.

The SSC's architecture was presented in (Canito et al., 2019), which was followed by an implementation and further demonstration presented in (Santos et al., 2021).

In addition to building upon the work presented in (Santos et al., 2021), this thesis also resulted in the submission of two conference papers:

- Rui Carvalho, Gabriel Santos, Zita Vale. *Pre-classification impact on intelligent search of semantic web services.*

- Rui Carvalho, Gabriel Santos, Zita Vale. *Semantic web services discovery and composition in the context of Multi-Agent Systems*

Additionally, the development of this work had applications in the following funded projects:

- MAS Society - Multi-Agent Systems SemantiC Interoperability for simulation and dEcision supporT in complex energY systems (FCT). This work provides a solution for increasing the interoperability between agents by presenting a platform that allows the configuration and publication of algorithms as web services and implementing an intelligent discovery and composition mechanism for the registered web services.
- PRECISE - Power and Energy Cyber-Physical Solutions with Explainable Semantic Learning (FCT). This project contributes to PRECISE, by offering a streamlined and intuitive way to configure and publish the ML algorithms developed in the context of this project.
- TradeRES - Tools for the Design and modelling of new markets and negotiation mechanisms for a ~100% Renewable European Power Systems (H2020). This project's contribution resides on the possibility of publishing all the python scripts developed in the context of TardeRES. As such, a pipeline of bid generation, market simulation, and analysis of results can be achieved by having the aforementioned scripts registered in the SSC.

1.4 Ethical Considerations

To ensure the integrity, transparency, and accountability of this project, it is essential to identify and address ethical issues. Based on these principles, end users will have access to information about the limitations inherent in the techniques and algorithms used in the process of semantic service discovery and composition. Additionally, algorithms that facilitate service discovery must avoid biases that could prioritize certain services unfairly. Furthermore, security measures are employed in order to ensure the privacy of the user's data. When users interact with the system it's crucial to ensure that any data collected is handled in accordance with privacy regulations like the general data protection regulation. Users are to be informed about what data is collected, how it's used, and given explicit consent before any data processing occurs. Lastly, security is paramount, particularly in systems that rely on web service composition. Strong encryption, access control mechanisms, and regular security audits are employed to protect user data and prevent unauthorized access or misuse.

1.5 Document Structure

In this introduction chapter, the context, problem, and objectives are described, allowing to understand the environment in which the problem has arisen and the objectives that were defined to respond to it.

Following the introduction, chapter 2 provides the foundational knowledge necessary to understand the research. It starts by introducing the concept of web services, then moves on

to discuss the Semantic Web and the technologies that support it. The section also covers agents in smart grids, the process of semantic services discovery, and the composition of these services. Additionally, it reviews related work in this field, examining various approaches, including syntactic-based, semantic-based, and pre-classification methods. SSC is also discussed, and the related work is analyzed. The background concludes with an overview of the applied techniques, covering topics such as text pre-processing, text encoding and embedding, vector similarity metrics, classification algorithms, and machine learning evaluation metrics.

Chapter 3 delves into the design and analysis of the system or research. It begins by outlining the requirements, with specific attention to the IDeS framework and SSC. This section also presents various use cases and discusses the system's overall architecture, with a focus on the IDeS architecture. Furthermore, it explores the integration of classification and discovery/composition modules, detailing the interaction between the discovery/composition module and the SSC, as well as between the classification module and the IDeS framework.

Chapter 4 provides a detailed account of the exploratory analysis conducted during the research, along with its results. It starts with the classification procedures, analyzing the results in depth. This is followed by the discovery techniques exploratory analysis and its corresponding outcomes. The chapter concludes with the composition experiments and a thorough analysis of the results obtained from these experiments.

In Classification: Exploratory Process and Results, the document addresses Web service classification, focusing specifically on the processes involved. It discusses both the training stage and the subsequent classification stage, providing insights into how web services are classified.

Chapters 4.3 and 4.4, explore the processes of discovering and composing semantic web services in detail. It outlines the methods used for semantic web service discovery and describes how these services are composed.

The Conclusion chapter summarizes the findings of the research, offering final thoughts and highlighting the key takeaways and future work.

2 Background

This chapter introduces the key concepts related to the technologies that form the foundation of this work.

2.1 Web Services

Service Oriented Computing is an interdisciplinary paradigm that is transforming the structure of distributed software development. Applications that employ a Service-Oriented Architecture (SOA) evolve over their lifetime and adapt more easily to changing and unpredictable environments (Yangui et al., 2021). Closely related with the service-oriented computing paradigm is the concept of web service. (Yangui et al., 2021). A web service is a software system intended to enable interoperable interactions between machines over a network. It is a modular, self-describing, and loosely coupled application that is independent of platform and programming language, allowing it to be published, discovered, and invoked across the Internet (Mosa & Abdelaziz, 2021). Web services use Uniform Resource Identifiers (URI) as a form of identification which can be accessed through the internet using its exposed interface (Yangui et al., 2021). In essence, web services can be described as modular applications that are self-contained and self-described, allowing them to be published, found, and used on the internet through the use of standard Web protocols (Paliwal et al., 2012). A common usage scenario is presented in Figure 1. The three entities involved in the processes of publishing, finding and biding of services are the service requester, the service provider, and the registry (Yangui et al., 2021).

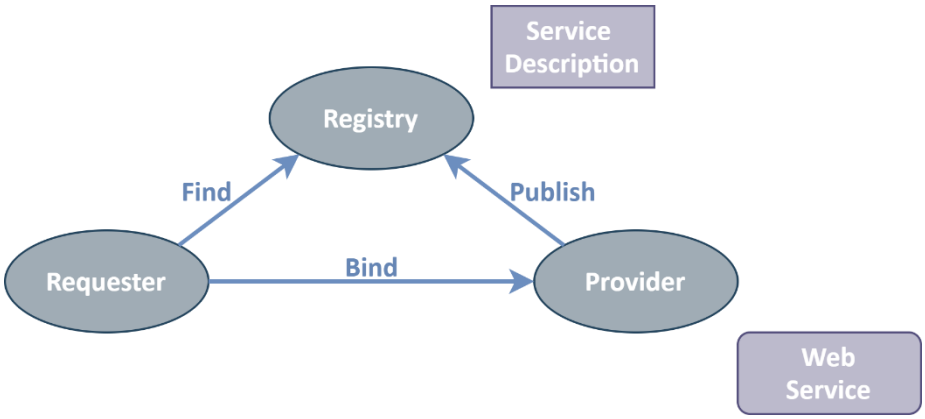


Figure 1 – Web service usage scenario. Adapted from (Cabral et al., 2004).

The service provider creates the web service, defines its description, and publishes it to a service registry. The service requester searches the registry, binds to the chosen service, and invokes the requested service. Finally, the service registry provides a searchable repository of service descriptions (Eid et al., 2008). When the need for a new service arises, the developer searches for the required service either by constructing a query or browsing the registry. Then, the meaning of the interface description is interpreted by the developer by analyzing meaningful

labels or variable names, comments, or additional documentation. The next step is to bind to the discovered service within the application that is being developed. This application that is now using a new web service is referred as the service requester. This service requester is, at this point, able to automatically invoke the discovered service offered by the service provider using web service communication protocols (Cabral et al., 2004). As such, web services can be discovered, invoked, and the composition of several services can be choreographed, using well defined workflow modeling frameworks (Cabral et al., 2004).

Deriving from these functionalities, the concept of composite web service emerges. A composite web service can be viewed as a collection of outsourced services (simple and/or composite) working together to provide a service with added value. An example of a composite service is a car selling service that is associated with car dealers, financing services, and insurance services to offer an integral car sale solution (Elmagarmid, 2016).

2.2 Semantic Web

Due to the increase of web services and lack of semantic parts in web service technologies such as Universal Description, Discovery and Integration (UDDI) and XML-based Web Services Description Language (WSDL) some difficulties have surfaced regarding the process of discovering web services (Çelik & Elçi, 2005) (Çelik & Elçi, 2006). Regarding WSDL, its descriptions incorporate information needed to invoke the services, which can be serialized using HTTP and SOAP protocols. However, WSDL doesn't present the possibility to systematically associate meanings to the messages and message arguments present in a given service description (Bussler et al., 2005). This lack of machine-readable semantics results in a required human intervention for automated service discovery and composition, which hampers their usage in complex business contexts (Cabral et al., 2004) (Kurniawan et al., 2018).

In turn, UDDI is a directory service that offers APIs for service providers to publish their service descriptions, and query APIs for service consumers to search services based on searching criteria(s). The use of UDDI has two main limitations regarding web service search (Bellur & Kulkarni, 2007). The first limitation resides in the fact that, although its mechanisms allow extensible service descriptions, it is limited to a syntax based search (Çelik & Elçi, 2006) (Bellur & Kulkarni, 2007) (Liu et al., 2016). The second limitation resides in the reliance on service providers to create and maintain service descriptions. This is perceived as a burden that has considerably discouraged service providers to use UDDI (Bellur & Kulkarni, 2007).

Besides the previously described search related problems, there are also difficulties in web service composition. These difficulties are related to the fact that the execution of complex tasks often requires the composition of several, atomic services which, in turn, faces various barriers to interoperability (Canito et al., 2019) (Tokmak et al., 2024). Although there is a great number of atomic services available on the web, the design and development of a workflow or composition between them can take a considerable amount of time. After the time taken in the discovery process, developers may realize that they may have different interaction protocols, and lacking description of their workings, inputs and outputs (Canito et al., 2019).

The concept of semantic web was created as response to these problems. It represents a vision of a Web of meaningful contents and services, which can be machine-interpreted. It can also be regarded as a substantial source of information, which can be modelled with the purpose of sharing and reusing knowledge (Cabral et al., 2004). As such, the semantic web represents a set of technologies that enhance communication enabling the interpretation of contextual information and infer new knowledge (Ochoa et al., 2023). In this regard, a semantic service description can be defined as a service's detailed explanation that is written semantically in order to provide a well-defined meaning to a service and it´s parameters so they can be understood by a machine. These descriptions are implemented through the use of semantic annotations. These annotations describe part of an electronic resource using metadata whose meaning is formally specified in an ontology (Kurniawan et al., 2018). Consequently, the Semantic Web provides the infrastructure for the publication of ontological descriptions and provides the necessary means for reasoning about concepts and terms described ontologically (Cabral et al., 2004).

2.3 Semantic Web Technologies

The current components of the Semantic Web framework are the Resource Description Framework (RDF), RDF Schema (RDF-S), the Web Ontology Language (OWL), and SPARQL as a query language (Patel & Jain, 2021). RDF is an XML-based standard from the World Wide Web Consortium (W3C) used for the description of resources on the Web. It adds some semantics to Extensible Markup Language (XML) data by allowing the representation of objects and their relationships through properties (Patel & Jain, 2021).

RDF-S is a schema language that extends RDF by providing a vocabulary to define the types and relationships between resources in RDF data (Patel & Jain, 2021). In this framework, RDF provides the underlying data model for creating triples, while RDF-S offers a way to describe how the nodes in an RDF graph are related through classes and properties. Figure 2 illustrates the RDF and RDF-S layers with a simple example.

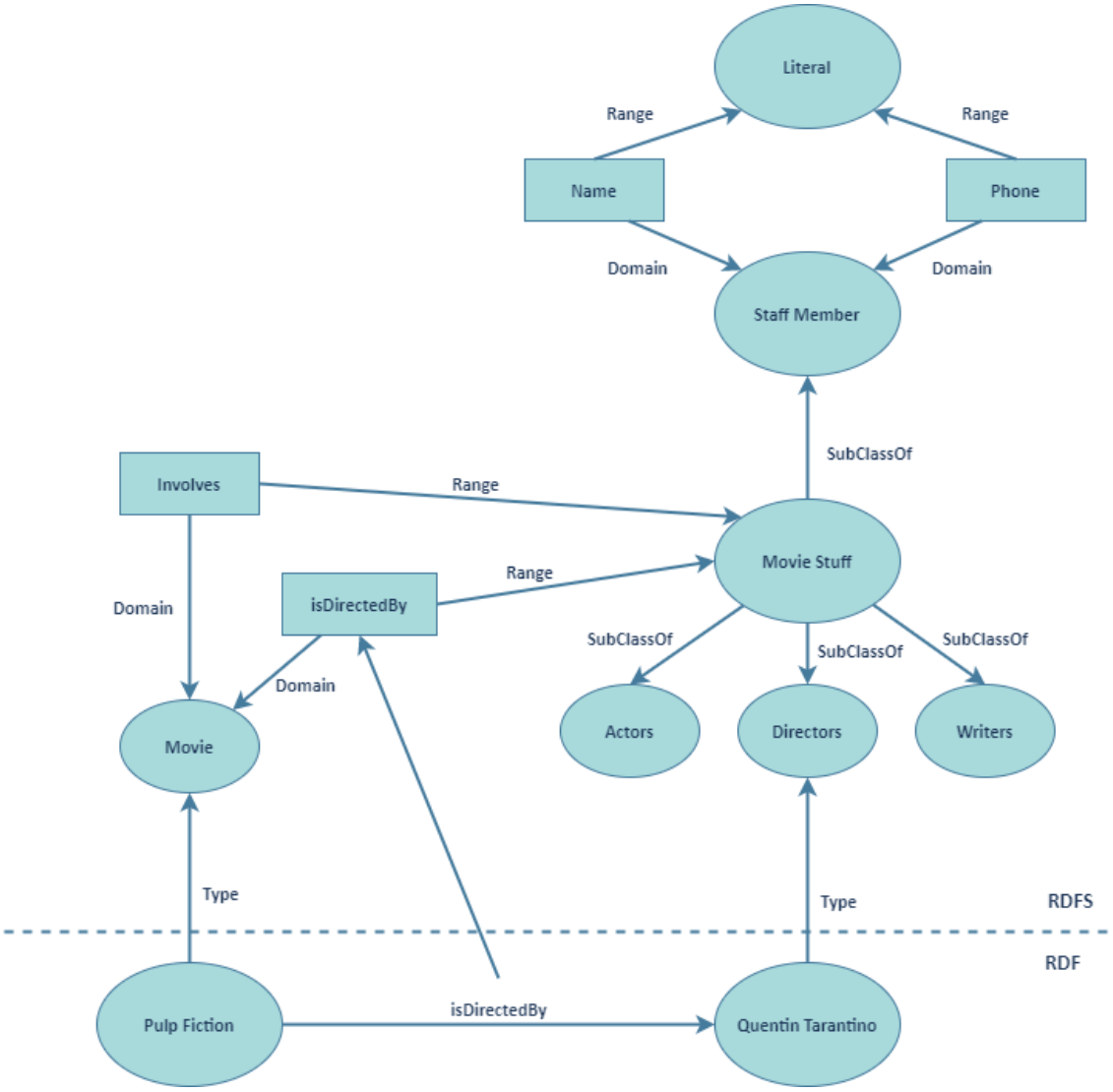


Figure 2 – RDF and RDFS layers example. Adapted from (Costa & Leal, 2013).

Ontologies were developed by the Knowledge Modelling research community aiming the facilitation of knowledge sharing and reusage. They provide a way of understanding the capabilities and limitations of services, as well as the knowledge needed to use them. It allows to combine information from web service standards such as UDDI and WSDL with information about the specific domain the service is dealing with. This would include things like the capabilities of the service, the costs associated with using it, and any knowledge needed to use it correctly (Costa & Leal, 2013). They build upon RDFS providing a source for greater expressivity when modelling domain knowledge, thus allowing to exchange this knowledge between users and heterogeneous and distributed application (Patel & Jain, 2021).

In the context of the semantic web, the service ontology combines all concept models associated to the description of a Semantic Web Service and establishes the knowledge-level model of the information describing and supporting the usage of the service. An ontology is,

thus, a knowledge model that describes a domain of interest using semantic aspects and structure (Ciaramella et al., 2009). It comprises of:

- Facts representing explicit knowledge, encapsulating concepts, their properties, and instances that represent entities described by concepts.
- Axioms and predicates representing implicit knowledge, through the use of rules that add semantics to facts and derive knowledge from them.

The standard for representation of ontologies on the web is defined by OWL ,with OWL-S being a particular OWL ontology created to establish a framework for the semantic description of web services encompassing the discovery, invocation and composition perspectives (Patel & Jain, 2021). In the process of service development, the abstract procedural concepts established by OWL-S ontology can be used in conjunction with the domain specific OWL ontologies which provide the terms, concepts, and relationships used to describe the various service properties (Ngan & Kanagasabai, 2013). In this regard, the OWL-S Service Profile defines a service in terms of its Inputs, Outputs, Pre-conditions and Effects (IOPE) and, in turn, the inputs and outputs presented in this tuple are used to reference the concepts in ontologies published on the Web (Bellur & Kulkarni, 2007). Generally speaking, the OWL-S service profile provides the functional and non-functional information required for service discovery by an agent, while the OWL-S service Model and OWL-S service grounding, in tandem, provide the adequate information for an agent to use a service, once it is found (Ngan & Kanagasabai, 2013). In summary, the OWL-S ontology is composed of three main parts as shown in Figure 3.

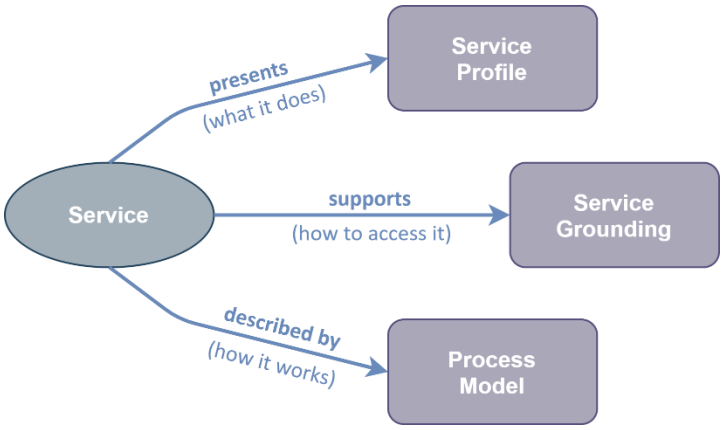


Figure 3 – Top level of the service ontology. Adapted from (*OWL-S*, 2022).

The service profile allows for service advertising and discovery. It describes what the service does, what benefits it offers, and what requirements are necessary to use it. It's designed to help matchmaking agents or service seekers determine whether a service meets its needs (Ngan & Kanagasabai, 2013);

The process model provides a detailed description about the way the service operates by detailing the semantic content of requests, the conditions under which particular outcomes will

occur and, what steps you need to take to achieve those outcomes. For more complex services, the model can be used to figure out what is needed to do to get the service started, what will happen during the service, and how the service will be monitored (Klusch et al., 2016).

Lastly, the service grounding details the protocols used in the communication with the service. This includes specifying the communication protocol, message formats, and other service-specific details such as port numbers used in contacting the service. Furthermore, the service grounding identifies, for each semantic type of input or output specified in the service model, an unequivocal way of exchanging data elements of that type with the service (Ngan & Kanagasabai, 2013).

In essence, ontologies are referenced as playing a central role in the Semantic Web, extending syntactic service interoperability to semantic interoperability (Elmagarmid, 2016).

2.4 Semantic Services and Multi Agent Systems

As previously stated, the semantic web provides the means to augment web services with detailed formal descriptions of their capabilities, therefore enabling automated composition, discovery, binding, and invocation of services. These augmented services (semantic services) have the objective of making web services machine understandable through the use of Semantic Web technologies for web service annotation and processing. In this regard, intelligent agents are able to automatically understand the function of a web service and it´s requirements in order to perform any given task. In short, semantic services can be seen as services that are made machine-understandable by the usage of formal ontologies. These ontologies enable the creation of the service's description. The service's description is composed by a semantic profile and a semantic process model that describes the semantics of the service it is grounded in, in terms of what the service does and how it works (Klusch et al., 2016). The main idea behind semantic services is then, to annotate web services with concepts defined in formal logic-based ontologies in a way that, from an artificial intelligence (AI) perspective, intelligent agents and service-based applications can reason on the service's formal semantic information. Ultimately, this facilitates not only the semantic interoperation between services but their automated logic-based composition planning and search (Klusch et al., 2016).

Within this framework, software agents often provide services in the context of MAS or a single agent-based system, with their interoperability and coordination principles following the same vision that was previously presented regarding the semantic web and semantic services. In the context of MAS, the concepts of agent and MAS require further clarification. According to (Goonatilleke & Hettige, 2022), an agent is an entity placed in a certain environment that senses different parameters that are used to make a decisions based on the goal established by the entity. The entity, then, executes the necessary actions on the environment based on these decisions. Within this frame of reference, the definition of agent reveals four concepts that require further exploration, namely the concepts of entity, environment, parameters, and actions.

An entity refers to the type of agent. It can be a software, e.g., daemon security agents, Internet of Things (IoT) devices, a hardware component, e.g., thermostat, or a combination of both, e.g., a robot (Dorri et al., 2018).

The environment refers to the location of the agent. It can be a network in the case of traffic monitoring agents, or a software application, where the agent is monitoring the actions of software components, for instance. Thus, an agent uses the information sensed from the environment in order to make decisions (Goonatilleke & Hettige, 2022).

Parameters refer to the different types of data that an agent can sense from the environment. For example, the parameters for a soccer robot agent are the position and speed of the ball, team members and opponents (Dorri et al., 2018).

Actions refer to activities that result in changes in the environment. For instance, if a soccer robot kicks the ball, it´s position changes. Actions can aggregate in sets of continuous or discrete actions. The continuous set of actions has no limit to the number of actions performed, whereas the discrete set has a finite number of actions e.g., an agent controlling a thermostat in a room (Dorri et al., 2018).

In turn, MAS technology is a key and promising area that consists of multiple decision-making agents operating in an environment to pursue either common or conflicting goals. The rapid growth and evolution of MAS technology can be attributed to its features, such as flexibility and intelligence, which are particularly effective in addressing complex distributed problems (Goonatilleke & Hettige, 2022).

Based on the concepts, (Dorri et al., 2018) presents a taxonomy of MAS applications which is useful to understand the scope in which these applications are used. The proposed taxonomy includes Computer networks, Robotics, Modelling, City and Built Environments, and Smart grids. A review of these applications is illustrated in Figure 4.

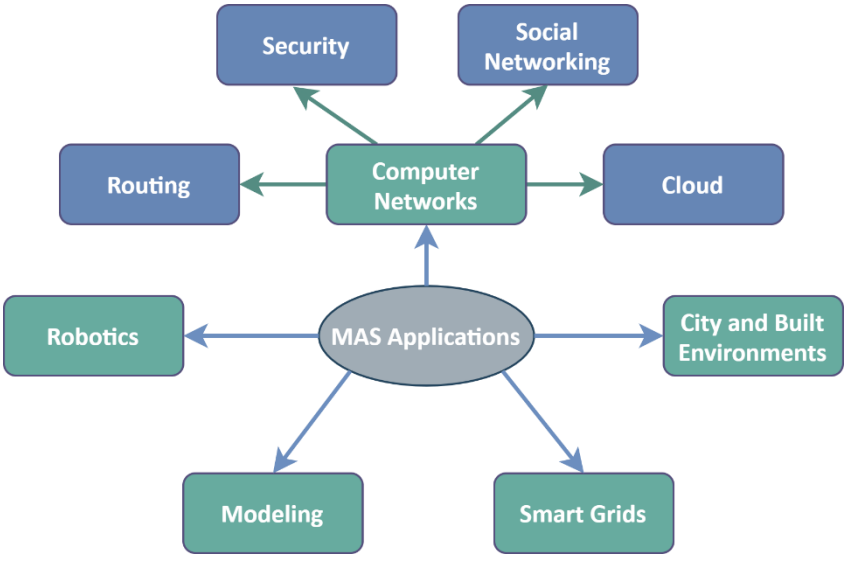


Figure 4 – Taxonomy of MAS applications. Adapted from (Dorri et al., 2018).

In the context of computer networks, the advent of new technologies and propagation of Internet-connected devices significantly increased the complexity. In this regard, agents are broadly used to overcome this complexity. Considering the wide range of applications of MAS in networks, four sub-categories are proposed by (Dorri et al., 2018). Cloud computing, social networking, security, and routing.

The usage of agents in robotics has been a topic of study for over twenty years. The first article found dates back to 1996 and focused on outlining the pros and cons of using agents in robotics (Dudek et al., 1996). According to (Garcia et al., 2013) there are two main challenges in robotics: (1) cooperation and coordination between robots, and (2) planning their movement trajectory. Over the years there have been a considerable number of articles written to address these challenges like the ones referenced in (Rizk et al., 2019).

Complex dynamic systems modelling is especially costly to achieve since they require significant processing overhead due to high complexity and demand for powerful modeling platforms. These problems find an answer in the use of agents. Their flexibility, autonomy, and scalability allow for Agent Based Modeling (ABM) of complex systems with low costs and low resources (Dorri et al., 2018).

Recently, agents have been used in the management of cities and buildings. In fact, a surge in the attention devoted by researchers to this topic was observed by (Dorri et al., 2018). Several topics in this context have been studied. These topics span from the organization and distribution of freights, the control and management of traffic and transportation systems, and the management of buildings, to name a few (Dorri et al., 2018). In this regard, IoT devices can be viewed as agents that perceive their environment and act autonomously to achieve specific goals. In a recent study, (Swapna Choudhary, 2024) propose a model that uses IoT to facilitate real-time data collection and seamless integration with smart grid infrastructures, thereby enabling a more responsive and adaptive energy management system. It employs a decentralized MAS that uses Auction-Based Mechanisms to enhance coordination among various energy resources. This integration of IoT not only promotes effective decentralized management of energy resources but also strengthens the resilience of the power grid, ultimately leading to improved efficiency in energy trading. By combining IoT capabilities with advanced artificial intelligence technologies, this approach can improve the management of renewable energy. It plays a crucial role in advancing global sustainability goals by optimizing how solar power systems operate and contribute to the broader energy ecosystem.

2.4.1 Agents in Smart Grids

Smart grids incorporate modern sensing technologies, control methodologies and advanced communications into the existing power grids in order to transform the current grid into one that functions more cooperatively, responsively and organically (Tuballa & Abundo, 2016). The concept of demand response is an integral part of a smart grid. It provides a way for consumers to use less electricity when the supply is scarce. This helps prevent prices from going up and keeps the electricity grid stable (Malik & Lehtonen, 2016). In this context, agents are used to

respond to several challenges inherent of smart grids. These challenges incorporate the balancing the generated and the demanded energy, the negotiation pricing, energy storage and restoration between energy consumers and producers (Dorri et al., 2018). As a response to these challenges, data mining (DM) approaches and agent-based simulation tools were proposed and proven to be a good match (Klusch et al., 2016).

Individual agents can be seen as service providers, and MAS are capable of solving more complex tasks. Both agents and MAS are capable of providing services, as can be observed in decision support agent-based systems (Pinto et al., 2015). Within the framework of a system that is able to seamlessly integrate web services and agent-based services, adding semantic properties to agents greatly improves their discovery and invocation given a set of specific requirements (Santos et al., 2021). It is then, possible to identify and suggest potential solutions for specific scenarios using queries, such as discovering which ML algorithms can be applied, using the information about their attributes and the anticipated outcomes (Santos et al., 2021). Thus, a semantic approach to MAS in the context of smart grids can unburden the load of designing and composing atomic services. Starting from this assumption, the next chapter focuses on reviewing the state of the art in semantic services discovery and composition.

2.5 Semantic Services Discovery

Dynamic discovery and invocation of Web Services is a key component of Service Oriented Architectures (Bellur & Kulkarni, 2007). Within this frame of reference, a semantic web service can be defined as a tuple $w = \{In_w, Out_w\} \in W$ where In_w is a set of inputs required to invoke w, Out_w is the set of outputs returned by w after its execution, and W is the set of all services available in the service registry. Each input and output is related to a semantic concept from an ontology O ($In_w, Out_w \subseteq O$). Semantic inputs and outputs can be used to discover relevant services as well as to compose the functionality of multiple services by matching their inputs and outputs together (Rodriguez-Mier et al., 2016). From this perspective, the discovery of services is a process that involves the semantic matching of the description of a service request with the description of published services. This matching occurs through the searches made on the semantic registry, using queries that usually include attributes such as, the service name, input, output, and preconditions. Additionally, the matching service can also be used to find the best solutions to tasks or goals, followed by a selection of services that can help achieve these goals. In turn, the degree in which a matching takes place can be based in an established criteria such as the relationship of types (Cabral et al., 2004).

According to (Ngan & Kanagasabai, 2013), existing methods of searching for semantic services can be classified as either centralized or decentralized directory-based, or as decentralized and directory-less. In centralized search, services are supposed to be registered by directory-based search service providers in one central and possibly replicated directory, or multiple distributed service directories at distinct nodes of the network. In turn, these directory nodes inform the service consumers about available services that are present in the network. With this search method, relevant semantic services can be found using contemporary web search engines,

specialized semantic web service search engines, or dedicated and authoritative semantic web service directories that include a query interface.

A possible downside of this approach is the possible existence of a single point of failure and performance bottleneck due to a central semantic service directory (Klusch et al., 2016). As an example, Figure 5 shows a possible scenario of a centralized semantic e-commerce system which uses a product ontology to provide a central source of information about product groups and their specific attributes.

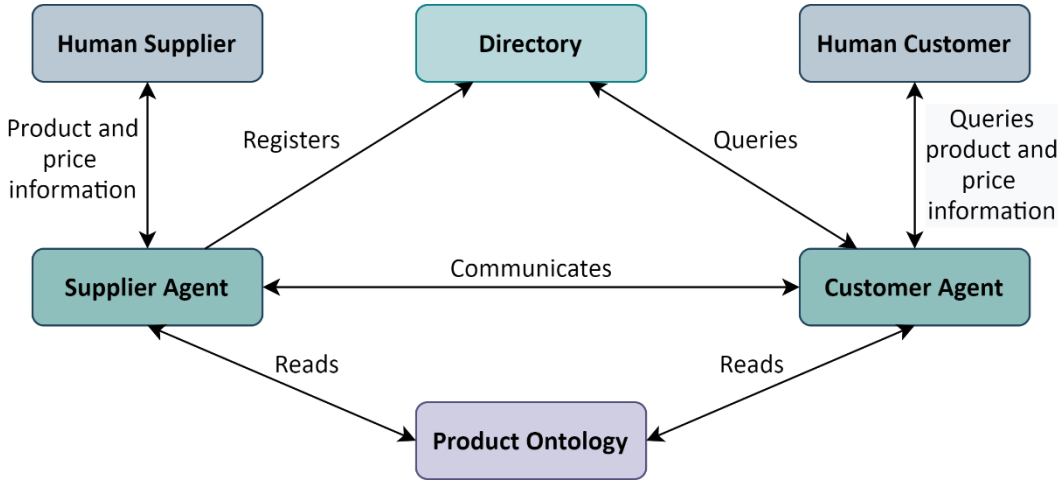


Figure 5 – Example of centralized approach. Adapted from (Ekelhart et al., 2007).

Supplier agents use this information to create user interfaces that allow human suppliers to input product and price information. Finally, the supplier agent registers itself with a central directory and offers the relevant product groups. On the customer side, the process is nearly identical. Depending on the desired product group, the customer agent accesses the product ontology to find out what products are available. Next, the customer agent creates a user interface to gather customer requirements. After the requirements are specified, the customer agent queries a central directory to find supplier agents that offer the desired product group. With a list of all available supplier agents, the customer can start the communication process with each supplier agent (Ekelhart et al., 2007).

Regarding decentralized directory-based search, a structured Peer-to-Peer (P2P) network overlay with a corresponding query routing protocol can be used to perform this type of search for semantic services. In this approach, a global service distribution or replication scheme, in conjunction with the location mechanism of the network, allows for semantic services placement and discovery by all peer nodes (Klusch et al., 2016). This type of search ensures the finding of highly replicated, as well as rare services. As a drawback, whenever peers are joining or leaving the network, or if the set of services which they provide changes, it results in a high communication overhead for publishing and maintaining the structured overlay (Klusch et al., 2016). P2P search can also be directory-less. In this case, the search for semantic services is performed in unstructured P2P networks without any given overlay structure. As such, each peer, initially, is only aware of services provided by its own peers or its direct neighbor's (Klusch

et al., 2016). Although this type of search acknowledged as being effective for finding popular services, the same cannot be stated for rare services. Additionally, it only provides search guarantees, i.e., incomplete recall (Han et al., 2010). Lastly, hybrid P2P networks comprise structured and unstructured overlay parts (Figure 6). Rare services can be found using super-peers in the structured overlay section and, relevant popular services are processed with restricted flooding or broadcasting to peers of the unstructured network section (Han et al., 2010).

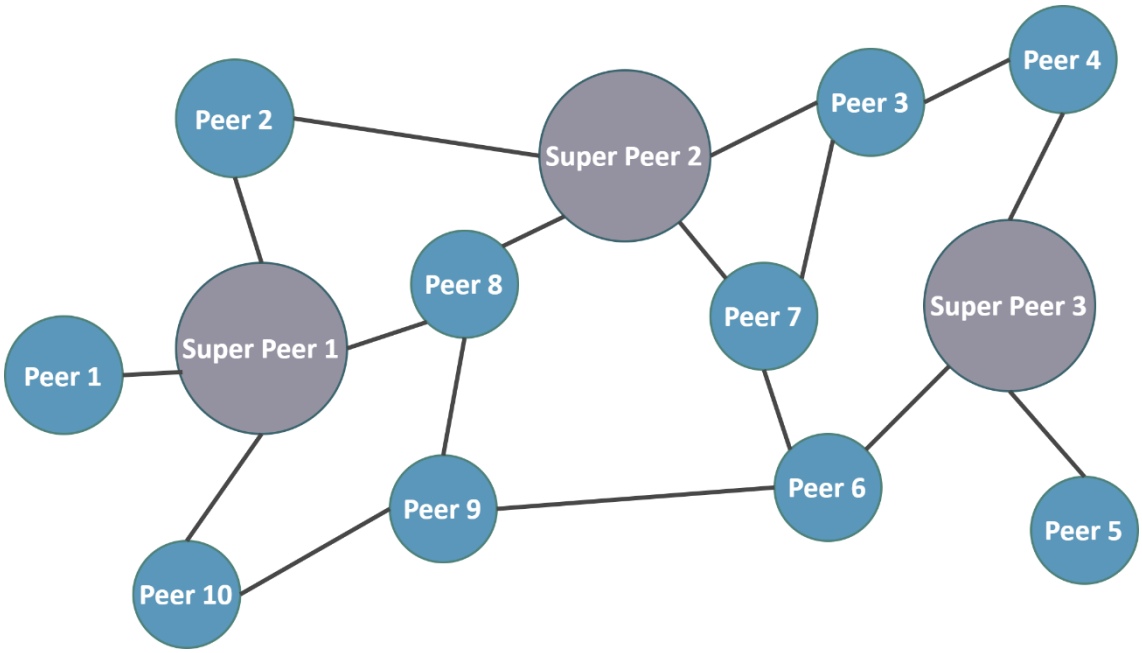


Figure 6 – Unstructured overlay. Adapted from (Han et al., 2010).

Concerning search patterns, web service discovery can be broken down into two main approaches: syntactic discovery, which relies on matching keywords in service descriptions and user queries, and semantic discovery, which relies on finding semantically similar services. The former is usually slower to retrieve information due to the lack of understanding of the semantic meaning of web service descriptions and user queries, while the latter can improve retrieval performance (Zhang et al., 2018). In this context, semantic-aware service discovery research is observed to branch into two categories: logical and non-logical discovery approaches. Logical approaches to service description rely on well-defined ontologies to formalize web service descriptions. These methods are difficult to apply because they require well-annotated semantic data about services and user queries (Li et al., 2022). Alternately, some non-logic latent factor models-based service discovery approaches have been proposed that can achieve better performance (Nabli et al., 2018).

(Klusch et al., 2016) offers another perspective regarding semantic service matching. According to the authors, existing approaches for semantic service matching are classified as either logic-based semantic matching or non-logic-based semantic matching, or hybrid semantic matching, depending on the kind of reasoning tools that are used to achieve this purpose. Non-logic-based semantic matching uses, for instance, methods for graph matching, schema matching, data

mining, and text similarity measurement. On the other hand, logic-based semantic matching focuses on performing logical reasoning on service descriptions (Klusch et al., 2016).

Pertaining to the components involved in the service discovery process, (Bussler et al., 2005) identifies three types of stakeholders:

1. Service Providers that inform about the services that are provided and use publish protocols to advertise their services with matchmakers.
2. Service Requestors that search for services that can perform a specific task using query protocols to ask the matchmakers which services are closer to satisfying their needs.
3. Matchmakers that accept descriptions of available services from providers and match them against requirements from requestors.

Within this framework, the requirements for service discovery can be divided into language requirements, functional requirements and, architectural requirements (Bussler et al., 2005). Language requirements allow to express the service's capabilities and goals, and include available service's preconditions and fulfillment limitations, protocols to be followed during interactions, and requestor requirements such as, goals, quality, security, and privacy. Functional requirements specify the tasks to be performed by each entity, namely:

1. Providers must describe the provided service's capabilities and constraints.
2. Abstract characterizations of required services must be presented by requesters in order to enable matching with the published capabilities.
3. Requestors must locate and interact with peers or matchmakers that can respond to queries for advertised service descriptions.
4. The comparison of descriptions of queries and capabilities must be made by the matchmakers.
5. Requestors must decide if they can satisfy the preconditions specified in a prospective service's self-description in order to use it.

As for the architectural requirements, the objective is to identify the various classes of agents necessary to produce the final result of the discovery phase. Discovery phase protocols include advertising protocols used by service providers to announce capability availability, and candidate service-discovery protocols used by requestors looking for services that satisfy their goals (Bussler et al., 2005).

Finally, on the subject of evaluation, (Klusch et al., 2016) proposes five key criteria for evaluating any semantic service search. First, it emphasizes the importance of supporting different service description formats and languages, ensuring flexibility and broad applicability. Second, the study highlights the system's usability, and the effort required for its configuration, stressing the need for a user-friendly and efficient setup process. Third, the study underscores the value

of service composition planning, which may include features like context-aware pruning of the search space or interactive recommendations for forward or backward chaining of services. Additionally, the presence of a user data privacy policy is considered essential, ensuring that user information is protected. Lastly, the study assesses service retrieval performance, focusing on the accuracy of results and the average query response time, particularly in comparison to other service test collections.

2.6 Semantic Service Composition

web service composition is the process of combining multiple web services to create a new service or application. It differs from traditional application integration in which components are tightly coupled and physically combined. Service composition incorporates roles and functionality to aggregate multiple services into a single composite service, which can be itself used as a basic service in further compositions (Kurniawan et al., 2018). As established in (Garriga et al., 2015), approaches in the field of service composition can be classified through the use of a set of eight features as illustrated in Table 1.

Table 1 – Features of service composition mechanisms. Adapted from (Garriga et al., 2015).

Feature	Accepted Values
Composition view	orchestration/choreography/workflow/other
Automation level	manual/semi-auto/auto
Composition time	static/dynamic
Standards conformance	yes/no
Verification and validation	testing/monitoring/simulation
Service (composition) specification	compositional/WS-/other/ad hoc
QoS awareness	NFP/QoS

In this composition context, orchestration focuses on the internals of the composite application and on the coordination of composition actions and components from a centralized perspective. It expresses the required actions that take place in the composition in order to integrate components. The result is an executable design of the composite application known as an orchestration (Lemos et al., 2015). Choreography refers to composite applications designed using a distributed perspective. In this approach, the providers of the components participating in a composition use a common communication protocol for their components which results in a contract between the co-operating partners. This protocol is not executable and must be implemented by each component individually. In turn, workflow specifies the information flow among work tasks. Composite service as a workflow includes a set of atomic services (or tasks) together with the control/data flow among the services (Garriga et al., 2015).

With respect to the automation level, in manual composition it is expected for users to edit the workflow scripts textually or graphically. This approach presents some shortcomings like reduced scalability, manual rebinding, and the need for users with expertise. Semiautomated compositions are built on the idea of service recommendation. In this case, users still need to select services and link them up to then execute the generated workflow. These systems also present problems of scalability and fault-handling (Wagner et al., 2011). Automated composition makes use of strategies like planning. This approach requires the knowledge of the context, semantics, and the problem domain space (Paik et al., 2014).

Regarding composition time, services can be composed in a proactive manner (static), or when requested by users (dynamic). Static composition is better suited for high-rate usage, resulting in a stable, available, and fast composition. Dynamic composition, on the other hand, take into account the user requests, preferences, and context, providing more flexible and adaptive applications (Garriga et al., 2015).

Concerning standards conformance, several important issues are currently being discussed by various standardization bodies in order to develop a better web service-based paradigm. Without widespread agreement on standards, it will be difficult to implement this new system properly. Standards are based on well-founded principles and following them will make it easier to integrate different composition systems. Some of these issues include quality of service (QoS) awareness, verification and validation, and adapters development (Garriga et al., 2015).

Verification and validation deal with the evaluation of the behavior of service compositions and the mechanisms applied to do so. Some methodologies include model checking, process algebra and theorem proving (Souri et al., 2018).

A composition system might support certain types of specifications for services and/or compositions, through standard or ad hoc languages, or their respective extensions. These approaches are based on Service (composition) specification. In this context, compositional specifications describe services interactions according to a Composition View and web service specifications describe functional/non-functional aspects at the atomic level. A drawback is the fact that non-functional descriptions may require extra effort on building domain ontologies and annotations (Garriga et al., 2015).

Non-Functional Properties (NFP) and QoS are essential aspects that must be considered in order to meet customer's requirements. According to (Garriga et al., 2015) NFPs-based composition approaches should ideally fit available industry standards. In turn, QoS-based composition approaches are usually focused on response time, results accuracy, completeness of covered data, price, availability, and reputation. Planned algorithms tend to neglect QoS concerns and, selection algorithms optimize the QoS but provide no functional flexibility (Wagner et al., 2011).

Irrespective of the approach, in service composition, the discovery of suitable services is a central task that needs to be frequently performed throughout the composition workflow. In this regard, a composition framework should provide relevant discovery mechanisms that could help to discover services able to consume or produce specific types of data that are usually

required. The integration of third-party service registries should also be supported since they represent key activity in the composition phase. Scalability is another factor that should be considered through the incorporation of optimizations that improve the scalability of the overall composition process. Finally, the composition framework should be able to find optimal service compositions by minimizing different criteria such as the number of services or the length of the composition to avoid complex, unmanageable solutions (Rodriguez-Mier et al., 2016).

2.7 Semantic Services Discovery and Composition: Related Work

As previously acknowledged, semantic service search approaches depend on the method used for semantic service selection (matchmaking). This process involves the pairwise semantic matching of a given service request with each service that is registered with the matchmaker in the respective local service directory, and the ranking of the semantic relevance of the services (Klusch et al., 2016). web service discovery is, in this setting, a fundamental task in Service-Oriented Architecture. Consequently, several web service discovery methodologies have been proposed over the last two decades (Garriga et al., 2015).

The related work review is presented in the next section and is organized according to the search patterns, i.e., syntactic discovery based on matching keywords in service descriptions and user queries, and semantic discovery based on finding semantically similar services.

2.7.1 Syntactic-Based Approaches

Syntactic-based Web Services are the original standard for developing services. Consequently, numerous research was made on the improvement of discoverability. The work presented by (Platzer & Dustdar, 2005) proposes a distributed scheme that combines Vector Space Model (VSM) with cosine similarity to retrieve services from many UDDI registries. The concept directing this approach is a combination of information retrieval methods and existing standards for the description of web services. First the existing information is analyzed and used it as comprehensively as possible. The author's objective is to create a search engine, where all the available information is gathered and used to find the best matching method for a specific request. To achieve this the VSM is used. This approach is generally used for search engines dedicated to search the repositories of Web pages. Using this strategy, a document is divided in keywords that represent dimensions in a n-dimensional vector space. Thus, a document can be seen as a vector within this "term space" and the position of this vector relative to other vectors within the same vector space describes their similarity to each other. The mathematical method used to evaluate how similar two documents are to each other and respectively match a given query is to calculate a cosine value for them and express the result as a percentage rating.

(Elshater et al., 2015) propose goDiscovery, a web service discovery approach that relies on statistical modeling and indexing techniques. It uses a Term Frequency – Inverse Document

Frequency (TF-IDF) scheme combined with VSM implemented as a K-Dimensional Tree structure. Each node in the tree splits on a particular hyperplane dimension given by each term. Then, when a user's query is received, goDiscovery transforms it into its equivalent TF-IDF vector using the TF-IDF model and the query vector is recursively compared to follow the appropriate subtree. Once a leaf node is reached, the nearest neighbor services associated to this leaf node are retrieved.

(Sotolongo et al., 2008) combine the lexical database WordNet along with a linear discriminant function. The algorithm calculates the degree of similarity between words and their relative importance to support the development of distributed applications based on web services by parsing the WSDL descriptions to extract documentation and name attributes. In this process, two separate VSM spaces are created, one for the original WSDL and another for the synonyms of each word using WordNet. Lastly, cosine similarity applied on both spaces is combined with a linear function that performs query–service similarity calculations. The algorithm uses the semantic information contained in the WSDL specifications, ranking web services based on their similarity to the term searched by the user. It is applied to a set of 48 real web services in five categories, then compared to four other algorithms based on information retrieval, resulting in an average improvement between 0.6% and 1.9% in precision and 0.7% and 3.1% in recall. Thus, enabling the reduction of the burden and time spent searching web services.

Another work (L. Chen et al., 2010), presents an approach to improve discoverability by proposing an approach for WordNet-powered web services discovery using kernel-based similarity matching mechanism. This approach employs WordNet lexical database to extend semantics and the kernel methods to modify similarity assessment mechanism. WordNet Dimensionality Reduction (WDR) tries to reduce the dimensionality of the term matrix by merging words that are synonyms, eliminating noise. In turn, a sigmoid kernel computes the similarity between the query and the service descriptions.

The work by presented by (Czyszczoń & Zgrzywa, 2014) proposes a modified TF-IDF scheme that considers sections in a service description as different bags of words. In this approach, document similarity scores change considerably as TF-IDF values are calculated for each bag, and multiple term vectors are created, which results in an increase in memory consumption. In order to mitigate the memory consumption issue, the approach merges the term vectors and computes the average weight of all service operation's parameters. Finally, the approach uses latent semantic indexing to index both SOAP and RESTful Web Services.

Recently, the work proposed by (Chebbi & Ayed, 2024) states that, the composition process involves searching for available services within a specific domain, selecting the appropriate service, coordinating the flow of composition, and activating the services. The objective of the research proposed by (Chebbi & Ayed, 2024) is to leverage artificial intelligence for personalizing web service composition. Specifically, using natural language processing (NLP) to analyze the descriptions and functionalities of existing online services. Based on user

profiles, AI algorithms can recommend and select suitable services that align with user's needs.

Using a similar approach, in a different context, (Murakami & Kimura, 2024) use machine learning techniques to analyze patterns in big data, enabling the identification of relevant services for data processing tasks. It also uses NLP is used to interpret and understand the descriptions of various services and data sources, facilitating the selection and composition of appropriate services based on user requirements. This way, the authors propose a framework that incorporates automated service discovery techniques to locate and retrieve available services that can be composed to meet specific analytical needs. The composition of services is managed through workflow automation, which allows for dynamic adjustments based on real-time data and user input. By integrating these techniques, the framework presented by (Murakami & Kimura, 2024) aims to provide a robust solution for big data analysis in smart city applications, allowing for efficient decision-making and resource management.

2.7.2 Semantic-Based Approaches

In (Talantikite et al., 2009)**,** the authors presented a model of semantic annotation for web service discovery, and calculated semantic similarity based on a defined ontology among web services. As such, this work provides a model for describing web services and an algorithm discovery and composition. As the authors stated, the composition of web services is useful in case a client's request cannot be satisfied by a single existing service, but by a composite service. This is obtained by a correct combination of several existing web services. The proposed approach takes advantage of a network of semantic Web services in which, before any submitted request, all the existing OWL-S described web services are registered. Each node in the network represents a semantic web service and an edge is labelled by a value which represents a degree of similarity between the output of the service and the input of the following service. However, there is no edge between non similar services. The obtained algorithm of discovery and composition draws its advantages from a graph structure, a chaining algorithm and also from a semantic annotations and similarity measures between concepts. The semantic network is explored in backward chaining and depth-first in a single pass to find a composition plan that satisfies a client's request. At the end of the exploration, only one optimal service is returned to the requester, which is characterized by the highest measure of similarity, the minimum execution time and the minimum allocated resources. In addition to this, the proposed technique offers various composition types: serial, dependent parallel and independent parallel. The major advantages identified by the authors are that it uses semantic annotations for the web service and for the request, and that it builds an interconnected network of Web services, based on the measure of similarity, in a single pass, reducing the time of replying to the request.

In another approach, (Meditskos & Bassiliades, 2010) used an ontology-based framework to retrieve Web services based on their subsumption relation and structural case-based reasoning. In this work, the authors (Meditskos & Bassiliades, 2010) describe and evaluate a web service

discovery framework using OWL-S advertisements. With this objective in mind, the authors follow a web service discovery model, which is based on abstract and lightweight semantic web service descriptions, using the Service Profile ontology of OWL-S. The main objective was to determine an initial set of candidate web services for a specific request as quickly as possible. Afterwords, this set could be used in more specific discovery approaches, based on richer web service descriptions. The proposed web service matchmaking algorithm extends object-based matching techniques used in Structural Case-based Reasoning, which allows the retrieval of web services not only based on subsumption relationships, but exploiting also the structural information of OWL ontologies and the exploitation of web services classification in Profile taxonomies, performing domain-dependent discovery. Furthermore, it is also described how the typical paradigm of Profile input/output annotation with ontology concepts can be extended, allowing ontology roles to be considered as well.

In (Khanam & Youn, 2016) the authors propose a new service discovery scheme that incorporates similarity methods using the WSDL specification and ontology to further automate the service discovery process, discover the best match rapidly, and improve the Hungarian algorithm used to find the minimum cost in assignment problems. This combination of methods includes structural similarity, semantic similarity and the concept similarity based on bipartite matchmaking techniques used to discover web services. The suggested scheme includes two phases involved in the discovery of the most suitable services to the request. In the first phase, a measure is taken regarding the similarity between the requested service and a set of advertised services. In the second phase, a bipartite graph of nodes based on the ontology is used to describe semantic web service matching. The obtained exploratory results are better than other existing scheme that are using the Hungarian algorithm, but it is lacking in precision, recall, and f-measure.

In (Rodriguez-Mier et al., 2016), the authors present a graph-based framework for automatic service composition and semantic service discovery that is centered on the semantic input-output parameter of service's interfaces for match making. This process starts with a composition request that specifies the user's requirements regarding inputs and outputs. With this information, the graph generation phase proceeds to build a graph with all the relevant services and the semantic relations between their inputs and outputs. Next, to find the relevant services, the composition graph phase is alternated with the discovery phase, which is responsible for retrieving the relevant services given the data available at different stages. The matchmaking phase deals with the relationships between the inputs and outputs of services, where the semantic matching degree between inputs and outputs is computed using a semantic reasoner. The service composition graph is eventually generated and contains all possible service compositions that satisfy the composition request. The service composition graph is optimized by grouping and reducing the number of services and relations. Next, an optimal search is performed over the graph to find the optimal composition. This process is followed by a search optimization phase that analyses and reduces the search space. The optimized composition workflow is then returned.

In an attempt to create a semantic similarity method that can be performed on both the textual description and the interface of web services, (F. Chen, Lu, et al., 2017) proposed a similarity measure, which integrates multiple conceptual relationships, and is developed on the basis of relational semantic distance among concepts in WordNet and other ontologies. Similarity measures for nouns/verbs organized in a hierarchal structure and adjectives organized in a bipolar structure are defined and differentiated. The proposed similarity measure integrating multiple conceptual relationships takes into consideration multiple conceptual relationships in order to offer a refined representation and matching of the service's capabilities. The evaluation presented in this study shows that although the exploratory results are promising in terms of precision, recall and f-measure, it is limited to the semantic similarity of the terms based on the generic WordNet ontology.

The ServiceModel concept in OWL-S provides a standard way to describe simple atomic and composite processes of a service. With this information in mind, some studies have been done on the transformation from OWL-S ontology to Petri nets, combining them with a programming language to obtain a scalable modeling language for concurrent systems (Jensen, 1991; Wang & Tepfenhart, 2019).

In (Ehrig et al., 2007) a Petri net based methodology is presented**,** in which the authors provide a solution for automatic discovery and composition. The approach consists of (semi-)automatic detection of synonyms and homonyms of process element names in order to support semantic process model interconnectivity and interoperability by measuring the similarity between business process models semantically modeled with the OWL. The traditional Petri nets are described with an Ontology Language-based format and this translation of Petri nets into OWL makes it possible to automatically compute similarities between business process models.

Another petri net based approach is presented in (Ni & Fan, 2010)**.** The authors propose a method for web service composition which presents an answer to the web services description inconsistency problem, allowing to compose services from different providers to fulfill business goals that cannot be fulfilled by a single service. In order to ensure the suitability of a Semantic Web Services composition, the authors proposed a model of Colored Petri Nets (CPN) transformed from OWL-S model and it is able to express the logical relations among the subprocesses of the service composition explicitly. This model can then be used to validate the composition using formalized methods of CPNs. Additionally, the authors proposed algorithms that are able to check the reachability and boundness, and also the semantic consistency.

Regarding the quality of service (QoS), (Benaboud et al., 2013) proposed a method for discovering and selecting web services that use OWL-S to describe themselves, quality of service, and customer demand. According to the authors, QoS becomes particularly important when the discovery engine returns multiple candidate web services that provide the same functionality. To help users make better decisions about using web services, it is required to have a model that includes QoS information and make it easier to evaluate the quality of services. To this end, the authors propose creating a quality-of-service evaluation process. To incorporate service QoS information in service discovery, the biggest identified challenge is the

specification and storage of the QoS information. In this paper, the authors describe a method for discovering web services using OWL-S, software agents, and domain ontologies. This approach is composed of four layers: a web service and Request description layer, a Functional match layer, a QoS computing layer, and a Reputation computing layer. Each layer uses the results of the previous layer to decrease the number of possible web services. The service consumer expresses his preferences by specifying that a certain QoS attribute is more important than another one and by evaluating the level of importance about each QoS attribute. This method is proven to be more efficient and more dynamic by exploiting the parallelism and the distribution given by agent technology.

Similarity computation can also be achieved by using Natural Language Processing (NLP) algorithms. (Sangers et al., 2013) started by creating a service context consisting of keywords that were extracted from the service description. Then, NLP techniques were incorporated into a keyword-based discovery process. This semantic search feature of the framework allows users to find semantic web services that match their specific needs by specifying keywords. This helps users avoid needing to know about semantic languages, making it easier to find the services they're looking for. Once the keywords are matched with the descriptions of the semantic web services, various techniques, such as part-of-speech tagging, lemmatization, and word sense disambiguation, were used to determine the correct meaning of the words. Finally, a matching process takes place to ensure the service is found.

In the context of NLP, (Fang et al., 2018) presented an approach that entails an ontology-based service preprocessor, reasoning-based service filter, and parameter-based service matcher. Ontology Filtering and Parameter Matching is centered around the function-oriented web service discovery which consists of 3 key modules: (1) ontology-based service preprocessor, (2) reasoning-based service filter, and (3) parameter-based service matcher. The first module utilizes the ontology referenced by services and requests to reduce the number of candidate services. The second module selects services which are matched with requests in logic. Consequently, it logically deduces the concepts and filters out the services that are insufficient to satisfy the user's parameter needs. The refined web services are then matched with user requests through the proposed parameter-based matching algorithm, which takes into account the relationship between the requested service and the requested parameters. An important feature of this approach is that the relationships among concepts in ontology are quantified and considered in the matching process, which results in high precision and recall. Additionally, the authors propose a filtering process based on logical reasoning to preprocess the large amount of web services. This filtering process allows for web services which are feasible in logic to be selected to be matched with user requests. This resulted in a great improvement in the run time performance of the service discovery approach, although it is lacking in terms of precision and recall.

Recently, an exploration of ChatGPT as a tool to perform service composition, was made by (Aiello & Georgievski, 2023). According to the authors, the objective was to execute tasks using loosely connected services, allowing for reliance on third-party implementations without prior

knowledge. While both syntactic and semantic approaches have drawbacks, the latter, involving additional service descriptions, is more prevalent. The assumption is that service implementations offer details on preconditions and outcomes. Nevertheless, semantic service annotations that could enhance automation, and detailed descriptions are not extensively available. The authors, view the advent of Large Language Models (LLM) as a technological breakthrough that can revive the field and elevate the levels of automation. Using ChatGPT, a composition example involving country-specific weather, currency conversion, and mapping can be transformed into Python code with appropriately phrased requests. Despite requiring some editing, the generated code illustrates the potential for fast development.

In conclusion, although progress in automated service composition has slowed over the past decade, tools like ChatGPT may renew the interest and practical applications, as long as current research challenges are taken into consideration.

2.7.3 Pre-Classification Approaches

Regardless of following a syntactic or semantic approach, ML has also been used to provide solutions based on service clusters, in order to organize services into groups and make the discovery and composition of semantic services easier and more efficient.

In (Ma et al., 2008) a novel approach is presented for efficiently finding web services on the Web. This approach is named Clustered Probabilistic Latent Semantic Analysis approach (CPLSA) and aims to combine syntactic analysis with a clustering semantic methodology. Given a query, first those web services whose contents are not compatible with a user's query are filtered out through a clustering algorithm to acquire an initial working dataset. With this objective in mind a k-means approach was implemented to eliminate irrelevant services. In the next step, a Probabilistic Latent Semantic Analysis approach (PLSA) is applied to the working dataset, which is further clustered into a finite number of semantically related groups so that service matching against the query can be carried out at the concept level. In this phase, PLSA is used to capture semantic concepts hidden behind the words in a query and in the semantic information of services, so that services matching can be implemented at an advanced concept level.

In (Wagner et al., 2011), the authors search for ways to compose service workflows automatically, based on specific factors. As such, there is a focus on resolving several key issues. Firstly, the reliability of a workflow decreases as the length of the workflow increases. If one of the services in the workflow fails, the success probability of the entire workflow is greatly reduced. In this context, the authors look for ways to increase the reliability of workflows without having to re-plan the workflow if one of the services fails. Apart from the reliability, to ensure the work's affordably and responsiveness, it is necessary to take into account the QoS associated with each of the services involved. This includes things like price and response time. Frequently, the value of a service composition is judged by a calculation that takes into account all of the QoS attributes. Additionally, QoS constraints specify limits on the workflows that can be used. If the constraints are violated, the workflow becomes useless. Regarding flexibility, manually composing services can be a tedious process, but planning tools can help make the

process easier by creating workflows on the user's behalf. Within this framework, the authors approach is to start by identifying clusters with functionally similar services. For that, the authors establish a data structure that clusters functionally similar services and provide methods to assess the QoS attributes of these clusters. Next, the service workflows are computed, considering only the root nodes of the service clusters and the assigned QoS attributes. As a result, the algorithm generates workflow templates, which provide the ability to consider a set of services for a given task.

In (Paliwal et al., 2012), a semantic categorization of web services by mining the associative relationship of terms is proposed. An integrated approach is developed for addressing the two major issues related to automated service discovery which are the semantic-based categorization of Web Services, and the selection of services based on semantic service description rather than syntactic keyword matching. The first step of the proposed approach involves semantic categorization of the Web Services. The next step is to select a Web Service for a given service request. In this step, the refinement of the set of Web Services is based on the input, output, and description parameters of the service. The aim is to select a set of services that perfectly matches the desired functionality according to these parameters. Additionally, the enhancement of the Web Service request with relevant ontology terms is made, and the matching of the enhanced service request with the set of candidate Web Services for selecting appropriate service is performed.

The work of (F. Chen, Li, et al., 2017) proposes a new web services discovery method based on semantic matching and service clustering for effective and practical web services discovery, integrating functional similarity with process similarity, assuming that service providers publish web services advertisements conforming to the semantic service description model. Based on these services published information, first, the system clusters web services into functionally similar groups. Then, a search engine searches through advertisements from the repository and groups semantically similar services into web service clusters according to the defined functional similarity measure. Typically, a user uses a query that provides a complete definition of the service in a specific language (like OWL-S). Then, in order to identify the most pertinent service to the request, the service search engine performs service matching in two steps. First the request is matched to a service cluster functionally most relevant to the user's request, then the services are ranked within the given cluster.

In a similar approach, (Surianarayanan & Ganapathy, 2016) focused on the computation of similarity, inter-cluster distance (ICD) and the selection of threshold for service discovery using clusters. A hierarchical agglomerative clustering-based approach is proposed for service discovery including two similarity models. These are, the Output Similarity Model (OSM) and Total Similarity Model (TSM) with additional levels for Degree of Match (DoM). With the objective of eliminating irrelevancy while clustering services, the OSM computes similarity between services using only the outputs of services. In turn, the TSM computes similarity between services using both inputs and outputs of services while discovering matched services of a given query. The proposed approach selects threshold-ICD in terms of DoM without altering the similarity demands of clients and ensures the appropriateness of clusters. Like the previous

work, results showed that pre-clustering can significantly increase the speed of the discovery process.

In (Cheng et al., 2017) the authors propose a web services discovery approach, focused on mining the underlying semantic structures of interaction interface parameters to help users with the discovery and deployment of web services. The process of matching interfaces can achieve high precision when the parameters of those interfaces contain meaningful synonyms, abbreviations, and combinations of disordered fragments. The proposed approach is composed up of three modules. The Web Service Model Extraction crawls web services on the Internet and extracts information using a standard web service description model. The Interface Semantic Mining module mines the underlying semantics and creates a semantics index library by clustering interaction interface names and fragments under the supervision of co-occurrence probability. The Main Discovery Process evaluates the user's request and searches the web services result set based on the index library. In this model, the authors propose a web service Operations Discovery algorithm (OpD). The OpD results include two types of web services: services with "Single" operations and services with "Composite" operations. As a result, the entire web services discovery approach can discover and compose results quickly with a high precision/recall rate.

From another perspective, (W. Gao & Wu, 2017) discuss that due to a vast number of web services online, recommending services for automatic mashup creation greatly facilitates the composition process of developers. The authors state that the various approaches that have been proposed for the task are focused on improving the recommendation accuracy of an individual service. This fact results in two problems. In one hand top-ranked services may be redundant presenting with the same functionality. In another, the cooperation relations among services are ignored. With this in mind, the authors infer that services should be recommended not individually, but collectively. To this end, a new recommendation framework composed of two stages (Service Set Generation Stage and Service Set Ranking Stage) is proposed. In this approach, first, services are clustered into different categories according to their functionalities. Next, the service categories that best match mashup requirement are selected. Then, one service from each selected service category is chosen to form a service set. This process guarantees that a service set includes services with distinct functionalities while satisfying the user's requirements. To address the cooperation relations among services issue, the authors propose the choice of a specific service set with three distinct factors, namely functional preferences, QoS preferences and composition preferences. For functional preferences, it is assumed that services in the optimal service set should match mashup requirement closely. For QoS preferences, services in the optimal service set should have high nonfunctional quality. For composition preferences, the authors presume that developers would choose a set of services that are more likely to be composed together. The comprehensive experiments conducted on have shown that the proposed framework is effective in recommending services.

In more recent research several approaches based on deep learning emerged. For example, (Yang et al., 2019) present a stacked deep neural network, named ServeNet, organized from the abstract low-level representation of service description to high-level features without

feature engineering. This approach predicts service classification on 50 service categories. With this approach it is also possible to extract sequential features from both past and future allowing for better performance on the texts with context information. The experiment results demonstrate that ServeNet achieves above average results in terms of accuracy.

(Bai et al., 2020) present a deep learning framework to perform accurate long-tail service (non-popular services) recommendations. According to the authors, the main challenge when recommending long-tail services is that historical usage data is scarce, and the descriptions provided by this data are frequently lacking in quality. To address the issue of poor quality of description, stacked denoising autoencoders (SDAE) are used in order to perform feature extraction. With regard to the sparsity of historical usage data, the patterns of the developer's preference were learnt instead of modeling individual services. Experimental results on a real-world dataset demonstrate that the proposed algorithm achieves good performances.

In another deep learning approach, (Zou et al., 2022) propose DeepWSC to cluster services through automatic feature extraction. In this approach, service composability features are integrated into deep neural network for web service clustering. First, DeepWSC determines the service composability network and generates the heuristics of service invocation relationships, which is then fed into an improved recurrent convolutional neural network to perform unsupervised training in a service feature extractor. Lastly, DeepWSC acquires the integrated implicit features of web services, which consists of deep semantic features and composability features of web services. Extensive experiments were conducted in order to evaluate the effectiveness of DeepWSC proving this approach effectiveness on multiple evaluation metrics.

According to (Li et al., 2022), the main limitations of the works based on a deep learning framework include the non-interpretability of features and the requirements of large-scale data for model fitting. Another kind of approach to semantics-aware service discovery are the ones based on a topic model. In this context, (Naim et al., 2018) propose a new content-based topic model to capture the maximal common semantic of sets of services. In this approach, it is assumed that all service descriptions were written in the WSDL. Additionally, topic models are used, in the context of this proposal, as efficient dimension reduction techniques, which are able to capture semantic relationships between word-topic and topic-service interpreted in terms of probability distributions. Based on the extracted topics, a set of matched services is returned by comparing the similarity between the query and the services related topic. The service discovery phase is responsible for retrieving the Top-R ranked web service set. First, the query is represented in topic space to discover the set of top-K implicit topics that are semantically associated to the query. Based on the selected topics, a set of web services is returned by comparing the similarity between the query and the related web services to these topics. Then, the similarity scores are used to rank and select the top web services. To minimize the redundancy in the search results, the top-ranked web services are re-ranked considering relevance, diversity, and density. Next, a search is performed over the service dependency network to find the optimal composition which results in the service dependency network deduction. The service dependency network takes the Top-R ranked web services set as input

and returns a sub-graph containing all possible service compositions. Then, a selection process is performed over the graph to find the optimal composition for the results of the query. Finally, the Top-R ranked web services and their optimized composition are returned.

In another example, (Shi et al., 2017) propose an augmented Latent Dirichlet Allocation (LDA) model, named WE-LDA, to improve the performance of web services clustering. The derived knowledge of topics extracted by topic models help to reveal the trend of service composition, understand the latent concepts of the services, and lead to better service recommendation. In the proposed method, first, the word vectors of all terms in description documents of web services based on the Word2vec tool are processed and learnt. Then, the K-means++ algorithm is executed to cluster all the words into different groups based on the similarities of their word vectors. Then, the information in the words clusters is used to train the distributed representations of web services based on the LDA model. Finally, all web services are clustered into different functional domains based on the clustering algorithms like K-means, or according to their latent topics.

Also using the LDA model, (Z. Gao et al., 2019) developed a method of service co-occurrence LDA to extract latent service co-occurrence topics. In this paper the authors start by defining the concept of service co-occurrence topic and a mechanism to construct service co-occurrence documents with the objective of treating each service as a document and its co-occurring services as the bag of words in that document. Service Co-occurrence LDA (SeCo-LDA) is also defined as a theoretical model along with four metrics to measure self-co-occurrence of a specific service. The SeCo-LDA allows the extraction of latent service co-occurrence topics, including representative services and words, temporal strength, and service's impact on topics. This information helps to reveal the trend of service composition, understand collaboration behaviors among services and lead to better service recommendation. Experiments in a real-world data set were conducted in order to verify the effectiveness and efficiency of this approach revealing promising results.

With a different perspective, (Cao et al., 2020) propose a mashup service clustering method that exploits a two-level topic model to mine the latent useful and novel topics. According to the authors, the implicit co-invocation records between Web APIs in common historical Mashups can be used to predict usage probability of unpopular Web APIs in Mashups. As such, these results can be used to diversify the Web APIs recommendation. To this end a two-phase framework is proposed. The main process of the first phase is the topic model-based Mashup service clustering in terms of both Mashup service content and Mashup service network. In this process K Mashup service clusters are obtained by applying the two-level topic model to derive topics of Mashup services. In the next phase, an active user submits a Mashup development requirement with textual description, which is first converted a corresponding Mashup requirement topic feature vector by using LDA technology. Meanwhile, the same process is executed to identify the K Mashup service clusters resulting from the first phase, and to obtain the Mashup service clusters topic feature vectors. Then similarity-based matching is used to identify a Mashup cluster with the highest similarity. Finally, a Web APIs recommendation based on the collaborative filtering (CF) algorithm is designed to recommend top-R Web APIs to the

user. Thus, it is possible to use historical invocation records between Mashups clusters and Web APIs to rank and recommend diverse Web APIs, including popular and unpopular available services.

Semantic matching is required in order to find satisfactory web services for a user query. However, this is an open research challenge due to the large variety of textual elements in a user query and the sentences in a service description. To address this challenge, (Li et al., 2022) proposed an unsupervised Bayesian probabilistic model, bi-directional sentence-word topic model (bi-SWTM), to learn the semantics of words and sentences in the same topic space. Topics extracted by the topic model can be used to represent the semantics of the words and sentences. The bi-SWTM takes advantage of the two-directional sequences of sentences and the words in each sentence to learn the latent topics. Words and sentences are defined in the same topic space, which builds a bridge to capture the similarities of the words and sentences from the semantic level. Thus, different textual types of user queries (keywords, phrases, and sentences) can be represented in a probabilistic simplex, which provides a flexible approach to extract the high-level semantics of the queries for service matching.

In a more recent publication (Merin et al., 2023), the authors used SVM and Random Forest classification models to predict web service quality based on ten attributes related to QoS. According to the authors, service discovery can usually be achieved through either syntactic or semantic methods. Syntactic discovery relies on keyword matching, but its simplicity results in low precision and recall rates. This approach often yields “no results found” when an exact match isn't identified. Additionally, it’s impractical to assume that all possible keywords are pre-fed into the system for each service. On the other hand, semantic discovery utilizes ontology, enhancing the clarity of web service description semantics. This requires a semantic tagged language like OWL-S or SAWSDL. Despite the advantages of semantic discovery, including better precision and recall, implementation faces challenges such as considering relationships in hierarchical ontologies and potential time consumption. Taking these considerations into account, in this project, the previously mentioned SVM and Random Forest classification models were developed to predict web service quality based on ten attributes related to QoS. The models use historical data, treating QoS attributes as explanatory variables and Overall quality as the dependent target variable. The collective effect of QoS attributes on service quality is represented as a non-linear relationship, normalized using chosen classification methods. Intelligent machine learning techniques are applied to predict the rank of a new web service, facilitating ranking for normal users. The system includes a QoS value-based filter to enhance the search experience for developers. Precision and recall are calculated and evaluated, producing higher results compared to previous schemes. The proposed system utilizes semantic annotation as a pillar, relating linguistic entities by classifying them into identical groups of word sets and establishing connections based on contextual relations (synonym, antonym, meronym, or hyponym). It calculates semantic similarity between the search phrase and each candidate meaning corresponding to every web service, going beyond mere tokenized search phrase and web service name comparison.

2.7.4 The Semantic Services Catalog

Still in the context of the related work review, it is essential to reference the work presented in (Santos et al., 2021), since the discovery and composition mechanism will interact directly with the Semantic Services Catalog (SSC) presented by (Santos et al., 2021). Figure 7 presents the home page of the SSC.

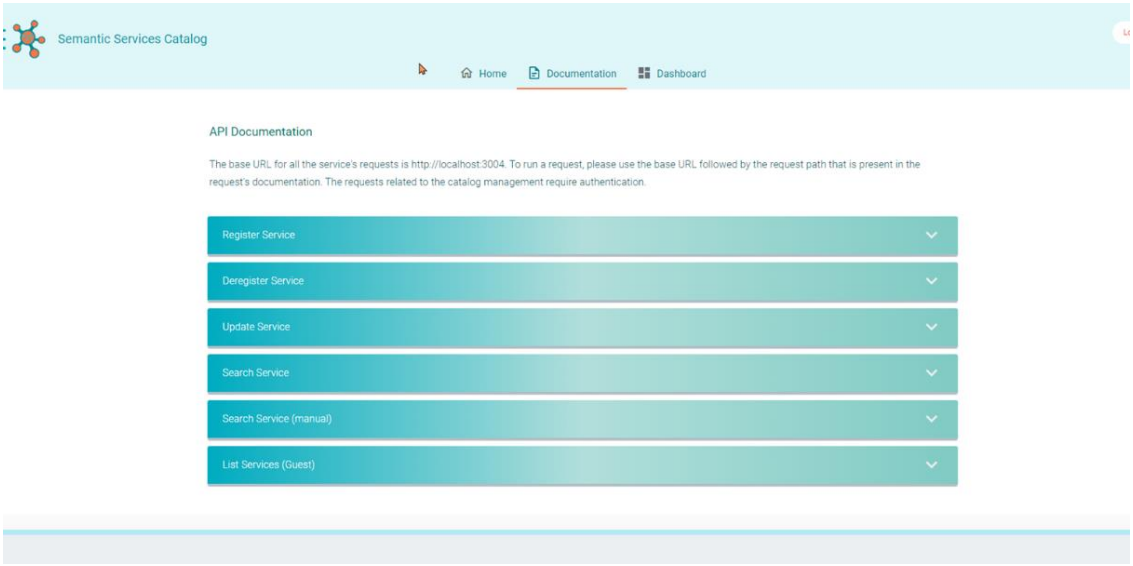


Figure 7 – Semantic Services Catalog home page.

As mentioned in (Santos et al., 2021), when dealing with interoperability between several MAS, it is critical that separately designed agent-based platforms communicate. As a result, agents in the MAS Society must be FIPA (Foundation for Intelligent Physical Agents) compliant, ensuring the interchange of messages between heterogeneous systems regardless of development platform. Being FIPA compliant allows a MAS agent to register in the Directory Facilitator (DF) agent of another agent-based platform. Nonetheless, this registration is generally hardcoded beforehand, either programmatically or through configuration or property files.

To reduce the effort and error prone manual configuration of distributed MAS, the SSC (Figure 7) was developed, providing a central location for service registration and discovery within the MAS Society. SSC has a modular and distributed architecture (Figure 8), featuring a backend that provides endpoints for registration, de-registration, update and deletion of services.

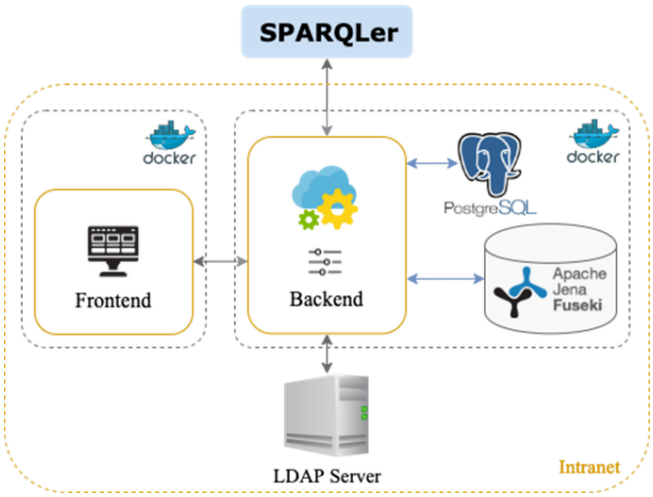


Figure 8 – SSC modular architecture (Santos et al., 2021).

It also enables simple and advanced search by keywords or through SPARQL query where the user sets the search constraints as needed. The components implemented for persistence are PostgreSQL for user data and documental information, and Fuseki, as a triple store for web services persistence. In turn, the frontend component lists services publicly, provides documentation for developers, allows for management and system configuration by administrators, and presents a search page where the simple and advanced search can be performed by the user. While SSC provides a foundation for registering and discovering semantic services, it does not offer intelligent search capabilities that take into account the semantic meaning of search terms, as well as the ability to suggest service compositions when a single service is insufficient to achieve the desired outcomes. In this regard, the present dissertation aims to implement a search and composition mechanism the works in tandem with the SSC upgrading the search mechanism so that it enables the mentioned intelligent search and composition of web services.

2.7.5 Related Work Analysis

Web services are becoming more popular as a way to share information because of a set of standards like WSDL, UDDI, and SOAP, that are flexible and able to be extended. These standards are based on XML, which makes them easy to understand and use. However, the XML-based specifications available for web services do not provide a full description of all the functionalities that can be supported by those services. Only a syntactic description is present. This means that human interaction is still required during the discovery process. Thus, a more reliable and effective web service discovery approach, that is suitable for automatic processing, was created.

The Semantic Web vision encourages developers to add machine-readable semantics to existing web services descriptions in order to make them more automatable and autonomous. By doing so, it becomes easier for software to find, compose, select, and invoke related web services. The goal of semantic web service technology is to make it easier for users to find and use web

services by letting software agents and applications work together to identify, integrate, and execute these resources automatically.

Many approaches for automatic web service discovery have been proposed, as discussed previously. However, they also present limitations. The following table summarizes the advantages and disadvantages of each approach.

Table 2 – Syntactic versus semantic approaches for web services discovery. Adapted from (Adala et al., 2011).

	Syntactic-based approaches for WS discovery	Semantic-based approaches for WS discovery
Matchmaking technique	A simple keyword-based search. Searching based on functional parameters. Searching based on syntax.	Exploit the semantic representation of concepts describing a Web Service and their relations using an ontology. Searching based on both functional and non-functional parameters.
Advantages	Simple and widely used technique. Standards like UDDI exist.	Minimize the manual discovery and usage of web service by allowing software agents to automatically and dynamically discover web services. Pledge the automation of WS discovery process. Effective and reliable technique.
Disadvantages	Do not allow retrieval of Web Services with similar functionality. Not suited for automatic processing. Still requires human interaction.	More complex technique. Semantic tagging of web services may be needed.

From Table 2 it is possible to verify that syntactic-based approaches have the advantage of being simpler to use and can be supported by standards like UDDI. On the other hand, automation is difficult to achieve since human interaction is often required. Semantic-based approaches, in turn, make automation easier and allow for the retrieval of web services with similar functionality, but are more complex to implement and require web services to be tagged.

2.8 ML and NLP Techniques Overview

As previously verified, web service discovery can be implemented in a variety of ways, such as, generating graphs that express the relationship between services, or by using NLP and ML techniques. Taking into account the need to periodically add new web services to the database by a heterogeneous group of users with different background knowledge, it became clear that

using NLP and ML would allow users to find semantic web services that match their specific needs by specifying keywords.

As highlighted by (Ngan & Kanagasabai, 2013), the initial matchmaking strategies can be greatly improved by applying standard ML algorithms such as regression, decision trees, or support vector machines (SVMs). This helps users to avoid needing to know about semantic languages, making it easier to find the most relevant services. Once the keywords are matched with the fields of the semantic web services, various techniques, such as tokenization, lemmatization, and word sense disambiguation, can be used to determine the correct meaning of the words. In NLP, Tokenization, Word Sense Disambiguation (WSD), and Lemma Extraction are foundational techniques that help computers understand, process, and analyze human language effectively.

Each of these techniques plays a critical role in enabling accurate and meaningful text analysis, which is essential for a wide range of NLP applications, from search engines and chatbots to sentiment analysis and machine translation (Daniel Jurafsky & Martin, 2024). After the pre-processing mechanisms, in order to encode categorical into a format suitable for ML models, methods like One-Hot Encoding, Word2Vec, and TF-IDF can be applied. These are three popular techniques used in NLP for converting text data into numerical format, which is essential for ML models (Goldberg, 2016). Furthermore, as observed in the section Pre-Classification Approaches, pre-classification strategies have proven to be useful in increasing the performance and accuracy of discovery mechanisms.

In the present section an overview is also made regarding the ML models for classification tasks that are commonly used. As stated in works like (Patidar et al., 2022) or (Caruana & Niculescu-Mizil, 2006) the following algorithms are widely recognized and used in classification tasks. The application of these algorithms was also observed in several of the studies presented in the previous section Semantic Services Discovery and Composition: Related Work. As such, the NLP techniques description is followed by the description the classification algorithms: Decision Tree; Random Forest; Logistic Regression; Gradient Boosting; k-Nearest Neighbors; Support Vector Machine; Naive Bayes; Multi-layer Perceptron; AdaBoost; XGBoost and CatBoost. In this regard, the following sub sections focus on describing the NLP and ML techniques best suited to achieve the expected results.

2.8.1 Text Pre-Processing

In this sub section a group of techniques that help computers understand, process, and analyze human language effectively is presented. A multitude of NLP applications depend on precise and insightful text analysis, which is made possible by each of these methods.

2.8.1.1 Tokenization

Refers to the process of breaking down text into smaller units called tokens, which can be words, phrases, symbols, or other meaningful elements: This allows to prepare the text for further processing by converting it into manageable pieces (D. Jurafsky & Martin, 2009).

2.8.1.2 Word Sense Disambiguation

Relates to the process of identifying which sense of a word is used in a given context when the word has multiple meanings. The objective is to enhance the understanding of the text by correctly interpreting words with multiple meanings (Navigli, 2015).

2.8.1.3 Lemma Extraction

Is the process of reducing words to their base or root form (lemma). Aims to group together different forms of a word so they can be analyzed as a single item through the use of morphological analysis of the words considering the context and part of speech of the word to determine the correct lemma (Manning et al., 2009).

2.8.2 Text Encoding and Embedding

One-Hot Encoding, Word2Vec, and TF-IDF are widely used techniques in the field of NLP and ML. They are particularly common for converting text data into numerical formats that can be used by ML models. Additionally, in order to expand the search terms with word embeddings, Word2Vec, GloVe, and S-Bert can also be used. This section provides a description of each.

2.8.2.1 One-Hot Encoding

One-hot encoding is a simple and intuitive method for representing categorical data as binary vectors. In the context of NLP, one-hot encoding represents each word in a vocabulary as a unique binary vector (Manning et al., 2009). For a vocabulary of size N, each word is represented as a vector of size N where all elements are 0 except the one corresponding to the word's position in the vocabulary, which is set to 1. This method is simple and easy to implement, but it is frequency-based, making no assumptions about the data. It is useful in small vocabularies or for categorical features where there are few possible values.

2.8.2.2 Word2Vec

Word2Vec is a group of related models used to produce word embeddings. These are dense vector representations of words that capture semantic meanings based on the context in which words appear (Mikolov et al., 2013). Word2Vec has two main architectures. Continuous Bag of Words (CBOW) which predicts a target word from its context words (words surrounding it). And Skip-gram, that is able to predict context words given a target word. These models use neural networks to learn word embeddings in a way that allows words with similar contexts in the training data result to have similar embeddings. As main advantages, Word2Vec captures semantic relationships i.e. words that have similar meanings or occur in similar contexts have similar vector representations; produces dense, low-dimensional vectors, making it computationally more efficient; and pre-trained Word2Vec models (like those from Google's Word2Vec model trained on Google News) can be used on different tasks, reducing the need for training from the beginning. It is usually used in semantic analysis, sentiment analysis, machine translation, and other NLP tasks where understanding the semantic relationships between words is needed (Mikolov et al., 2013). Additionally, Encoding with Word2Vec allows text data to be transformed into numerical vectors that can be fed into ML models. This encoding is crucial for downstream tasks like text classification, clustering, sentiment analysis, and more.

2.8.2.3 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents (corpus), and is a product of two metrics (Salton & Buckley, 1988). Term Frequency (TF) which measures the frequency of a word in a document. It is the number of times a word appears in a document divided by the total number of words in that document. And Inverse Document Frequency (IDF), that is able to measure the importance of a word across all documents in the corpus. It is the logarithmically scaled inverse fraction of the documents that contain the word, which helps in down-weighting commonly used words (like "the", "is", etc.) and up-weighting words that are more unique to each document. TF-IDF reflects Importance by helping to identify words that are important to a specific document within a corpus. Like one-hot encoding, TF-IDF vectors can be high-dimensional if the vocabulary is large. It is frequently used in document classification, information retrieval, search engines, and any application where identifying the relative importance of words in documents is crucial (Salton & Buckley, 1988).

2.8.2.4 GloVe

Developed by the Stanford NLP Group in 2014, is another word embedding model that differs from Word2Vec primarily in its training approach. GloVe is based on matrix factorization techniques. It constructs a large matrix where each entry represents the co-occurrence frequency of two words in a corpus (Pennington et al., 2014)**.**

2.8.2.5 Sentence-BERT

Introduced by UKP Lab in 2019, extends BERT (Bidirectional Encoder Representations from Transformers) by adding a pooling operation over the output of BERT to generate a fixed-size vector for each sentence. BERT itself is a deep transformer-based model that captures rich contextual information by processing sentences bidirectionally. This way, it generates semantically meaningful sentence embeddings that can be compared using cosine similarity (Devlin & Liu, 2014)**.**

2.8.3 Vector Similarity Metrics

Similarity metrics can be used to calculate various similarity scores between the original search term’s embeddings and the expanded term’s embeddings, as well as the similarity between the results embeddings and the user query embeddings. With these values it is possible to select the best embeddings to expand the user queries and to rank the search results in order to improve their accuracy.

2.8.3.1 Cosine Similarity

Measures the cosine of the angle between two vectors in a multi-dimensional space. It is frequently used in text analysis and information retrieval to measure document similarity. Cosine Similarity is ideal for measuring the semantic similarity of vectors representing text (Manning et al., 2009)**.**

2.8.3.2 Euclidean Similarity

Measures the inverse of the Euclidean distance between two vectors. It is commonly used in clustering and classification tasks. Euclidean Similarity (or inverse distance) provides a sense of the absolute closeness in a continuous space (Stork et al., 2001).

2.8.3.3 Manhattan Similarity

Measures the distance between two points by summing the absolute differences of their coordinates. Manhattan Similarity captures similarity based on the sum of absolute differences, useful for sparse data (Kumar & Minz, 2014).

2.8.3.4 Jaccard Similarity

Jaccard Similarity offers a measure for set-based comparison, highlighting overlap between discrete sets of attributes or terms. It measures the similarity between two sets as the size of the intersection divided by the size of the union. It is used in in binary attribute comparison, such as in recommender systems and clustering categorical data (Kumar & Minz, 2014).

2.8.4 Classification Algorithms

This section encompasses the description of the previously mentioned classification algorithms: Decision Tree; Random Forest; Logistic Regression; Gradient Boosting; k-Nearest Neighbors; Support Vector Machine; Naive Bayes; Multi-layer Perceptron; AdaBoost; XGBoost and CatBoost.

2.8.4.1 Decision Tree

A Decision Tree is a non-parametric model that splits the data into subsets based on the value of input features. It represents decisions in the form of a tree structure where each internal node represents a feature (or attribute), each branch represents a decision rule, and each leaf represents an outcome (class label). It recursively splits the data at each node based on the feature that provides the maximum information gain or the least Gini impurity (Quinlan, 1986). In this context, the concepts of information gain, entropy and Gini impurity require further exploration.

Information Gain (IG): Information Gain (IG) is based on the concept of entropy from information theory. It measures the reduction in entropy (or uncertainty) after splitting a dataset based on a feature. A feature with high information gain is chosen for a split because it provides a more informative (less random) classification of the dataset (Chollet, 2017).

Entropy: Entropy is a measure of the randomness or impurity in a dataset. In the context of decision trees, it quantifies the amount of disorder or impurity in the classes of a dataset. Information Gain helps select the attribute that results in the maximum reduction of entropy (i.e., the attribute that best separates the classes in the dataset). A higher Information Gain means a more informative attribute (Chollet, 2017).

Gini Impurity: Gini Impurity is another measure used in decision trees to assess the quality of a split. It measures the probability of incorrectly classifying a randomly chosen element if it were randomly labeled according to the distribution of class labels in the dataset. The Gini Impurity

ranges from 0 (perfect purity, where all elements belong to a single class) to a maximum value of 0.5 (binary classification, where the classes are perfectly mixed in equal proportions). A lower Gini Impurity indicates a better split, meaning the data in each subset is more homogeneous in terms of class distribution (Chollet, 2017).

2.8.4.2 Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode (most common output) of the classes of the individual trees. It works by randomly selecting subsets of features and samples to create each decision tree, which reduces variance and helps prevent overfitting. Each decision tree is built using a different subset of the training data and features. The final classification is based on the majority vote from all trees (Breiman, 2001).

2.8.4.3 Logistic Regression

Logistic Regression is a statistical model that models the probability of a binary outcome (1/0, Yes/No, True/False) based on one or more predictor variables. It uses the logistic function to return the output of a linear equation between 0 and 1. It works by computing a weighted sum of input features and applies a logistic function to model the probability of a certain class. This model is simple to implement, efficient to train, and outputs well-calibrated probabilities. However, it assumes linear relationships between independent variables and the log odds, sensitive to outliers (Scott et al., 1991).

2.8.4.4 Gradient Boosting

Gradient Boosting is an ensemble technique that builds models sequentially. Each new model is trained to correct the errors made by the previous models. It minimizes the loss function by using gradient descent, making it highly accurate for both regression and classification tasks. Gradient Boosting sequentially adds weak learners to minimize a loss function, often using decision trees as weak learners. It offers high accuracy, handles different types of data and, has less overfitting compared to other models (Friedman, 2001).

2.8.4.5 k-Nearest Neighbors (k-NN)

k-NN is an instance-based learning algorithm that classifies a sample based on the majority class among its k-nearest neighbors from the training data. It is a non-parametric method that makes no assumptions about the data distribution. It is simple to implement, has no training phase, and it is good for small datasets. However, it can be computationally expensive during prediction (Cover & Hart, 1967).

2.8.4.6 Support Vector Machine (SVM)

SVM is a classification algorithm that finds the hyperplane which best separates the classes in the feature space. It can be extended to non-linear boundaries using kernel functions. It works by maximizing the margin between the closest points (support vectors) of different classes by finding an optimal hyperplane (Cortes & Vapnik, 1995).

2.8.4.7 Naive Bayes

Naive Bayes is a probabilistic classifier based on Bayes theorem[1]. It assumes that the presence of a feature in a class is independent of the presence of any other feature. Operates by computing the posterior probability of each class given the input features and assigns the class with the highest probability. It presents as advantages, the fact that it is simple, fast, efficient with high-dimensional data, and works well with small datasets (McCallum & Nigam, 1998).

2.8.4.8 Multi-layer Perceptron (MLP)

MLP is a class of feedforward artificial neural network (ANN) that consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. It uses backpropagation to train the network. The process consists in transforming input data through a series of weighted layers using activation functions to predict the output class. As such, it can model complex relationships. Nevertheless, MLP requires a large amount of data and computational power, and it can be prone to overfitting (Bishop, 2007).

2.8.4.9 AdaBoost

AdaBoost, or Adaptive Boosting, is an ensemble technique that combines multiple weak learners to create a strong classifier. It adjusts the weights of misclassified instances so that subsequent models focus more on difficult cases. It works by sequentially adding weak learners through weight adjustments according to the errors of the previous models. This technique proved to be effective for binary classification, it is relatively easy to implement, and works well with various weak learners (Freund & Schapire, 1995).

2.8.4.10 CatBoost

CatBoost is a gradient boosting algorithm that is particularly effective with categorical features. It automatically handles categorical variables and prevents overfitting with minimal hyperparameter tuning. It works by using ordered boosting and a special technique called Ordered Target Statistics (ordered TS). The idea behind Ordered TS is to calculate the target statistics (like the mean target value for each category) in an ordered manner to minimize bias when using categorical features. As a consequence, it can handle categorical variables directly, it is robust to overfitting, and efficient on large datasets (Prokhorenkova et al., 2018).

2.8.4.11 XGBoost

XGBoost (Extreme Gradient Boosting) is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It is based on gradient boosting and provides more regularization to reduce overfitting by using a gradient boosting framework with additional regularization to optimize both speed and performance. This makes it efficient and robust to overfitting (T. Chen & Guestrin, 2016).

[1] Baye's Theorem provides a way to update the probability of a hypothesis A based on new evidence B. The theorem shows how the prior probability $P(A)$ is adjusted by multiplying it by the likelihood $P(B|A)$, and normalizing by the overall probability of the evidence $P(B)$ (Mohri et al., 2018).

2.8.5 Machine Learning Evaluation Metrics

Determining the most significant metric depends on the specific context of the classification problem and the goals of the analysis. Accuracy, Recall, and F1 score are widely used to evaluate classification ML models, such as the ones used in the classification mechanism, because they provide insights into different aspects of each model's performance (Hassan et al., 2022).

Accuracy: Accuracy is the most straightforward metric and is often used as a primary measure of a model's performance. Accuracy is significant when all classes are equally important, and false positives and false negatives have similar costs (Hassan et al., 2022).

Recall: Recall is essential when identifying all positive instances is more critical than avoiding false positives. Recall is significant in scenarios where missing positive instances (false negatives) is more costly than incorrectly labeling negative instances as positive (false positives). For example, in medical diagnosis, it's crucial to capture all instances of a disease, even if it means more false positives (Hassan et al., 2022).

F1 Score: The F1 score balances precision and recall and is useful when there is an imbalance between the classes or when the cost of false positives and false negatives is significantly different. The F1 score is significant when you want to find a balance between precision (the ability of the classifier not to label as positive a sample that is negative) and recall (the ability of the classifier to find all positive samples). It's particularly valuable in binary classification problems with imbalanced classes (Hassan et al., 2022).

3 Analysis and Design

This chapter is focused on an architectural proposition for the web service configuration and publishing platform. Since one of its objectives is also to register services in the SSC, the use cases for the SSC and for the web service configuration and publishing platform are also identified and presented.

3.1 Requirements

The formalization of functional and non-functional requirements was guided by the FURPS+ model. This model allows capturing the functional and non-functional requirements from the user's perspective and provides a clear description of the system's expected behavior (Kruchten, 1995). Functional requirements represent the main functionalities of the application. Non-functional requirements describe relevant architectural aspects that are not directly related to the operation of the application (Eeles, 2005).

3.1.1 Intelligent Decision Support (IDeS) Framework

In Table 3 the functional requirements regarding the IDeS framework are summarized with information about the type of user that can perform them.

Table 3 – Service publishing system functional requirements.

Requirement	Guest	Author	Admin
Configure service	✗	✓	✓
Edit service configuration	✗	✓	✓
Register Service	✗	✓	✓
Create service's home page	✗	✓	✓
View services homepage	✓	✓	✓
Test the service's execution	✓	✓	✓
View activity logs	✗	✗	✓
Classify the web service before registering it	✗	✓	✓

Non-functional requirements define overall qualities or attributes of the application rather than specific behaviors and include the following:

- Intuitive interface that allows quick access to service settings (viewing and editing).
- Guided step by step configuration process.
- Errors must be detected in advance, providing relevant information to the user.

- The system must remain operational after a failure occurs.
- The impact of a failure should be minimized through modular and segmented implementation.
- Only users from host institution's Lightweight Directory Access Protocol (LDAP) directory should be able to use the system.

3.1.2 Semantic Services Catalog (SSC)

In Table 4 the functional requirements regarding the service registering system are summarized with information about the type of user that can perform them.

Table 4 – Service registering system functional requirements.

Requirement	Guest	Admin
Register a service.	✓	✓
Deregister service.	✓	✓
Delete service.	✗	✓
View list of services currently registered.	✓	✓
View a documentation page about the system's usage.	✓	✓
Search services through manually entered SPARQL queries and through intelligent semantic service discovery techniques.	✓	✓
Obtain Service composition suggestions in case the search does not return a unique service that satisfies the solicitor's requirements.	✓	✓
Configure templates for SPARQL queries.	✗	✓
Configure of RDF languages permitted.	✗	✓

In the case of the service registering system Non-functional requirements are defined as follows:

- **Authentication**: only authenticated users will be able to perform management tasks.
- **Scalability**: The system must be able to increase its capacity and functionality based on the needs of its users.
- **Reliability**: The application must present the least possible flaws in its use and must have high availability in the functionalities that it presents.
- **Maintainability**: Good software analysis and design practices must be adopted.
- **Design constraints**: Adoption of good design practices.

3.2 Use Cases

Based on the previous functional requirements analysis, several use cases were identified for each proposed application. The use case view (Figure 9) intends to capture the use cases and respective actors, reflecting the main features of the system from the user's perspective (Kruchten, 1995).

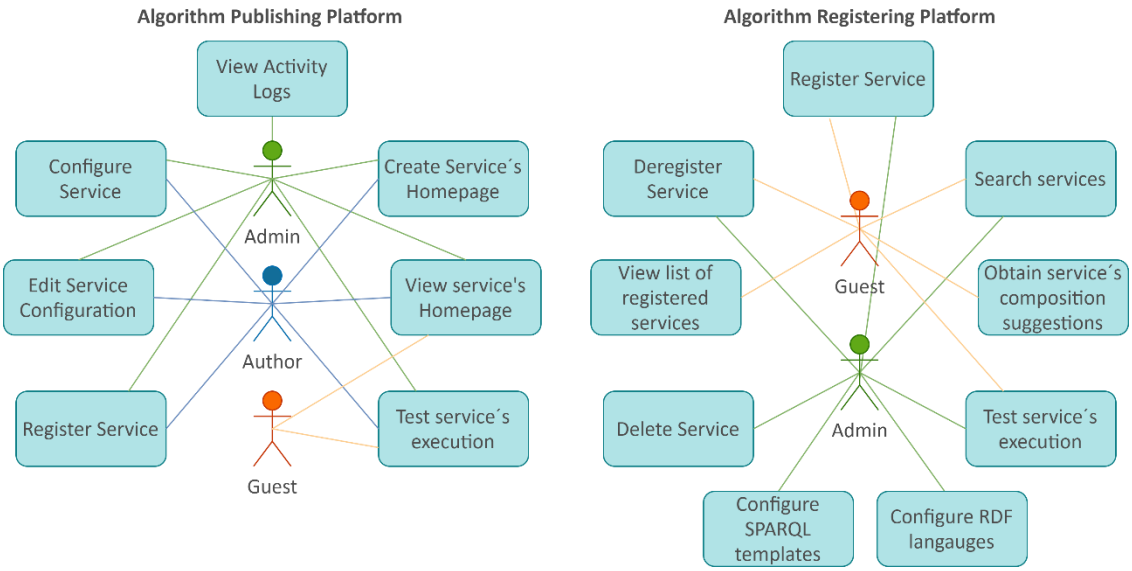


Figure 9 – Use case view for each platform.

As can be observed in Figure 9, the algorithm publishing platform includes use cases related to the configuration of the service to be published, the creation of the services home page and the testing of the execution of the service. The algorithm registering platform comprises use cases related to service registering, discovering, and composing. It also includes administrative use cases, such as, SPARQL templates and RDF languages configuration.

3.3 System's Architecture

The system's architecture includes the architecture of the two previously mentioned platforms. In this context, Figure 10 summarizes de architecture of both systems and how they interact.

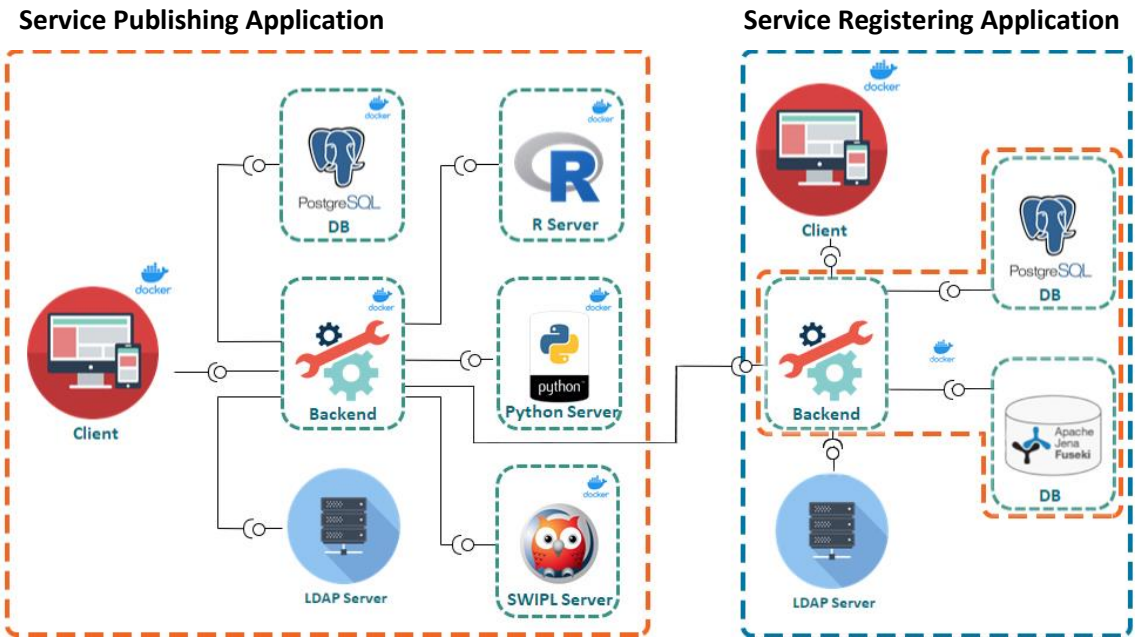


Figure 10 – Publishing and registering application's architecture.

The IDeS framework has three main pages:

- The configuration page, where a guided step by step configuration of the service is presented.
- The service page which, after the configuration is completed and the publishing process is finalized, provides a summary of the service, along with its documentation and test area.
- The reconfiguration page, where a previously published service can be edited. The reconfiguration page also allows admins to view logs regarding operations and errors about the service. These functionalities are illustrated in Annex B: IDeS Framework User Interface.

The SSC is a web application where the list of the registered services can be viewed, along with the documentation about the application and a reserved admin area for configuring SPARQL queries and type of languages allowed. A dedicated area for services search and composition suggestion is also present allowing to search for a specific service or retrieve a set of services that, together, provide the same functionalities as the searched service.

Since a service can be registered in the publishing process, the publishing application communicates with the registering application to do so.

3.3.1 IDes Framework Architecture

The main idea behind IDeS framework (Figure 11) is to provide a user-friendly service configuration tool that enables authors to configure and publish a service and register it in the SSC.

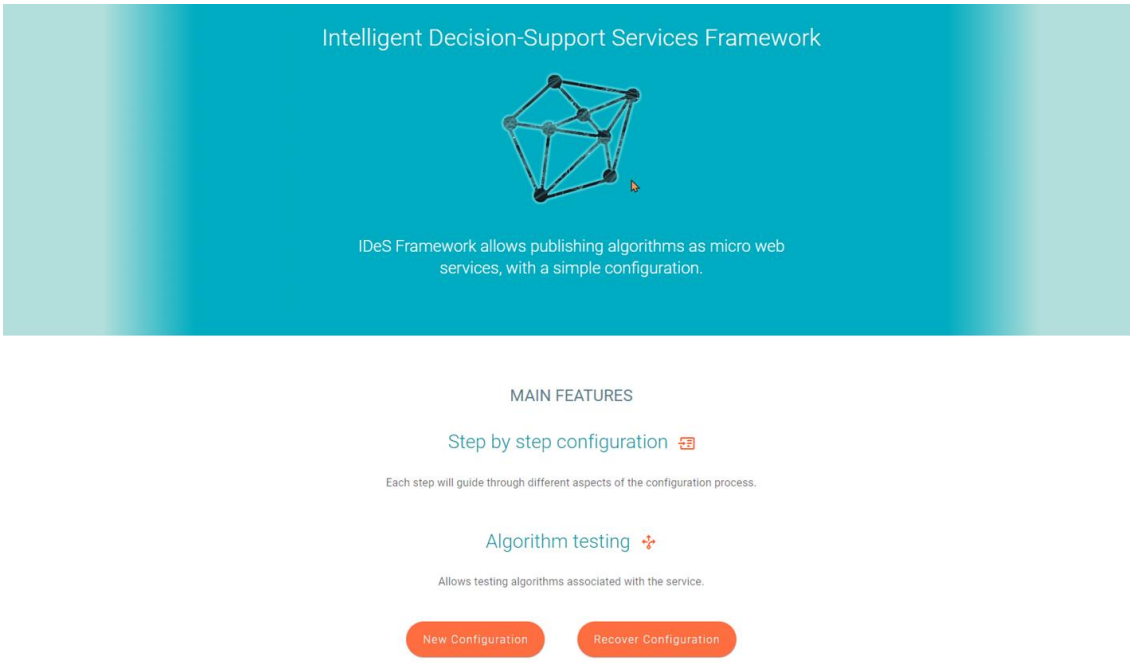


Figure 11 – IDeS Framework home page.

The configuration process is comprised of six steps with tool tips to aid in the configuration. In the first step, the user defines the service name, uploads the execution file(s), defines the development language, establishes a base request path, uploads the services logo (optional), and provides a service's summary. In step two the authors information is provided by the user. Steps 3 and 4 are dedicated to defining all aspects related to the services requests. Step 5 allows to enter the information needed to register the service in the SSC and proceeds to register the service. And, in the final step a preview of the page of the service is provided along with a section to test the service's execution. Since the main focus of this work resides in the integration of the classification and discovery/composition algorithms, the architectural and implementation details described in this section will focus on the mentioned integration. The remaining detailed functionality diagrams are present in the Annex A: Process Diagrams.

As illustrated by Figure 12, the IDeS framework features a modular and distributed architecture.

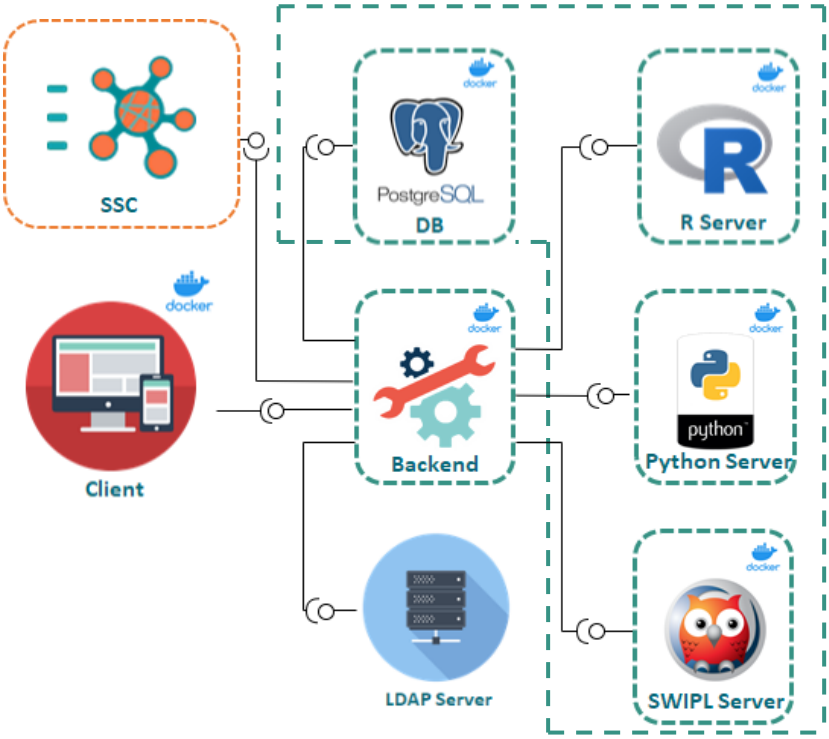


Figure 12 – IDeS Framework component view.

Regarding the IDeS framework's components, que architecture in comprised of a client interface, which interacts with the backend services for the purpose of configuring and publishing services. In the configuration process, it presents a step-by-step service configuration, service page pre-visualization, service page generation, service registration in the SSC, and service execution. The backend handles the core business logic and communication between components, providing various endpoints, including those for authentication, create, read, update and delete service configuration, service page data, and service registration (which sends a registration request to the SSC). Also including endpoints for service execution, and administrator configuration. The technology used for persistence is the PostgreSQL which stores data related to the web service configuration and user data. There is also a component for each supported programming language (Python, R, SWI-Prolog), allowing for the published web service's execution and testing.

3.3.2 Classification and Discovery/Composition Modules Integration Architecture

Having previously described the way the IDeS framework and the SSC interact; this section describes how the classification module is integrated with the IDeS framework and how the discovery/composition module interacts with the SSC.

Figure 13 illustrates the interaction between the classification module and the IDeS framework so that, in the moment before the web service registration in the SSC, the web service being created is classified according to the set of groups which are described further ahead. It also presents how the discovery/composition module is integrated with the SSC in order to allow for an intelligent web service discovery and composition, thus enhancing the search functionality of the SSC. Both classification and discovery modules are implemented as python scripts that include the establishment of endpoints for the necessary operations.

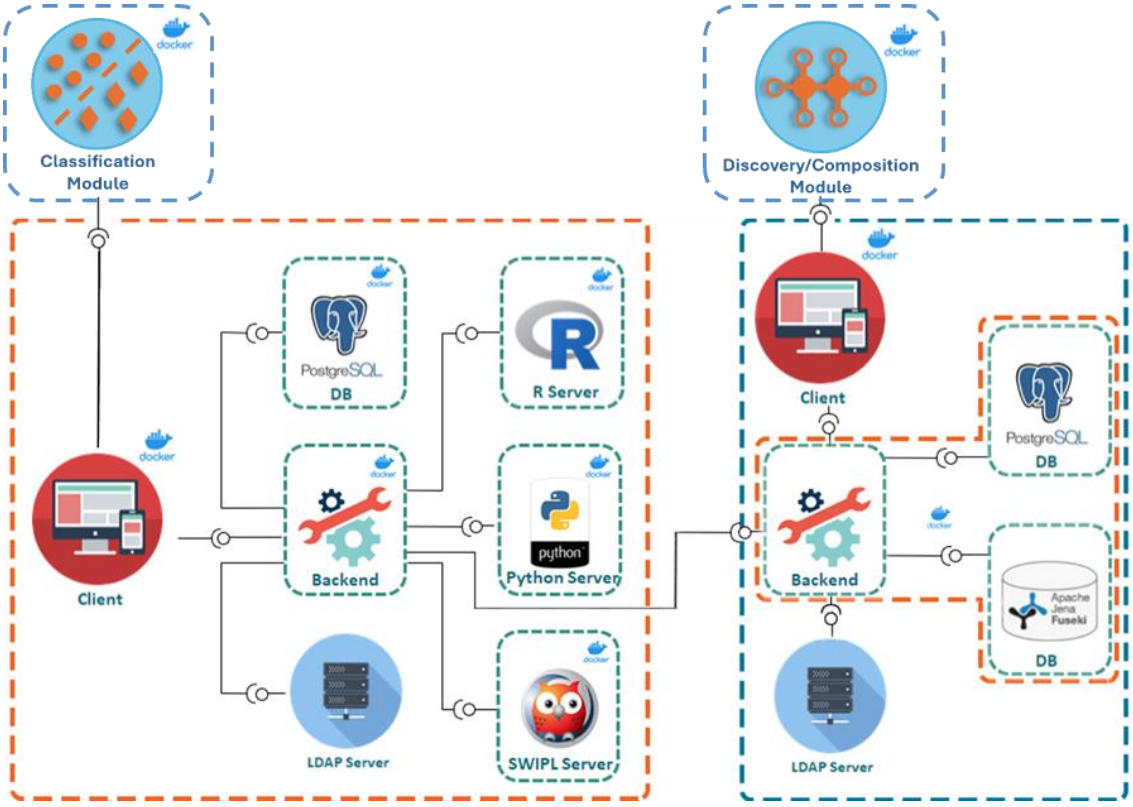


Figure 13 – IDeS framework and SSC integration with the classification and discovery/composition modules.

As Figure 13 indicates, the classification module interacts directly with the IDeS framework UI in order to present the web service classification suggestions, as well as receiving the accepted classification by the user.

3.3.3 Discovery/Composition Module and SSC Interaction

The diagram in Figure 14 illustrates the SSC Process View for Service Discovery/Composition. It outlines the process where a user searches for services within SSC, focusing on the interaction between the UI, the Discovery/Composition component, the SSC and Fuseki.

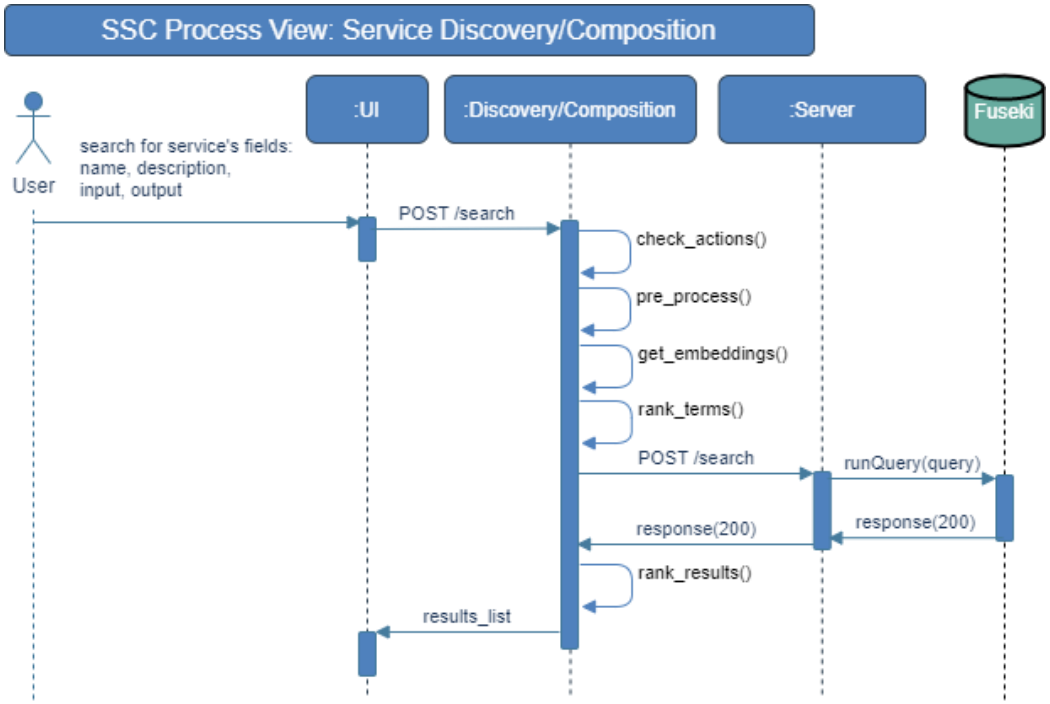


Figure 14 – Process View – Service discovery and registration.

In this sequence the process of searching for services (or compositions of services) within the SSC is described. The user starts by performing a search for services, specifying fields such as name, description, input, and output parameters through the UI. Once the search criteria are set, the UI sends a POST request to the Discovery/Composition component at the /search endpoint.

The Discovery/Composition component, then, processes the search request through several steps. First, it uses the check_actions() method to verify the request and identify any possible multiple actions that will result in a query for each action, thus enabling eventual compositions. Then, it pre-processes the search query using NLP, generating embeddings for more efficient matching against stored services through the get_embeddings() step. The search terms are ranked according to their relevance using the rank_terms() method.

After this pre-processing and ranking, the Discovery/Composition component forwards the processed query to the SSC component via another POST request to the /search endpoint. The SSC component interacts with the triple store, through the runQuery(query) method which sends a request to find services that match the processed query in the database. Fuseki responds with a list of matching services, and the SSC sends a success response (response 200) along with the list(s) of results back to the Discovery/Composition component.

The Discovery/Composition component, then, ranks the returned list of results by running the rank_results() method, ensuring that the services retrieved are ordered based on their relevance to the user's search query. Once the results are ranked, the Discovery/Composition component sends the list of ranked services back to the UI, where they are presented to the user.

3.3.4 Classification Module and IDeS Framework Interaction

Figure 15 shows the end-to-end flow of registering a service using the IDeS framework, starting from the user request to the successful registration in SSC, with interactions across multiple layers such as classification, routing, controlling, and service execution.

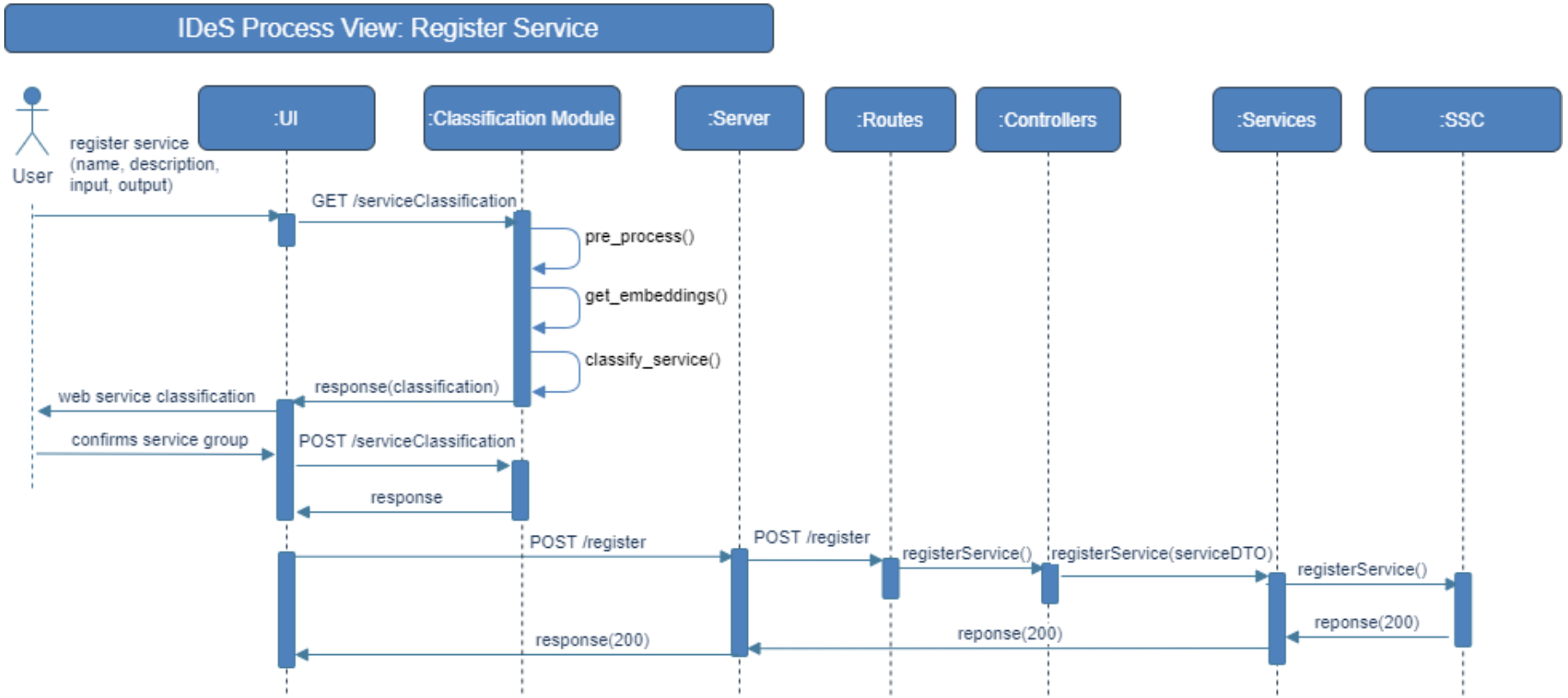


Figure 15 – Process View – Service classification and registration.

The user begins by registering a service through the UI, providing details such as the service name, description, input, and output parameters. After receiving a suggested group for the service, the user confirms the suggested service group or enters a new one.

Next, the UI sends a GET request to the Classification Module at the /classify endpoint to retrieve relevant classifications for the service. The Classification Module processes the request and sends a classification response back to the UI. The UI then displays the classification suggestion, asking for user confirmation. Along with the predicted group classification, the user is presented with a list of groups, each accompanied by a brief description to verify the prediction and aid in group selection, as shown in Figure 15.

After confirming the classification, the UI sends a POST request to the /addNewService endpoint to submit the service classification details defined by the user, which triggers the retraining stage with the new classification in place.

Following the confirmation of the service classification, the UI sends a POST request to the SSC Server at the /register endpoint to initiate the service registration process. The server forwards this request to the Routes component, which redirects it to the appropriate Controller. The Controller receives the request and invokes the registerService() method with the serviceDTO (a Data Transfer Object containing all the service details).

In the next step, the Controller calls the registerService() method in the Services layer to handle the business logic of service registration. The Services layer then interacts with the SSC to complete the registration by calling its registerService() method. Finally, the SSC responds with a success message (response 200), confirming that the service has been successfully registered.

4 Exploratory Analysis of ML and NLP Techniques

Given the need to frequently add new services to the database by users with varying levels of expertise, it became clear that integrating NPL and ML techniques would allow users to find semantic web services tailored to their needs through keyword searches. As noted by (Ngan & Kanagasabai, 2013), employing common ML algorithms like regression, decision trees, or SVMs can greatly improve initial matchmaking methods. Once the keywords are aligned with the categories of semantic web services, different methods can be used to accurately interpret the terms. Furthermore, as previously mentioned, using classification techniques to categorize web services can help refine the search results. Consequently, the exploratory process for web service discovery and composition was centered on developing solutions based on these principles.

Having analyzed both problems and opportunities and presented the architectural design of the proposed systems, the next step is focused on the exploratory aspects of implementing the classification, and discovery/composition modules. Thus, this chapter illustrates the exploratory work and development of ideas, directed at selecting the best strategies to implement the classification mechanism along with the search/composition mechanism.

4.1 Dataset Characterization

To advance through the exploratory phase, a dataset of web services was first created. This dataset includes a variety of web services, each described by several attributes (organized in columns): "Name of Service," "Description of Service," "Input of Service," "Output of Service," and "Group." The content of the dataset consists of various services grouped into categories indicated in the "Group" column, with each group containing services related to a specific domain or functionality.

The "Name of Service" refers to the name of the web service, while the "Description of Service" provides a brief overview of what the service does. The "Input of Service" outlines the input parameters required by the service, and the "Output of Service" details the output provided by the service. The "Group" serves as a numeric identifier categorizing the service into a specific group. The dataset consists of the categories described in Table 5.

Table 5 – Dataset group characterization.

Group	Designation	Description
1	Mathematical Operations	APIs to perform mathematical operations

Group	Designation	Description
2	Stock Market Data	APIs for retrieving stock market data, including real-time and historical prices.
3	Authentication Services	Services for verifying user credentials and providing access tokens
4	Online payment related operations	APIs for online payment related operations
5	Event and Calendar Management	Tools for managing events, checking availability, and setting reminders
6	File Management	Services for archiving, backing up, and securely sharing files
7	Currency Conversion	Services for converting currencies, including real-time and multi-step conversions
8	Data Aggregation and Analytics	Tools for aggregating, analyzing, and visualizing data
9	Travel and Distance Calculation	APIs for calculating travel distances, durations, and planning routes
10	Email Services	Services for sending, receiving, and verifying emails
11	Image Compression	Tools for compressing images, with options for batch processing and lossless compression
12	Job Search and Listings	Platforms for searching job openings based on companies, industries, or specific skills
13	Language Translation	Services for translating text or documents between languages
14	Music and Concert Services	Services related to purchasing concert tickets, retrieving lyrics, and recommending playlists
15	News Services	APIs for fetching and alerting news based on user-defined criteria
16	Survey Tools	Tools for creating, distributing, and analyzing surveys.
17	Payment and Billing Services	Services for managing cryptocurrency transactions, digital wallets, and subscription billing
18	SMS Services	APIs for sending SMS messages and verifying phone numbers
19	Social Media Services	Tools for managing social media ads, analyzing social media data, and sharing content
20	Document and Optical Character Recognition Services	Services for adding annotations to documents and converting images to text
21	URL Management	Tools for managing, customizing, and validating URLs

Group	Designation	Description
22	Video Services	APIs for integrating video chat, analyzing video content, and streaming videos
23	Weather Information	APIs for retrieving current weather information and weather maps
24	Customer Support and Shipping	Services for providing customer support, managing shipping logistics, and handling subscriptions
25	Airport and Flight Information	Services providing airport weather information, tracking baggage, and notifying gate changes

For each group of Table 5, at least 12 web service examples were included (Figure 16). Group 5 (File Management) has the higher number of examples (15).

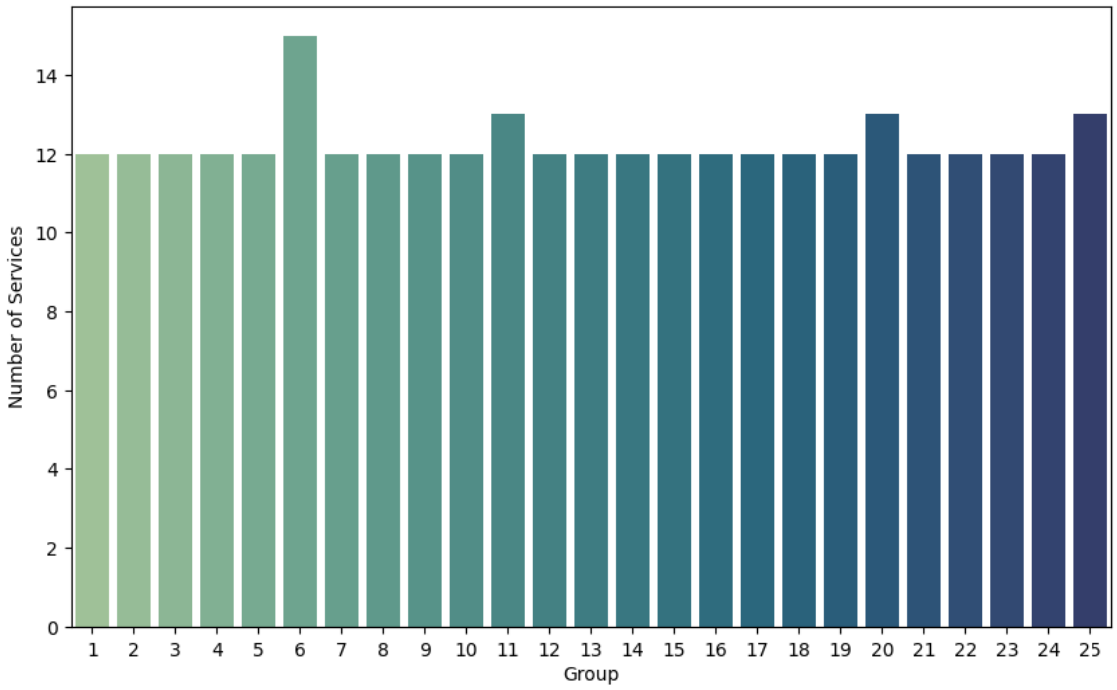


Figure 16 – Number of Services by Group.

Figure 17 depicts a word cloud based on each web service's description. From the analysis of the word cloud, the most prominent words are, the word "Manage" stands out as the most prominent in the word cloud, indicating that many services in the dataset are related to management functions, such as data management, user management, or resource management. Following this, the words "User" and "Data" are also frequent, suggesting that numerous services focus on user data management, manipulation, or access, potentially involving user authentication, data retrieval, or storage. The word "Provide" further indicates that many services are designed to deliver or supply specific functions or data, aligning with the purpose of service APIs.

Other significant words include "Payment," which implies a substantial focus on financial transactions, payment processing, or management within the service descriptions. Additionally, the words "Convert," "File," "Image," "Flight," and "Event" suggest that there are services related to file conversions, image processing, flight information, and event management.

Figure 17 – Word cloud based on web services descriptions.

From the word cloud, it is also possible to infer common themes. The presence of words like "Manage," "Data," "User," "Document," and "File" points to a significant number of services involved in managing and processing various types of data. The words "Payment" and "Transaction" highlight a focus on financial services, including payment handling, transaction verification, and related activities. The terms "Image," "Video," and "Convert" suggest services dealing with multimedia content, such as conversion, compression, or streaming. Lastly, the words "Flight" and "Location" indicate services related to travel, logistics, or real-time tracking.

Another relevant aspect is the occurrence of several action-oriented words. Words like "Retrieve," "Send," "Store," "Track," "Handle," "Calculate," "Convert," and "Analyze" are action verbs that indicate the kinds of operations the services perform. This shows that many services are designed to perform specific actions on data or user inputs.

From the word cloud it is also possible to get some additional insights. Firstly, the diversity of web services becomes clear, since the word cloud reflects a diverse range of services offered, from user management to data processing, payment handling, content conversion, and more. Lastly, there is a user and data-centric focus, since a significant emphasis on terms related to "User" and "Data" suggests that a majority of the services revolve around user-centric operations or data handling.

As such, this dataset is used with two main purposes. To train ML models that categorize web services into predefined groups based on their descriptions, inputs and outputs. And to test discovery and composition strategies in order to achieve results that align with the established requirements.

4.2 Classification: Exploratory Process and Results

In order to test the best encoder to be able to improve the classification results, three encoding approaches were used. One-Hot Encoding, Word2Vec, and TF-IDF.

The following is an example of the type of services added in the tests that were performed:

- “Name of Service”: “FileStore”,
- “Description of Service”: “File storage”,
- “Input of Service': “File to be stored.”,
- “Output of Service”: “Confirmation of storage.”

Using services with this structure, the three encoding approaches were tested and evaluated. The tests followed the same procedure, differing only in the encoding stage. As such, the testing pipeline can be summarized as follows:

1. **Text Preprocessing**: Tokenization, stop word removal, and lemmatization.
2. **Categorical Variables Encoding**:
    a. One-Hot Encoding to transform the preprocessed text attributes into encoded features suitable for machine learning models.
    b. Word2Vec Vectorization: Vectorizing text using a pre-trained Word2Vec model and averaging word vectors.
    c. TF-IDF Vectorization: A pre-trained TF-IDF Vectorizer transforms the combined text of the new service into TF-IDF features. This converts text data into numerical format suitable for machine learning.
3. **Supervised Classification**: Using multiple pre-trained and retrained classification models to categorize the new service.
4. **Evaluation and Filtering**: Selecting high-performing models based on accuracy, F1 score, and recall.
5. **User Interaction**: Allowing user input for final group selection.
6. **Model Updating and Retraining**: Regularly updating the dataset and retraining models to incorporate new data.

4.2.1 Classification Results Analysis

In Table 6 the results obtained using example provided in the beginning of this chapter (“Name of Service”: “FileStore”, “Description of Service”: “File storage”, “Input of Service': “File to be stored.”, “Output of Service”: “Confirmation of storage.”) are presented. It can be observed that only Naïve Bayes was able to predict the expected group. Naive Bayes has the highest accuracy among all the models at 0.772. This means it correctly classifies 77.2% of all cases, including this particular instance where it correctly identified the group as 5. Naive Bayes also has a good recall score of 0.72, suggesting it is effective at correctly identifying true positives (i.e., correctly classifying web services into their appropriate groups). While some models like Random Forest and Multi-layer Perceptron have similar accuracy scores (0.74), their predictions

were incorrect for this specific case. This implies that their high accuracy might come from correctly predicting other instances but failing in this particular one. Hence, given these mixed results, One Hot Encoding may not be the ideal encoding approach.

Table 6 – Results above 0.6 for each metric using One-Hot Encoding (Correct prediction:5).

Model	Prediction	Accuracy	F1 Score	Recall
Random Forest	19	0.74	0.77	0.74
Multi-layer Perceptron	23	0.74	0.77	0.74
Logistic Regression	15	0.72	0.77	0.72
Naive Bayes	**5**	**0.772**	**0.74**	**0.72**
Gradient Boosting	15	0.70	0.75	0.70
Decision Tree	15	0.61	0.67	0.61

Table 7 presents the results obtained using Word2Vec encoding which denote a noticeable difference between the previous results.

Table 7 – Results above 0.6 for each metric using Word2Vec (Correct prediction:5).

Model	Prediction	Accuracy	F1 Score	Recall
CatBoost	**5**	**0.87**	**0.88**	**0.870**
Support Vector Machine	5	0.85	0.87	0.85
Random Forest	5	0.81	0.81	0.81
XGBoost	5	0.73	0.70	0.73
k-Nearest Neighbors	5	0.71	0.70	0.71
Gradient Boosting	5	0.68	0.70	0.68

CatBoost performed the best, with the highest accuracy (0.87), F1 score (0.88), and recall (0.87), closely followed by the Support Vector Machine (Accuracy 0.85). The general performance across models improved with Word2Vec, indicating its effectiveness in capturing semantic relationships within the text data. This shows that using Word2Vec encoding produces better results than using One Hot Encoding.

In Table 8 the results gathered using TF-IDF encoding are presented. As it can be observed, this was the encoding that provided the best results.

Table 8 – Results above 0.6 for each metric using TF-IDF (Correct prediction:5).

Model	Prediction	Accuracy	F1 Score	Recall
Logistic Regression	**5**	**0.935484**	**0.944905**	**0.935484**
Random Forest	5	0.919355	0.931541	0.919355
Support Vector Machine	5	0.919355	0.923477	0.919355
Naive Bayes	5	0.903226	0.914337	0.903226
k-Nearest Neighbors	5	0.903226	0.898797	0.903226
Multi-layer Perceptron	5	0.887097	0.891398	0.887097
CatBoost	5	0.887097	0.889606	0.887097
Gradient Boosting	5	0.854839	0.873297	0.854839
Decision Tree	5	0.838710	0.842832	0.838710
XGBoost	5	0.806452	0.804301	0.806452

Logistic Regression achieved the highest accuracy (0.935), F1 score (0.944), and recall (0.935), suggesting that TF-IDF is the most effective encoding method among the three. Other models such as Random Forest and Support Vector Machine also performed well, with accuracies above 0.9. As it becomes clear, using TF-IDF results in an even higher number of correct classifiers, accompanied by an increase in the score of all metrics applied.

To finalize the results analysis, a summary focusing encoding effectiveness and model performance follows. In term of encoding effectiveness, One-Hot Encoding showed limited effectiveness. Among the models tested, only Naive Bayes successfully predicted group 5, and even then, the performance metrics were moderate. This method may not effectively capture complex relationships within the text data, resulting in subpar model performance. On the other hand, Word2Vec demonstrated improved effectiveness. All models that utilized Word2Vec were able to correctly predict group 5, and the models displayed higher accuracy and F1 scores compared to those using One-Hot Encoding. This improvement is likely due to Word2Vec's ability to capture semantic relationships, which aids in better classification. The most effective encoding method was TF-IDF. All models using TF-IDF correctly predicted group 5, and their performance metrics generally surpassed those of models using Word2Vec and One-Hot Encoding. This indicates that TF-IDF more effectively captures the key features and terms within the service descriptions, leading to enhanced classification performance.

Regarding model performance, Logistic Regression with TF-IDF encoding emerged as the best-performing model across all encoding methods, achieving an accuracy of 0.935, an F1 score of 0.944, and a recall of 0.935. The general performance trend observed was that models performed best with TF-IDF encoding, followed by Word2Vec, and performed least effectively with One-Hot Encoding. This trend suggests that for text-based classification tasks, TF-IDF may be the superior feature extraction method, as it captures relevant discriminative information more effectively than both One-Hot Encoding and Word2Vec. As such, in the implementation of this module, only the TF-IDF encoder was used due to the fact that this encoder had consistently improved the results obtained in the performed tests.

4.3 Discovery: Exploratory Process and Results

The approach followed to implement the search mechanism was incremental and based on empirical experimentation. This experimentation was organized according to search terms that were used to compare different strategies for preprocessing and extending data. Firstly, tokenization and word sense disambiguation (WSD) were used in order to split text into words or tokens and find the correct sense of words in the user query. This process produced results that were lacking in returning web services with related terms or synonyms. Although this approach provided precise results by correctly interpreting the user's intent, it was limited to exact matches. It failed to retrieve web services that contained related terms or synonyms, leading to a narrow scope of search results.

To overcome this limitation, word embeddings were used to expand on the search words. This greatly improved the spectrum of results, but also resulted in the use of terms that where clearly out of scope, returning the expected results, plus a large number of results that were not expected. Finally, with the objective of removing unwanted terms, similarity measures were applied, firstly to rank the word embedding terms and choose just the top n percent, secondly to order the results provided by the database in relation to the search terms. Another modification made, was the replacement of Word2Vec with SBERT which improved the accuracy of the embeddings and the overall performance of the system. The tables presented in Discovery Results Analysis illustrate the experimentation process and its results. The top left corner of each table indicates the search field and the corresponding search term or phrase.

4.3.1 Discovery Results Analysis

Table 9 shows that using tokenization and WSD alone retrieved a very limited number of precise results (true positives), while the addition of word embeddings increased the number of results but also introduced a high number of false positives. Table 10 to Table 14 demonstrate the effectiveness of the semantic similarity ranking. The application of similarity measures consistently reduced the number of false positives while maintaining or slightly reducing the number of true positives, thus balancing precision and recall. For instance, in Table 10 with the search term "Sum," the combination of tokenization, WSD, word embeddings, and semantic similarity ranking resulted in a balanced outcome of 22

results with 19 true positives and only 3 false positives, significantly improving over the embeddings-only approach. Similarly, in Table 11 to Table 13, semantic similarity ranking proved effective in reducing false positives across different search terms, such as "forecast," "price," and "cost," while retaining the true positives. This demonstrates the robustness of the refined search mechanism in handling a variety of search queries and contexts.

Table 9 – Results analysis for search term “Addition” in the “Name of Service” field.

Name: “Addition”	number of results	true positive	false positive	false negative
Tokenization + WSD	1	1	0	0
Tokenization + WSD + Word Embeddings	306	1	305	0
Tokenization + WSD + Word Embeddings + Semantic Similarity Ranking	2	1	1	0

Table 10 – Results analysis for search term “Sum” in the “Name of Service” field.

Name: “Sum”	number of results	true positive	false positive	false negative
Tokenization + WSD	2	2	0	19
Tokenization + WSD + Word Embeddings	65	20	45	1
Tokenization + WSD + Word Embeddings + Semantic Similarity Ranking	22	19	3	2

Table 11 – Results analysis for search term “forecast” in the “Description of Service” field.

Description: “forecast”	number of results	true positive	false positive	false negative
Tokenization + WSD	1	1	0	1
Tokenization + WSD + Word Embeddings	7	2	5	0
Tokenization + WSD + Word Embeddings + Semantic Similarity Ranking	2	2	0	0

Table 12 – Results analysis for search term “price” in the “Description of Service” field.

Description: “price”	number of results	true positive	false positive	false negative
Tokenization + WSD	6	5	1	11
Tokenization + WSD + Word Embeddings	28	16	12	0
Tokenization + WSD + Word Embeddings + Semantic Similarity Ranking	24	16	8	0

Table 13 – Results analysis for search term “cost” in the “Description of Service” field.

Description: “cost”	number of results	true positive	false positive	false negative
Tokenization + WSD	6	5	1	30
Tokenization + WSD + Word Embeddings	49	36	13	0
Tokenization + WSD + Word Embeddings + Semantic Similarity Ranking	48	36	12	0

Table 14 – Results analysis for search term “cost of shipping” in the “Description of Service” field.

Description: “cost of shipping”	number of results	true positive	false positive	false negative
Tokenization + WSD	9	4	5	0
Tokenization + WSD + Word Embeddings	63	4	59	0
Tokenization + WSD + Word Embeddings + Semantic Similarity Ranking	23	4	19	0

This experimentation phase allowed for the development of a search mechanism that represents a comprehensive approach to the usage of various NLP and ML techniques to enhance a semantic search system. It combines WSD, word embeddings, and semantic similarity computations to rank web services from a SPARQL endpoint based on user queries.

4.4 Composition: Exploratory Process and Results

As previously stated, the discovery/composition process starts with the user's input. The composition mechanism is only activated when the user fills at least, either the service's input and output fields, or the description field. The input is, then, processed using tokenization, word sense Tokenization, Word Sense Disambiguation, and Lemma Extraction. GloVe and SBERT models provide the tools to expand the terms entered for each field of the service's characterization. After this initial processing, if the description field is filled, further processing is made with the objective of identifying semantically distinct actions separated by verbs and conjunctions in the user's input (this process is explained with further detail in Semantic Web Service Discovery and Composition next section). If more than one action is identified, a query for each action is constructed with the expanded terms of each action. The result of each query (a list of web services) represents one stage of the composition, and each web service of the list represents a possible match for that stage. If the user also filled the input and output fields, then, the lists will be ordered according to the similarity between each service's input and output, and the information entered by the user. In case, only the input and output fields are filled, the matching process focuses on the degree of similarity between the user's input and output and each service's input and output.

4.4.1 Composition Results Analysis

In this example (Table 15), the input, output and description fields where filled, allowing for an attempt to split the description into actions. The function with this task uses the "en_core_web_sm" spaCy language model to split the description according to conjunctions and verbs. The identified actions in this example were: "adds two numbers", and "gets the absolute value of the result". From these two actions, two queries are constructed, using both input and output fields and the corresponding action.

Table 15 – Composition result for example 1.

	number of results	true positive	false positive	false negative
Input: "two numbers"	9	1	8	0
Output: "number's absolute"				
Description: "adds two numbers and gets the absolute value of the result"				

The result consists of two lists where the elements in the first list are composed with the elements on the second list according to the input and output similarities. Table 16 shows the 5 top results:

Table 16 – Composition top 5 results for example 1.

Name: Addition	→	**Name: Absolute Value**
Name: FactorialMultiplication	→	Name: Addition
Name: Absolute Value	→	Name: Multiplication
Name: Subtraction	→	Name: Exponentiation
Name: Exponentiation	→	Name: Factorial

Table 17 shows an example where the description field was not filled. In this case, a query is made with the input and output search terms. After the results the list is split according to output and input similarity. The resulting lists represent the stages of the composition, and each element of each list represent a possible solution for the corresponding stage. Note that, in this example, 58 false positives were returned. Nevertheless, the ranking process ensured that the true positive results were followed by the false positives.

Table 17 – Composition result example 2.

	number of results	true positive	false positive	false negative
Input: "symbol or company"	62	4	58	0
Output: "converted amount"				
Description: ""				

If the same query is completed with a brief description (example: "get stock price and convert it to a specific currency"), the composition mechanism is able to reduce the number of results to 52. In the case the top results remained the same, but there was a reduction in the number of false positives. The top 5 results are shown in Table 18.

Table 18 – Composition top 5 results for example 2 with and without description.

Name: StockPriceAPI 1	→	**Name: RealTimeCurrencyConverter**
Name: StockPriceAPI 2	→	**Name: Currency Converter1**
Name: StockPriceAPI 3	→	**Name: Currency Converter2**
Name: StockPriceAPI 4	→	**Name: MultiCurrencyConverter**
Name: WeatherForecast	→	CryptoCurrencyConverter

Table 19 and Table 20 show additional examples of queries made with only the description field filled, and with input, output and description fields filled. These examples focused on using a different type of phrasing for the description where the distinction between actions is not as clear as in the first example.

Table 19 – Composition result example 3.

	number of results	true positive	false positive	false negative
Input: “”	24	7	17	0
Output: “”				
Description: “'plan flight routes according to weather information”				

Table 20 – Composition result example 4.

	number of results	true positive	false positive	false negative
Input: “file”	1	1	0	4
Output: “status”				
Description: “store files synchronizing across multiple devices”				

Table 21 and Table 22 show the results for the queries performed in Table 19 and Table 20. The results are consistent with the previous examples where the most relevant results appear first, being followed by less relevant results sequentially.

Table 21 – Composition top 5 results for example 3.

Name: FlightRoutePlanner	→	**Name: WeatherAPI1**
Name: Route Planning Service 3	→	**Name: WeatherAPI2**
Name: Route Planning Service2	→	**Name: WeatherForecast**
Name: Route Planning Service1	→	**Name: Weather Service**
Name: Flight Tracker	→	**Name: WeatherAPI3r**

Table 22 – Composition top 5 results for example 4.

Name: Encrypted File Storage	→	Name: Remote File Sync

The Exploratory Analysis of ML and NLP Techniques chapter illustrates the flexibility and effectiveness of the proposed search and composition mechanisms in handling various types of user input. By combining multiple NLP techniques, such as tokenization, WSD, lemma extraction, and word embeddings, the system can effectively expand and refine user queries, enhancing search results. Moreover, the use of similarity measures to rank and filter results ensures that the most relevant services are prioritized, thereby improving the overall quality and usability of the search mechanism. The incremental and multi-faceted approach allowed for a more precise understanding of user intent, enabling a more accurate and efficient service composition. This comprehensive methodology aims to be used in dynamic environments where users may provide varied and complex search inputs, necessitating a robust and adaptive search mechanism.

5 Web service classification

This chapter is dedicated to describing the process of web service classification and how it is incorporated in the web service configuration process. Regarding the classification process, there are two stages in the classification phase. Firstly, the base dataset is used to train the clustering models in a comprehensive ML training pipeline for text classification. This happens only once. Next, every time a new service is added, the models are used to classify its group, and the user confirms the new service's group. The dataset with the new service is then used to retrain the models.

5.1 Training Stage

The main operations involved include several key steps, as shown in Figure 18. First, a function is defined for text preprocessing, which handles tokenization, removal of stop words, and lemmatization.

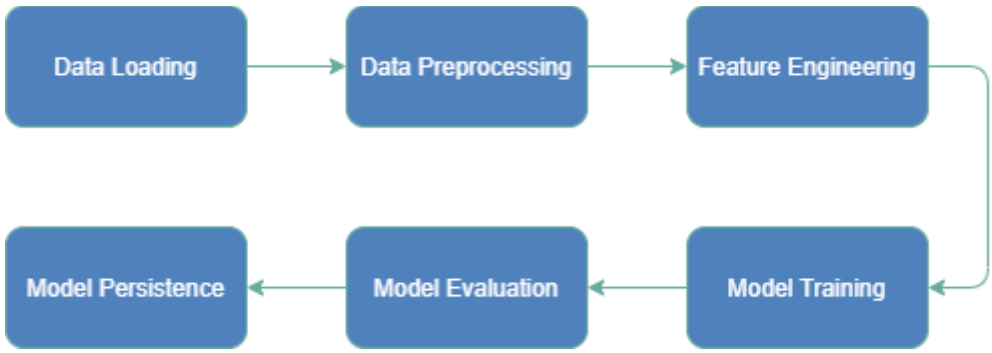


Figure 18 – Training stage summary.

Next, categorical variables are encoded using appropriate encoders to convert features such as 'Name of Service,' 'Description of Service,' 'Input of Service,' and 'Output of Service' into a format that can be used by machine learning models.

Following this, the encoded features are assigned to the variable X, while the target variable 'Group' is assigned to y. The data is then split into training and testing sets, with 85% of the data used for training and 15% reserved for testing.

Model initialization follows, where a dictionary is created to store various ML models that will be used for classification. The models included are Random Forest, Decision Tree, Logistic Regression, Gradient Boosting, k-Nearest Neighbors, Support Vector Machine, Naive Bayes, Multi-layer Perceptron, AdaBoost, XGBoost, and CatBoost. Each model is trained using the training data, and predictions are subsequently made on the test data. The model's performance is evaluated by calculating metrics such as Accuracy, F1 score, and Recall.

Finally, the encoder used for preprocessing, along with the evaluation results for all the models, is saved locally for future use.

5.2 Classification Stage

In this stage, ML models are employed to categorize a new service into either an existing or a new group. As depicted in Figure 19, the process begins with the user adding a new service by providing information for all required fields, including "Name of Service," "Description of Service," "Input of Service," and "Output of Service." The text data is then preprocessed, and categorical variables are encoded to prepare the data for model prediction.

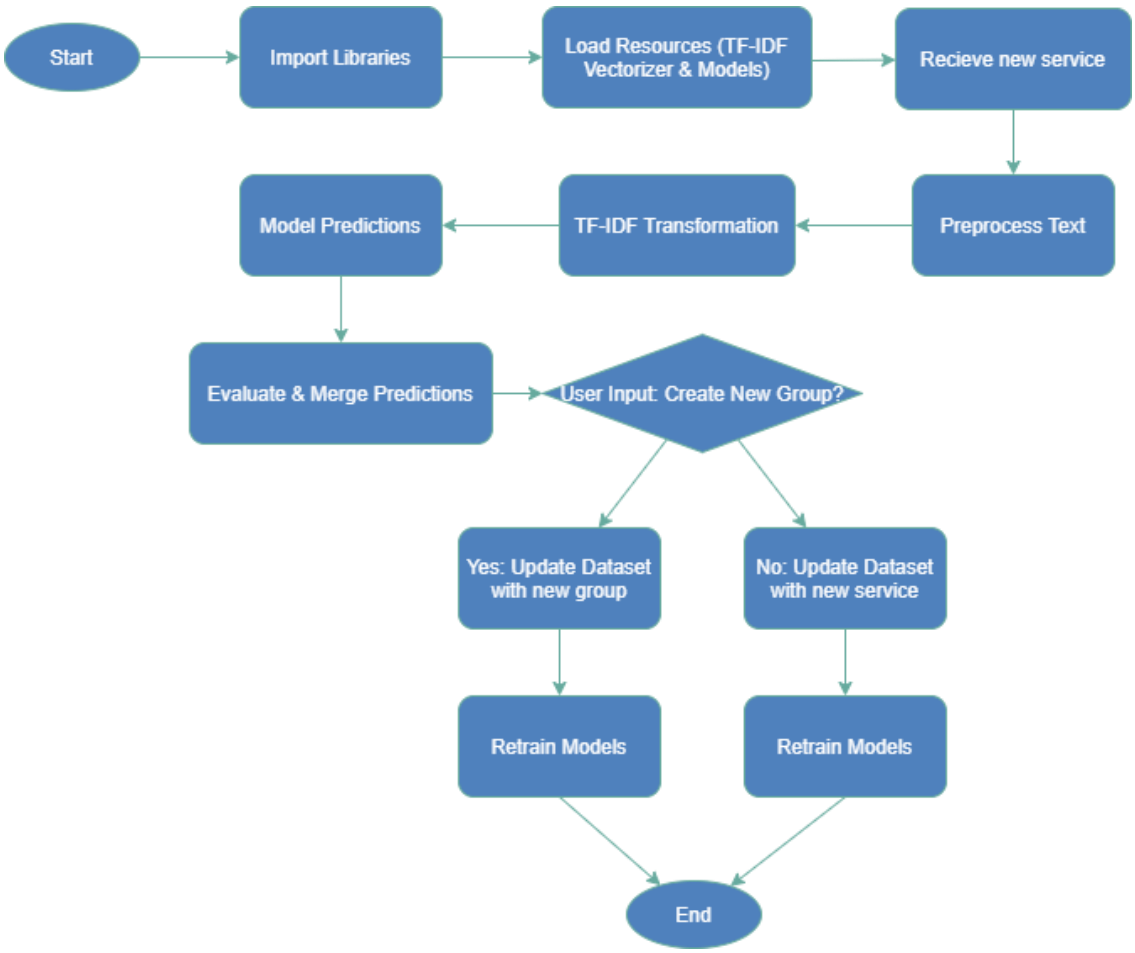


Figure 19 – Classification flow diagram.

The next step involves group prediction, where pre-trained classification models are loaded, and each model predicts the group for the new service. Following this, model evaluation and selection take place. A comma separated values file containing performance metrics (such as accuracy, F1 score, and recall) for each model is loaded, and models with high scores, specifically, those with accuracy, F1 score, and recall greater than 0.6 are filtered and sorted. The unique predicted groups from these high-performing models are then identified.

User interaction follows, where the user is given the option to select one of the proposed groups or create a new group. Finally, the process of updating and retraining begins. The new service is added to the original dataset with the selected group. The text attributes of the updated

dataset are preprocessed, and categorical variables are re-encoded. Each model is then retrained using the updated dataset, and new performance metrics are calculated and saved along with the updated models. The updated performance results and dataset are also saved for future use.

5.3 Integration with IDeS Framework

Upon defining and implementing the classification processes, the classification module was implemented and the communication between the IDes framework and the classification module was established. In this regard, in order to contextualize the way the classification process is integrated in the web service's configuration pipeline, a brief description of each configuration step is presented in this section. The present section includes figures related to step 5 of the configuration pipeline, where the web service classification takes place. The figures illustrating the remaining steps can be found in Annex B: IDeS Framework User Interface.

5.3.1 Step 1: Basic Information

In the first step, the user defines the service name, uploads the execution file(s), defines the development language, establishes a base request path, uploads the services logo (optional), and provides a service's summary, as can be verified in Figure 30 and Figure 31 presented in Annex B: IDeS Framework User Interface.

When defining the service's name, the base request path will be automatically updated, and the service version will be added.

IDeS framework currently supports Python, R or Prolog as script development languages. The development language of the uploaded script must be the same as the development language chosen in the dropdown list. If the script has associated files, they must be uploaded as well as a part of a compressed directory.

A logo of the service can be uploaded to be featured in the service's page. It should be a png image file and its size is be adjusted to the service's home page app bar. In case, no logo is uploaded, the app bar will automatically show the IDeS Framework logo.

As it is shown in the example, the service's summary must describe the main aspects of the service. There are some basic formatting tools to adjust the text to the user's preferences. The objective of this summary is to be presented in the service's home page.

5.3.2 Step 2: Authors Information

In step two the authors information is provided by the user. The user interface for this step is represented by Figure 32 and Figure 33 included in the Annex B: IDeS Framework User Interface.

After clicking the plus icon to add an author, a form for each author must be filled. In order to be able to edit the service after its configuration, the manager checkbox must be checked and a username from the GECAD-ISEP LDAP account must be provided.

After adding a manager, a validation is required. This is done by clicking in the validation button in order to check if the manager's data is valid.

After clicking the validate button, a login form will be presented to fill with the authentication data. If more than one manager is added, the framework is able to check each one without the need to authenticate each new manager.

5.3.3 Step 3 / Step 4: Requests Definition

Next, step 3 and 4 are dedicated to defining all aspects related to the services requests, as can be verified in Figure 34 and Figure 35 in the Annex B: IDeS Framework User Interface.

In case several files were uploaded in a compressed directory, a list of paths/files will be presented in order to determine which is the file that contains the script. Otherwise, a single path/file will be presented for selection.

The path field requires the definition of a path for the request. This field is initially automatically filled with the base URL and an example path that must be replaced.

In the script call field, the name of the function and arguments (if required) must be provided. In case there are no arguments the parentheses must be included, nevertheless.

The input schema file selection is only available if the script call has arguments. This schema should serve as a template to validate the input of the function.

The output schema is always present and allows the definition of a template to validate the output of the function.

Step 4 is dedicated to the header and parameter definition for each created request, as depicted in Figure 36 in the Annex B: IDeS Framework User Interface. By clicking the plus icon in the headers list, a form is presented to define an array of headers represented by key/value pairs as shown in the example of Figure 36. In case there is no need for headers, the list must be left empty.

If the script call defined in step 3 has parameters, further information about them is required. By clicking the edit button, the request parameter form is presented. The information provided in this form is crucial to the generation of the documentation about the service.

5.3.4 Step 5: SSC Configuration

Step 5 allows to enter the information needed to register the service in the SSC and proceed to register the service. This step’s form is illustrated in Figure 20.

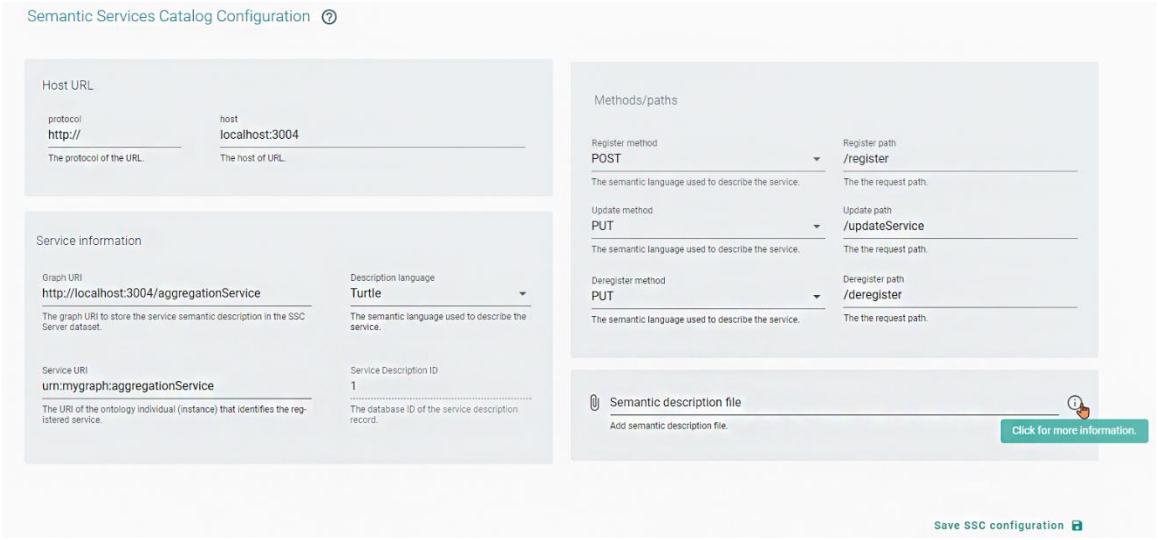


Figure 20 – Semantic Services Catalog configuration form.

The IDeS Framework allows the registration of the configured service in the Semantic Services Catalog. To register a service in SSC, the service providers must provide the service's semantic description, the language used in the semantic description, and a Uniform Resource Identifier to identify the RDF graph in the triple store.

This framework allows the creation of a configuration for the Semantic Services Catalog that is stored within the framework's data base in order to be associated with the service itself.

The service description must be uploaded in a JSON file with the description associated with the property serviceDescription.

After the form is filled, the web service's classification is suggested to the user as presented in Figure 21, where the dialog box implemented the IDeS framework's UI to present the classification suggestions is depicted.

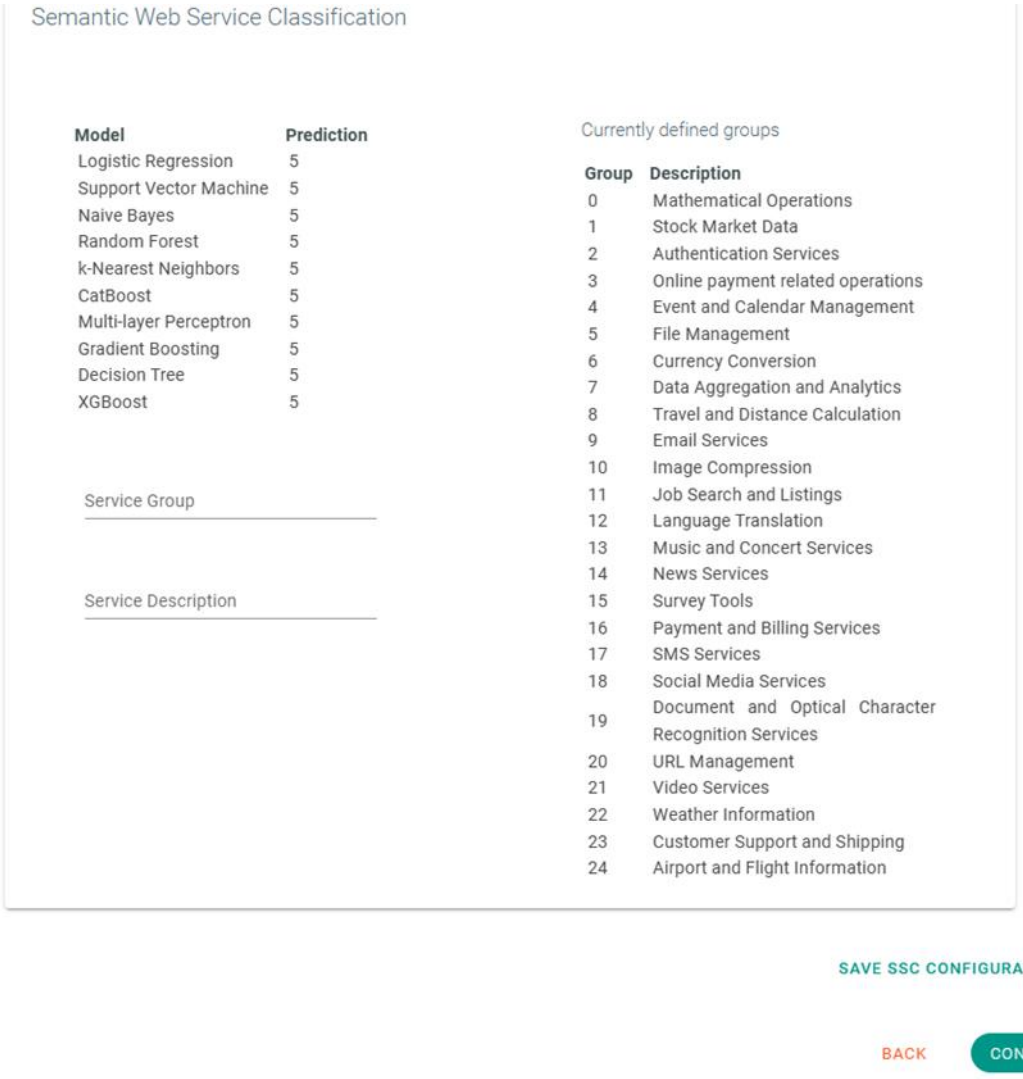


Figure 21 – IDeS framework step to configure the data necessary to register the service in SSC.

This dialog box enables the user to receive a service group suggestion list where the prediction made by each of the used classification algorithms is presented. With the objective of aiding the user in the group selection, a list of the various existing groups and respective characterization is also presented. The input fields allow to enter the web service's group, and the group's description, in case a new group is being added.

5.3.5 Step 6: Service's Web Page Preview

Finally, in step 6 the user is able to review all the information provided by accessing a preview of the web page of the web service being created. This preview allow to access the web service's home page (Figure 37, Figure 38 in the Annex B: IDeS Framework User Interface), the documentation page with all the information needed to use the service (Figure 39 in the Annex B: IDeS Framework User Interface), and a test page where the services execution can be tested (Figure 40 in the Annex B: IDeS Framework User Interface).

6 Semantic Web Service Discovery and Composition

Having discussed which techniques produce the best results in chapter 4, the present chapter describes the overall processes implemented in order to make semantic web services discovery and composition possible.

6.1 Semantic Web Service Discovery

The main execution pipeline can be summarized as the group of sequential steps illustrated in Figure 22.

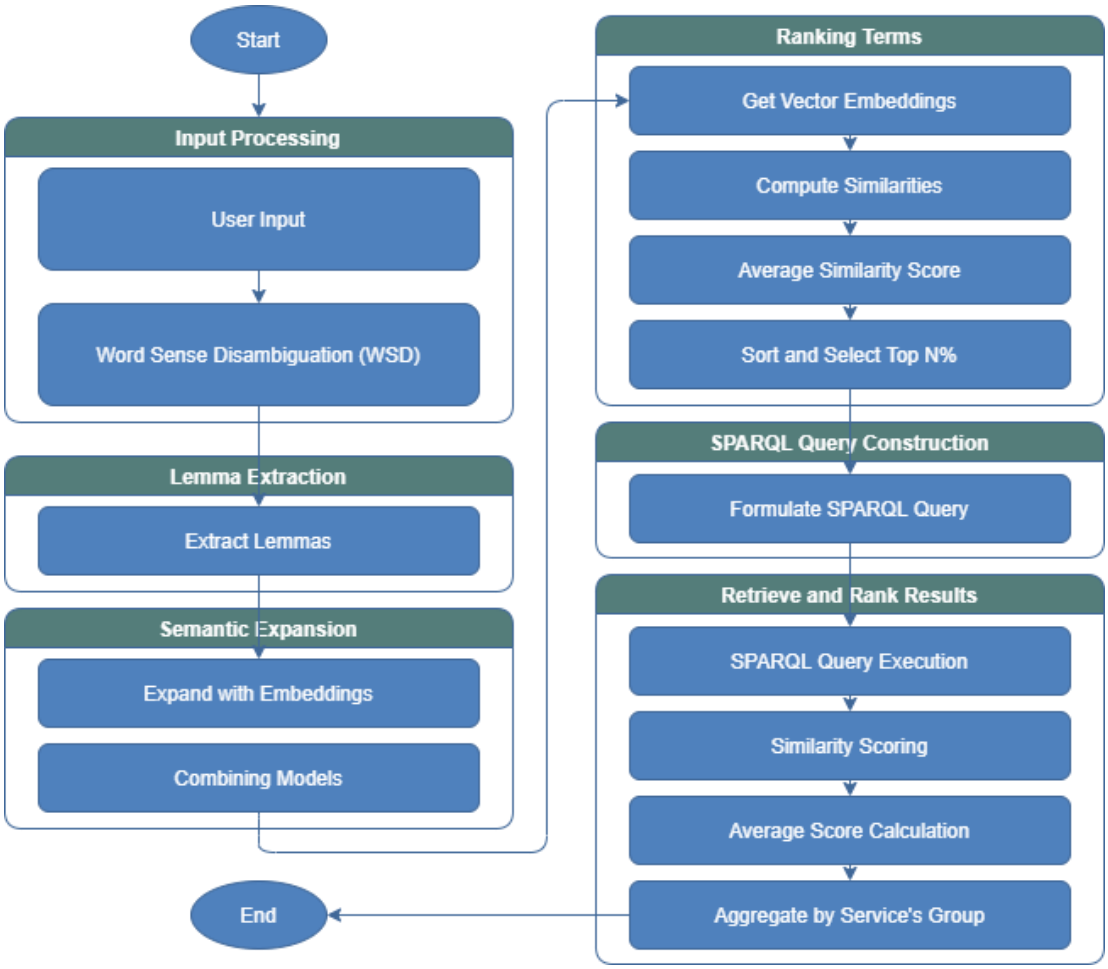


Figure 22 – Discovery flow diagram.

In this process, the user provides the search terms for a field or group of fields that characterize the services. The search terms are, then, tokenized and lemmanized. Next, the generated tokens and lemmas, are expanded and the expanded terms are ranked according

to the similarity to the search terms. Then, the SPAQRL query is performed, and its results are ranked and presented to the user.

Delving deeper into the search mechanism illustrated by Figure 22, the following paragraphs describe the discovery mechanism in further detail.

Input Processing: The process starts when the user inputs queries related to the service name, input type, output type, or description. To ensure the input is interpreted correctly, Word Sense Disambiguation (WSD) is applied using the Lesk algorithm from NLTK library[2]. This step identifies the most appropriate sense for each word in the query based on its context.

Lemma Extraction: Once the word senses are determined, lemmas, which are the canonical forms of the words, are extracted. This extraction ensures that the terms are consistently represented throughout the process.

Semantic Expansion: Next, the mechanism expands the set of terms using pre-trained models (SBERT and GloVe). These models identify the top similar words to the extracted lemmas, thereby broadening the search terms. The expansion is carried out using both SBERT and GloVe to leverage different representations of word similarities.

Ranking Terms: The expanded and original terms are then converted into vector embeddings. Similarity scores such as Cosine, Euclidean, Manhattan, and Jaccard are calculated between these embeddings. For each expanded term, an average similarity score is computed, and the terms are ranked based on this score. The top percentage of terms are selected for the next stage.

SPARQL Query Construction: Using the top-ranked expanded terms, a SPARQL query is formulated to search the Fuseki endpoint.

Retrieve and Rank Results: The SPARQL query is executed to retrieve relevant web services from the Fuseki endpoint. For each result, semantic similarity scores are calculated between the result descriptions and the original query. These scores are computed using the same similarity measures applied earlier. The average of these scores is then calculated for each result. The ranking of the results also considers the number of common lemmas between the result and the search query, as well as the service's group obtained in the classification stage. Ultimately, the results are sorted in descending order based on the number of common lemmas, the average similarity score, and the service group, producing a final ranked list of results.

In summary, the described execution pipeline outlines a systematic process for service discovery, involving several key stages. Initially, the user provides search terms, which undergo tokenization, lemmatization, and semantic expansion using SBERT and GloVe. The

[2] The Lesk algorithm, available in the Natural Language Toolkit (NLTK) library, is an algorithm for word sense disambiguation. The algorithm's goal is to determine the correct meaning of a word based on its context within a sentence (*NLTK Library*, n.d.).

expanded terms are ranked based on their similarity to the original search terms using the previously mentioned similarity metrics. Next, a SPARQL query is generated and executed through the triple store to retrieve relevant web services. The retrieved results are then ranked by calculating similarity scores between the result descriptions and the user's query, while also considering common lemmas and service group classifications. The results are finally presented in a ranked list, ensuring the most semantically relevant services are prioritized.

6.2 Semantic Web Service Composition

As verified in section Semantic Services Discovery and Composition: Related Work, there are several ways to implement semantic web services discovery and composition. Methods include annotating web services with relevant information like a description, input parameters and output parameters, generating graphs that convey relationships between services, or using NLP and ML techniques. Since, the discovery mechanism already employs techniques such as tokenization, lemmatization, and word sense disambiguation, to determine the correct meaning of the words and perform a matching process, the composition process expands this strategy, focusing on whether the query entered by the user can by segmented into actions, and then, matching the inputs and outputs of each action. The main execution pipeline can be summarized as the group of sequential steps illustrated in Figure 23.

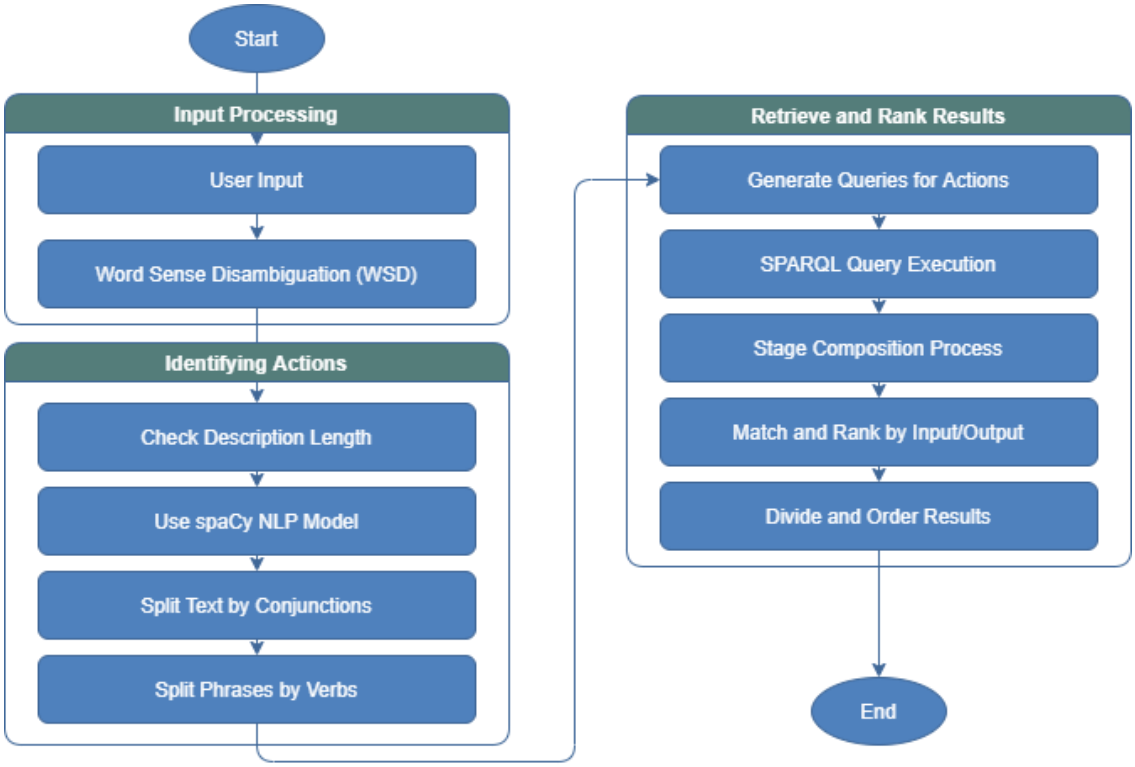


Figure 23 – Composition flow diagram.

In the composition process, each step is executed as follows, as illustrated by Figure 23. First, actions are identified. If the description field contains fewer than two words, no action identification occurs, and the description is treated as a single action. The spaCy NLP model "en_core_web_sm" is then used to process the input text, converting it into a doc object with linguistic annotations. The text is split into phrases based on coordinating conjunctions. Each token in the doc object is examined in a loop; if a token is a coordinating conjunction, the current set of tokens is joined into a string and recorded as an action. The phrases are further split by verbs, using verbs to indicate the start of a new action. Each identified action results in a separate query if more than one action is present.

For retrieving and ranking results, SPARQL queries are executed for each action to fetch relevant web services from the Fuseki endpoint. When multiple actions are identified, each set of results from Fuseki represents a different stage in the composition process and is ordered according to its similarity to the input and output. If only one action is identified—either due to a missing description field or the failure to identify multiple actions—the matching process focuses on the similarity between the user's input and output and each service's input and output. The results are then categorized into those with higher similarity to the input and those with higher similarity to the output and are ordered based on the degree of similarity.

6.3 Integration with SSC

Upon defining and implementing the aforementioned discovery and composition processes, the discovery and composition module was implemented and the communication between the SSC and the discovery/composition module was established. Figure 24, presents the search form implemented in SSC that enables the user to input the search terms.

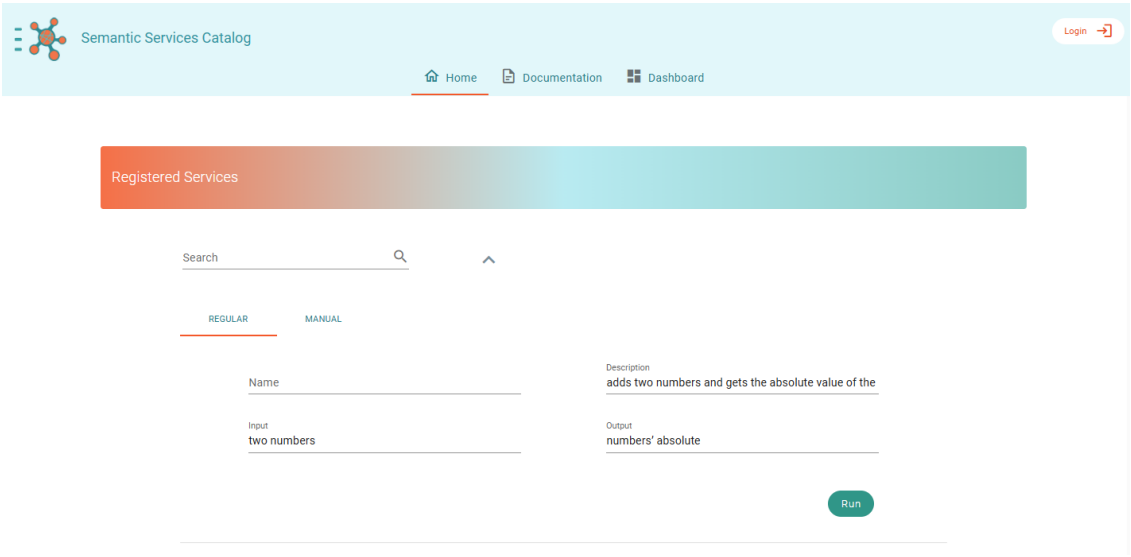


Figure 24 – SSC Search form.

After clicking the ‘Run’ button, the user is presented with a list of results. In this case, the user’s query resulted in a semantic service composition. The results are presented as shown in Figure 25.

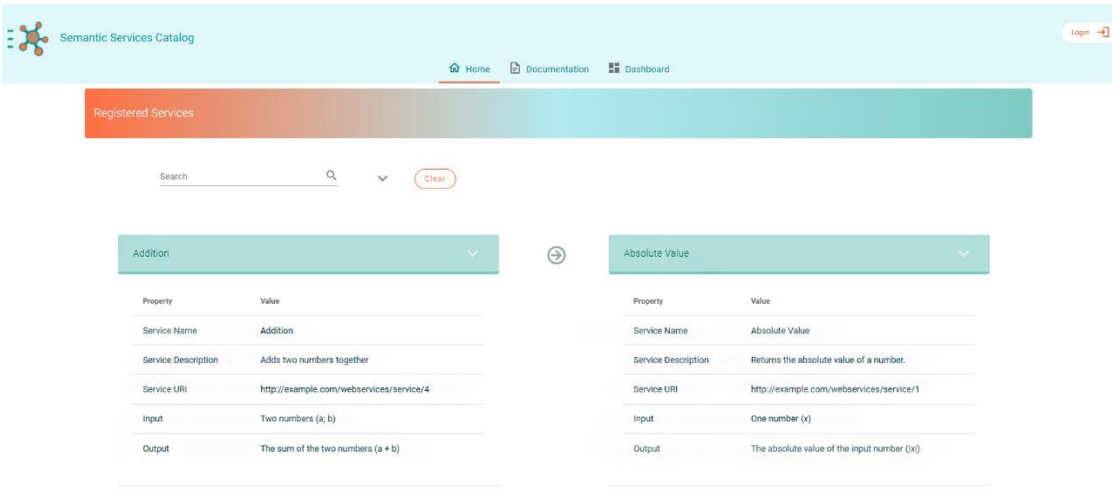


Figure 25 – SSC search results page.

The results presented in Figure 25 match the search terms and results presented in Table 16, but are shown in the user’s perspective when interacting with the SSC’s user interface.

In summary, the composition process follows a structured sequence of steps aimed at identifying actions from the service description. Each identified action leads to a separate query if multiple actions are present. In cases where multiple actions are found, the composition process handles each set of results as separate stages, ranking them by their similarity to both the input and output. If only one action is identified, the ranking focuses on comparing the input and output similarities of the user’s query with those of the web services. Results are then categorized and ranked based on their degree of input/output similarity. As such, this process enables an efficient and user-friendly interface for semantic service composition and discovery.

7 Conclusion

This work acknowledges that significant changes in power and energy systems have led to shifts in consumer behavior, making them more engaged in the process. To model these complex systems, agent-based simulation tools are commonly used, especially in the power and energy sectors. However, limitations in interoperability between independently developed systems continue to obstruct the understanding of important interrelationships. Service-Oriented Architectures (SOA) and Multi-Agent Systems (MAS) offer a solution by enabling more complex tasks and providing decision-support services within agent-based systems.

In order to understand the state of the art regarding web services search and composition, a literature review was conducted. This review allowed to understand that there are several approaches to the problem, like semantic and syntactic based approaches along with pre-classification approaches. From this stage it became clear that performing a pre-classification can improve the search performance by grouping services in categories. Additionally, given the need to frequently add new services to the database by users with varying levels of expertise, it became clear that integrating NPL and ML techniques would allow users to find semantic web services tailored to their needs through keyword searches.

As verified in the literature review, the use of ML algorithms can significantly impact the initial matchmaking methods. Once the keywords are introduced in the search fields, different methods can be used to accurately interpret the terms. Furthermore, as previously mentioned, using classification techniques to categorize web services by type can help refine the search results. Consequently, the exploratory process for web service discovery and composition was centered on developing solutions based on these principles. The results demonstrated that the classification process is essential in determining the semantic web service's corresponding group, which is then used to refine the search results. These results are sorted in descending order based on the number of common lemmas, the average similarity score, and the service group, producing a final ranked list of results. In this context, the ability to attribute a group during the process of configuration is introduced in the configuration process performed in IDeS framework. This allows for a more relevant retrieval of search results when performing web service discovery/composition using the SSC search form.

In turn, the discovery exploratory process allowed for the development of a search mechanism that represents a comprehensive approach to the usage of various NLP and ML techniques. It combines WSD, word embeddings, group pre-classification and semantic similarity computations to rank web services from a SPARQL endpoint based on user queries. The discovery process begins with tokenization and WSD to split user queries into tokens and accurately interpret word meanings. However, this initial strategy was limited, as it only returned exact matches and failed to capture synonyms or related terms. To address this, word embeddings were introduced to expand search terms, which broadened the scope of results but also introduced irrelevant ones. To refine this, similarity measures were applied to rank the

embedding terms and prioritize the most relevant search results. Additionally, replacing Word2Vec with SBERT improved both the accuracy of embeddings and overall system performance.

Building upon these findings the composition exploratory process and results illustrate the flexibility and effectiveness of the proposed search and composition mechanisms in handling various types of user input. For web service composition, the system processes the description field of user queries, identifying distinct actions separated by verbs and conjunctions. Each action generates its own query with expanded terms, and the resulting list of services represents possible matches for different stages of the composition. If the user provides input and output fields, the system ranks the services based on the similarity between the user's input/output and those of the services. In conclusion, by combining the previously mentioned NLP techniques with the identification of actions within the user's query, the system can effectively expand and refine user queries, enhancing search results and providing composition suggestion if applicable.

In this context, the contributions provided by this work provide valid answers to the identified gaps. IDeS framework allows users to configure and publish a semantic web service in a streamlined and guided approach, also performing a pre-classification of the service. In turn, improvements made in the search functionality of the SSC allowed for more precise and relevant search results, as well as the possibility to suggest service compositions when a single service to perform the required operation is not found. Other important contributions are the submissions of two articles in the International Conference on Autonomous Agents and Multiagent Systems (AAMAS):

- Rui Carvalho, Gabriel Santos, Zita Vale. *Pre-classification impact on intelligent search of semantic web services.*
- Rui Carvalho, Gabriel Santos, Zita Vale. *Semantic web services discovery and composition in the context of Multi-Agent Systems*

The development of this work also had applications in the following funded projects:

- MAS Society - Multi-Agent Systems SemantiC Interoperability for simulation and dEcision supporT in complex energY systems (FCT)
- PRECISE - Power and Energy Cyber-Physical Solutions with Explainable Semantic Learning (FCT)
- TradeRES - Tools for the Design and modelling of new markets and negotiation mechanisms for a ~100% Renewable European Power Systems (H2020)

7.1 Future Research

In future work, expanding the current dataset can be beneficial, particularly by integrating the web services that will be configured and published through the IDeS Framework. A larger dataset would allow for more comprehensive testing and validation of the classification,

discovery and composition mechanisms. Furthermore, incorporating a wider variety of services could enhance the robustness of the system, allowing it to handle a broader range of scenarios and edge cases, improving its overall effectiveness.

Regarding the classification, process, future research can focus on improving the classification results by exploring and testing new machine learning and natural language processing techniques. Moreover, conducting comparative studies of different approaches would help in identifying the most effective techniques.

To further improve the discovery and composition mechanisms, it would be beneficial to research ways to optimize the performance of the search and composition processes, particularly in terms of speed and scalability, would be essential as the dataset grows. These improvements could significantly enhance user experience, especially when handling large-scale or complex queries.

8 References

Adala, A., Tabbane, N., & Tabbane, S. (2011). A framework for automatic web service discovery based on semantics and NLP techniques. *Advances in Multimedia*, *2011*. https://doi.org/10.1155/2011/238683

Aiello, M., & Georgievski, I. (2023). Service composition in the ChatGPT era. *Service Oriented Computing and Applications*, *17*(4), 233–238. https://doi.org/10.1007/s11761-023-00367-7

Bai, B., Fan, Y., Tan, W., & Zhang, J. (2020). DLTSR: A Deep Learning Framework for Recommendations of Long-Tail Web Services. *IEEE Transactions on Services Computing*, *13*(1), 73–85. https://doi.org/10.1109/TSC.2017.2681666

Bellur, U., & Kulkarni, R. (2007). Improved matchmaking algorithm for semantic web services based on bipartite graph matching. *Proceedings - 2007 IEEE International Conference on Web Services, ICWS 2007*, *Icws*, 86–93. https://doi.org/10.1109/ICWS.2007.105

Benaboud, R., Maamri, R., & Sahnoun, Z. (2013). Agents and OWL-S Based Semantic Web Service Discovery With User Preference Support. *International Journal of Web & Semantic Technology*, *4*(2), 57–75. https://doi.org/10.5121/ijwest.2013.4206

Bishop, P. (2007). Pattern Recognition and Machine Learning. *Journal of Electronic Imaging*, *16*(4), 049901. https://doi.org/10.1117/1.2819119

Breiman, L. (2001). Random forests. Random Forests, 1–122. *Machine Learning*, *45*(45), 5–32. https://link.springer.com/article/10.1023/A:1010933404324

Bussler, C., Zaremba, M., Finin, T., Huhns, M. N., Paolucci, M., Sheth, A. P., & Williams, S. (2005). *Mark Burstein BBN Technologies A Semantic Web Services Architecture*. *October*, 72–81. www.computer.org/internet/

Cabral, L., Domingue, J., Motta, E., Payne, T., & Hakimpour, F. (2004). Approaches to semantic web services: An overview and comparisons. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3053*, 225–239. https://doi.org/10.1007/978-3-540-25956-5_16

Canito, A., Santos, G., Corchado, J. M., Marreiros, G., & Vale, Z. (2019). Semantic Web Services for Multi-Agent Systems Interoperability. *EPIA 2019: Progress in Artificial Intelligence*, *11805*, 606–616.

Cao, B., Liu, X., Rahman, M. D. M., Li, B., Liu, J., & Tang, M. (2020). Integrated Content and Network-Based Service Clustering and Web APIs Recommendation for Mashup Development. *IEEE Transactions on Services Computing*, *13*(1), 99–113. https://doi.org/10.1109/TSC.2017.2686390

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. *ACM International Conference Proceeding Series*, *148*, 161–168. https://doi.org/10.1145/1143844.1143865

Çelik, D., & Elçi, A. (2005). A semantic search agent approach: Finding appropriate semantic web services based on user request term(S). *ITI 3rd International Conference on Information and Communications Technology, ICICT 2005 - Enabling Technologies for the New Knowledge Society*, *2005*, 675–688. https://doi.org/10.1109/ITICT.2005.1609659

Çelik, D., & Elçi, A. (2006). Discovery and scoring of semantic web services based on client requirement(S) through a semantic search agent. *Proceedings - International Computer Software and Applications Conference*, *2*, 273–278. https://doi.org/10.1109/COMPSAC.2006.127

Chebbi, I., & Ayed, L. Ben. (2024). *Personalized Web Services Composition Using Arti cial Intelligence*. 0–17.

Chen, F., Li, M., Wu, H., & Xie, L. (2017). Web service discovery among large service pools utilising semantic similarity and clustering. *Enterprise Information Systems*, *11*(3), 452–469. https://doi.org/10.1080/17517575.2015.1081987

Chen, F., Lu, C., Wu, H., & Li, M. (2017). A semantic similarity measure integrating multiple conceptual relationships for web service discovery. *Expert Systems with Applications*, *67*, 19–31. https://doi.org/10.1016/j.eswa.2016.09.028

Chen, L., Yang, G., Wang, D., & Zhang, Y. (2010). WordNet-powered web services discovery using Kernel-based similarity matching mechanism. *Proceedings - 5th IEEE International Symposium on Service-Oriented System Engineering, SOSE 2010*, 64–68. https://doi.org/10.1109/SOSE.2010.42

Chen, T., & Guestrin, C. (2016). XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-Augu*, 785–794. https://doi.org/10.1145/2939672.2939785

Cheng, B., Zhao, S., Li, C., & Chen, J. (2017). A Web Services Discovery Approach Based on Mining Underlying Interface Semantics. *IEEE Transactions on Knowledge and Data Engineering*, *29*(5), 950–962.

Chollet, F. (2017). Machine learning. In *Machine Learning* (Vol. 45, Issue 13). https://books.google.ca/books?id=EoYBngEACAAJ&dq=mitchell+machine+learning+1997&hl=en&sa=X&ved=0ahUKEwiomdqfj8TkAhWGslkKHRCbAtoQ6AEIKjAA

Ciaramella, A., Cimino, M. G. C. A., Lazzerini, B., & Marcelloni, F. (2009). Situation-aware mobile service recommendation with fuzzy logic and semantic web. *ISDA 2009 - 9th International Conference on Intelligent Systems Design and Applications*, 1037–1042. https://doi.org/10.1109/ISDA.2009.242

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Costa, T., & Leal, J. P. (2013). Publishing linked data with DaPress. *OpenAccess Series in Informatics*, *29*(August 2013), 67–81. https://doi.org/10.4230/OASIcs.SLATE.2013.67

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on*

Information Theory, *13*(1), 21–27. https://doi.org/10.1109/TIT.1967.1053964

Czyszczoń, A., & Zgrzywa, A. (2014). Latent semantic indexing for web service retrieval. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8733*, 694–702. https://doi.org/10.1007/978-3-319-11289-3_70

Devlin, B., & Liu, R. (2014). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*.

Dorri, A., Kanhere, S. S., & Jurdak, R. (2018). Multi-Agent Systems: A Survey. *IEEE Access*, *6*, 28573–28593. https://doi.org/10.1109/ACCESS.2018.2831228

Dudek, G., Jenkin, M. R. M., Milios, E., & Wilkes, D. (1996). *A Taxonomy for Multi-Agent Robotics *. 397*, 375–397.

Eeles, P. (2005). Capturing Architectural Requirements What Is an Architectural Requirement? *IBM Rational Developer Works*. http://www.therationaledge.com/content/nov_01/t_architecturalRequirements_pe.html

Ehrig, M., Koschmider, A., & Oberweis, A. (2007). Measuring similarity between semantic business process models. *Conferences in Research and Practice in Information Technology Series*, *67*.

Eid, M., Alamri, A., & El Saddik, A. (2008). A reference model for dynamic web service composition systems. *International Journal of Web and Grid Services*, *4*(2), 149–168. https://doi.org/10.1504/IJWGS.2008.018885

Ekelhart, A., Fenz, S., Tjoa, A. M., & Weippl, E. R. (2007). Security issues for the use of Semantic Web in E-Commerce. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4439 LNCS*(September 2015), 1–13. https://doi.org/10.1007/978-3-540-72035-5_1

Elmagarmid, A. K. (2016). *Composing web services on the Semantic web . VLDB J Composing Web services on the Semantic Web*. *May*, 333–351.

Elshater, Y., Elgazzar, K., & Martin, P. (2015). GoDiscovery: Web Service Discovery Made Efficient. *Proceedings - 2015 IEEE International Conference on Web Services, ICWS 2015*, 711–716. https://doi.org/10.1109/ICWS.2015.99

Fang, M., Wang, D., Mi, Z., & Obaidat, M. S. (2018). Web service discovery utilizing logical reasoning and semantic similarity. *International Journal of Communication Systems*, *31*(10), 1–13. https://doi.org/10.1002/dac.3561

Freund, Y., & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *904*, 23–37. https://doi.org/10.1007/3-540-59119-2_166

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals*

of Statistics, *29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Gamha, Y. (2023). A framework for REST services discovery and composition. *Service Oriented Computing and Applications*, *17*(4), 259–275. https://doi.org/10.1007/s11761-023-00376-6

Gao, W., & Wu, J. (2017). A Novel Framework for Service Set Recommendation in Mashup Creation. *Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017*, 65–72. https://doi.org/10.1109/ICWS.2017.17

Gao, Z., Fan, Y., Wu, C., Tan, W., Zhang, J., Ni, Y., Bai, B., & Chen, S. (2019). SeCo-LDA: Mining Service Co-Occurrence Topics for Composition Recommendation. *IEEE Transactions on Services Computing*, *12*(3), 446–459. https://doi.org/10.1109/TSC.2018.2821149

Garcia, C., Cardenas, P. F., Saltaren, R., Puglisi, L., & Aracil, R. (2013). Expert Systems with Applic ations A cooperative multi-agent robotics system : Design and modelling q. *Expert Systems With Applications*, *40*(12), 4737–4748. http://dx.doi.org/10.1016/j.eswa.2013.01.048

Garriga, M., Flores, A., Cechich, A., & Zunino, A. (2015). Web services composition mechanisms: A review. *IETE Technical Review (Institution of Electronics and Telecommunication Engineers, India)*, *32*(5), 376–383. https://doi.org/10.1080/02564602.2015.1019942

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, *57*, 345–420. https://doi.org/10.1613/jair.4992

Goonatilleke, S. T., & Hettige, B. (2022). Past, Present and Future Trends in Multi-Agent System Technology. *Journal Europeen Des Systemes Automatises*, *55*(6), 723–739. https://doi.org/10.18280/jesa.550604

Han, H., He, J., & Zuo, C. (2010). A hybrid P2P overlay network for high efficient search. *Proceedings - 2010 2nd IEEE International Conference on Information and Financial Engineering, ICIFE 2010*, 241–245. https://doi.org/10.1109/ICIFE.2010.5609293

Hassan, S. U., Ahamed, J., & Ahmad, K. (2022). Analytics of machine learning-based algorithms for text classification. *Sustainable Operations and Computers*, *3*(April), 238–248. https://doi.org/10.1016/j.susoc.2022.03.001

Jensen, K. (1991). Coloured Petri Nets: A High Level Language for System Design and Analysis. In *High-level Petri Nets* (pp. 44–119). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-84524-6_2

Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing*. Pearson.

Jurafsky, Daniel, & Martin, J. H. (2024). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. https://web.stanford.edu/~jurafsky/slp3/

Khanam, S. A., & Youn, H. Y. (2016). A web service discovery scheme based on structural and semantic similarity. *Journal of Information Science and Engineering*, *32*(1), 153–176.

https://doi.org/10.6688/JISE.2016.32.1.9

Klusch, M., Kapahnke, P., Schulte, S., Lecue, F., & Bernstein, A. (2016). Semantic Web Service Search: A Brief Survey. *KI - Kunstliche Intelligenz*, *30*(2), 139–147. https://doi.org/10.1007/s13218-015-0415-7

Kruchten, P. (1995). Architectural Blueprints — 4+1 View Model of Software Architecture. *IEEE Software*, *12*(6), 42–50.

Kumar, V., & Minz, S. (2014). *Feature Selection : A literature Review*. *4*(3). https://doi.org/10.6029/smartcr.2014.03.007

Kurniawan, K., Ekaputra, F. J., & Aryan, P. R. (2018). Semantic Service Description and Compositions: A Systematic Literature Review. *2018 2nd International Conference on Informatics and Computational Sciences, ICICoS 2018*, *June 2019*, 37–42. https://doi.org/10.1109/ICICOS.2018.8621686

Lemos, A. L., Daniel, F., & Benatallah, B. (2015). Web service composition: A survey of techniques and tools. *ACM Computing Surveys*, *48*(3). https://doi.org/10.1145/2831270

Li, S., Luo, H., Zhao, G., Tang, M., & Liu, X. (2022). bi-directional Bayesian probabilistic model based hybrid grained semantic matchmaking for Web service discovery. *World Wide Web*, *25*(2), 445–470. https://doi.org/10.1007/s11280-022-01004-7

Liu, X., Agarwal, S., Ding, C., & Yu, Q. (2016). An LDA-SVM active learning framework for web service classification. *Proceedings - 2016 IEEE International Conference on Web Services, ICWS 2016*, 49–56. https://doi.org/10.1109/ICWS.2016.16

Ma, J., Zhang, Y., & He, J. (2008). Efficiently finding web services using a clustering semantic approach. *ACM International Conference Proceeding Series*, *292*. https://doi.org/10.1145/1361482.1361487

Malik, F. H., & Lehtonen, M. (2016). A review: Agents in smart grids. *Electric Power Systems Research*, *131*, 71–79. https://doi.org/10.1016/j.epsr.2015.10.004

Manning, C. D., Raghavan, P., & Hinrich, S. (2009). *Information Retrieval - Online edition (c) 2009 Cambridge UP*. *c*, 569. http://www-nlp.stanford.edu/IR-book/

McCallum, A., & Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. *AAAI/ICML-98 Workshop on Learning for Text Categorization*, 41–48. https://doi.org/10.1.1.46.1529

Meditskos, G., & Bassiliades, N. (2010). Structural and role-oriented web service discovery with taxonomiesin OWL-S. *IEEE Transactions on Knowledge and Data Engineering*, *22*(2), 278–290. https://doi.org/10.1109/TKDE.2009.89

Merin, J. B., Aisha Banu, W., & Fahima Sanobar Shalin, K. (2023). Semantic Annotation Based Effective and Quality Oriented Web Service Discovery. *Journal of Internet Services and Information Security*, *13*(2), 96–116. https://doi.org/10.58346/JISIS.2023.I2.006

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word

representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, 1–12.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of Machine Learning* (Issue 112). Massachusetts Institute of Technology.

Mosa, A., & Abdelaziz, A. (2021). A Survey on Web Service Discovery Approaches. *Fusion: Practice and Applications*, *5*(2), 62–69. https://doi.org/10.54216/FPA.050202

Murakami, Y., & Kimura, K. (2024). Human-Centered Services Computing for Smart Cities. In *Human-Centered Services Computing for Smart Cities*. https://doi.org/10.1007/978-981-97-0779-9

Nabli, H., Ben Djemaa, R., & Ben Amor, I. A. (2018). Efficient cloud service discovery approach based on LDA topic modeling. In *Journal of Systems and Software* (Vol. 146, pp. 233–248). https://doi.org/10.1016/j.jss.2018.09.069

Naim, H., Aznag, M., Quafafou, M., Durand, N., Naim, H., Aznag, M., Quafafou, M., Durand, N., Approach, P., Naim, H., Aznag, M., Quafafou, M., & Durand, N. (2018). *Probabilistic Approach for Diversifying Web Services Discovery and Composition To cite this version : HAL Id : hal-01465112 Probabilistic Approach for Diversifying Web Services Discovery and Composition*.

Navigli, R. (2015). *Word Sense Disambiguation : A Survey Word Sense Disambiguation : A Survey*. *February 2009*. https://doi.org/10.1145/1459352.1459355

Ngan, L. D., & Kanagasabai, R. (2013). Semantic Web service discovery: State-of-the-art and research challenges. *Personal and Ubiquitous Computing*, *17*(8), 1741–1752. https://doi.org/10.1007/s00779-012-0609-z

Ni, Y., & Fan, Y. (2010). Model transformation and formal verification for Semantic Web Services composition. *Advances in Engineering Software*, *41*(6), 879–885. https://doi.org/10.1016/j.advengsoft.2010.01.005

Niu, L., Ren, F., & Zhang, M. (2018). Feasible negotiation procedures for multiple interdependent negotiations. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 641–649. https://doi.org/https://doi.org/10.5555/3237383.3237479

NLTK library. (n.d.). https://www.nltk.org/

Ochoa, W., Larrinaga, F., & Pérez, A. (2023). Context-aware workflow management for smart manufacturing: A literature review of semantic web-based approaches. *Future Generation Computer Systems*, *145*, 38–55. https://doi.org/10.1016/j.future.2023.03.017

OWL-S. (2022). https://www.w3.org/Submission/OWL-S

Paik, I., Chen, W., & Huhns, M. N. (2014). A scalable architecture for automatic service composition. *IEEE Transactions on Services Computing*, *7*(1), 82–95. https://doi.org/10.1109/TSC.2012.33

Paliwal, A. V., Shafiq, B., Vaidya, J., Xiong, H., & Adam, N. (2012). Semantics-based automated service discovery. *IEEE Transactions on Services Computing*, *5*(2), 260–275. https://doi.org/10.1109/TSC.2011.19

Patel, A., & Jain, S. (2021). Present and future of semantic web technologies: a research statement. *International Journal of Computers and Applications*, *43*(5), 413–422. https://doi.org/10.1080/1206212X.2019.1570666

Patidar, S., Kumar, D., & Rukwal, D. (2022). Comparative Analysis of Machine Learning Algorithms for Heart Disease Prediction. *Advances in Transdisciplinary Engineering*, *27*, 64–69. https://doi.org/10.3233/ATDE220723

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1532–1543. https://doi.org/10.3115/v1/d14-1162

Pinto, T., Vale, Z., Praça, I., Pires, E. J. S., & Lopes, F. (2015). Decision support for energy contracts negotiation with game theory and adaptive learning. *Energies*, *8*(9), 9817–9842. https://doi.org/10.3390/en8099817

Platzer, C., & Dustdar, S. (2005). A vector space search engine for Web services. *Proceedings - Third European Conference on Web Services, ECOWS2005*, *2005*, 62–70. https://doi.org/10.1109/ECOWS.2005.5

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). Catboost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, *2018-Decem*(Section 4), 6638–6648.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1023/A:1022643204877

Rizk, Y., Awad, M., & Tunstel, E. W. (2019). Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys*, *52*(2). https://doi.org/10.1145/3303848

Rodriguez-Mier, P., Pedrinaci, C., Lama, M., & Mucientes, M. (2016). An integrated semantic web service discovery and composition framework. *IEEE Transactions on Services Computing*, *9*(4), 537–550. https://doi.org/10.1109/TSC.2015.2402679

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 513–523. https://doi.org/10.1007/springerreference_63962

Sangers, J., Frasincar, F., Hogenboom, F., & Chepegin, V. (2013). Semantic Web service discovery using natural language processing techniques. *Expert Systems with Applications*, *40*(11), 4660–4671. https://doi.org/10.1016/j.eswa.2013.02.011

Santos, G., Canito, A., Carvalho, R., Pinto, T., Vale, Z., Marreiros, G., & Corchado, J. M. (2021). Semantic Services Catalog for Multiagent Systems Society. In F. Dignum, J. M. Corchado, & F. De La Prieta (Eds.), *Advances in Practical Applications of Agents, Multi-Agent Systems, and Social Good. The PAAMS Collection* (pp. 229–240). Springer International

Publishing.

Scott, A. J., Hosmer, D. W., & Lemeshow, S. (1991). Applied Logistic Regression. In *Biometrics* (Vol. 47, Issue 4). https://doi.org/10.2307/2532419

Shi, M., Liu, J., Zhou, D., Tang, M., & Cao, B. (2017). WE-LDA: A Word Embeddings Augmented LDA Model for Web Services Clustering. *Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017*, 9–16. https://doi.org/10.1109/ICWS.2017.9

Sotolongo, R., Kobashikawa, C., Dong, F., & Hirota, K. (2008). Algorithm for Web Service Discovery Based on Information Retrieval Using WordNet and Linear Discriminant Functions. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, *12*(2), 182–189. https://doi.org/10.20965/jaciii.2008.p0182

Souri, A., Rahmani, A. M., & Jafari Navimipour, N. (2018). Formal verification approaches in the web service composition: A comprehensive analysis of the current challenges for future research. *International Journal of Communication Systems*, *31*(17), 1–27. https://doi.org/10.1002/dac.3808

Stork, D. G., Duda, R., & Hart, P. (2001). *Pattern Classification*. John Wiley & Sons.

Surianarayanan, C., & Ganapathy, G. (2016). An approach to computation of similarity, inter-cluster distance and selection of threshold for service discovery using clusters. *IEEE Transactions on Services Computing*, *9*(4), 524–536. https://doi.org/10.1109/TSC.2015.2399301

Swapna Choudhary. (2024). Design of an Iterative Method for Optimizing Solar Power Systems using Quad LSTM with IoT Integration Operations. *Journal of Electrical Systems*, *20*(6s), 2831–2846. https://doi.org/10.52783/jes.3290

Talantikite, H. N., Aissani, D., & Boudjlida, N. (2009). Semantic annotations for web services discovery and composition. *Computer Standards and Interfaces*, *31*(6), 1108–1117. https://doi.org/10.1016/j.csi.2008.09.041

Tokmak, A. V., Akbulut, A., & Catal, C. (2024). Web service discovery: Rationale, challenges, and solution directions. *Computer Standards and Interfaces*, *88*(March 2023). https://doi.org/10.1016/j.csi.2023.103794

Tuballa, M. L., & Abundo, M. L. (2016). A review of the development of Smart Grid technologies. *Renewable and Sustainable Energy Reviews*, *59*, 710–725. https://doi.org/10.1016/j.rser.2016.01.011

Wagner, F., Ishikawa, F., & Honiden, S. (2011). QoS-aware automatic service composition by applying functional clustering. *Proceedings - 2011 IEEE 9th International Conference on Web Services, ICWS 2011*, 89–96. https://doi.org/10.1109/ICWS.2011.32

Wang, J., & Tepfenhart, W. (2019). Colored Petri Nets. *Formal Methods in Computer Science*, *May 2021*, 271–289. https://doi.org/10.1201/9780429184185-10

Yang, Y., Ke, W., Wang, W., & Zhao, Y. (2019). Deep learning for web services classification. *Proceedings - 2019 IEEE International Conference on Web Services, ICWS 2019 - Part of*

the 2019 IEEE World Congress on Services, 440–442. https://doi.org/10.1109/ICWS.2019.00079

Yangui, S., Goscinski, A., Drira, K., Tari, Z., & Benslimane, D. (2021). Future generation of service-oriented computing systems. *Future Generation Computer Systems*, *118*, 252–256. https://doi.org/10.1016/j.future.2021.01.019

Zhang, N., Wang, J., Ma, Y., He, K., Li, Z., & Liu, X. (Frank F. (2018). Web service discovery based on goal-oriented query expansion. In *Journal of Systems and Software* (Vol. 142, pp. 73–91). https://doi.org/10.1016/j.jss.2018.04.046

Zou, G., Qin, Z., He, Q., Wang, P., Zhang, B., & Gan, Y. (2022). DeepWSC: Clustering Web Services via Integrating Service Composability into Deep Semantic Features. *IEEE Transactions on Services Computing*, *15*(4), 1940–1953. https://doi.org/10.1109/TSC.2020.3026188

Annex A: Process Diagrams

Figure 26 presents the process diagram for configuring a service and generating the corresponding documentation using the IDeS Framework.

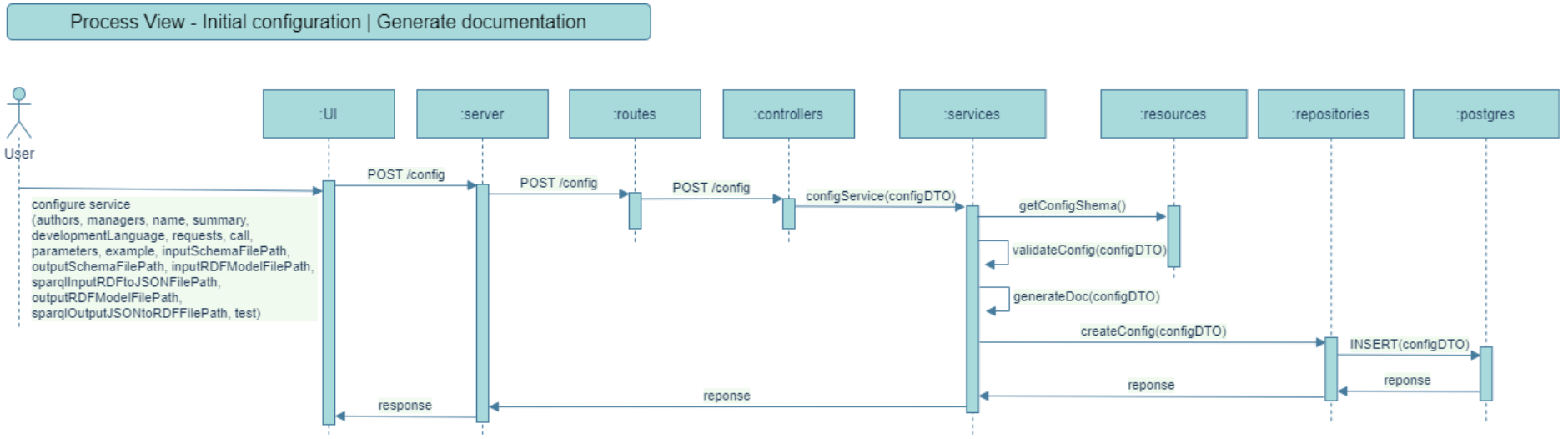


Figure 26 – Process diagram for the service's configuration.

Figure 27 presents the process diagram for registering a service in the SSC.

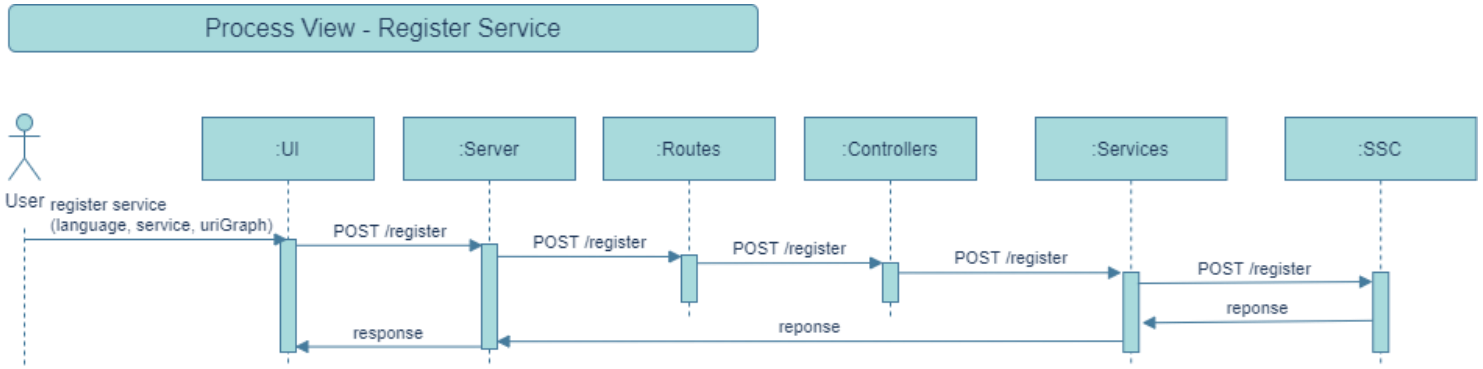


Figure 27 – Process Diagram for service registration in SSC.

Figure 28 presents the process diagram for testing the web service's execution.

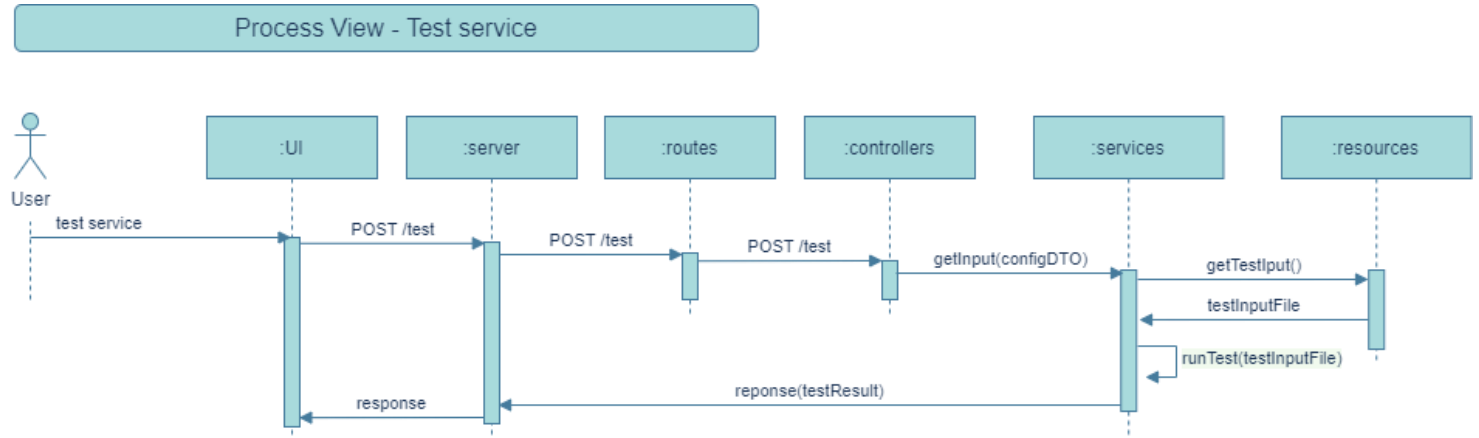


Figure 28 – Process diagram for testing the service's execution.

Figure 29 presents the process diagram for creating logs that register the user's actions through the configuration process.

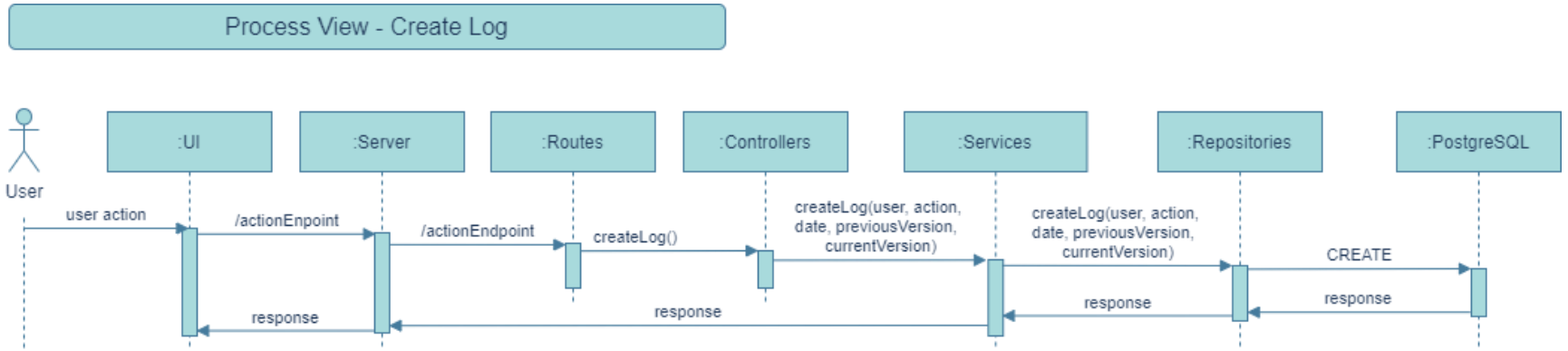


Figure 29 – Process diagram for log creation.

Annex B: IDeS Framework User Interface

As mentioned in the Analysis and Design section, IDeS framework has a guided configuration process divided in 6 steps illustrated in the following figures.

Figure 30 depicts the first step of the configuration process. In Figure 31 each field is explained in detail.

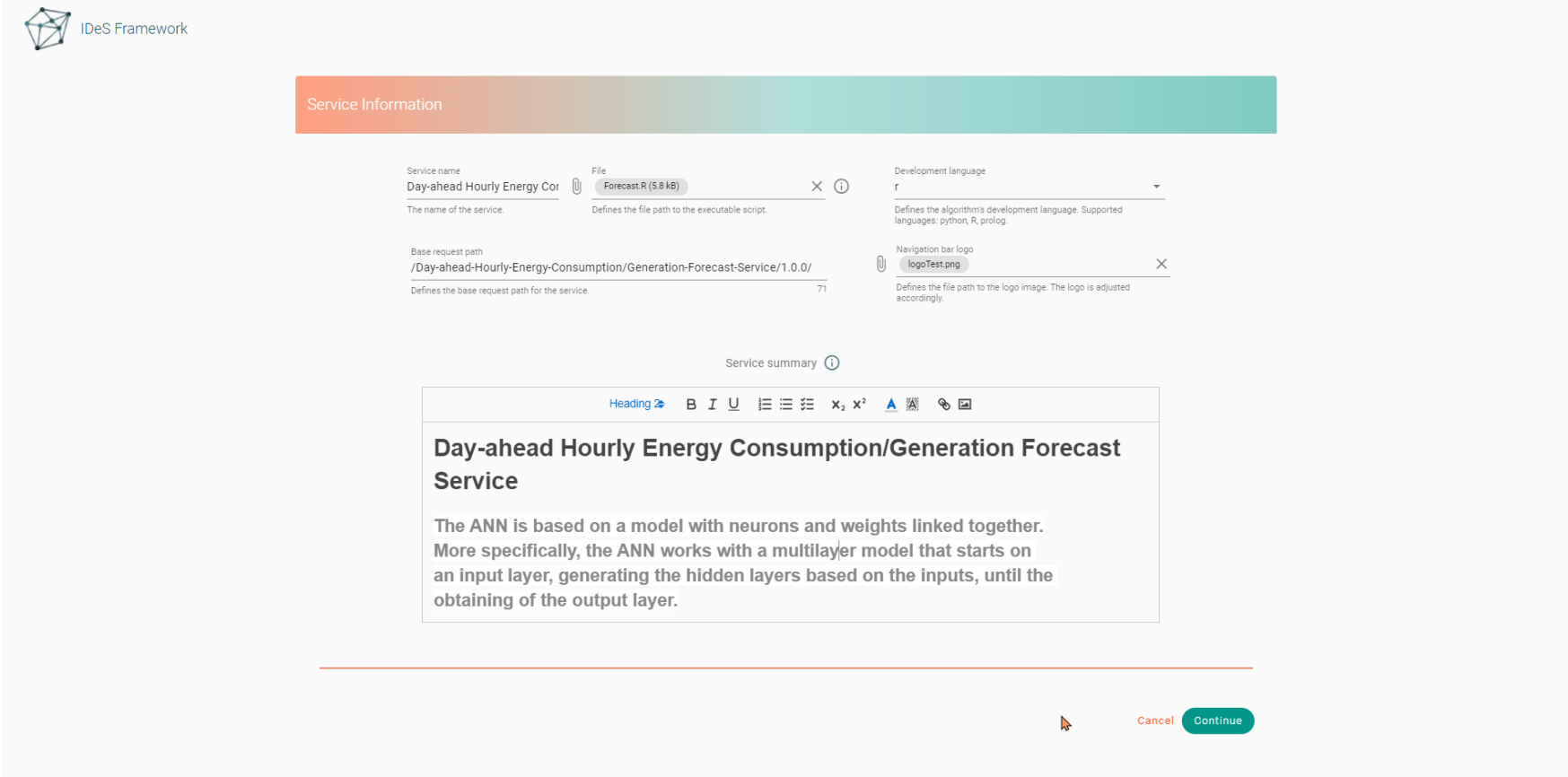


Figure 30 – Step 1 form for the configuration process.

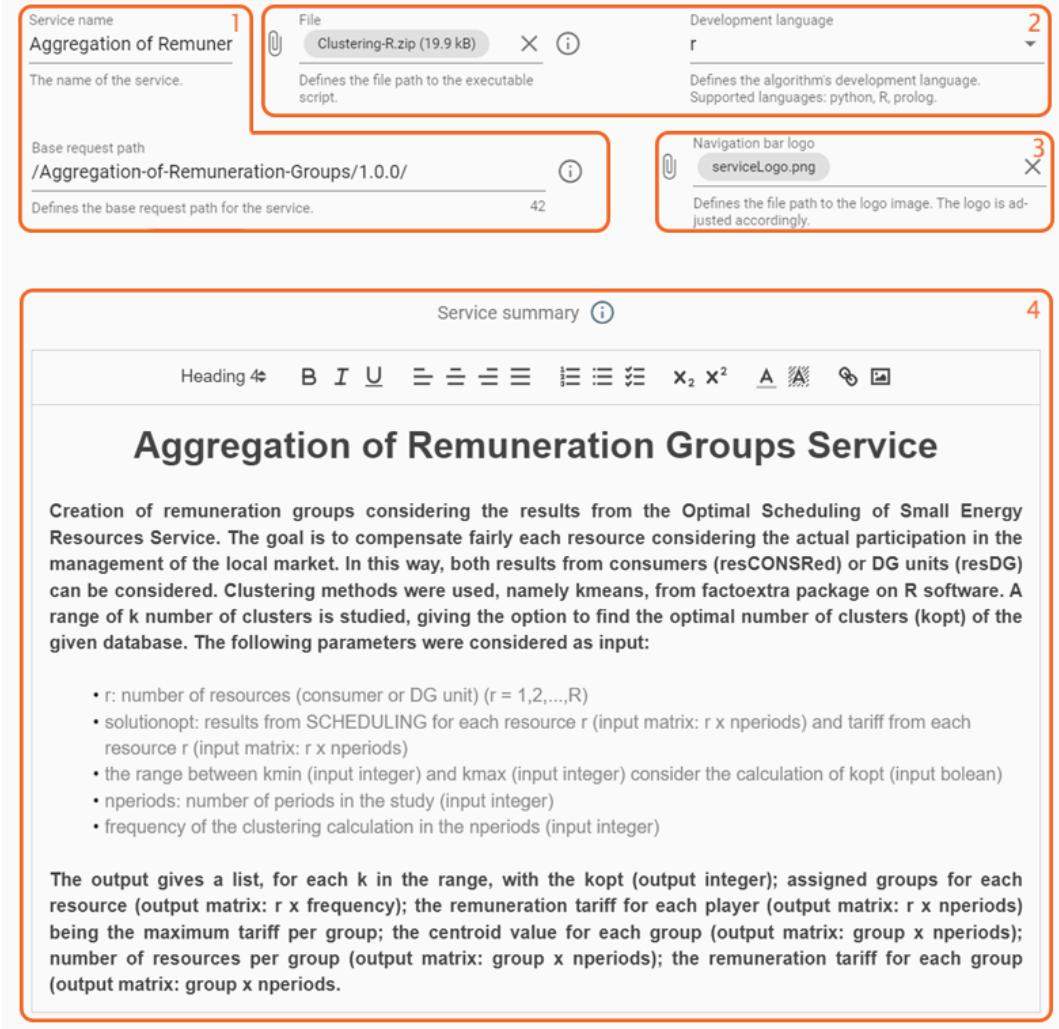


Figure 31 – Step 1 field tool tips.

Figure 32 depicts the second step of the configuration process where the authors are defined. In Figure 33 each field is explained in detail.

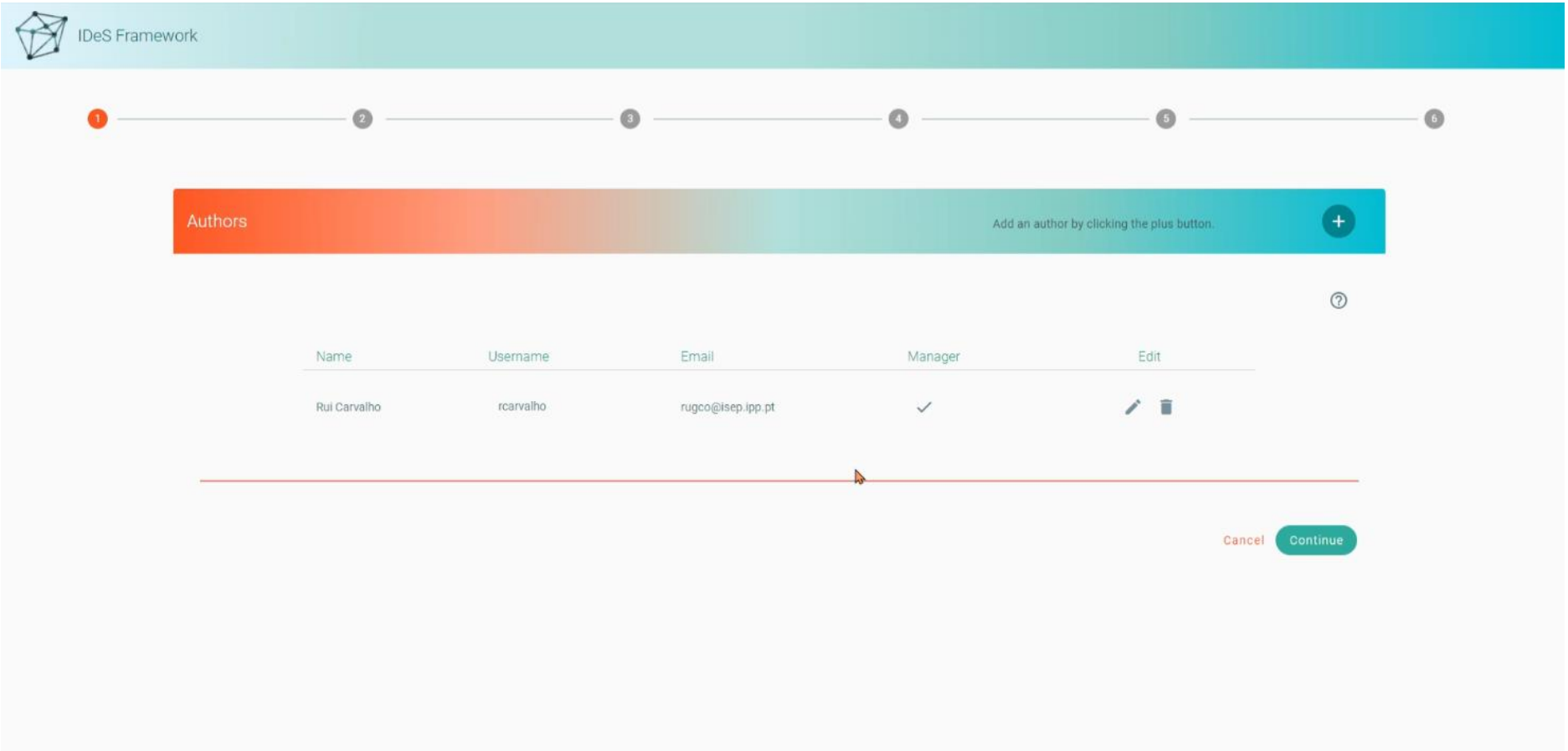
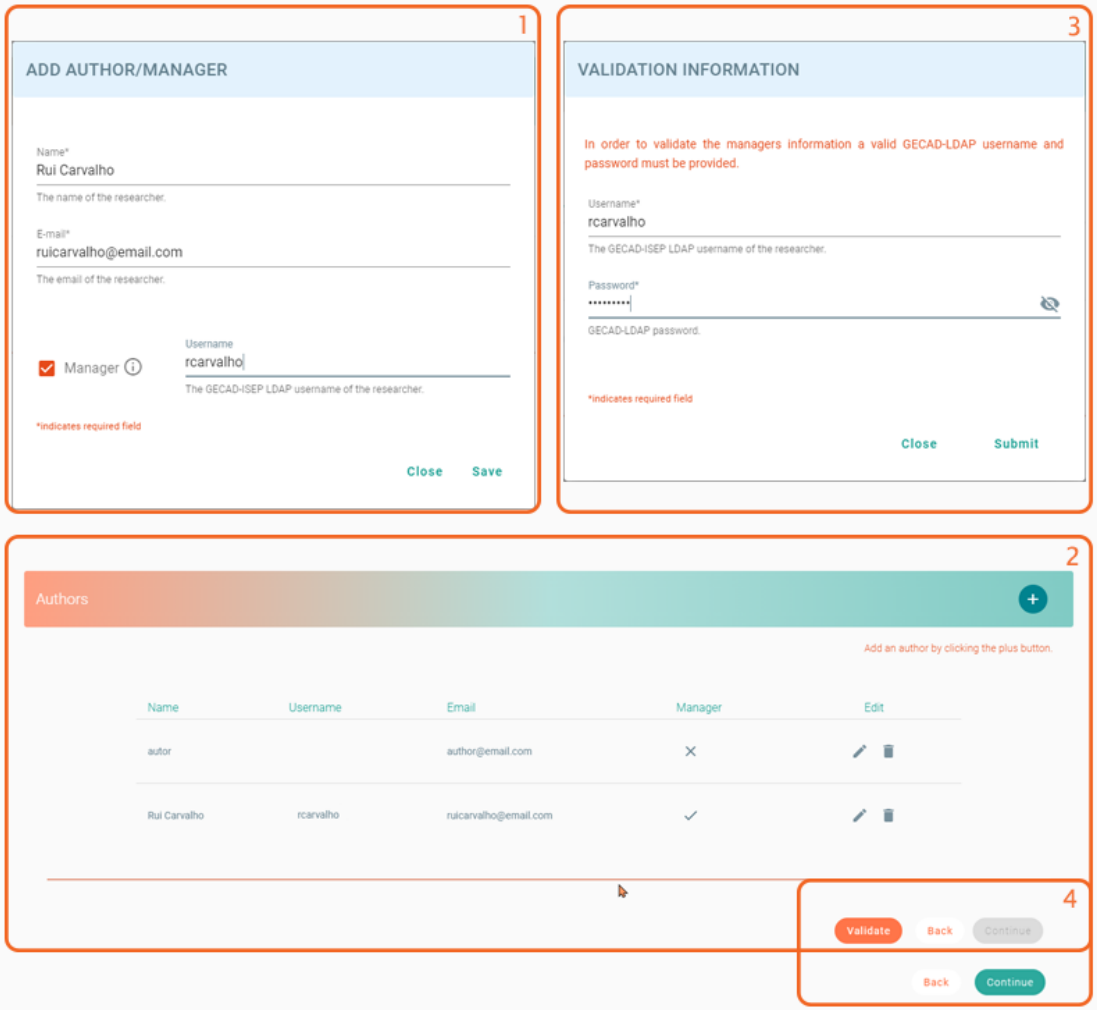


Figure 32 – Step 2 Author definition form.



1 After clicking the plus icon a form for each author must be filled. In order to be able to edit the service after it's configuration, the manager checkbox must be checked and a username from the GECAD-ISEP LDAP account must be provided.

2 After adding a manager, a validation is required. Please click the validation button in order to chek if the manager's data is valid.

3 After clicking the validate button, a login form will be presented. Please fill with the authentication data of the GECAD-ISEP LDAP account. If more than one manager is added, the framework will check each one without the need to authenticate each new manager.

4 If the authentication is successfull, the validate button will disappear and the continue button will become enabled. Otherwise, the manager(s) that don't exist in the GECAD-ISEP LDAP database will be highlighted.

Figure 33 – Step 2 tool tips.

Figure 34 depicts the third step of the configuration process where the requests to interact with the service are defined. In Figure 35 and Figure 36 each field is explained in detail.

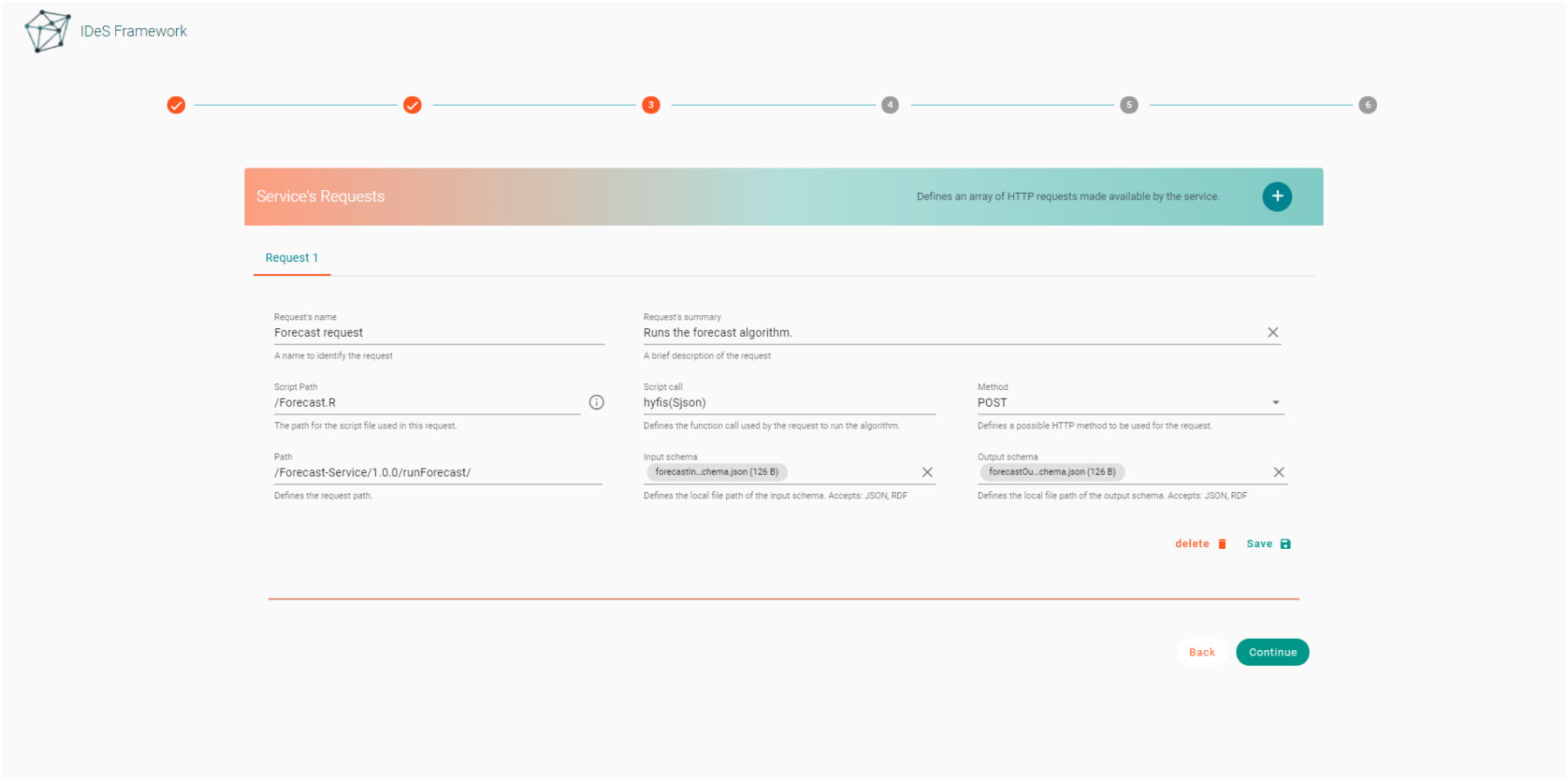


Figure 34 – Step 3 requests definition form (part 1).

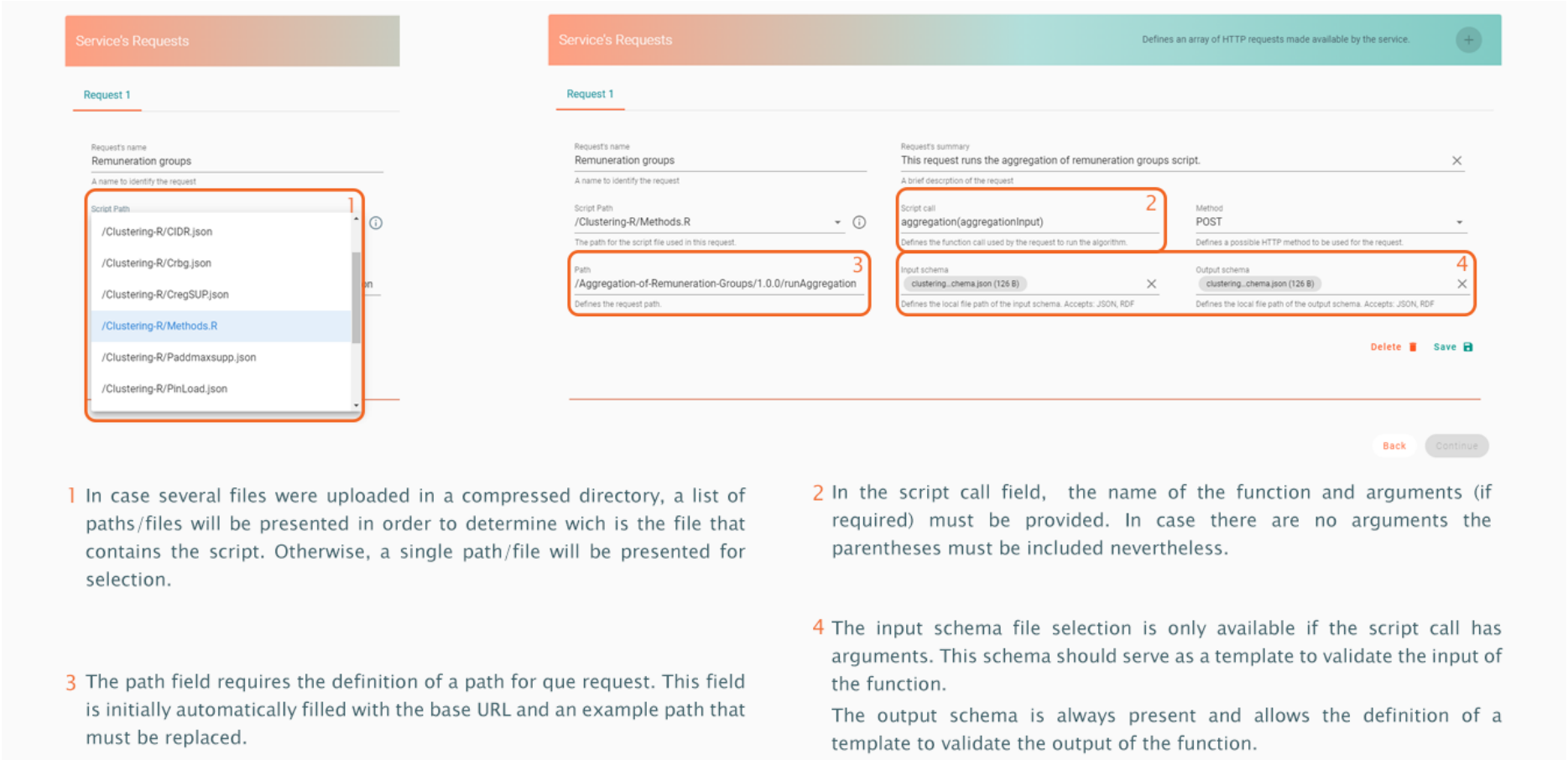
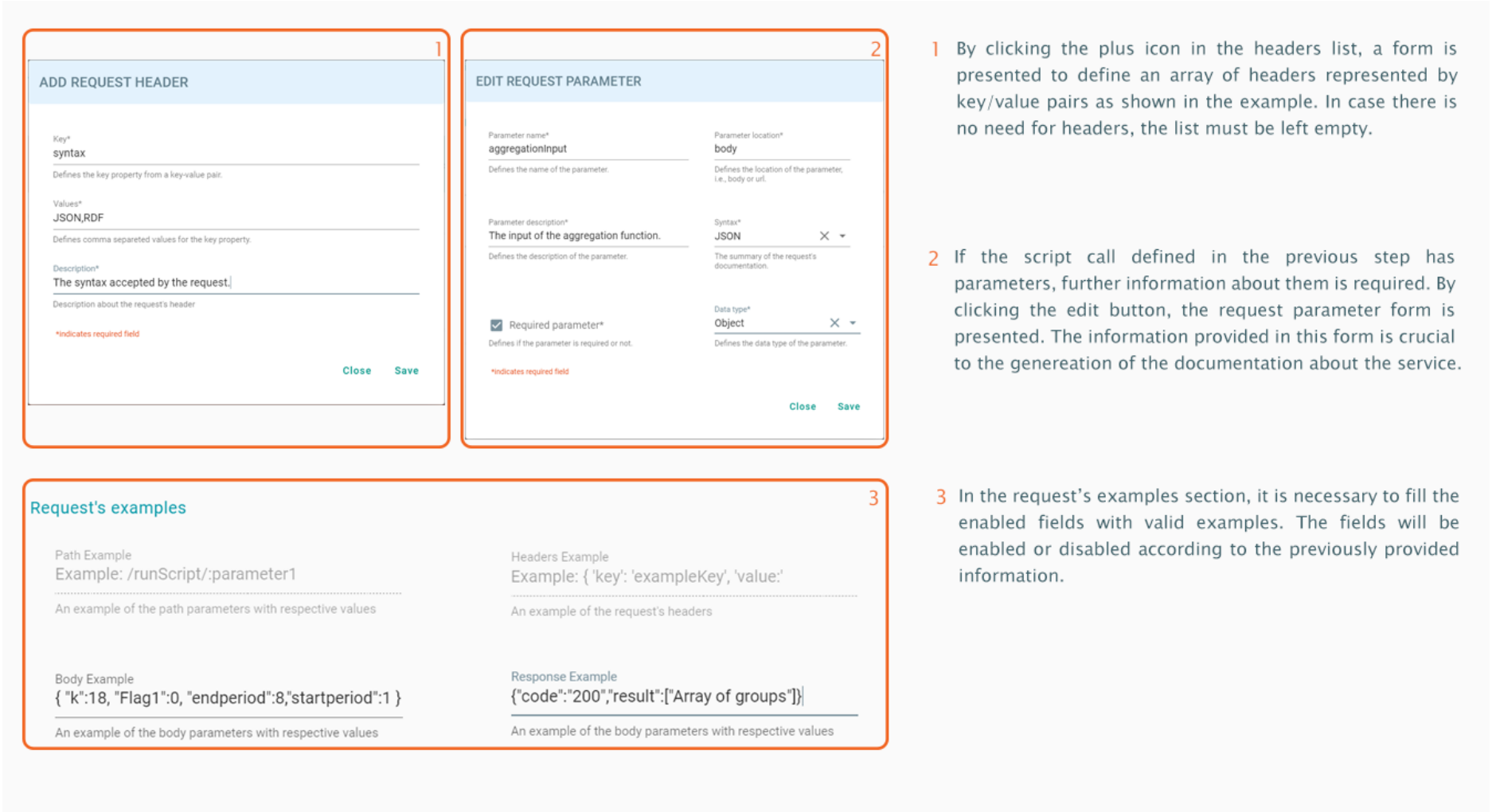


Figure 35 – Step 3 tool tips.



1 By clicking the plus icon in the headers list, a form is presented to define an array of headers represented by key/value pairs as shown in the example. In case there is no need for headers, the list must be left empty.

2 If the script call defined in the previous step has parameters, further information about them is required. By clicking the edit button, the request parameter form is presented. The information provided in this form is crucial to the genereation of the documentation about the service.

3 In the request's examples section, it is necessary to fill the enabled fields with valid examples. The fields will be enabled or disabled according to the previously provided information.

Figure 36 – Step 4 requests definition part 2.

Figure 37 Depicts the final step where a preview of the web services page is presented.

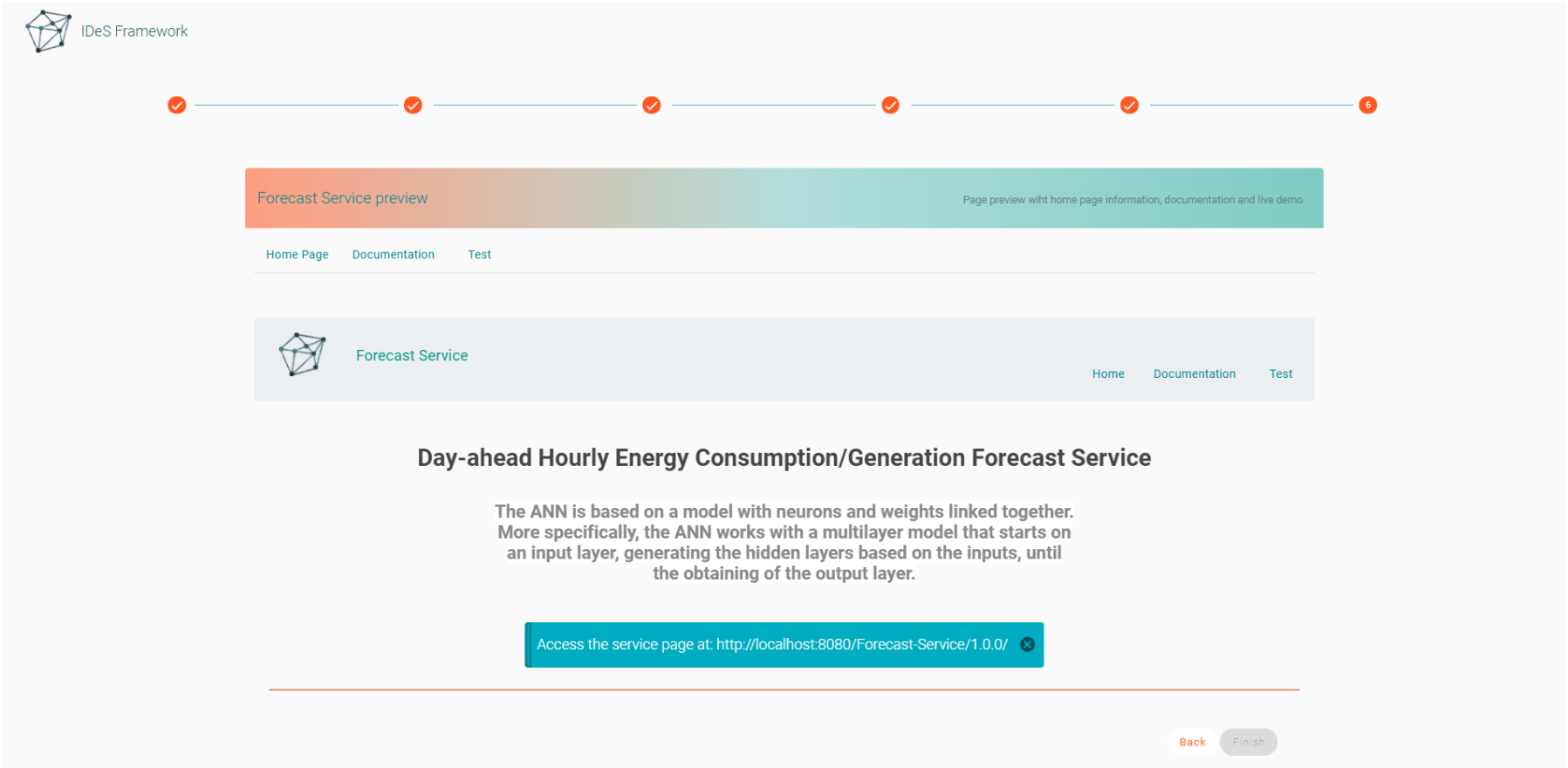


Figure 37 – Step 6 service page preview.

Figure 38 Depicts the generated web services page where the service is described, documented, tested and managed.

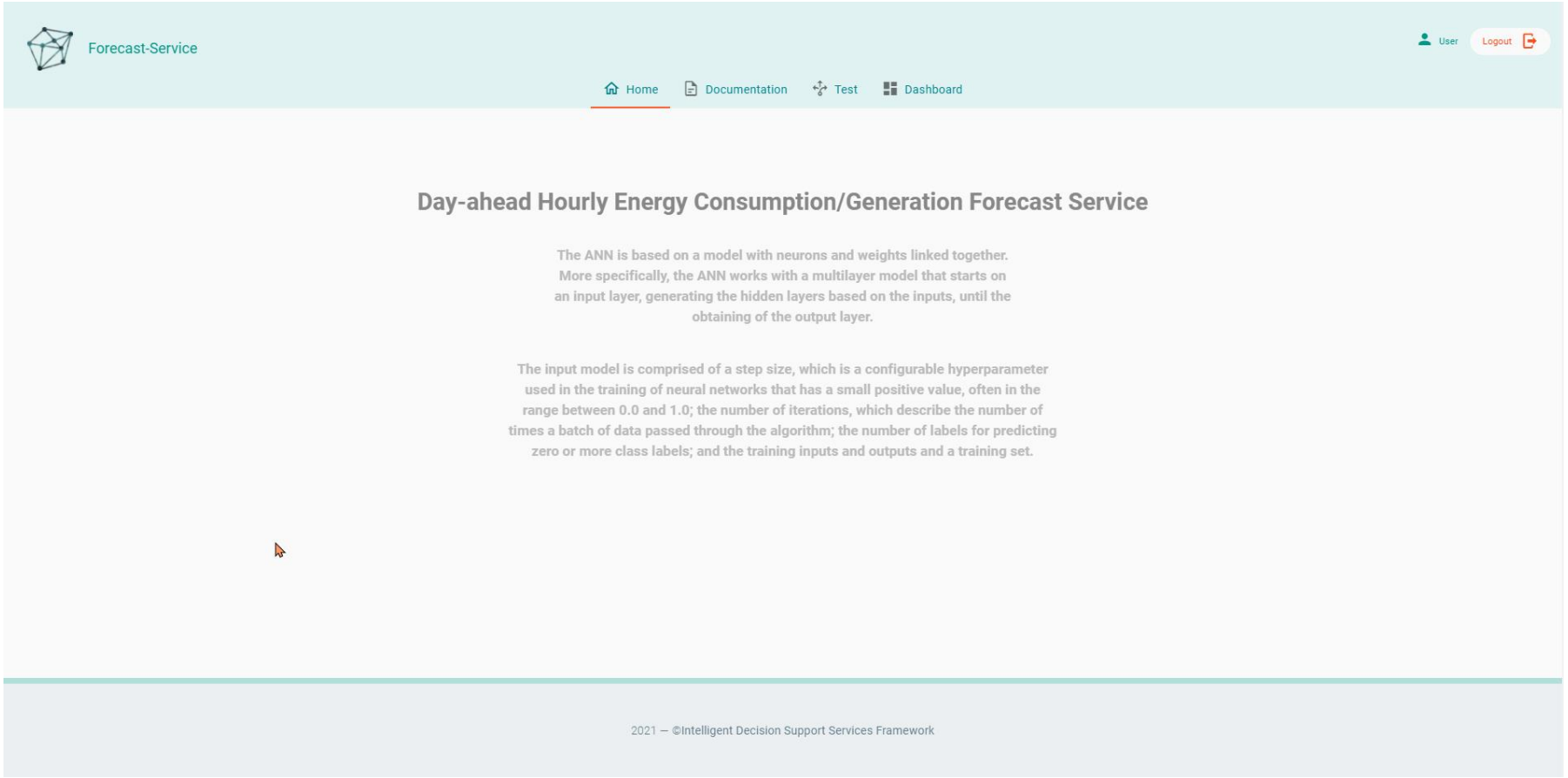


Figure 38 – Service page generated after the configuration process.

In the documentation tab, a user can acess all required information to use the web service (Figure 39).

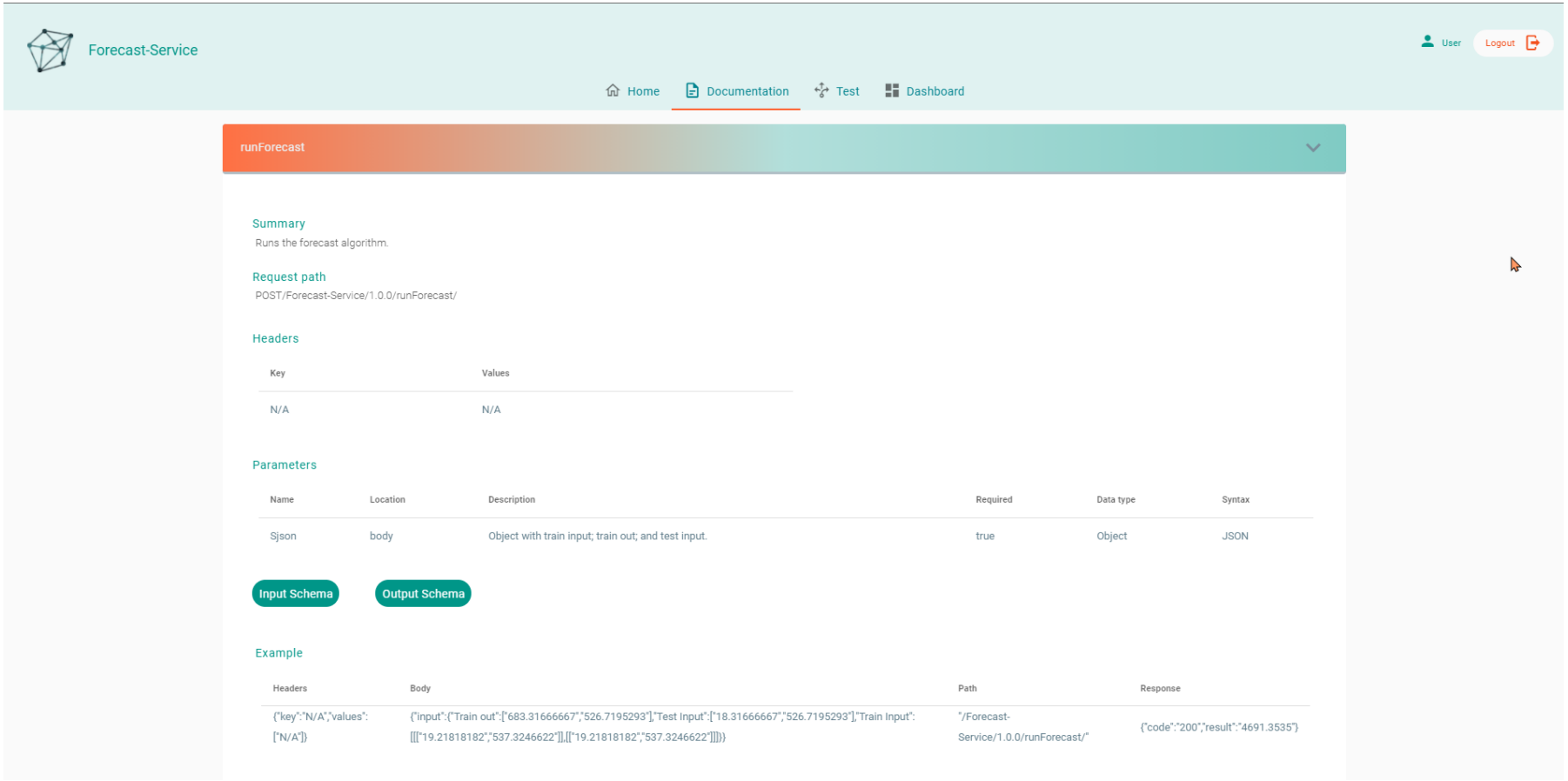


Figure 39 – Service page documentation tab.

In the test tab, a user can test the execution of the service by entering the service's required input (Figure 40).

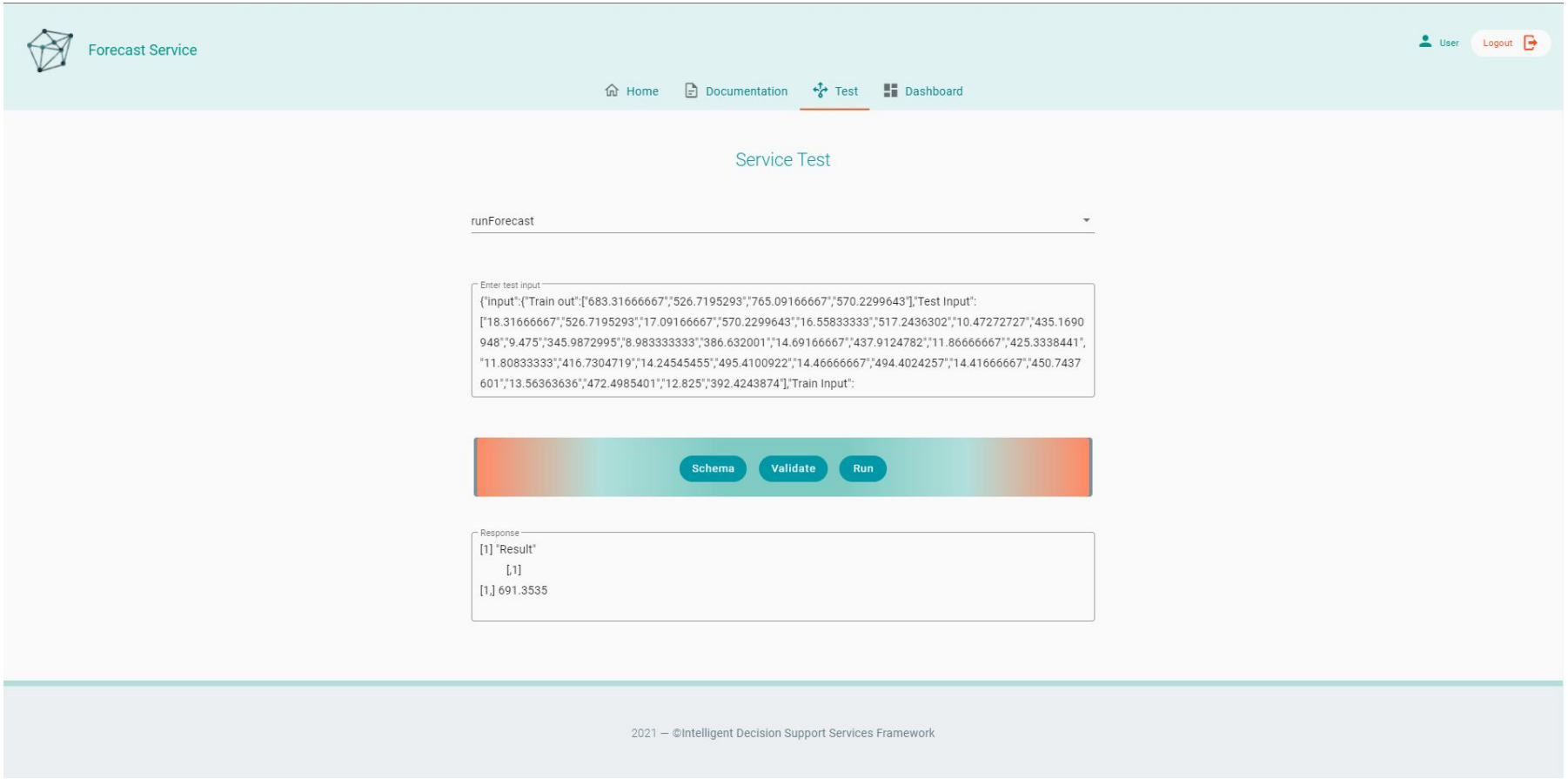


Figure 40 – Service page execution test page.

In the dashboard tab, the user can manage the services configuration, edit the services configuration, or create different versions of the same service (Figure 41).

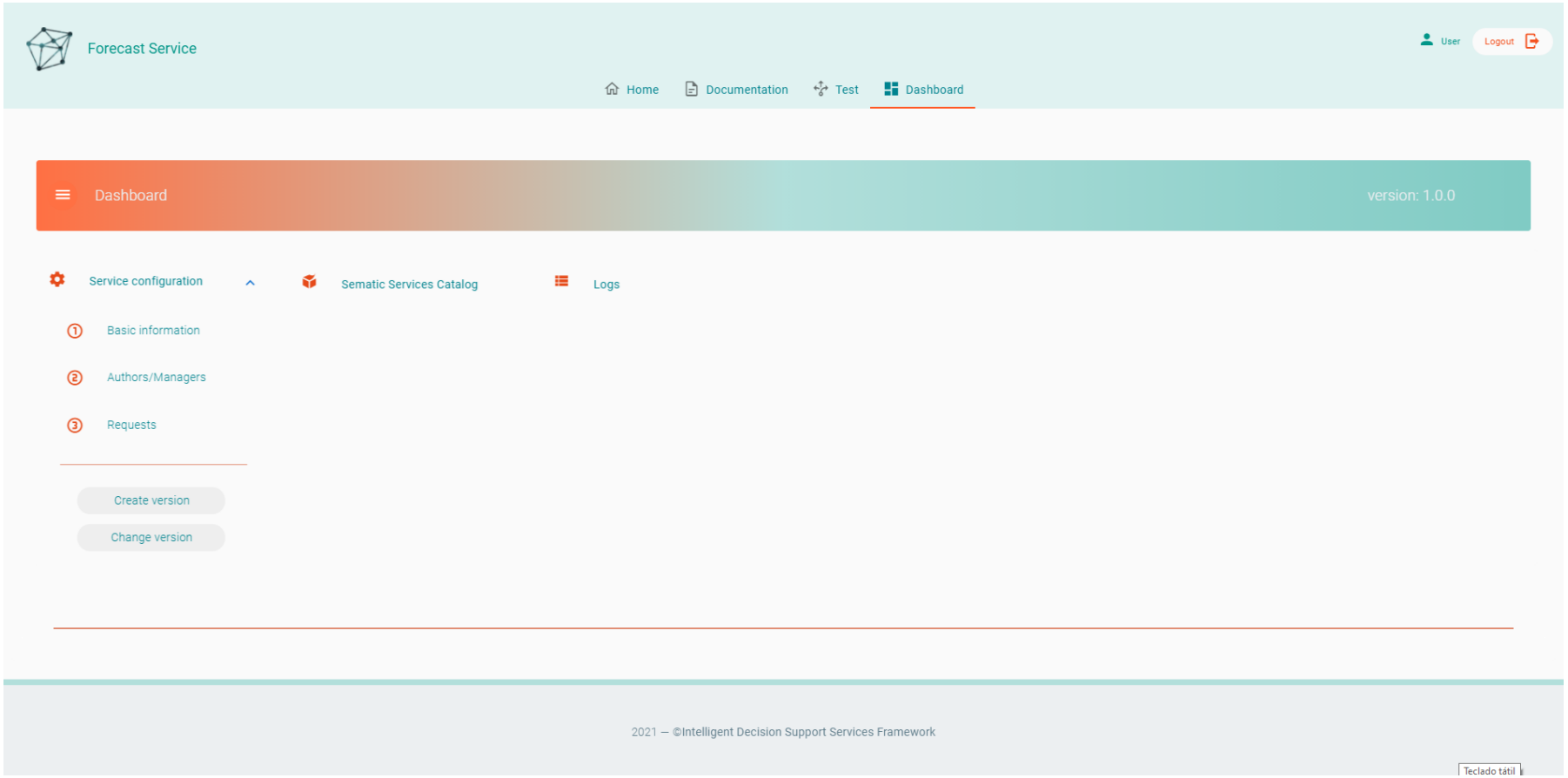


Figure 41 – Service page dashboard for managing and editing.