REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ DES MATHEMATIQUES ET DE l'INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Génie Logiciel

Par :

LAOUMIR Mustapha Abdelkader
RAHLAOUI Lahcen

Sur le thème

Towards a Deep Learning Approach for Social Web Services Discovery

---

Soutenu publiquement le .. / .. / 2022 à Tiaret devant le jury composé de :

Mr BEKKI Khadir	MAA	Université Tiaret	Président
Mr MEGHAZI Hadj Madani	MAA	Université Tiaret	Encadrant
Mr MOKHTARI Ahmed	MAA	Université Tiaret	Examinateur

2021-2022

We dedicate our graduation to all of the Resources

that enabled us to succeed.

Acknowledgements

All thanks and praises go to Allah, the Almighty and the Merciful, for granting us the wisdom and the health to complete this work. A thesis is a long journey. It is a long journey but unrealizable in solitude. For the past eight months.

We would like to thank the Supervisors who have done us the honor of participating in this thesis jury. It is an honor for us that **Mr. Bekki Khadir** has agreed to chair this jury, and that **Mr. Mokhtari Ahmed** has accepted to examine our work.

Last but not least, We would like to express our gratitude to **Mr. Meghazi Hadj Madani**, who supervised our undergraduate thesis. additionally, We would want to appreciate the guidance, support, and advice he offered throughout our time as students. He has since been a good father and mentor, whose solid advice we can always depend on.

Lastly, We would be remiss in not mentioning our family, our friends, and our colleagues. Their belief in us has kept our spirits and motivation high during this process.

Abstract

When it comes to web services, the most obvious issue is always determining the best approach to allow clients to discover the most relevant services for a possible demand. Several approaches have been proposed in this regard, but the majority of them ignore the interaction history of web services. The purpose of this research is to study a new approach that takes into account and exploits these previous interactions in order to implement a solution from the field of Deep Learning that allows for a better discovery web services.

Key Words: Web Services, Semantic Web Services, Social Web Services, Deep Learning.

Résumé

En matière de services Web, la question la plus évidente est toujours de déterminer la meilleure approche pour permettre aux clients de découvrir les services les plus pertinents pour une éventuelle demande. Plusieurs approches ont été proposées à cet égard, mais la majorité d'entre elles ignorent l'historique d'interaction des services Web. Le but de cette recherche est d'étudier une nouvelle approche qui prend en compte et exploite ces interactions précédentes afin de mettre en place une solution issue du domaine du Deep Learning qui permette une meilleure découverte des services web.

Mots Clés: Services Web, Services Web Sémantiques, Services Web Sociaux, Deep Learning.

الملخص

عندما يتعلق الأمر بخدمات الويب ، فإن المشكلة الأكثر وضوحا هي دائما تحديد أفضل نهج للسماح للعملاء باكتشاف الخدمات الأكثر ملائمة للطلب الملائم.
وقد اقترحت عدة نهج في هذا الصدد، ولكن معظمها يتجاهل تاريخ التفاعل بين خدمات الويب. الغرض من هذا البحث هو دراسة نهج جديد يأخذ في الاعتبار و يستغل هذه التفاعالت السابقة من اجل تنفيذ حل من مجال التعلم العميق يسمح بالتوصية باستهالك خدمات الويب الملائمة

الكلمات المفتاحية: خدمات الويب، خدمات الويب الدلالي، خدمات الويب االجتماعية، التعلم العميق.

Contents

List of Figures

List of Tables

General Introduction

"To accomplish great things, we must not only act, but also dream; not only plan, but also believe". »

— Anatole France

General Introduction

Several innovative technologies have been introduced in recent years to improve communication, resource sharing, and system interoperability. These tasks have compelled domain experts to invent and discover new methods for resolving a variety of computer problems. Web services technology is one of the technologies that emerged at the beginning of the twenty-first century and has since been adopted by businesses. Web services are software-based platforms that are free of any software or hardware compatibility constraints. Web services have several advantages; they can be accessed remotely through any sort of platform, they can be used to develop distributed systems, and they are available from any type of client. Web services are programs that may collaborate with one another in a transparent manner for the user, and their implementation is based on a distributed architecture. This service-oriented architecture (SOA) is based on the definition of services and their interactions. The services are published in repositories by the providers that host them. They are available over a network to users who discover, select, invoke, and use them, but they often confront the challenge of automatically determine the best service compositions that best meet their needs. One approach would be to create a social network where the communities are able to evolve with a very minimal human interaction which has many advantages:

- A very minimal dependence on human interactions to survive and evolve.
- Creating the room for new communities to emerge and evolve under the relationship of competition/substitution and collaboration.
- Having historical data of the network which means being able to observe the state of the communities and it's evolution with time.

Despite the fact that a Social Network offers several benefits, it is a difficult task that must be overcome. Creating a social network for Web Services is confronted with several challenges:

- Finding the right methods to determine the relationship between two Web Services.
- The lack of data that is relevant to a Social Network for web services.
- There is a scarcity of quality predictive models on which to base recommendations. Unfortunately, linear models result in poor predictions.

In this regard, the goal of our work is to propose with the help of Deep Learning a Social Network for web services that uses social interaction data between these services, to analyse it, exploit it and observe the evolution of its communities in order to improve the discovery; to that end, the following manuscript has been structured as follows:

The first chapter entitled "Web Services" which consists of five sections, in the first and the second sections we present some definitions, characteristics and operation of web services and APIs. In the third sections, we present some limits of the web services. In the fourth section, we present the semantic web services by evoking their evolution from syntactic to semantic, as well as the standards used in the semantic description of the web services, and in the fifth section, we give a definition of the Social Web Services and we talk about the added value of a social aspect to the web services.

In the second chapter entitled "Machine Learning for NLP" we present the methods most commonly used for data analysis in the field of Natural Language Processing.

In the third chapter entitled "Building a social Network of Web Services to improve WS discovery process" we introduce our approach adopted for the discovery of web services and the methods used in the implementation of our own solution, explaining all the steps and the results obtained by the implementation of our solution.

Chapter 1

Web Services

« If you are unable to understand the cause of a problem it is impossible to solve it.. »

— Naoto Kan1

State of the art

1.1 Introduction

Service-Oriented Architecture (SOA) has emerged as an architectural approach that enhances the service delivery performance of existing traditional systems while still retaining their most important features. This approach, due to its flexibility of adoption, has gained the attention of both academic and business entities, especially in the development of world-leading technologies such as Cloud Computing (CC) and the Internet of Things (IoT). Although many studies have listed the success factors of SOA, a few minor failures have also been reported in the literature. Despite the availability of rich material on SOA, there is a lack of systematic reviews covering the different aspects of the SOA concept in Information Systems (IS) research. Therefore, the central objective of this study is to review existing issues of SOA and share the findings with academia[1].

Service-oriented architecture is the term used to describe the model architecture for the execution of distributed software applications, the latest two models (CORBA and DCOM) are architectures of distributed software components, not service-oriented architecture, where the term "service" is generally absent from their terminology (except, for example, in CORBA where its services are referred to as features offered by the middleware platform to application components), on the other hand, one of the advantages of SOA is that distributed applications n39; need generic distributed middleware system no longer to communicate, but only with interoperable protocols and communication technologies on the Internet, these understandings and methods will be explored in details later.

The emergence of Web services technology should bring a very high level of interoperability and very low coupling, in order to establish the basis of a service-oriented architecture with an advanced dynamic configuration, some sort of distributed application architecture that is weakly coupled (or highly coupled) by a set of decentralized, autonomous distributed applications (web services), interacts based on communication protocols and implements work using open and non-invasive techniques[2].

1.2 Web Services

1.2.1 History

As the World-Wide Web (WWW) exploded into the lives of the public in the 1990s, people suddenly had vast amounts of information placed at their fingertips. The system was developed to allow information sharing within internationally dispersed working groups. The original WWW consisted of documents (i.e., Web pages) and links between documents. The initial idea of the WWW was to develop a universal information database to publish information that could be accessed in a reliable and simple way by consumers.

The information would not only be accessible to users around the world, but the information would be linked so that it could be easily browsed and quickly found by users. Organizations soon realized the importance of this technology to manage, organize, and distribute their internal data and information to customers and partners. As organizations started to implement business-to-customer and e-commerce solutions, they realized that the initial technologies associated with the WWW were not sufficient to sell products over the Internet. Additional functionality was required to guarantee that transactions were conducted in a secure way. To this end, SSL (Secure Sockets Layer), a protocol defined by Netscape, was developed for transmitting private documents via the Internet. Using SSL, organizations were able to implement a solution to obtain confidential user information, such as credit card numbers.

With globalization, organizations were progressively undertaking mergers and acquisitions.

This has created organizations with an IT environment composed of disparate legacy systems, applications, processes, and data sources. In order to meet increasing customer and business partner expectations for real-time information, organizations were required to link their heterogeneous, autonomous and distributed systems to improve productivity and efficiency. This important requirement led to the development and deployment of EAI (enterprise application integration) solutions. EAI platforms were used for integrating incompatible and distributed systems.

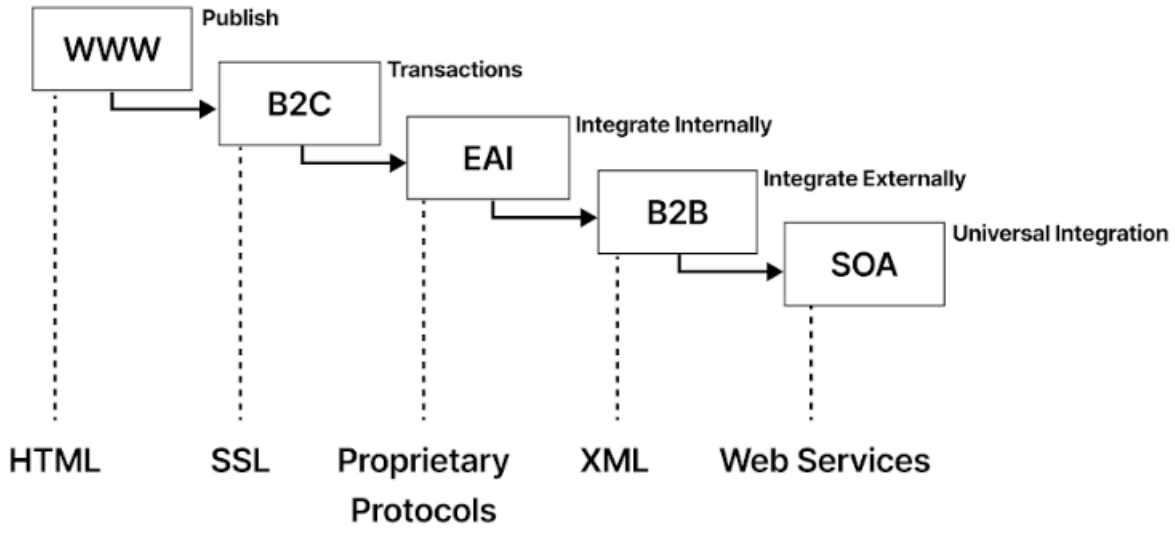

Figure 1.1: The evolution of business usage on the WWW

The limitations of EAI solutions made most organizations realize that integrating internal systems with external systems to business supply chain members was a key to staying competitive, since the majority of business processes spanning across several organizations. Internal and external systems are needed to communicate over networks to allow businesses to complete a transaction or part of a transaction. To achieve this level of integration, business-to-business (**B2B**) solutions were developed. B2B infrastructures were directed to help organizations to streamline their processes so they could carry out business transactions more efficiently with their business partners (such as resellers and suppliers). To reach a higher level of integration, most B2B solutions have relied on the use of XML as the language to represent data.

The figure 1.1 above shows the evolution of businesses usage on the WWW, XML allows one to model data at any level of complexity since it is extensible with the addition of new tags. Data can be published in multiple formats. In contrast to the proprietary protocols used by EAI platforms, XML is vendor and platform independent allowing standard commercial software to process any conforming document.

Many organizations have already seen and experienced the advantages of using XML to represent data for Web-based information exchanges (such as B2B communications). Nevertheless, organizations realized that their B2B strategies have to lead the development of architectural solutions that often exhibited a tight-coupling among interacting software applications which limited the flexibility and dynamic adaptation of IT systems. As a result and to overcome these

limitations, the concept of service-oriented architecture (SOA) was introduced and defined as a method of designing, developing, deploying and managing discrete pieces of computer logic (i.e., services) within the WWW[3].

1.2.2 Definition

We can define a web service as a software that could make itself available on the internet and identified by a **URI** and designed to enable communication and data exchange between heterogeneous applications and systems in distributed environments via the Internet. where It has a defined interface described in a machine processable format (*WSDL*). Other systems can then interact with the web service in the manner specified by its description using SOAP messages; we use XML to encode all transmissions to a web service. For example, a web service can be called by a client by sending an XML message and waiting for a response.

Applications of different languages can intercommunicate with other applications through the web service over the network. Web services are therefore the most efficient way to share methods and functionality. In addition, they reduce implementation time by allowing you to leverage existing services directly.

In a web service, machine-readable file formats like *XML* and *JSON* are transferred using a web technology like HTTP.and it is described by an XML interface named **WSDL**, it can exchange documents with other services using the **SOAP** protocol, it can be searched in a directory such as **UDDI**.

In practice, a web service is a web-based object-oriented interface to a database server that can be used by another web server or a mobile app to provide a user interface to the end-user. Many organizations that provide data in HTML format will also provide it as XML or JSON on their server, often via a Web service to allow syndication. A mashup is a Web application in which a Web server consumes multiple Web services from different machines and compiles the content into a single user interface[4].

Figure 1.2 shows the general facets of a web service.

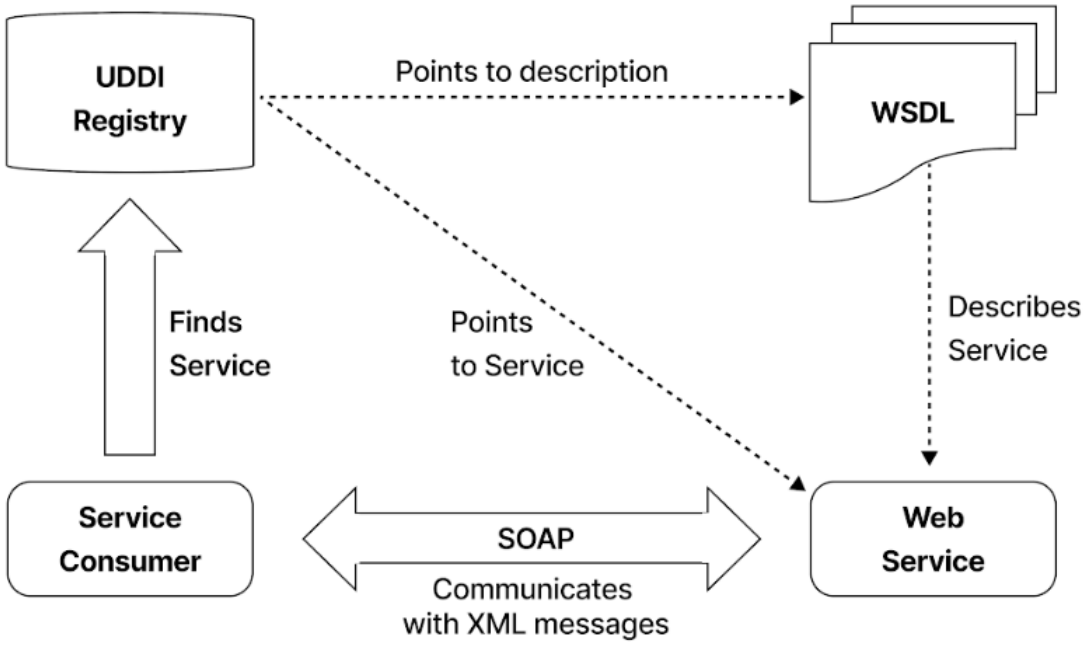

Figure 1.2: The general facets of a web service

1.2.3 Web service architecture

Web services apply the principles of the web to devices, An example of this is the World Wide Web.Web services can communicate through a set of technologies that share a common architecture designed to be implemented on different systems. We can count these technologies as follows: $HTTP$, $WSDL$, $REST$, $XML-RPC$, $SOAP$ and $UDDI$.

There are also three roles where publishing, finding and linking are done, and these three roles are:

- **Service Provider :** A web service provider creates and hosts web services. The web service is implemented using WSDL language, and clients can invoke the service also by deploying the WSDL interface as well.
- **Service Request :** The service requester, using XML, issues a search request and calls the web service and this role is considered as a consumer of the web service.
- **Service Registry :** Web services are published with the help of a service registry, Which is based on the UDDI specification, a description of the WSDL service and a URL are provided.

As we can see in the figure 1.3 the architecture of the web service and also its roles

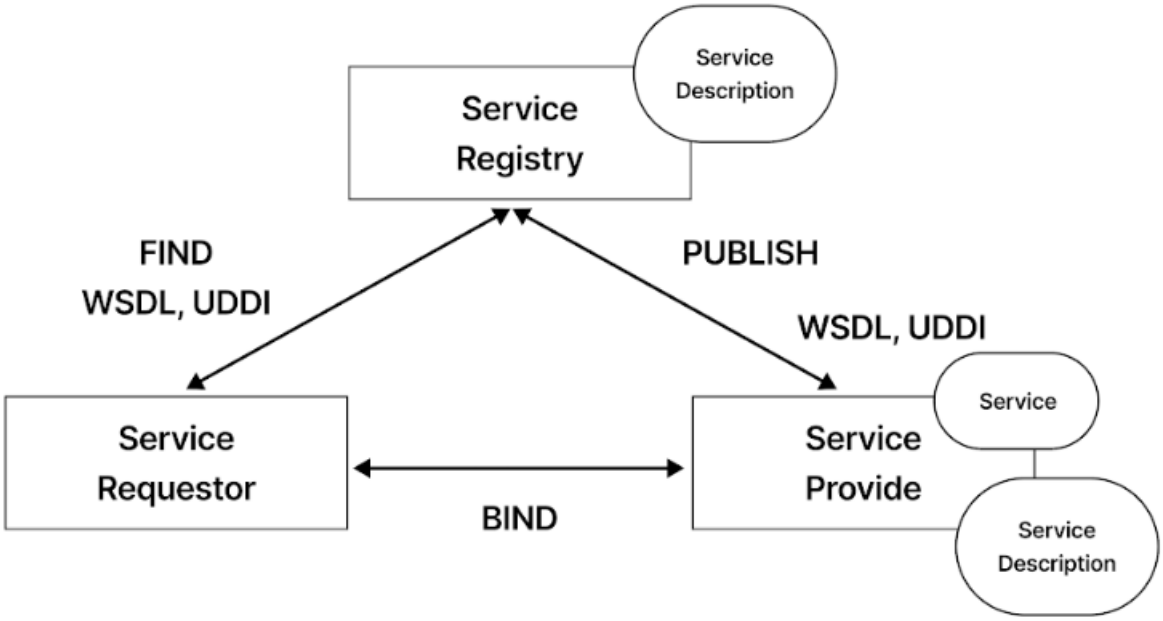

Figure 1.3: Web Service Architecture

1.2.4 Web service characteristics

Web services have the following special characteristics:

1. **XML-based :** The information representation and record transportation layers of a web service use XML. There is no need for networking, operating system, or platform binding when using **XML**.

2. **Coarse-grained :** it offers more functionality. It combines one or more *fine-grained* services into a coarse-grained one.

3. **Loosely Coupled :** A web service allows for loosely coupled system connections. Web APIs add an abstraction layer to the environment, allowing the connection to be more adaptable and flexible. Capability to be synchronous and asynchronous : A client invokes synchronous Web services over existing Web protocols and waits for a response. RPC-oriented messaging is used to serve synchronous Web services .Asynchronous and synchronous endpoints are implemented using Servlets, *HTTP*, and *XML/SOAP*.

4. **Supports RPC :** A web service supports **RPC** by providing personal services that are equivalent to those provided by a traditional aspect. A web service (also known as a web API) is a program that provides an interface that can be called from another program. Any application can call for application-to-application programming.

1.2.5 The Layers of a Web Service

Web services adhere to a two-tiered structure composed of four major layers:

- **Transport layer :** This layer is in charge of transporting XML messages between apps. Essentially, this layer includes HTTP, SMTP, FTP, and new protocols such as BEEP.
- **Communication layer :** This layer is in charge of formatting the changed data so that messages may be understood at all times. Currently, two completely different architectural styles are used for these data exchanges. On the first hand, we have the distributed operations architecture (RPC protocols) based on XML, which includes XML-RPC and SOAP, and on the other, we have the web resources architecture, REST (Representational State Transfer), which is based only on good Web principles (HTTP protocol in particular).
- **Service description layer :** This layer is in charge of describing the public interface of the Web service. The language used to describe a Web service is WSDL, which is a standard XML-based notation for creating a service's interface description. This specification defines an XML grammar for describing Web services as collections of communication endpoints (ports) via which messages are exchanged.
- **Service discovery layer :** This layer is in charge of centralizing services in a common registry and simplifying the search and publishing of Web services. Currently, service discovery is ensured via a UDDI (Universal Description, Discovery, and Integration) calendar[5].

1.2.5.1 XML

XML is a protocol for containing and managing information. On another level, it's a family of technologies that can do everything from formatting documents to filtering data. And on

the highest level, it's a philosophy for information handling that seeks maximum usefulness and flexibility for data by refining it to its purest and most structured form. A thorough understanding of XML touches all these levels. Let's begin by analyzing the first level of XML: how it contains and manages information with markup. This universal data packaging scheme is the necessary foundation for the next level, where XML becomes really exciting: satellite technologies such as stylesheets, transformations, and do-it-yourself markup languages. Understanding the fundamentals of markup, documents, and presentation will help you get the most out of XML and its accessories[6].

And here it is a list of XML's features: XML can store and organize just about any kind of information in a form that is tailored to your needs. XML offers many ways to check the quality of a document, with rules for syntax, internal link checking, comparison to document models, and data typing. With its clear, simple syntax and unambiguous structure, XML is easy to read and parse by humans and programs alike. XML is easily combined with stylesheets to create formatted documents in any style you want. The purity of the information structure does not get in the way of format conversions [6].

1.2.5.2 SOAP

SOAP (Simple Object Access Protocol) is a long-standing standards-based web services access protocol. SOAP, which was created by Microsoft, is not as straightforward as the moniker suggests. SOAP only uses XML to deliver communications services. SOAP was created by Microsoft to replace earlier technologies that do not operate well on the internet, such as the Distributed Component Object Model (DCOM) and Common Object Request Broker Architecture (CORBA). These technologies fail due to their reliance on binary communications. SOAP's XML communications is more reliable over the internet. The key is that SOAP is very expandable, but you just utilize the parts that you require for a certain purpose. For example, when utilizing a public online service that is publicly open to everybody, you don't actually require WS-Security[7].

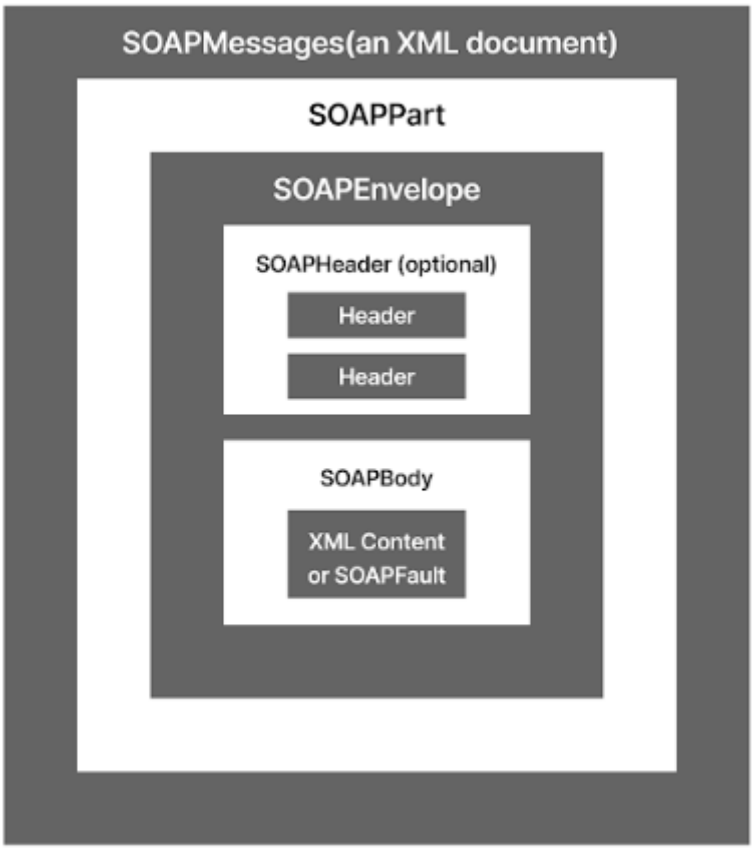

Figure 1.4: SOAP Protocol

Figure 1.4 shows the protocol used in SOAP.

1.2.5.3 REST

REST stands for representational state transfer and was created by computer scientist Roy Fielding, it s an architecture style for designing network applications , making it easier for systems to communicate with each other, we can also execute ***CRUD*** methods and get access to the data on the server , depending on the client server protocol mostly http by sending requests. than it could transfer server objects as resources .

All web services are based on the REST; hence it is called a *RESTful* service. The purpose of developing a RESTful web service is to make the web service more effective.

Fielding arrived at REST by evaluating all networking resources and technologies available for creating distributed applications. Without any constraints, anything and everything goes, leading us to develop applications that are hard to maintain and extend. With this in mind, he looks at document distributed application architectural styles beginning with what he calls the

null space—which represents the availability of every technology and every style of application development with no rules or limits[8].

1.2.5.4 HTTP

HTTP (Hypertext Transfer Protocol), designates in computer language a communication protocol between a client and a server for the World Wide Web. It was invented in the 1990s by Tim Berners-Lee. It establishes a link between a computer (client) and a Web server. The client, via a Web browser, sends a request to the server, which provides an almost instantaneous response. In other words, the HTTP communication protocol is what allows an Internet user to access content (a Web page, a CSS file, etc.), by ordering the server to perform an action. Requests and responses are composed of a header and a message body expressed in *ASCII* text. On a more technical level, the HTTP protocol (Figure 1.5) is an application layer protocol, 7th layer of the OSI model. It uses by default the software port 80.

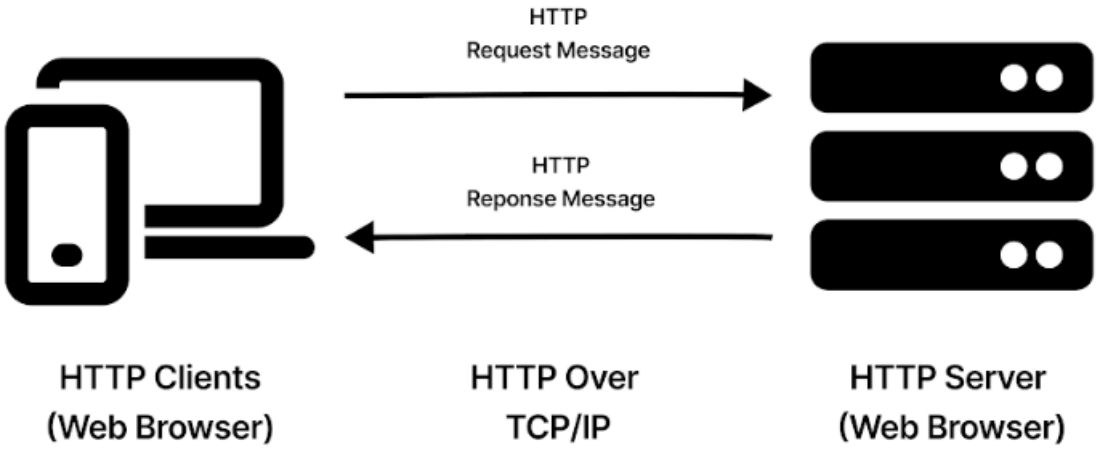

Figure 1.5: HTTP protocol

1.2.5.5 UDDI

The Universal Description, Discovery and Integration (*UDDI*) service directory is a standard for publishing and standard for publishing and discovering information about Web services. It enables providers to present their Web services to customers. The UDDI specification aims to create an independent platform, an open Framework for the description, discovery and integration of business services. UDDI focuses on the discovery process of the Service Oriented Architecture (*SOA*) discovery process and uses standard technologies such as **XML**, **SOAP**

and **WSDL** to simplify simplify collaboration between partners in the context of business exchanges. The repository can be accessed in several ways.

- The white pages include the list of companies and information associated with them (contact details, company description, identifiers, etc.).
- The yellow pages list the Web services of each company under the WSDL STANDARD.
- The green pages provide precise technical information on the services provided.

The classic UDDI usage scenario is illustrated in 1.6. Company *B* has published the Web service *S* and Company *A* is a customer of this service.

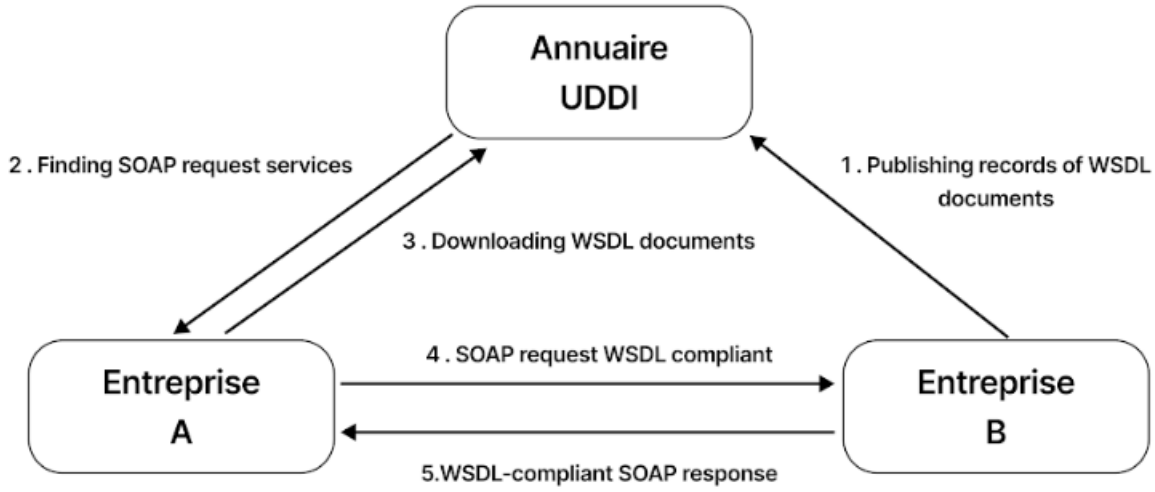

Figure 1.6: General diagram of the UDDI directory.

1.2.5.6 WSDL

WSDL stands for Web Services Description Language, it is a standard description language, it is a system-neutral programming language, as it is the interface that is provided to users, and the XML schema is used as a standard for describing message formats.
Also, the aforementioned language provides a detailed description of web services, including protocols, ports used, location of the service, methods that can be used, and so on.
The WSDL language file consists of six basic elements (Figure 1.7) and other optional elements, of which we enumerate the following:

1. **Definitions :** the main element of the document, containing the elements of the web service and giving its name, as well as declaring the name-spaces used.

2. **Types :** WSDL is not tied to a specific writing system, it uses the XML specification, and all types of data that will be used between the client and the provider are described.

3. **Message :** the name of the message is determined by the element, either a distress message or a response message, and it can contain the elements of the part or not, so the values carried by the message return to us.

4. **Prototype :** the prototype collects several messages to compose a process, and each process represents an input and output message for messages.

5. **Binding :** this component contains information specific to the soap, and it writes specifications centered on the way the service is implemented, represented by the communication protocol and messages specified by a specific port.

6. **Service :** you specify addresses to invoke a particular service, where they are used to group related ports, and the URL will be who calls the soap service.

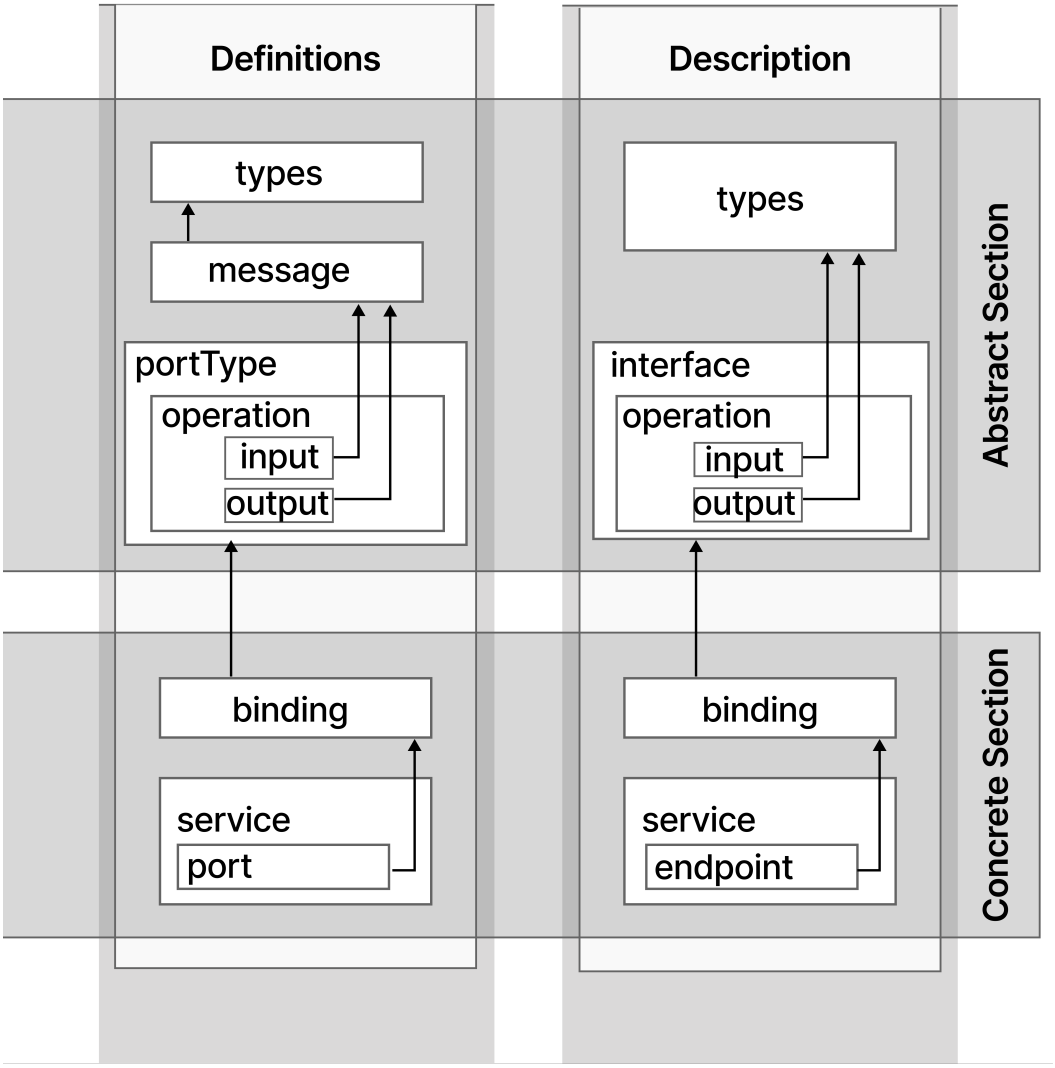

Figure 1.7: WSDL

1.3 APIs

An API is simply a point of contact between a development environment and the developers, allowing them to take advantage of the environment's services without having to build everything from the ground up. In general, the API's goal is to hide the details, provide Encapsulation, and highlight how to use the code.

To give an example from reality, when car engine companies create a new engine, they package it and sell it to companies so that they can use it to build their own products, such as cars, without having to know the details of how it was made.

A connection between computers or computer programs is known as an application programming interface (*API*). It's a type of software interface that provides a service to other programs. An API specification is a document or standard that describes how to create or use a connection or interface. An API is implemented or exposed by a computer system that meets this standard. The term API can be used to refer to either the specification or the implementation.

An application programming interface, in contrast to a user interface, which connects a computer to a person, connects computers or pieces of software to each other. It is not intended for use by anyone other than a computer programmer who is incorporating it into software. An API is made up of various parts that serve as tools or services for programmers. A program or programmer who makes use of one of these parts is said to be calling that portion of the API. Subroutines, methods, requests, and endpoints are all terms used to describe the API calls. These calls are defined by an API specification, which explains how to use or implement them.

One of the goals of APIs is to hide the internal workings of a system, exposing only the parts that a programmer will use and keeping them consistent even if the internal workings change. An API can be custom-built for a specific pair of systems, or it can be a shared standard that allows many systems to communicate with one another.

Web APIs, which allow communication between computers connected by the internet, are commonly referred to as APIs. Programming languages, software libraries, computer operating systems, and computer hardware all have APIs. APIs date back to the 1940s, but the term didn't become popular until the 1960s and 1970s. Micro-services, which are loosely coupled services accessed through public APIs, have grown in popularity as a result of recent developments in the use of APIs[9].

1.4 Web service challenges:

The process of exchanging information seamlessly between internal business units, customers, and partners is critical to success; however, most organizations use a variety of disparate applications that store and exchange data in different ways, making it impossible for them to "talk" to one another productively. Web services have evolved as a practical, cost-effective solution for connecting information distributed between critical applications across

the previously insurmountable operating system, platform, and language barriers. Web services have a number of advantages, but they also have a number of drawbacks. Most importantly, despite the fact that Web services are designed to be simple, developing and implementing them can be difficult[10].

- the proposed approaches only deal with specific aspects of the SBA adaptation The issues of identifying and modeling cross-layer dependencies, as well as designing cross-cutting adaptation strategies, remain unsolved.
- The proposed solutions are not adaptable enough to a wide range of scenarios or use cases. The majority of approaches are concerned with changes in *QoS* and related adaptation strategies, such as service replace-ability, or with a specific set of application faults and recovery actions.
- Target the adaptation types in which the goal is to modify the system in response to changes that have already occurred, rather than adapting the system before these changes occur, i.e. preventive adaptation This necessitates the development of novel techniques and methods for application diagnosis in order to anticipate potential changes and their consequences.
- The role of hybrid and highly dynamic open systems continuously grows, The problem of modeling various aspects of the application context and its evolution should be the focus of the adaptation design. Only a few adaptation proposals address context-aware methods and techniques at the moment, limiting the applicability of existing approaches to new requirements and needs[11].

1.5 Semantic Web Services

A semantic web service is the result of incorporating semantic concepts into the web service description. Figure 1.8 shows how the concept of semantic web services combines two important research domains in the field of technology: the semantic web and web services. These two trends will combine to create a fully automated Web for computer interaction, where people can organize their collaboration and business relationships. To make the Web more dynamic, this convergence task is accomplished by making Web services automatically

exploitable by machines and realizing interoperability between applications via the Web. The concept of semantic Web services is the same as the automation of tasks such as finding, selecting, composing, and implementing appropriate services.

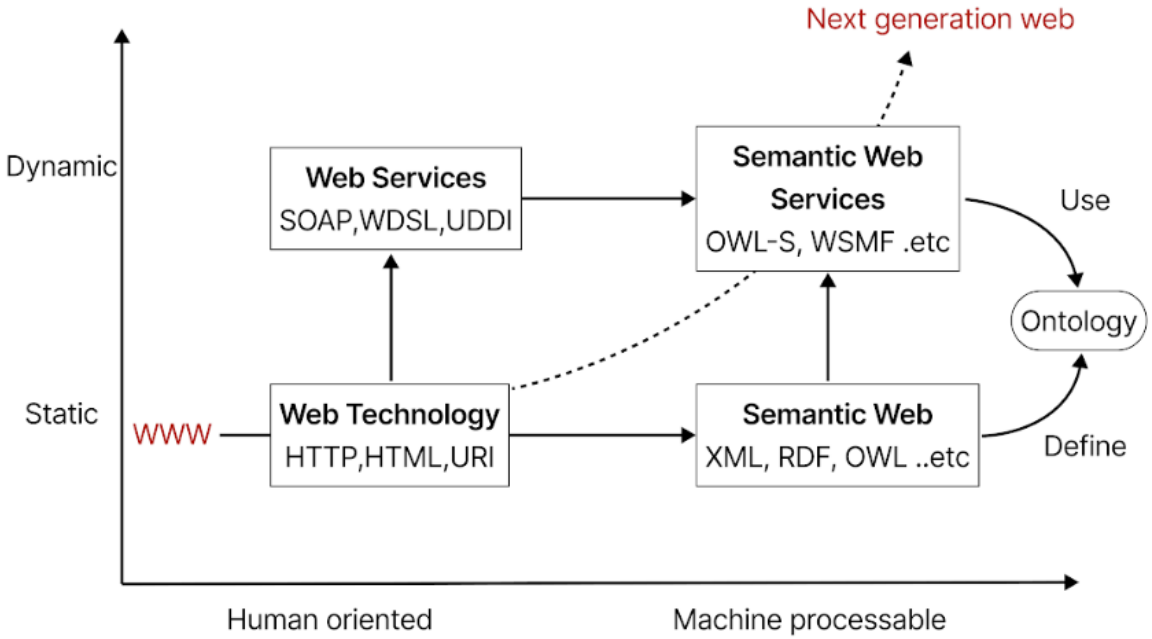

Figure 1.8: Semantic web service

The main goal of Semantic Web Services approaches is the automation of service discovery and service composition in a **SOA**.

There are others objectives of semantic web services is to create a semantic web of web services whose properties, capabilities, interfaces and effects are described in an unambiguous and machine-readable way, using the technical layers without being conceptually dependent.

And about it's architecture is founded on web services, and the stack is made up of multiple layers, each of which adheres to a different standard. The three layers that make up the transport layer, as well as the three levels that make up the basic infrastructure described above, can be found above the transport layer, as shown in Figure1.9

These layers are based on the new *SOAP*, *WSDL*, and *UDDI* standards. A business process layer is added, allowing business processes to be accessed both within and outside of an organization. The business layer is completed by cross-functional layers such as security, transactions, administration, and quality of service. The semantic layer connects to each of the four upper layers, allowing these processes to be automated.

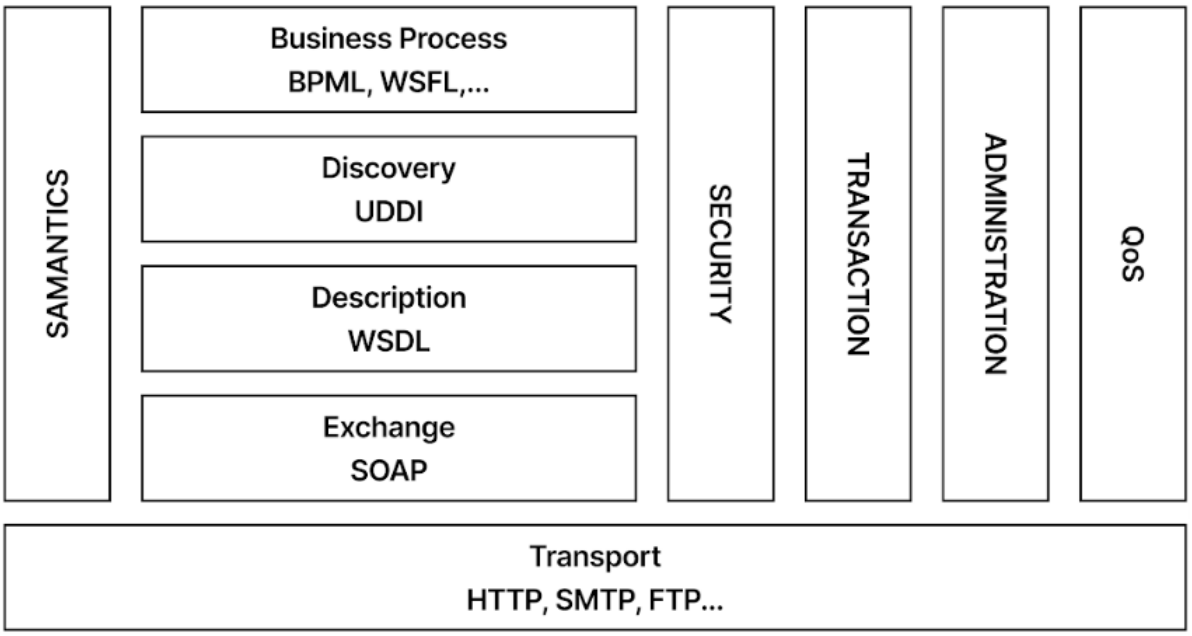

Figure 1.9: The transport layers

1.6 Social Web Services

Social web service is the result of combining Social Computing and Service-Oriented Computing **(SOC)** (oriented computing). On the one hand, social computing refers to the computerized facilitation of social studies and human social dynamics, as well as the design and use of information and communication technologies that are socially aware. Collective actions, content sharing, and information dissemination are all aspects of social computing. SOC service-oriented computing, on the other hand, is based on the principles of service supply and demand, weak coupling, and cross-organizational flows. It can "know" who they've worked with in the past and who they'd like to work with in the future thanks to the combination of social computing and service-oriented computing. The "*memory*" of actions that **SWSoc** can accumulate over time and apply in the future is made up of these two time-stamped elements. They also show a group of **SWSoc** working together to share their experiences in response to requests for the development of complex value-added composite services.

SWSocs should be at the forefront of advising users on how to create and reuse value-added services.

When companies discover and use Web services to meet their needs, they are included in service compositions based on their needs, they are included in service compositions based

on the functionality they offer and the quality of service (**QoS**) they can guarantee, which implies a need for contracts[12].

Social networks illustrate the enormous popularity of Web 2.0 applications, which help users to become proactive. Familiarly, we can now refer to users as prosumers, suppliers and consumers. The prosumers post definitions on wikis, establish interest groups and share tips and tricks. These various operations illustrate the principles of "*I offer services that someone else might need*" and "*I need services that someone else could provide*" that **SOA** is based on. The offers and requests for services perfectly demonstrate the behavior of people in today's society, imposing a social dimension on the way Web services should be managed in terms of Web services must be managed in terms of description, discovery, binding and composition[12].

1.7 Conclusion

Given the growing applications of service computing and cloud computing, a large number of web services are deployed on the Internet, which triggers the search for web service recommendation. Despite the quality of service, the use of user feedback has become the current trend in service recommendations. At the same time, with the prevalence of social networks, users are actively interacting with various computers and users, resulting in a huge volume of available data, such as service information, user service ratings, interaction logs and user relationships. Therefore, capturing and analyzing this data via AI techniques will substantially improve the quality of recommendation of Web services to users and businesses. In the following chapter, we will introduce the major techniques that allow us to analyze large volumes of data and thus make relevant decisions for the business development of companies.

Chapter 2

Machine Learning for NLP

"Nothing is particularly hard if you divide it into small jobs".
- Henry Ford

2.1 Introduction

We have seen Machine Learning as a huge trend for the past few years, the reason for this might be the huge amount of data production by applications, the increase of computation power in the past few decades and the development of better algorithms.

Machine Learning is used anywhere from automating dull tasks to offering intelligent insights, industries in every sector try to benefit from it. You may already be using a device that utilizes it. For example, a wearable fitness tracker like *Fitbit* , or an intelligent home assistant like Google Home[13].

Figure 2.1 Shows the subfields of Automation.

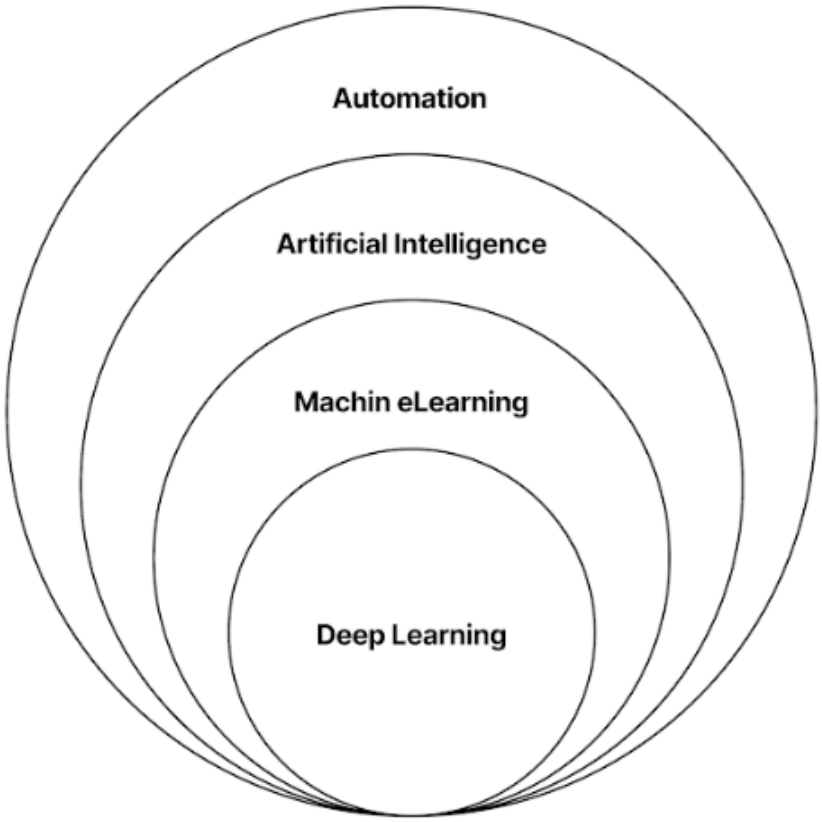

Figure 2.1: Machine Learning

2.2 Data Analysis Techniques:

Simply described, data analysis is the process of examining data in order to discover meaningful information. This is accomplished through the use of analytical and statistical tools to

inspect, clean, convert, and model data[14]. **The most implemented Data Analysis techniques are :**

2.2.1 Simple and multivariate linear regression

2.2.1.1 Simple Linear Regression

An analysis that is suited for a single quantitative explanatory variable and a quantitative outcome. Simple linear regression is the most widely used analysis method when looking at the relationship between a quantitative outcome and a single quantitative explanatory variable. (The "simple" part indicates that we're only looking at one explanatory variable). We frequently have many distinct values of the explanatory variable in linear regression, and we normally assume that values between the observed values of the explanatory variables are also viable values. We assume that the population mean of the outcome and the value of the explanatory variable have a linear relationship. We can express the structural model using the equation if Y is some outcome and x is some explanatory variable.

$$E(Y|x) = \beta 0 + \beta 1x \tag{2.1}$$

where $E()$, or "*expected value of*," denotes a population mean; Yx, or "*Y given x*" denotes that we're looking at the possible values of Y when x is limited to a single number; $\beta 0$ or "*beta zero*" denotes the intercept parameter; and $\beta 1$, or "*beta one*," denotes the slope parameter. The term coefficient refers to any parameter or parameter estimate utilized in an equation to predict Y from x. The name of the explanatory variable, or an abbreviation of it, is frequently substituted for the "1" subscript in $\beta 1$ [15].

2.2.1.2 Multivariate Linear Regression

The majority of statistically analyzed data does not necessarily have one response variable and one explanatory variable. The number of variables can vary based on the investigation in most circumstances. Multivariate regression is used to assess the correlations between various multidimensional variables.

Multivariate regression is a technique for determining the degree to which the independent and dependent variables are linearly connected to one another. Because the vari-

ables are correlated, the relationship is considered to be linear. After applying multivariate regression to the dataset, this method is used to predict the response variable's behavior based on its associated predictor variables. In machine learning, multivariate regression is a supervised algorithm that predicts the behavior of dependent variables and many independent variables[16].

2.2.2 Principal Component Analysis (PCA)

Principal Component Analysis, or **PCA**, is a dimensionality-reduction approach for reducing the dimensionality of large data sets by transforming a large collection of variables into a smaller one that retains the majority of the information in the large set. The loss of precision when reducing the number of variables in a dataset is a trade-off, but the answer to reducing dimensionality is to sacrifice a little precision for simplicity. Because smaller datasets are easier to study and visualize, analyzing data for machine learning algorithms is easier and faster without having to deal with superfluous factors. To summarize, **PCA**'s goal is to reduce the number of variables in a dataset while preserving as much information as possible [17].

2.2.3 Classification

In machine learning, classification is a supervised learning concept that divides a set of data into classes. Speech recognition, face recognition, handwriting recognition, document categorization, and other classification issues are the most common challenges. It could be a problem with multi-class or binary classification. Machine learning algorithms for classification come in a variety of shapes and sizes [18].

Classification algorithms in machine learning employ input training data to predict the likelihood or probability that the data will fall into one of the established categories. Classifying emails into "spam" and "non-spam" categories, as employed by today's main email service providers, is one of the most common uses of classification[19].

Classification is a type of "*pattern recognition*," to put it another way. In this case, the same pattern (`similar number sequences`, `words` or `sentiments`, and so on) is found in subsequent data sets by classification algorithms that were applied to the training data. We can learn how text analysis software can perform tasks such as sentiment analysis,

which classifies unstructured text based on polarity of opinion (*positive, negative, neutral, and the like*)[20].

2.2.4 Clustering

Clustering is the process of dividing a dataset into groups, or clusters. The goal is to divide the data into clusters with very similar points within each cluster and very different points between them. Similar to classification systems, clustering algorithms assign (or predict) a number to each data point, indicating which cluster it belongs to. k-nearest neighbors (**k-NN**) is used to classify new data points into existing clusters, and k-means clustering is used as an unsupervised learning technique to discover discrete groups of data points. Although there are other types of clustering algorithms, these two are the most commonly used in machine learning and data mining[21].

2.2.5 Dimensionality Reduction

A massive amount of data being generated as a result of digitalization in a variety of industries, including healthcare, manufacturing, sales, IoT devices, the Web, and businesses. The algorithms for machine learning are employed. to discover patterns among the data's properties As a result, they can be utilized to create predictions. can be utilized to make executive decisions by medical practitioners and persons in management positions. Not all of them The properties of the created datasets are crucial for the machine learning algorithms to be trained.Some features may be meaningless, while others may have no impact on the prediction's outcome. The strain on machine learning algorithms is reduced by ignoring or deleting these unnecessary or less important attributes.

Linear Discriminant Analysis (***LDA***) and Principal Component Analysis (***PCA***) are two popular dimensionality reduction approaches (**PCA**)[22].

2.2.6 Learning Association Rules

Association rule learning is another key unsupervised data mining method for finding interesting associations (relationships, dependencies) in large sets of data items. The data is saved as transactions, which can be generated by an external process or extracted from re-

lational databases or data warehouses. Association rules are an important data mining tool for extracting knowledge from data because of their good scalability characteristics and the ever-growing size of the accumulated data.

The discovery of interesting associations provides a source of data that businesses frequently use to make decisions. Market-basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, data preprocessing, genomics, and other applications of association rules are just a few examples. Personalization and recommendation systems for browsing web pages (*such as Amazon's recommendations of related/associated books*) and genomic data analysis are two interesting examples[23].

2.3 Categories of Machine Learning

2.3.1 Supervised Learning

By connecting variables to known outcomes and working with labeled datasets, the supervised learning method focuses on learning patterns as shown in Figure 2.2. We can use supervised learning to acquire or produce data from a previous machine learning deployment. Supervised learning is fascinating because it operates in a similar fashion to how humans learn[24].

For example, a supervised algorithm can forecast the market rate for the purchase of a used car by examining the relationship between car attributes (*such as year of production, car brand, mileage, and so on*) and the selling price of other cars sold in the past. The supervised algorithm can work backwards from the ultimate price of other cards sold to discover the relationship between the car's attributes and its value.

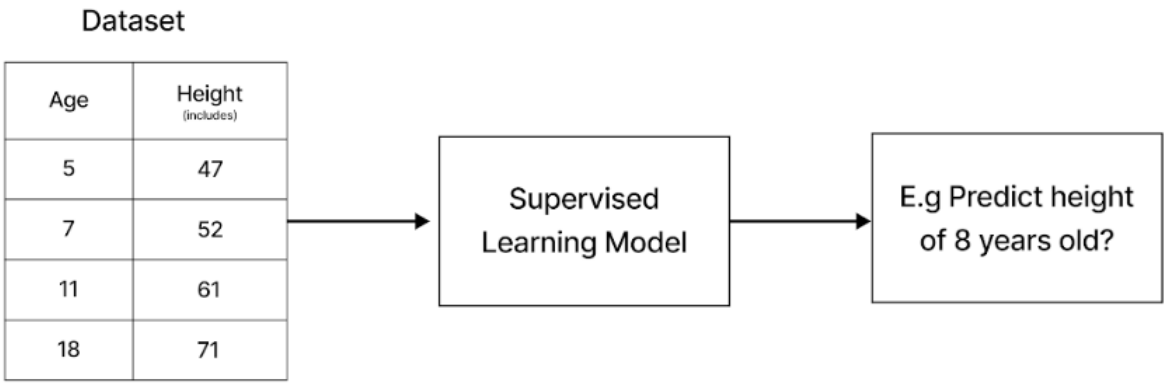

Figure 2.2: Supervised learning example

2.3.2 unsupervised Learning

Unsupervised learning can be used to discover previously unknown patterns in data. In unsupervised learning, the algorithm tries to learn some inherent structure to the data using only unlabeled instances. Unsupervised learning tasks like as clustering and dimensionality reduction are prevalent. Furthermore, we will not classify all variables and data patterns. Instead, the machine must employ unsupervised learning methods to detect hidden patterns and generate labels. Unsupervised learning is exemplified by the k-means clustering technique. This basic algorithm groups data points that have similar characteristics [25].

When a data scientist wants to better understand the data, unsupervised algorithms are usually utilized in an exploratory setting rather than as part of a larger automatic system. Unsupervised algorithms are widely employed as a stage in the preprocessing of supervised algorithms. Learning a new representation of the data can help supervised algorithms perform better or use less memory and time.

2.3.3 Reinforcement Learning

Reinforcement learning is a technique for teaching an autonomous agent that detects and acts in its surroundings to select the best actions to achieve its goals. This broad problem includes learning to manage a mobile robot, optimizing factory operations, and learning to play board games. A trainer may deliver a reward or penalty each time the agent performs an action in its surroundings to signify the desirability of the ensuing state[26].

For instance, consider self-driving cars: Reinforcement learning is utilized for a variety of objectives in self-driving automobiles, including the following. To test **RL** on physical tracks, DeepRacer, an Amazon cloud service, can be used.

Also, *how does it work?* An **AI** agent is taught by issuing orders, and for each action, the agent receives a reward as a reward. The agent's performance improves as a result of these feedbacks.

Reward feedback can be positive or negative, meaning the agent receives a positive reward for each successful activity and a negative reward for each bad action.

2.4 Artificial Neural Networks

An artificial neural network is a data-processing system that shares some of the same performance characteristics as biological neural networks. As generalizations of mathematical models of human cognition or neural biology, artificial neural networks (Figure 2.3) have been developed. based on the following assumptions:

1. *Information processing occurs at many simple elements called neurons.*
2. *Signals are passed between neurons over connection links.*
3. *Each connection link has an associated weight. which. in a typical neural net, multiplies the signal transmitted.*
4. *Each neuron applies an activation function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output signal[27].*

Our brain has billions of neurons that process data in the form of electrical signals. External information/stimuli are received by the dendrites of the neuron, which is processed in the neuron cell body, turned to an output, and transferred through the Axon to the next neuron. The following neuron can accept or reject the signal, depending on its strength. Artificial Neural Networks (**ANNs**) function similarly to biological neural networks. Neurons correspond to nodes, and connections between neurons correspond to weighted edges, therefore they can be thought of as weighted directed networks. The processing element of a neuron receives a large number of signals (*both from other neurons and as input signals from the external world*).

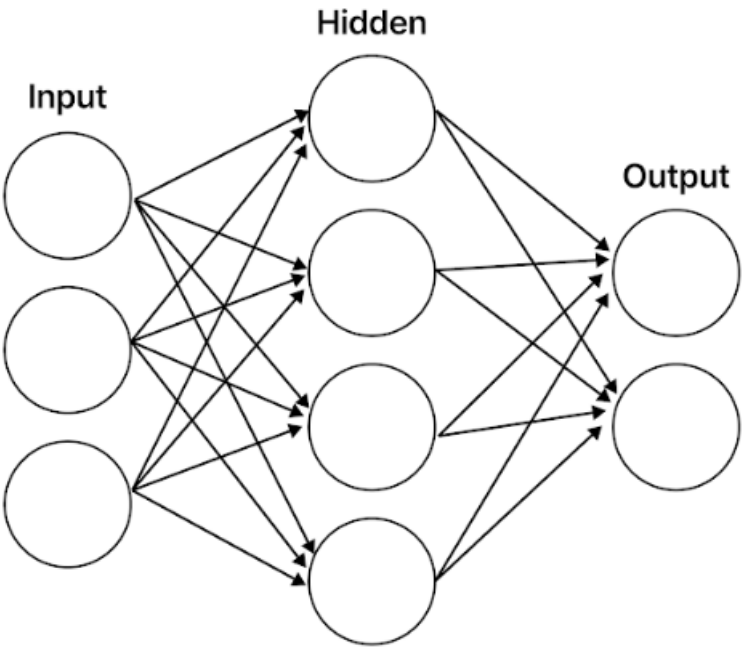

Figure 2.3: Neural Networks

2.5 Deep Learning

In order to extract and manipulate features, deep learning algorithms employ a cascade of several layers of nonlinear processing units. A hierarchy of concepts is formed by the layers, each of which learns numerous levels of representations that correspond to various degrees of abstraction using the output from the preceding layer as input[28].

2.6 Deep Learning for NLP

Deep learning frameworks have advanced rapidly in recent years, similar to **NLP** techniques. There are numerous frameworks, each with its own specialization. **TensorFlow**, **PyTorch**, **Keras**, **MXNet**, **CNTK**, **Chainer**, **Caffe2**, **PaddlePaddle**, and **Matlab** are the most popular deep learning frameworks. 2 As this applies to deep learning, the main component of a modern deep learning framework is efficiency in linear algebra capabilities, with support for CPU and GPU computation (*dedicate hardware like **TPUs*** [Jou16] are becoming increasingly popular as well). All relevant Python frameworks include support for both CPU and GPU. From the ones mentioned previously, we look at the top deep learning frameworks and give a brief description of them. To do so, we compare the Google Trends for each framework

and concentrate on the top 3.3. **Keras** is the most popular framework (*globally*), followed by TensorFlow, and finally **PyTorch** [29].

2.6.1 Deep learning Models for NLP

2.6.1.1 Transformers

A transformer is a deep learning model that uses the self-attention process and weights the importance of each component of the input data differently. It is largely utilized in the disciplines of computer vision and natural language processing (NLP) (CV). Transformators are used in activities like translation and text summarizing to handle sequential input data, such as natural language. when the entire input is processed at once by transformers. Any place in the input sequence can have context thanks to the attention mechanism[30].

2.6.1.2 Bert

BERT is a machine learning framework for natural language processing that is open source (**NLP**). ***BERT*** is a program that uses surrounding text to help computers understand the meaning of ambiguous language in text. The *BERT* framework was trained on Wikipedia text and can be fine-tuned using question and answer datasets.

Language models could previously only read text input in one of two ways: *left-to-right* or *right-to-left*, but not both at the same time. *BERT* is unique in that it can read in both directions at the same time. Bidirectionality is a capability made possible by the introduction of Transformers.

BERT is pre-trained on two different but related *NLP* tasks using this bidirectional capability: Masked Language Modeling and Next Sentence Prediction.

Only an unlabeled, plain text corpus was used to train *BERT* (namely the entirety of the English Wikipedia, and the Brown Corpus). Even as it is used in practical applications, it continues to learn unsupervised from unlabeled text and improve (ie `Google search`). Its pre-training serves as a foundation of "knowledge" on which to build. BERT can then be fine-tuned to a user's specifications and adapt to the ever-growing body of searchable content and queries. Transfer learning is the term for this process.

To prevent the word in focus from "seeing itself" — *that is, having a fixed meaning independent of its context* — *BERT* employs a masked language modeling technique. *BERT* is then forced to identify the masked word solely on the basis of its context. Words in BERT are defined by their context rather than by a pre-determined identity [31].

2.6.1.3 GCN

GCNs are a type of neural network that can be used to learn from graphs. They're so powerful that even a *2-layer GCN* started at random can produce useful node feature representations in networks. or we can also say A graph convolutional network **(GCN)** is a type of neural network that works with graphs. Given a graph $G = (V, E)$, a **GCN** takes as input

- "An input feature matrix $N \times F^0$ feature matrix, X, where N is the number of nodes and F^0 is the number of input features for each node"
- "An $N \times N$ matrix representation of the graph structure such as the adjacency matrix A of G [32]."

2.6.1.4 Bert-GCN

We use a *BERT-style* model to initialize representations for document nodes in a text graph in the proposed **BertGCN** model *(e.g., BERT, RoBERTa). GCN* takes these representations as inputs. The outputs of *GCN* are treated as final representations for document nodes and sent to the $softmax$ classifier for predictions, after which the document representations will be iteratively updated based on the graph structures. We can use the complementary strengths of pretrained models and graph models in this way [33]. "Specifically, we construct a heterogeneous graph containing both word nodes and document nodes following **TextGCN** [34]. We define word-document edges and word-word edges based on the term frequency-inverse document frequency ($TF - IDF$) and positive point-wise mutual information ($PPMI$), respectively. The weight of an edge between two nodes i and j is defined as:" [33]

$$A_{i,j} = \begin{cases} \text{PPMI}(i,j), & i, j \text{ are words and } i \neq j \\ \text{TF} - \text{IDF}(i,j), & i \text{ is document , } j \text{ is word} \\ 1, & i = j \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

2.7 Conclusion

In this chapter, we discussed several automatic learning strategies for text classification. We are particularly interested in the supervised learning . In the following chapter, we introduce the conceptual study and implementation steps of our proposed Web Service discovery approach, as well as the various stages of implementation of our proposed solution.

Chapter 3

Building a social Network of Web Services to improve WS discovery process

« You can't control what you can't measure. »

— Tom DeMarco

Problems and Contributions

3.1 Introduction

The web is a method of viewing and sharing data over the internet. The emergence of web services led to a great development in the Internet, as it greatly helped users in terms of facilitating work and the development . As a result, finding a suitable service for users' complex needs is crucial, Service engineers frequently struggle to find the appropriate Web services to fulfil users' requests , The difficult thing that remains is how to find these services and access them easily, especially with The increasing number of cloud services and **APIs** deployed on the web . As a result, users must sift through a plethora of services before deciding which ones best meet their requirements. When it comes to discovering inter-operable services that can be combined or replaced, the selection task takes on a new level of complexity.
Despite the growing interest in incorporating social elements, modern businesses still face challenges in terms of agility and competitiveness, as they must respond to market changes and unexpected opportunities. Web applications must be inter-operable in order to meet this challenge [35].
Several studies(*Related work*)have introduced recommendation systems that help users find the services that are most relevant to their needs.Relying on these studies and using deep learning to solve this problem and help users. but older studies depended on the ontology which its disadvantages exceeded the benefits and couldn't solve the problem of making a social network as it requires massive efforts to be done properly. Furthermore the implicit properties of web services have been overlooked in the search for ways to improve web service clustering accuracy. And

3.2 Related work

In this paper [36] , Abdelouahab et al. are interested in **svm**, The goal is to improve the accuracy of *SVM* by combining this classifier with a stochastic local search as a feature selection method, where they selected features using the stochastic local search (**SLS**) metaheuristic and then categorize them using **SVM**. They put their technique to the test on the *QWS* Dataset. According to their tests, their method produces better results than **SVM** [37]. where they did the following points :

- *"The search space **S** (the set of solutions) is defined by the subsets of relevant attributes*

(selected attributes). "

- *"The objective function (fitness) f **S** —> **IR** is such that for any Is **E S**, / (**s**) represents the Accuracy (classification rate) of a classifier built with the subset of attributes s."*

- *"The neighborhood of a solution: this concept indicates the possible move in the search space **S**. The exploration of search space is to browse a path that goes with every step of a solution to a neighboring solution. The objective function of our problem is to maximize the fitness. We precise that the aim of this work is to optimize the Accuracy (classification rate) of the SVM classifier by selecting a subset of the best attributes (relevant attributes). The basic idea is to improve iteratively an existing solution by exploring a neighborhood"* [36].

Rozina et al. in [38], The majority vote based ensemble model is used in their study to classify web services using *WSDL* features. and they used a combination of three techniques (**Naïve Bayes**, **SVM**, and **Decision Tree J48**) to make up their ensemble classifier. Preprocessing is used to refine the information that has been downloaded.and every trained classifier classifies the web service into one category. then using a majority voting approach, the output of all three classifiers is combined. and the output of the testing web service will be the category with the most votes.

They tested the framework's accuracy using a publicly available dataset called "**WSExpress**: **Dataset 4**, **WSDL dataset**" [39], which contains **3738** real-world web services.
Where they had to train All of the classifiers on pre-classified web service datasets. The accuracy of the models is determined using tenfold cross validation. For all identified categories, this approach could yield the highest precision and recall. This increase in precision can be attributed to two main factors that are *"enhanced pre-processing with focused feature selection and ensemble classification based on the majority"* [38].

This method specifies which attributes and documentation contents are dependent on *WSDL content*. where They combine *the service name*, service documentation, *WSDL types*, *messages*, and *ports* to form the WSDL contents. Says Abdelwahab et al. in [40] they named their approach with name **SoCo** (for `Social Composer`) that offers two components:

- A social knowledge extraction and modeling component, in a social network. Information that can be used in the context of service composition is referred to as knowledge.

- The recommendation manager where this knowledge is captured in order to define and build a set of rules and actions that describe composition behavior in order to improve and customize a composition for a specific user.

In this regard, they discuss two different approaches to forming social bonds:

1. **Explicit**: The user is given the option of declaring a relationship with another user. This is akin to today's social media.
2. **Implicit**: A social relationship is inferred based on the activities of different users, such as when one person uses services created by another.

Finally, we can state that they have created a novel concept that uses data from a user's social networks to dynamically recommend service composition strategies to him. The end-social user's network knowledge, the expertise of social network members, and contextual social proximity between people, including trust.
Xiaoqin Xie et al. in [41] designed a prototype system for service composition . The semantic extraction and social network construction module extract the trust semantic and construct the **SCSN** and store it by Social network storage module , and doing the following points :

- "*definition and presentation of trust concepts and trust properties*".
- "*extraction of association relation such as using, providing, evaluating and recommending relationships in service composition*" [41].

According to the Guobing Zou et al. in [42] has been suggested a framework named it "*DeepWSC: A novel framework with deep neural network for web service clustering*" ,where it represents a web service with a semantic matrix and depends on convolutional neural network (**RCNN**) to get better result, this based done by a conventional clustering algorithm, and this to obtain implicit contextual information of web services. we can say that they depend on these steps as below:

- *"We propose a novel framework with deep neural network, **DeepWSC**, for web service clustering. It fundamentally boosts service discovery, composition and recommendation by effectively providing the desired web services."*

- *"We propose an expansion strategy for more accurately extracting features of web services, which integrates an improved recurrent convolutional neural network (**RCNN**), an augmented probabilistic topic model (**WE-LDA**) and a trained word embedding model (**Word2vec**)".*

- *"Extensive experiments are conducted on large number of real-world web services crawled from ProgrammableWeb, which demonstrate that DeepWSC outperforms state-ofthe-art approaches for web service clustering"* [42].

As we can see in the results that the DeepWSC could excel on the previous researches on many evaluation metrics.

3.3 Motivating Scenario

Let's consider a software engineer at a software development company where he is working on a mobile application that tells the weather depending on the user's location, we consider the worst case scenario that is a close deadline he needs to find two crucial web services to his application, the first one determines the location of the app user and the second one determines the weather of said location, it's already hard enough to find only one web service as he has got to go through a plethora of web services in any existing platform (programmableweb.com, apilist.fun...) to find the exact one that meets all his needs and to find two of them that work seamlessly together is impossible to do in a short time as there is no tool or service to make sure that they can collaborate.

A better scenario would be having a fast and reliable source of information that helps him discover web services such as a Social Network for Web Services that not only brings out web services that are relevant to the whole community that needed it before but also bring out web services that could be reliable but new which makes the chance of them appearing in a search very low on the already existing platforms, also one of the main features of the social network is bringing out the web services that successfully collaborated with a said web service and that would save so much time by not only finding a web service for weather but also web services that work seemlessly with it such as location ones and many others.

3.4 Approach Overview

In this research we seek to find the most efficient way to discover web services and even though there are many works on this only a few of them have implemented the social concept and none of them worked on a real life example and from these two ideas have emerged our approach of a social network for web services discovery. But before we say anything about that, what is a social network and why is it relevant in our case?

3.4.1 Why online Social Networks

Online Social Networks (OSNs) have gained popularity as a result of their astounding success, as demonstrated by the sharp rise in user activity and subscriptions. Over 55 million people are currently using Facebook.com, a popular OSN, and 250,000 new users sign up on average every day.

Facebook in 2007 The most popular OSN in Germany, StudiVZ.netin 2007 saw a significant surge, after its debut in 2005, with more than 3 million users currently subscribed. Alexa.com in 2007 reports that four OSN websites (MySpace.com, Facebook.com, Orkut.com and Hi5.com) are in the top ten of the global traffic ranking[43].

And therefore it would be a great idea to use the OSN in our advantage since we believe that it will out-stand the conventional methods (as shown in the comparison table 3.1). We have picked Programmanbleweb.com as a model for the conventional method since it's the most used platform for web services discovery.

Feature	programmableweb.com	WS Social Network
Post, Delete, Search	X	X
Feedback		X
Competition		X
Simple Colaboration (Mashups)	X	X
Complex Colaboration (Based on the relations created by users feedback)		X

Table 3.1: Comparison between programmableweb.com and WS Social Network

Our goal then is to create and use Deep Learning models to classify Web-Services based on their categories and that would be a crucial part of the making of our own Social Network

3.5 The Proposed Architecture

Our goal is to build and initialize our Social network that will make it usable and most importantly updatable in a way that emerges the best web services regardless of how many users it has or its competitors. in this section we will explain the process of creation of our Social Network. As shown in figure 3.1 below Deep Learning is a very important and indispensable to our architecture as it is used for classification and measuring the similarity between web services.

For the first part we used deep learning models to be able to classify web services to their category that will result in the creation of an adjacency matrix that will be used in the social network.

The second part is using our formulas **Colab(**S_i, S_j**)** and **Comp(**S_i, S_j**)** to generate the related adjacency matrices that will be also used in the social network (the formulas will be thoroughly described in the next section).

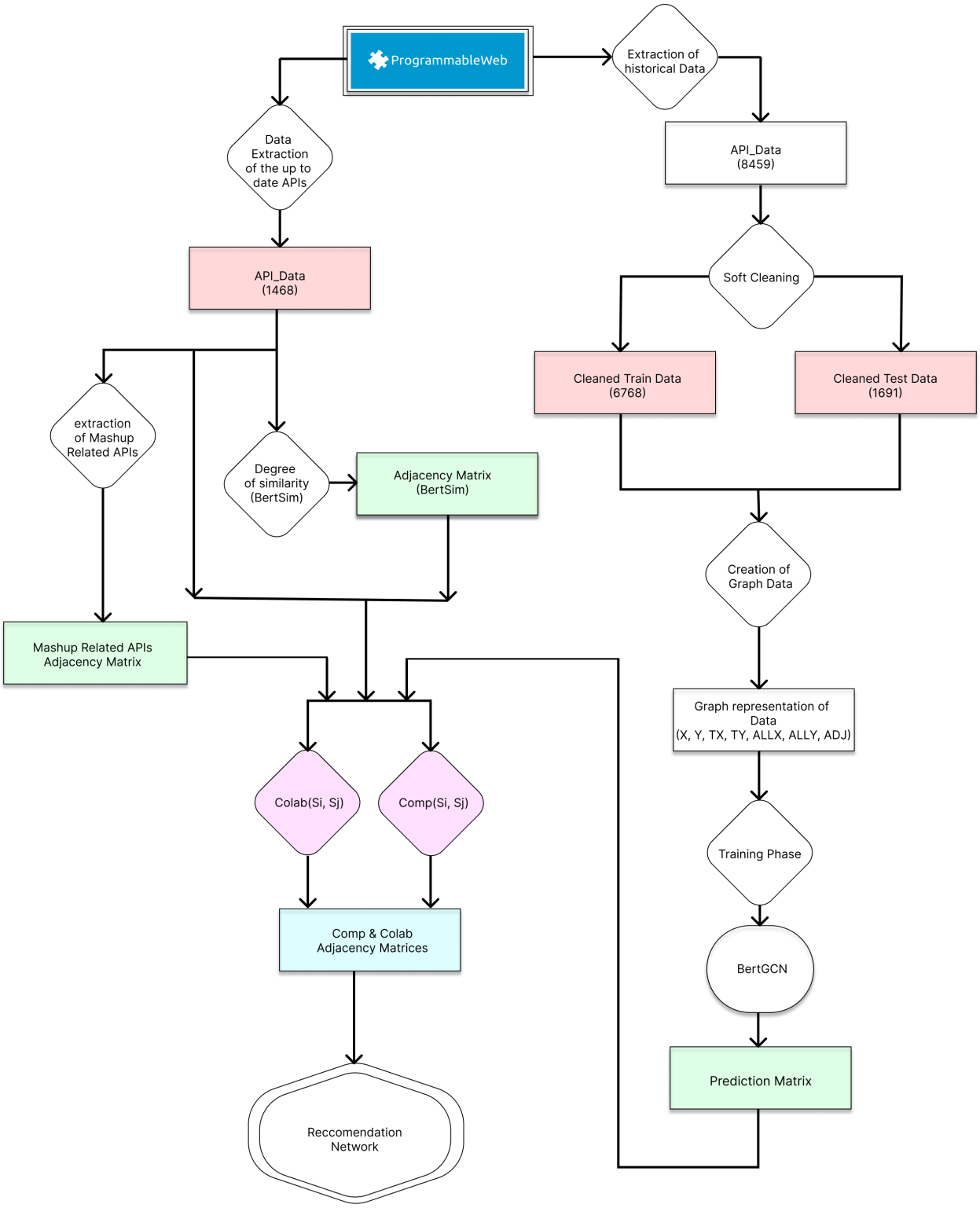

Figure 3.1: Design of our proposed architecture

3.5.1 Data Gathering for the Classification model

In our approach of recommending web services we have used two real data sources collected from the most famous collection of web services that is programmableweb.com. The first one was already provided, cleaned and ready to use from the "DeepWSC: A Novel Framework with Deep Neural Network for Web Service Clustering" paper which contains 8,459 web services conducted on the top 20 categories (table 3.2), this data will be used in the training process of the BertGCN model [42].

Class Name	#Services	Class Name	#Services
Financial	757	Security	312
Messaging	591	Reference	304
ECommerce	553	Email	299
Payments	553	Search	290
Social	510	Travel	294
Enterprise	509	Video	281
Mapping	429	Education	277
Government	371	Advertising	274
Science	357	Transportation	269

Table 3.2: Distribution of the web services in 20 categories

3.5.2 Data Gathering for the Proposed Equations

We have developed a tool using python and BeautifulSoup library to automatically crawl all the services (atomic and mashups) published in the same collection (programmableweb.com) and get all the attributes of the service. The repertoire indicates if the service is a mashup or not. In the case where it's an atomic service it stores all the attributes of the service and in the case where it's a mashup it stores all services associated in the mashup with some important attributes of the mashup.

At the end of the process of the collection of the data we ended up with two files <API-DATA> that contained 21834 atomic web services and <MASHUP-DATA> that contained 6425 mashups, we proceeded then to make a third file <API-MASHUP-COMBINED> that contained 1468 web service that is the intersection of the two files which means all the web services and the mashup they belonged to as shown in the figure 3.2, this dataset will be divided into two

parts each part will represent the services at an instance T_n (APIs published before 2011) and T_{n+1} (APIs published after 2011) and they will be used in the proposed equations hence the initialization of the Social Network.

	A	B	C	D	E	F	G
1	Title	Categories	Api_Original_Desc	Url	Versions	Followers	Mashup
2	MediaTempl	['Hosting']	Web hosting compa The API uses RES	/api/m	[{'version_	7	(mt) Server Monitor
3	MediaFire A	['File Sharing',	MediaFire is a clou	/api/m	[{'version_	159	[t]Space
4	Google Stor	['Storage', 'Pla	Google Storage for	/api/g	[{'version_	55	[t]Space
5	Telegram Bc	['Bots', 'Cloud',	Telegram, the clouc	/api/te	[{'version_	114	@trafficRobot
6	LiveConnect	['Calendars', 'E	LiveConnect lets de	/api/li	[{'version_	12	#docs ("pound docs")
7	Embedly AP	['Video', 'Photo	Get embed code vi	/api/e	[{'version_	23	#tweet-a-lot
8	Google Gma	['Email', 'OAutl	Google Gmail now	/api/g	[{'version_	110	0boxer
9	Impact Radi	['Tools', 'Adver	(Note Impact Radiu	/api/in	[{'version_	23	1dollarthings.com
10	Google App	['Tools', 'Cloud	[This API is no long Use the Google Ap	/api/g	[{'version_	288	2lingual Google Search

Figure 3.2: API-MASHUP-COMBINED.csv Data Sample

The file <API-DATA> contain a number of attributes, we only kept a small categorical attributes, «Title, Categories, Api-Original-Desc, Url, Versions, Followers» which are the most relevant attributes to our equations.

After the vertical cleaning we do a horizontal cleaning by taking the intersection of the files <API-DATA> and <MASHUP-DATA> as shown in the algorithm bellow.

Algorithm 1 Algorithm That create <API-MASHUP-COMBINED> dataset

```
Services = listOfServices()
Mashups = listOfMashups()
Nb_Ser = Length(Services) length of web services
Nb_Mush = Length(Mushups)                          ▷ length of mashups
R_APIs = Mushups(Related_APIs)                     ▷ Related APIs in mashups
for each service i in Services do
    NewColumn[i] = −1
    Name = getApiName(i)
    for each mashup j in Mushups do
        R_APIs = Mushups[i](Related_APIs)
        if Name in R_APIs then
            mashup = Mashups(j)
            NewColumn[i] = mashup
for each mashup j in NewColumn do
    for each service i in Services do
        if NewColumn[j] == −1 then
            delete Services[i]
            delete NewColumn[i]
```

Description of Algorithm 1

First we list all the API names and the length of list of APIs as well as the Mashup's then we loop through each service we guard the name and we compare it through another loop to each related API in each mashup that will tell us if the API exists in the mashup, if it does then we keep it's name in the new column that will be added later to the Services Matrix otherwise we fill it with a flag "-1" so we can delete it later along with the service that doesn't belong to any mashup.

3.5.3 Our Proposed equations

As shown in the figure 3.1 the **Colab(**S_i, S_j**)** and **Comp(**S_i, S_j**)** equations are highly inspired from the "Universal Gravitation Equation" :

$$F = g \times \frac{m1 \times m2}{r^2}$$

and they are an important brick in the creation of the Social Network, in this section we will thoroughly explain it.

$$Comp(S_i, S_j) = BERT_{sim}(S_i, S_j) + (c + \mu) \times g \times \frac{followers(S_i)\|S_i\| \times followers(S_j)\|S_j\|}{\|S_i, S_j\|^2}$$

$$Colab(S_i, S_j) = (1 - BERT_{sim}(S_i, S_j)) + (c + \mu) \times g \times \frac{followers(S_i)\|S_i\| \times followers(S_j)\|S_j\|}{\|S_i, S_j\|^2}$$

$$followers(S_i) = 1 + \frac{total\_followers\_S_i}{total\_followers}$$

Colab(S_i, S_j**)** is the equation that calculates rate of Collaboration between a given API i and an API j and **Comp(**S_i, S_j**)** is the equation that calculates rate of Competition between a given API i and an API j, and since they are inspired from the "Universal Gravitation Equation" $followers(S_i) \times ||S_i||$ represents the mass of the API and $||S_i, S_j||$ represents the distance between two APIs both of these equations use the help of some other functions and parameters:

- **Google Doc2Vec** is a model used to create a vectorized representation of a group of words taken collectively as a single unit, this will help us to represent the descriptions of the APIs mathematically so we can perform many operations on them such as calculate the **magnitude** of an API and the **distance** between two APIs.

- $BERT_{sim}(S_i, S_j)$ is a function that uses a Deep Learning model (Bert) to calculate percentage of the similarity between two APIs (S_i and S_j) using their the descriptions, similar APIs means they compete against each other and being less similar increases the chances of them collaborating together.

- **followers(**S_i**)** is the ratio an API's followers divided by the sum of all users, we add 1 to give it more impact on the **Colab(**S_i, S_j**)** or **Comp(**S_i, S_j**)** equations.

- $||S_i, S_j||$ is the distance between two given apis, it's easily calculated since the descriptions of the APIs are represented as vectors using Doc2Vec.

- $||S_i||$ is the magnitude of a given API it is also easily calculated as long as the descriptions are vectorized.

- c is a parameter that can affect the state of the network it's value depends on whether the APIs i and j are in the same class predicted by the classification model.

- μ is also a parameter that can affect the state of the network and it's value depends on whether the APIs belong to the same Mashup[1]

- g is a parameterizable constant that represents the gravitational force between two APIs.

3.6 Experimental Study

In this section we will go through the process of training our classification models, using the models and the gathered data to create the adjacency matrices and discussion of our results.

[1]Mashup : is a collection of APIs made in programmableweb.com

3.6.1 The Classification Model

Before we go ahead with the classification we will apply a soft cleaning on the gathered data, transform it so it fits our used models and discuss the evaluation metrics of our models.

3.6.1.1 Extract Actions from Web Services Descriptions

Another problem we faced is that since the majority of descriptions are short, it is hard to have a significant number of verbs to work with when using these models. To overcome this, we used, in this step, the ***'Extract-actions***' algorithm, to extract what we qualify as actions (Words of service actions).

Algorithm 2 Extract Actions

```
    Parameters: Web service textual description
    Result: List of Actions[ ]
 1: function SGet_Action(Sentence)
 2:     if (Item is Verb) and (has dobj) then
 3:         concatenate the verb with dobj ;
 4:         return as Action;
 5:     else
 6:         if (Item is Verb) and (has conjuction) then
 7:             go to the conjunction item;
 8:             once you reach the Noun: concatenate the Verb with the Noun;
 9:             if (this Noun has conjuction) then
10:                 concatenate also the Verb with the next Noun;
11:         return Actions;
12:
13: procedure Get_Action(Sentence)
14:     if (Item is Verb) and (has xcomp) then
15:         go to the xcom node;
16:         concatenate the Verb with each action of SGet_Action(xcomp);
17:         Add all as Actions to the list of Actions[ ];
18:     else
19:         Add all Actions of SGet_Action(Sentence) to the list of Actions[ ];
20:
21: for each sentence in Web service textual description do
22:     Get_Action(Sentence);
```

Where **xcomp** is *An open clausal complement* and **dobj** represents *the direct object.* The

algorithm is illustrated by the following examples:

Example 1:

- Description: Service X **translates** documents.
- Actions: [**translates** documents].

Example 2:

- Description: Service X **translates** paragraphs and documents.
- Actions: [**translates** paragraphs, **translates** documents].

Example 3:

- Description: Service X uses FacebookApi to subscribe users.
- Actions: [**uses** facebookApi, **uses subscribe** users].

3.6.1.2 Data Cleaning

The attribute «description» designates the description of the web services that was written by the authors, frequently used words could be an advantage in the NLP process but that doesn't apply to stop words such as «the», «for» and «you» as they are not significant to the subject of the Web Service and therefore we have to do some sort of cleaning to the description text to only leave the most significant words in it. figures 3.4 and 3.5 show the difference between a cleaned description vs a non-cleaned one.

```
LinkedIn is the world's largest business social networking hub.
Launched in 2003, LinkedIn has millions of users and is
implemented in over 200 countries. One purpose of the site is
to allow registered users to maintain a list of contact details
of people with whom they have some level of relationship, called
Connections. Users can invite anyone (whether a site user or
not) to become a connection. \n\LinkedIn actually provides 2
APIs:\n-The JavaScript API is a rich client library enabling
you to build dynamic applications in the web browser. Use OAuth
2 to easily authorize users via the "Sign in with LinkedIn"
button, access LinkedIn data with native objects, and interact
with Plugins.\n\n-The REST API provides a simple, consistent
representation of people, companies, jobs, and the interactions
and relationships between them. Our query language lets you
read data in XML and JSON at the granularity and aggregation
that you choose. Use OAuth 1.0a to authorize users and begin
making REST API calls using any programming language.
```

Figure 3.4: Description of a Web Service before cleaning

```
linkedin worlds largest business social networking hub launched
2003 linkedin millions users implemented 200 countries one
purpose site allow registered users maintain list contact
details people level relationship called connections users
invite anyone whether site user become connection linkedin
actually provides 2 apis javascript api rich client library
enabling build dynamic applications web browser use oauth 2
easily authorize users via sign linkedin button access linkedin
data native objects interact plugins rest api provides simple
consistent representation people companies jobs interactions
relationships query language let read data xml json granularity
aggregation choose use oauth 1.0a authorize users begin making
rest api calls using programming language
```

Figure 3.5: Description of a Web Service after cleaning

3.6.1.3 Data Representation

Our approach counts a lot on the performance of the machine learning model and that's why we chose one of the best algorithms in NLP which is a combination between the famous google's BERT and Graph Convolutional Networks (GCN) but in order to be able to use them we need to do some data graph representation as the GCN module takes in graphs as input, what we have as an initial input is a text that needs to be transformed into a graph representation.

Specifically, we make a heterogeneous graph that contains both word and documents nodes, we define word-document and word-word edges based on the TF-IDF[2] and the PPMI[3][42].

After the transformation of our data set (figure) we now have x, y, tx, ty, allx, ally, adj files that has the following representations :

- **x** is the feature vectors of the training instances as scipy.sparse.csr.csr matrix object.
- **tx** is the feature vectors of the test instances as scipy.sparse.csr.csr matrix object.
- **allx** is the feature vectors of both labeled and unlabeled training instances (a superset of ind.programmableweb.x) as scipy.sparse.csr.csr matrix object.
- **y** is the one-hot labels of the labeled training instances as numpy.ndarray object.
- **ty** is the one-hot labels of the test instances as numpy.ndarray object.
- **ally** is the labels for instances in ind.programmableweb.allx as numpy.ndarray object.
- **adj** is adjacency matrix that contains TF-IDF(i,j) and PPME(i,j).

3.6.1.4 Evaluation Metrics

We will evaluate the performance of our models using Purity, NMI, Precision and F1-Score.

- **Purity :** This index compares each cluster created by the clustering algorithm to the ground-truth one with the greatest overlap and attempts to determine the proportion of "corrected" clustered samples, resulting in the formula below:

$$Purity(B, A) = \frac{1}{n} \sum_{k} \underset{r}{max} |B_k \cap A_r|$$

- **NMI :** The Normalized Mutual Information (NMI) score is a normalization of the Mutual Information (MI) score that scales the findings between 0 (no mutual information) and 1 (complete mutual information) (perfect correlation).

[2]TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.

[3]PPMI (Positive Point-wise mutual information) score tells us how much more/less likely the co-occurrence is than if the words were independent.

- **Recall :** It is the precise number of positive results divided by the total number of relevant samples.

$$Recall = \frac{truepositives}{truepositives + falsenegatives}$$

- **F1-Score :** The F1 score is a balanced average of precision and recall. The F1 score range is [0, 1]. It indicates the precision of your classifier (whether it classifies correctly) as well as its robustness (There is a significant number of cases.). A high precision but a low recovery rate results in a high precision but a large number of difficult-to-classify cases. The higher the F1 score, the better the performance of our model. It may be expressed mathematically as follows:

$$F1 - Score = \frac{1}{\frac{1}{2}(\frac{1}{recall} + \frac{1}{precision})}$$

3.6.2 Classification Results

3.6.2.1 Justification of the BertGCN Choice of Model

BertGCN is a very Strong model that was trained by us to be capable of classifying Web Services based on their descriptions to their categories this will be crucial to us as the classified data will create two adjacency matrices, the first one is based on the categories of the web service and the second one is based on the mashup that the web service belongs to. These adjacency matrices will be used further in our proposed equations.

We specifically chose BertGCN because of the lack of data, as will be shown in results BertGCN can give a decent accuracy even with small amounts of data.

We have used the the Combined Bert and GCN algorithms to obtain a classification model. it was then trained on two different datasets, the first one contains only the descriptions of the APIs and the second one has the same descriptions concatenated with the actions (verbs) used in the same description. The model has been tweaked and optimized to the following parameters:

- Max_Length = 433

- Batch_size = 16
- Lambda = 0.6 (the factor balancing BERT and GCN prediction)
- Epochs = 8
- GCN layers = 2
- Dimension of a Hidden Layer in GCN = 100
- Dropout = 0.5
- GCN Learning Rate = 6e-7
- Bert Learning Rate = 9e-6
- Bert Model = "bert-base-uncased"

After many executions and a lot of tweaking to our classification models we have found the best results we could, we have evaluated the classification results according to four widely used measures (Purity, NMI, Recall, F1-Score) the table 3.3 shows the performance of all competing methods, we will compare this result with the results of paper [42] and paper [44] as well as some of the unpublished works of our colleagues which has already surpassed the results of the state of the art:

Methods	Purity	NMI	Recall	F1-Score
DeepWSC (RCNN, WE-LDA)	0.5708	0.4856	0.3821	0.3969
DeepWSC (RCNN, WE-LDA, Heuristics)	0.6379	0.5273	0.4186	0.4356
RCNN	0.6438	0.5704	0.6438	0.6247
RCNN_Merged	0.6595	0.5795	0.6588	0.6512
BERT	0.7746	0.6924	0.7746	0.7712
BERT_Merged	0,7785	0,6931	0,7786	0,7759
BertGCN (Api_original_desc)	**0,7843**	**0,7028**	**0,7843**	**0,7770**
BertGCN (Api_original_desc + actions)	**0,7848**	**0,7051**	**0,7849**	**0,7774**

Table 3.3: Performance comparisons of web service Classification among competing methods

With our BertGCN(api_original_desc) model, we found an improvement with an average of 55.49% on all mesures compared to the classification results of the DeepWSC model (RCNN, WE-LDA, Heuristics) in the paper [44]. Then, once we integrated the actions in the data graph our BertGCN (api_original_desc + actions) model allowed us to have an improvement compared to the classification results of the BertGCN(api_original_desc) model with 0.34%,

we also compared it to some unpublished work of our colleagues and as shown in the [table] above where the BertGCN (api_original_desc + actions) has shown an improvement of 1.44% compared to the BERT model and 1.32% compared to the BERT_Merged model.

Discussing the Results

We have chosen the "BertGCN (api_original_desc + actions)" to create the adjacency matrices since it gave the best results compared to all works.

3.6.3 Creating the Adjacency Matrices

In this section we will discuss the creation of the adjacency matrices using our proposed equations.

3.6.3.1 API-API Adjacency matrix (Predicted Category)

After Predicting the category of the gathered Datasets that we crawled using BertGCN we went ahead and made the adjacency matrix of the APIs that has the same predicted category as shown in the figure 3.6 below this matrice will be used in our proposed equations.

```
[ ]  #creating api-api adjacency matrix

     api_api_adj = np.zeros((len(df),len(df)))

     for i in range(0, len(api_api_adj)):
       for j in range(0, len(api_api_adj)):
         if df.predicted_cat[i] == df.predicted_cat[j]:
           api_api_adj[i][j] = 1
```

```
     api_api_adj
```

```
array([[1., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 1.],
       [0., 0., 1., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 1., 0., ..., 0., 0., 1.]])
```

```
[ ]  #saving it as a npy file for later use
     np.save("api-api-adj-matrix.npy", api_api_adj)
```

Figure 3.6: Code Snippet of the creation of the Predicted Category APIs Adjacency matrix

3.6.3.2 API-API Adjacency matrix (Mashup)

With the help of this same Dataset that we crawled that also had the name of the Mashups we were able to create this adjacency matrix of the APIs that belong to the same mashup as shown in the figure 3.7 below this matrice will be used in our proposed equations.

```
[ ]  #creating api-api-mashup adjacency matrix

     api_api_mashup_adj = np.zeros((len(df),len(df)))

     for i in range(0, len(api_api_mashup_adj)):
       for j in range(0, len(api_api_mashup_adj)):
         if df.mashup[i] == df.mashup[j]:
           api_api_mashup_adj[i][j] = 1
```

```
[ ]  api_api_mashup_adj

     array([[1., 0., 0., ..., 0., 0., 0.],
            [0., 1., 0., ..., 0., 0., 0.],
            [0., 0., 1., ..., 0., 0., 0.],
            ...,
            [0., 0., 0., ..., 1., 0., 0.],
            [0., 0., 0., ..., 0., 1., 0.],
            [0., 0., 0., ..., 0., 0., 1.]])
```

```
[ ]  #saving it as a npy file for later use
     np.save("api-api-mashup-adj-matrix.npy", api_api_mashup_adj)
```

Figure 3.7: Code Snippet of the creation of the Mashup related APIs Adjacency matrix

3.6.3.3 API-API Adjacency matrix (Comp(S_i, S_j)

With the help of the number of APIs followers in the new Dataset, BertSIM and both of the previous adjacency matrices we were able to create this function that calculates the degree of competition between two APIs and create an APIs adjacency matrix as shown in the figure 3.8 below, this method will aslo be used to create the collaboration adjacency matrix using the Colab(S_i, S_j) equation.

```
#function that calculates the value of competition between two given APIs
def competition(i,j):
  global alpha, mu, g
  return (sim_matrix[i][j] + ((alpha + mu) * g * (( followers(i) * magnitude(i) + followers(j) * magnitude(j) ) / 2)))
```

```
print(competition(1,2))
```

```
0.9874825089978229
```

```
#initializing the network with 0 then filling it using the competition function
network_comp =  np.zeros((len(df),len(df)))

for i in range(0, len(network_comp)):
  for j in range(0, len(network_comp)):
    network_comp[i][j] = competition(i,j)
```

```
#saving it as a npy file for later use
np.save("competition-network.npy", network_comp)
```

Figure 3.8: Code Snippet of the creation of competitive APIs Adjacency matrix

3.6.4 The Evolution of the Social Network

To further more showcase the evolution of our social network and compare it to the conventional methods we represented the communities of the APIs from the collected datasest to show the development of the communities. figure 3.9 shows the development of the communities for in two instances of time and figure 3.10 and 3.11 show the development of the communities in our social network in the same instances of time.

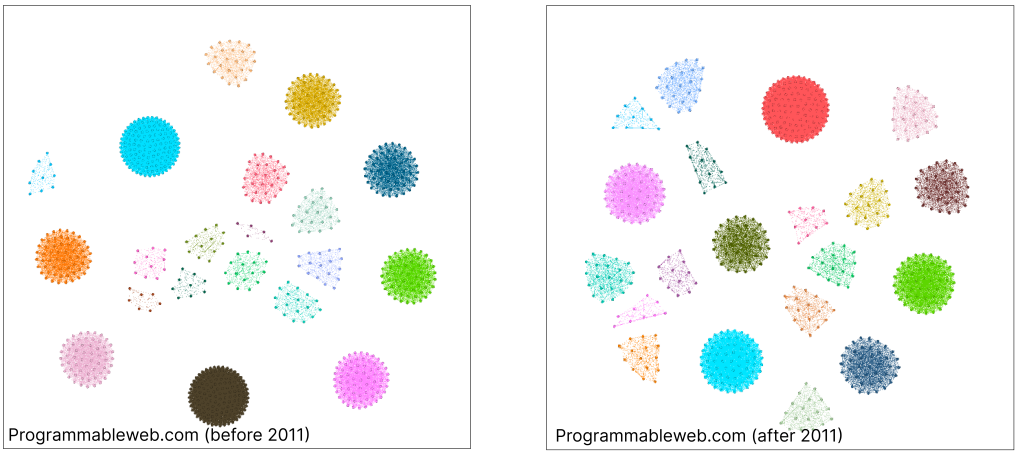

Figure 3.9: The state of the classes in the programmableweb.com repository at the instance T_n and T_{n+1}

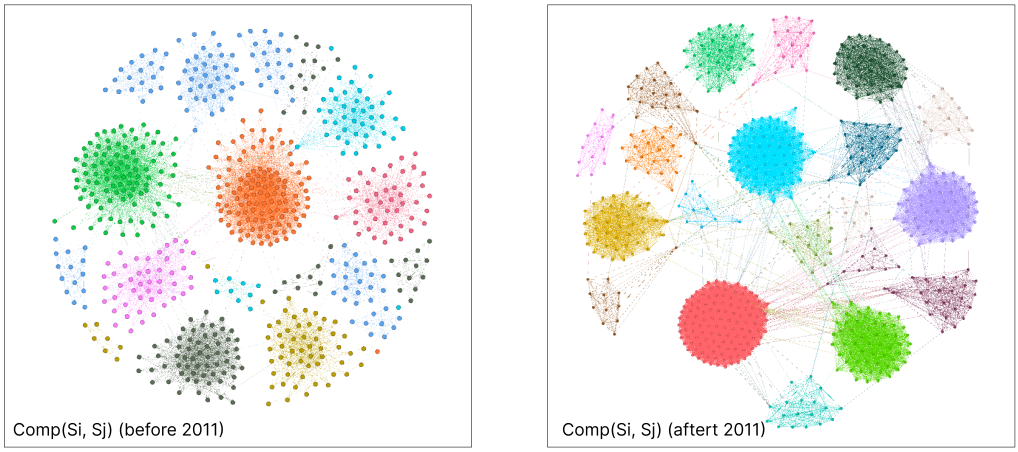

Figure 3.10: The state of the Competition Network communities at the instance T_n and T_{n+1}

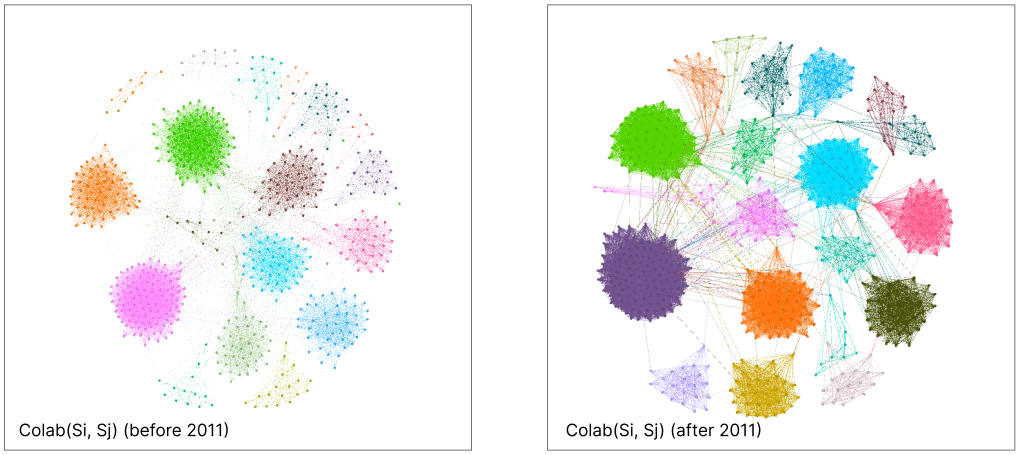

Figure 3.11: The state of the Collaboration Network communities at the instance T_n and T_{n+1}

As we can see the classes used in the conventional method don't have room for development the classes stay the same, there is no relationship between them and no memory of the state of the classes, whereas we can see that in our approach the number of communities can change and there is a relationship between them which makes it better in the process of discovery, we can also use the historical state of the communities to observe their evolution over time.

3.7 Conclusion

In this chapter we went through the process of designing and explaining our social web services architecture and the steps that we made to achieve it from the very first step that is collecting the data all the way to building the classification model and creating the social network, explaining the results and the evolution of the social network as well as comparing it to the conventional method.

"This is not the end, it's not even the beginning of the end, but it may be the end of the beginning. »

— Winston Churchill.

3.8 General Conclusion

In this work, we attempted to provide a new approach for web service discovery based on the service and its history of interaction with its semblances (collaborations and competitions). We introduced web service technology as well as the subject of research projects that gave rise to its two versions: semantic and social. Then, we went through the various strategies for analyzing large amounts of data, focusing specifically on deep learning (Deep Learning), which allows us to propose models for analyzing system memory, which is established using data from web service interactions.

Previous works have only gone so far by either using the ontologies which is only applicable on the RESTful APIs or by using random or constant values to represent gthe relation between the APIs in the process or creating the networks and that's problematic since the relationship between the APIs will have no meaning in the beginning which will increase the chances of its failure and consume far more time in the case successfully building the communities.

To overcome all the mentionned issues we used Natural Language Processing to analyse and create meaningful relationships between the APIs and by that firstly, we build a multi-class classification model using machine learning that is capable of classifying web services, by training it on a Dataset derived from the "programmableweb.com" API collection we also collected two datasets containing the two different timestamps (before 2011) and (after 2011) this datasets were used in the process of creating and initializing the Social Network by creating adjacency matrices using our proposed equations, then observe their evolution and compare them to the conventional method.

As with any research project in this field, we encountered difficulties in obtaining a Dataset with historical data, as well as a Dataset large enough to allow for a better learning phase that would lead for better predictions hence a better network.

In light of our work, it would be more interesting to see a further exploit the communities, analyse them and observe their evolution over the course time by gathering their state as our work has only been to create and initialize the state of the communities.

Bibliography

[1] Naghmeh Niknejad et al. "Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation". In: *Information Systems* 91 (2020), p. 101491.

[2] Nicolas Salatgé. "Conception et mise en oeuvre d'une plate-forme pour la sûreté de fonctionnement des services Web". PhD thesis. Institut National Polytechnique de Toulouse-INPT, 2006.

[3] Jorge Cardoso. *Semantic Web Services: Theory, Tools and Applications: Theory, Tools and Applications*. IGI Global, 2007.

[4] Wikipedia contributors. *Web service — Wikipedia, The Free Encyclopedia*. [Online; accessed 4-June-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Web_service&oldid=1090337961.

[5] Wikipedia contributors. *Web service — Wikipedia, The Free Encyclopedia*. [Online; accessed 11-September-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Web_service&oldid=1099358362.

[6] Erik T Ray. *Learning XML: creating self-describing data*. " O'Reilly Media, Inc.", 2003.

[7] SmartBear. *SOAP vs Rest*. URL: https://smartbear.com/blog/soap-vs-rest-whats-the-difference/.

[8] Jose Sandoval. *RESTful Java Web Services*. Vol. 1. Packt Publishing, 2009.

[9] Wikipedia contributors. *API — Wikipedia, The Free Encyclopedia*. [Online; accessed 3-June-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=API&oldid=1090801103.

[10] Erin Cavanaugh. "Web services: Benefits, challenges, and a unique, visual development solution". In: *white paper, Feb* 10 (2006).

[11] Chrysostomos Zeginis and Dimitris Plexousakis. "Web service adaptation: State of the art and research challenges". In: *Institute of Computer Science, FORTH-ICS, Technical Report* 410 (2010).

[12] Zakaria Maamar, Hakim Hacid, and Michael N Huhns. "Why web services need social networks". In: *IEEE Internet Computing* 15.2 (2011), pp. 90–94.

[13] Ayush Pant. "Introduction to machine learning for beginners". In: *Preuzeto* 19 (2019), p. 2021.

[14] Ashesh Anand and JK Prasad. "An Empirical Study of Foreign Direct Investment in Top Eight in INDIA". In: ().

[15] Howard J Seltman. *Experimental design and analysis*. 2012.

[16] *What is multivariate regression*. https://www.voxco.com/blog/multivariate-regression-definition-example-and-steps/#:~:text=Multivariate%20regression%20is%20a%20technique,the%20correlation%20between%20the%20variables., note = Accessed: 2022-06-06.

[17] Zakaria Jaadi. "A step-by-step explanation of Principal Component Analysis (PCA)". In: *Retrieved June* 7 (2021), p. 2021.

[18] *classification in machine learning*. https://www.edureka.co/blog/classification-in-machine-learning/. Accessed: 2022-06-06.

[19] Emmanuel Gbenga Dada et al. "Machine learning for email spam filtering: review, approaches and open research problems". In: *Heliyon* 5.6 (2019), e01802.

[20] *classification algorithms*. https://monkeylearn.com/blog/classification-algorithms/. Accessed: 2022-06-06.

[21] Elisa Indriasari et al. "Application of Predictive Analytics at Financial Institutions: A Systematic Literature Review". In: *2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI)*. IEEE. 2019, pp. 877–883.

[22] G Thippa Reddy et al. "Analysis of dimensionality reduction techniques on big data". In: *IEEE Access* 8 (2020), pp. 54776–54788.

[23] Krzysztof J Cios et al. "Unsupervised learning: Association rules". In: *Data Mining*. Springer. 2007, pp. 289–306.

[24] *how machine learning works*. https://www.datarobot.com/blog/how-machine-learning-works/. Accessed: 2022-06-06.

[25] Oliver Theobald. *Machine learning for absolute beginners: a plain English introduction*. Vol. 157. Scatterplot press, 2017.

[26] Rob Hierons. *Machine learning. Tom M. Mitchell. Published by McGraw-Hill, Maidenhead, UK, International Student Edition, 1997. ISBN: 0-07-115467-1, 414 pages. Price: UK£ 22.99, soft cover*. 1999.

[27] Laurene V Fausett. *Fundamentals of neural networks: architectures, algorithms and applications*. Pearson Education India, 2006.

[28] WJ Zhang et al. "On definition of deep learning". In: *2018 World automation congress (WAC)*. IEEE. 2018, pp. 1–5.

[29] Uday Kamath, John Liu, and James Whitaker. *Deep learning for NLP and speech recognition*. Vol. 84. Springer, 2019.

[30] Wikipedia contributors. *Transformer (machine learning model) — Wikipedia, The Free Encyclopedia*. [Online; accessed 11-September-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Transformer_(machine_learning_model)&oldid=1109265096.

[31] *BERT language model*. https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model/. Accessed: 2022-06-06.

[32] *How to do Deep Learning on Graphs with Graph Convolutional Networks*. https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780. Accessed: 2022-06-06.

[33] Yuxiao Lin et al. “Bertgcn: Transductive text classification by combining gcn and bert”. In: *arXiv preprint arXiv:2105.05727* (2021).

[34] Liang Yao, Chengsheng Mao, and Yuan Luo. “Graph convolutional networks for text classification”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 7370–7377.

[35] Zakaria Maamar et al. “Using social networks for web services discovery”. In: *IEEE internet computing* 15.4 (2011), pp. 48–54.

[36] Abdelouahab Laachemi and Dalila Boughaci. “A stochastic local search combined with support vector machine for Web services classification”. In: *2016 International Conference on Advanced Aspects of Software Engineering (ICAASE)*. IEEE. 2016, pp. 9–16.

[37] Sidra Shafi and Usman Qamar. “[WiP] Web Services Classification Using an Improved Text Mining Technique”. In: *2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE. 2018, pp. 210–215.

[38] Usman Qamar et al. “A majority vote based classifier ensemble for web service classification”. In: *Business & Information Systems Engineering* 58.4 (2016), pp. 249–259.

[39] Yilei Zhang, Zibin Zheng, and Michael R. Lyu. “WSExpress: A?acronym title="Quality of Service">QoS-aware Search Engine for Web Services”. In: *Proc. IEEE Int’l Conf. Web Services (ICWS’10)*. 2010, pp. 83–90.

[40] Abderrahmane Maaradji et al. “Towards a social network based approach for services composition”. In: *2010 IEEE International Conference on Communications*. IEEE. 2010, pp. 1–5.

[41] Hiba Fallatah, Jamal Bentahar, and Ehsan Khosrowshahi Asl. “Social network-based framework for web services discovery”. In: *2014 international conference on future internet of things and cloud*. IEEE. 2014, pp. 159–166.

[42] Guobing Zou et al. “Deepwsc: A novel framework with deep neural network for web service clustering”. In: *2019 IEEE International conference on Web services (ICWS)*. IEEE. 2019, pp. 434–436.

[43] Hanna Krasnova et al. “Why participate in an online social network? An empirical analysis”. In: (2008).

[44] Guobing Zou et al. “DeepWSC: clustering web services via integrating service composability into deep semantic features”. In: *IEEE Transactions on Services Computing* (2020).